MPC8641D
Integrated Host Processor Family Reference Manual

Supports
MPC8640
MPC8640D
MPC8641
MPC8641D

MPC8641DRM
Rev. 2, 07/2008

freescale™
semiconductor

How to Reach Us:

Home Page:
www.freescale.com

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

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

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

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

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

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

Document Number: MPC8641DRM
Rev. 2, 07/2008

Contents

Contents

Chapter 2
Memory Map

Contents

Contents

Contents

Chapter 6
e600 Core Registers and Instruction Set Summary

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Chapter 10
I^2C Interfaces

Contents

Chapter 11
DUART

Contents

Chapter 12
Local Bus Controller

Contents

Contents

Contents

Chapter 13
Enhanced Three-Speed Ethernet Controllers

Contents

Contents

Contents

Contents

Contents

Contents

Chapter 14
DMA Controller

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Chapter 16
PCI Express Interface Controller

Contents

Contents

Contents

Contents

Contents

Contents

Chapter 18
Device Performance Monitor

Contents

Chapter 19
Debug Features and Watchpoint Facilities

Contents

Appendix A
Complete List of Configuration, Control, and Status Registers

Contents

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

About This Book

This reference manual defines the functionality of the MPC8641 processor family, which integrates either one or two e600 Power Architecture™ cores with system logic required for networking, storage, and general purpose embedded applications. The e600 core implements the PowerPC™ instruction set architecture (ISA) and is described in the *Programming Environments Manual for 32-Bit Implementations of the PowerPC™ Architecture*. This book is intended as a companion to the *PowerPC™ e600 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 MPC8641D integrated host processor at an overview level. The following chapters are included:

- Chapter 1, "MPC8641D Overview," provides a high-level description of features and functionality of the MPC8641D integrated host processor, including its interfaces, and its programming model. The functional operation of the MPC8641D with emphasis on peripheral functions is also described.
- Chapter 2, "Memory Map," describes the memory map of the MPC8641D. An overview of the local address map is followed by a description of how local access windows, address translation and mapping units, and the configuration, control, and status register space are used to define the local address map and the configuration, control, and status registers within that map.
- Chapter 3, "Signal Descriptions," lists all the external signals, cross-references for signals that serve multiple functions, output signal states at reset, and reset configuration signals (and the modes they define).
- Chapter 4, "Reset, Clocking, and Initialization," describes the hard and soft resets, the power-on reset (POR) sequence, power-on reset configuration, clocking, and initialization of the MPC8641.

Part II, "e600 Core," describes the many features of the MPC8641D core processors at an overview level and the interaction between the core complex and the L2 cache. The following chapters are included:

- Chapter 5, "e600 Core Overview," provides an overview of the e600 core processor and the L1 caches and MMU that, together with the core, comprise the core complex.
- Chapter 6, "e600 Core Registers and Instruction Set Summary," provides a listing and description of the e600 registers in reference form.

Part III, "Memory, Peripherals, and I/O Interfaces," defines the memory and I/O interfaces of the MPC8641D and how these blocks interact with one another and with other blocks on the device. The following chapters are included:

- Chapter 7, "MPX Coherency Module (MCM) Overview," defines the e600 coherency module and how it facilitates communication between the e600 core complex, the L2 cache, and the other blocks that comprise the coherent memory domain of the MPC8641.

  The MCM provides a mechanism for I/O-initiated transactions to snoop the MPX bus of the e600 cores in order to maintain coherency across cacheable local memory. It also provides a flexible, easily expandable switch-type structure for e600- and I/O-initiated transactions to be routed (dispatched) to target modules on the MPC8641.
- Chapter 8, "DDR Memory Controllers," describes the DDR SDRAM memory controllers of the MPC8641. These fully programmable controllers 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. Special features like ECC error injection support rapid system debug.
- Chapter 9, "Programmable Interrupt Controller (PIC)," describes the embedded programmable interrupt controller (PIC) of the MPC8641. The PIC 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 10, "I^2C Interfaces," describes the inter-IC (IIC or I^2C) bus controller of the MPC8641. 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 MPC8641D powers up in boot sequencer mode which allows the I^2C controller to initialize configuration registers.
- Chapter 11, "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 12, "Local Bus Controller," describes the local bus controller of the MPC8641. The main component of the local bus controller (LBC) is its memory controller which provides a seamless interface to many types of memory devices and peripherals. The memory controller is responsible for controlling eight memory banks shared by a high performance SDRAM machine, a general-purpose chip-select machine (GPCM), and up to three user-programmable machines (UPMs). As such, it supports a minimal glue logic interface to synchronous DRAM (SDRAM), SRAM, EPROM, Flash EPROM, burstable RAM, regular DRAM devices, extended data output DRAM devices, and other peripherals.
- Chapter 13, "Enhanced Three-Speed Ethernet Controllers," describes the enhanced three-speed Ethernet controllers on the MPC8641. 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 MPC8641D memory coherency module.

- Chapter 14, “DMA Controller,” describes the four-channel general-purpose DMA controller of the MPC8641. The DMA controller transfers blocks of data independent of the e600 core or external hosts. Data movement occurs within the local address space. The DMA controller has four high-speed channels. Both the e600 core and external masters can initiate a DMA transfer. All channels are capable of complex data movement and advanced transaction chaining.
- Chapter 15, “Serial RapidIO Interface,” describes the Serial RapidIO controller, which conforms to Revision 1.2 of the *Parallel RapidIO Interconnect Specification*. This chapter describes the implementation of the serial RapidIO controller.
- Chapter 16, “PCI Express Interface Controller,” complies with the *PCI Express™ Base Specification, Revision 1.0a* (available from http://www.pcisig.org). It is beyond the scope of this manual to document the intricacies of the PCI Express protocol. This chapter describes the PCI Express controller of this device and provides a basic description of the PCI Express protocol.

Part IV, “Global Functions and Debug,” defines other global blocks of the MPC8641. The following chapters are included:

- Chapter 17, “Global Utilities,” defines the global utilities of the MPC8641. These include power management, I/O device enabling, power-on-reset (POR) configuration monitoring, general-purpose I/O signal use, and multiplexing for the interrupt and local bus chip select signals.
- Chapter 18, “Device Performance Monitor,” describes the performance monitor of the MPC8641. Note that the MPC8641D performance monitor is similar to but separate from the performance monitor implemented on the e600 core.
- Chapter 19, “Debug Features and Watchpoint Facilities,” describes the debug features and watchpoint monitor of the MPC8641.

This reference manual also includes the following two appendices, a glossary, and an index.

- Appendix A, “Complete List of Configuration, Control, and Status Registers,” provides a complete listing of the CCSR registers in the order they appear in the CCSR map.
- Appendix B, “Revision History,” lists the major differences between revisions of the *MPC8641D Integrated Host Processor Reference Manual*.

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 ISA and computer architecture in general:

- *The PowerPC Architecture: A Specification for a New Family of RISC Processors*, Second Edition, by International Business Machines, Inc.

  For updates to the specification, see http://www-1.ibm.com/support/docview.wss?uid=pub1sa14208300

- *Computer Architecture: A Quantitative Approach*, Third Edition, by John L. Hennessy and David A. Patterson.
- *Computer Organization and Design: The Hardware/Software Interface*, Second Edition, by David A. Patterson and John L. Hennessy.

Related Documentation

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

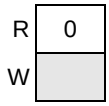
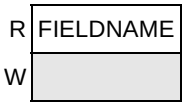
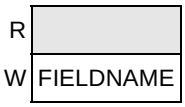
- Reference manuals (formerly called user's manuals)—These books provide details about individual implementations.
- Addenda/errata to reference or user's manuals—Because some processors have follow-on parts, an addendum is provided that describes the additional features and functionality changes. These 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 processors.
- *PowerPC™ e600 Core Complex Reference Manual* (e600CORERM)
- *Programming Environments Manual for 32-Bit Implementations of the PowerPC™ Architecture* (MPCFPE32B)

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

Conventions

This document uses the following notational conventions:

cleared/set	When a bit takes the value zero, it is said to be cleared; when it takes a value of one, it is said to be set.
mnemonics	Instruction mnemonics are shown in lowercase bold.
italics	Italics indicate variable command parameters, for example, **bcctr***x*.
	Book titles in text are set in italics
	Internal signals are set in lowercase italics, for example, $\overline{core\_int}$
0x0	Prefix to denote hexadecimal number
0b0	Prefix to denote binary number
rA, **r**B	Instruction syntax used to identify a source GPR
rD	Instruction syntax used to identify a destination GPR

REG[FIELD]	Abbreviations for registers are shown in uppercase text. Specific bits, fields, or ranges appear in brackets. For example, MSR[LE] refers to the little-endian mode enable bit in the machine state register.
x	In some contexts, such as signal encodings, an unitalicized x indicates a don't care.
x	An italicized *x* indicates an alphanumeric variable
n	An italicized *n* indicates a numeric variable
¬	NOT logical operator
&	AND logical operator
|	OR logical operator
||	Concatenation, for example, TCR[WP]||TCR[WPEXT]
	Indicates a reserved bit field in an e600 register. Although these bits can be written to as ones or zeros, they are always read as zeros.
	Indicates a reserved bit field in a memory-mapped register. Although these bits can be written to as ones or zeros, they are always read as zeros.
	Indicates a read-only bit field in a memory-mapped register.
	Indicates a write-only bit field in a memory-mapped register. Although these bits can be written to as ones or zeros, they are always read as zeros.

Signal Conventions

$\overline{\text{OVERBAR}}$	An overbar indicates that a signal is active-low.
lowercase_italics	Lowercase italics is used to indicate internal signals.
lowercase_plaintext	Lowercase plain text is used to indicate signals that are used for configuration. For more information, see Section 3.1, "Signals Overview."

Acronyms and Abbreviations

Table i contains acronyms and abbreviations used in this document.

Table i. Acronyms and Abbreviated Terms

Term	Meaning
ADB	Allowable disconnect boundary
ATM	Asynchronous transfer mode
ATMU	Address translation and mapping unit
BD	Buffer descriptor

Table i. Acronyms and Abbreviated Terms (continued)

Term	Meaning
BIST	Built-in self test
BRI	Basic rate interface
BTB	Branch target buffer
BUID	Bus unit ID
CAM	Content-addressable memory
CCB	Core complex bus
CCSR	Configuration control and status register
CEPT	Conférence des administrations européenes des postes et télécommunications (European Conference of Postal and Telecommunications Administrations)
COL	Collision
CRC	Cyclic redundancy check
CRS	Carrier sense
DDR	Double data rate
DMA	Direct memory access
DPLL	Digital phase-locked loop
DRAM	Dynamic random access memory
DUART	Dual universal asynchronous receiver/transmitter
EA	Effective address
ECC	Error checking and correction
EEST	Enhanced Ethernet serial transceiver
EHPI	Enhanced host port interface
EPROM	Erasable programmable read-only memory
FCS	Frame-check sequence
FEC	10/100 fast Ethernet controller
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

Table i. Acronyms and Abbreviated Terms (continued)

Term	Meaning
IEEE	Institute of Electrical and Electronics Engineers
IPG	Interpacket gap
IrDA	Infrared Data Association
ISDN	Integrated services digital network
ITLB	Instruction translation lookaside buffer
IU	Integer unit
JTAG	Joint Test Action Group
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
MCM	MPX coherency module
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
PCMCIA	Personal Computer Memory Card International Association
PCS	Physical coding sublayer
PIC	Programmable interrupt controller
PMA	Physical medium attachment
PMD	Physical medium dependent

Table i. Acronyms and Abbreviated Terms (continued)

Term	Meaning
POR	Power-on reset
PRI	Primary rate interface
RGMII	Reduced gigabit media independent interface
RISC	Reduced instruction set computing
RTOS	Real-time operating system
RWITM	Read with intent to modify
RWM	Read modify write
Rx	Receive
RxBD	Receive buffer descriptor
SCC	Serial communication controller
SCP	Serial control port
SDLC	Synchronous data link control
SDMA	Serial DMA
SFD	Start frame delimiter
SI	Serial interface
SIU	System interface unit
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

Table i. Acronyms and Abbreviated Terms (continued)

Term	Meaning
VA	Virtual address
ZBT	Zero bus turnaround

Diagrams

The diagrams in this publication are provided to aid in understanding the overall functionality and features of this product. They do not depict the implementation details of the product, which are subject to change.

Part I
Overview

Part I describes the many features of the MPC8641D integrated processor at an overview level. The following chapters are included:

- Chapter 1, “MPC8641D Overview,” provides a high-level description of features and functionality of the MPC8641D integrated processor. It describes the MPC8641D, its interfaces, and programming model. The functional operation of the MPC8641D, with emphasis on peripheral functions, is also described.
- Chapter 2, “Memory Map,” describes the MPC8641D 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 with cross references to the sections detailing descriptions of each.
- Chapter 3, “Signal Descriptions,” provides a listing of all the external signals, cross-references for signals that serve multiple functions, output signal states at reset, and reset configuration signals (and the modes they define).
- Chapter 4, “Reset, Clocking, and Initialization,” describes the hard and soft resets, power-on reset sequence, power-on reset (POR) configuration, clocking, and initialization of the MPC8641D.

Chapter 1
MPC8641D Overview

The MPC8641 processor family integrates either one or two e600 cores, built on Power Architecture™ technology, with system logic required for networking, storage, wireless infrastructure, and general-purpose embedded applications. The MPC8640 and MPC8641 integrate one e600 core while the MPC8640D and MPC8641D integrate two cores. When referring to the MPC8641 throughout the reference manual, the functionality described applies to the MPC8640, MPC8641, MPC8640D, and the MPC8641D. Any differences in the MPC8640D/MPC8641D from the MPC8640/MPC8641 are noted specifically in the reference manual. The MPC8641 implements resources defined for embedded processors by the PowerPC ISA (instruction set architecture). This chapter provides a high-level description of the features and functionality of the MPC8641 family of integrated processors.

The MPC8641 family use e600 cores and high-speed interconnect technology to balance processor performance with I/O system throughput. The e600 core was previously referred to as the G4 core. The e600 core includes 32-Kbyte separate L1 (level-one) instruction and data caches, and a 1-Mbyte L2 cache. The core is a high-performance superscalar design supporting multiple execution units, including 4 independent units that execute the AltiVec™ instruction set architectural extension for support of 128-bit vector operations. It implements the 32-bit portion of the PowerPC ISA, which provides 32-bit effective addresses; integer data types of 8, 16, and 32 bits; and floating-point data types of 32 and 64 bits. It also provides virtual memory support for up to 4 Pbytes (petabytes) of virtual memory (2^{52}) and real memory support for up to 64 Gbytes (gigabytes) of physical memory (2^{36}). The MPC8641D provides an optional low-memory address offset mode between the dual cores, allowing each processor to see a distinct and private physical address space for a range of addresses starting at address 0x0_0000_0000.

That is, the device provides the ability to translate (remap) a range of addresses; the two cores can have private address space without any special software involvement.

The high level of integration in the MPC8641 helps simplify board design and offers significant bandwidth and performance increase. The MPC8641 has a DDR/DDR2 SDRAM memory controller, a local bus controller (LBC), a programmable interrupt controller (PIC), two I^2C controllers, a four-channel DMA controller, and a dual universal asynchronous receiver/transmitter (DUART). For high speed interconnect, the MPC8641 provides two sets of multiplexed pins that support 2 interface standards: 1x/4x Serial RapidIO (with message unit) and 1x/2x/4x/8x PCI Express. Also, the MPC8641 has 4 integrated 10 Mbps, 100 Mbps, and 1 Gbps Ethernet controllers with TCP offload and classification capabilities.

The two 64-bit (72-bit with ECC) DDR2 SDRAM controllers support most JEDEC standard x8 or x16 DDR2 memories available today, including both buffered and unbuffered DIMMs at up to a 300-MHz clock rate and a 600-MHz data rate. This provides a peak bandwidth of 5.3 Gbytes/s per controller. The built-in error correction codes (ECC) helps ensure data integrity for reliable high-frequency operations. The memory controllers are highly configurable to allow performance tuning to the specific work load. This includes features like memory interleaving on both a bank and controller level, auto precharge or page

modes, and transaction level queueing and reordering. Special features like ECC error injection support rapid system debug. The memory controllers support device densities of 64 Mbits to 4 Gbits. Each controller has four chip select lines, so four banks are supported by each controller. Using 4-Gbit devices, each memory controller can address up to 16 Gbytes, or 32 Gbytes per MPC8641 device.

The MPC8641 offers multiple flexible high-speed I/O ports. These ports can be configured in one of the following ways:

- Dual x1, x2, x4, or x8 PCI Express
- Single x1, x2, x4, or x8 PCI Express and a single 1x or 4x serial RapidIO
- Single x1, x2, x4, or x8 PCI Express
- Single 1x or 4x serial RapidIO

The serial RapidIO port is compliant with the *RapidIO Interconnect Specification, Revision 1.2*. It supports the I/O and message-passing logical specifications, both the 8- and 16-bit common transport specification, and the 1x/4x LP-Serial physical layer specification. It supports 4 priority levels and ordering within a priority level, CRC error management, and 32- to 256-byte transactions. The physical layer of the SRIO unit can operate at 1.25, 2.5, and 3.125 Gbaud per lane. This translates to data rates of up to 10 Gbits/s in each direction with 4 lanes.

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

The PCI Express port is compatible with the *PCI Express Base Specification Revision 1.0a*. It supports either x1, x2, x4, or x8 lane widths with each lane at 2.5 Gbaud (2.0 Gb/s). It is configurable at boot time to act as either a root complex or endpoint. The maximum supported packet payload size is 256 bytes. The interface supports virtual channel 0 (VC0) and traffic class 0 (TC0) only.

The enhanced three-speed Ethernet controllers (eTSECs) each interface to 10/100/1000 Ethernet/IEEE 802.3 networks and devices featuring generic 8-/16-bit FIFO ports. The eTSECs support several standard interfaces to connect to an external Ethernet transceiver: MII, RMII, GMII, RGMII, TBI, and RTBI. Each TSEC or a pair of TSECs can be configured in FIFO mode, which bypasses the Ethernet MAC. This may be used for connecting to ASICs without the overhead of the Ethernet protocol. Four identical eTSECs are available, providing flexible options for connectivity and control access at different speeds.

The eTSEC provides the flexibility to accelerate the identification and retrieval of standard and non-standard protocols carried over Ethernet, including: IPV4, IPV6, TCP, UDP, and VLAN. Checksum operations can be optionally off-loaded to an eTSEC to accelerate existing TCP/IP stacks. On transmission, varying fractions of link bandwidth can be allocated to each of multiple transmit queues through a modified weighted round-robin scheduler. On receive, an arbitrary set of queue selection rules can be programmed into each eTSEC to implement flexible quality of service or firewall strategies based on high-level protocol identification. The eTSECs also provide a register-based statistical module that supports management information base (MIB) remote monitoring (RMON). Without enabling advanced features, each eTSEC can emulate a PowerQUICC III™ three-speed Ethernet controller (TSEC), allowing existing driver software to be re-used.

1.1 Preface

The performance requirements placed on processors continue to increase faster than Moore's law due to the need to support additional bandwidth and additional services. A processor at today's frequencies is not adequate for tomorrow's requirements and process shrinks do not deliver enough additional performance on their own. For applications that can take advantage of thread-level parallelism, a dual core device is an excellent solution.

Both cores of the MPC8641 can be used in a symmetric multiprocessing mode to achieve higher performance. The cores can also be used for separate tasks, potentially running independent operating systems. This gives application developers great flexibility in assigning distinct processing resources to distinct tasks that need guaranteed performance. For instance, one core can manage a data plane and the other a control plane.

The value proposition of a dual core device is further enhanced by including a high degree of peripheral integration typically found on system controllers such as DDR controllers. A device with faster on-chip buses can entirely replace the system controller where a discrete processor without integration was used previously. In the case of this device, the on-chip MPX bus operates three to four times faster than an external MPX bus could. Because MPX bus speed is proportional to memory bandwidth and inversely proportional to memory latency, this has the potential to provide a performance increase to applications limited by either condition.

Many existing applications use two processors running in symmetric multiprocessor mode connected to a system controller by a system bus, as shown in Figure 1-1. These three devices can be replace by a single dual core integrated device, resulting in savings in board space as well as significant performance enhancements.

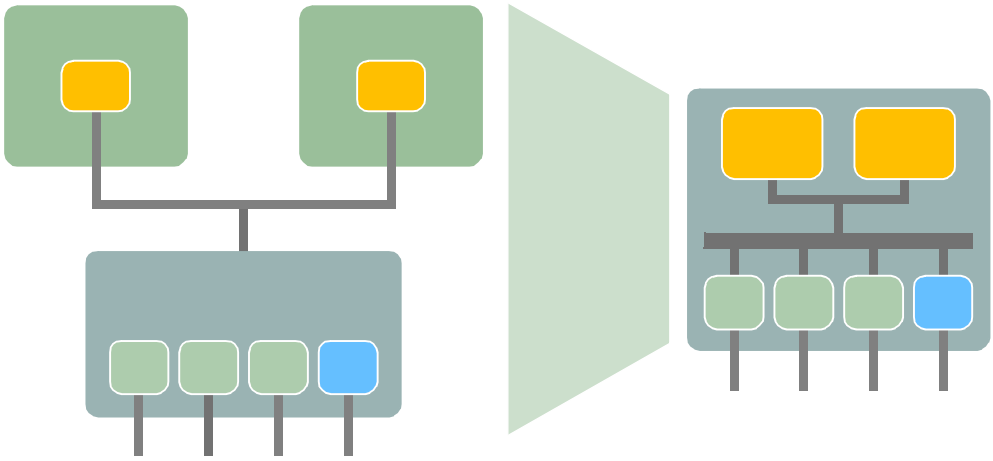


Figure 1-1. Two Processors Running in Symmetric Multiprocessor Mode

1.2 MPC8641 Overview

This section provides a high-level overview of the MPC8641 features. Figure 1-2 shows the major functional units within the MPC8640D and MPC8641D. Note that although the figure shows two cores for the MPC8640D/MPC8641D, the MPC8640 and MPC8641 have just one core.

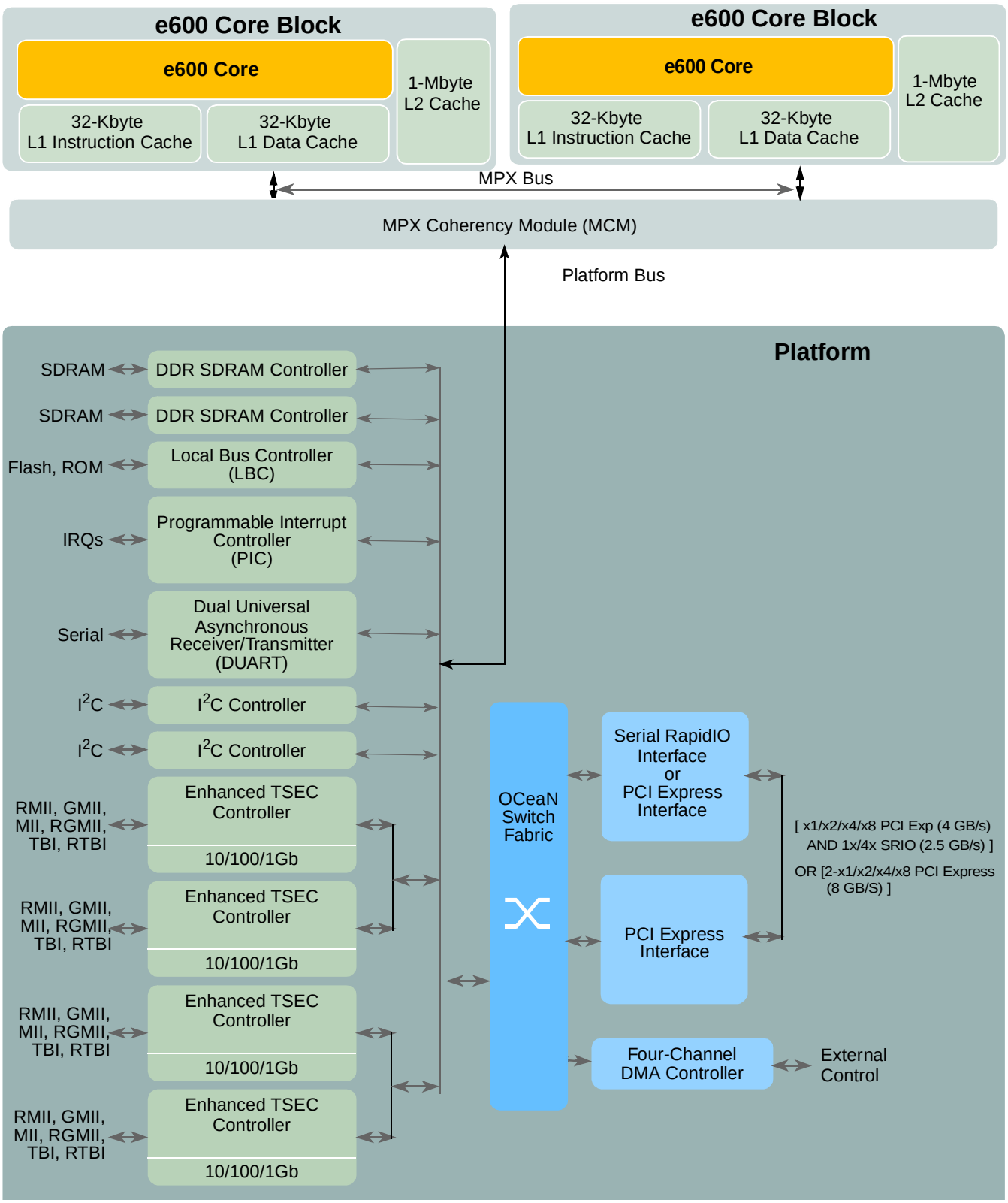


Figure 1-2. MPC8640D/MPC8641D Block Diagram

1.2.1 Key Features

The following lists an overview of the MPC8641 key feature set:

- Major features of the e600 core are as follows:
  - High-performance, 32-bit superscalar microprocessor that implements the PowerPC ISA
  - Eleven independent execution units and three register files
    - Branch processing unit (BPU)
    - Four integer units (IUs) that share 32 GPRs for integer operands
    - 64-bit floating-point unit (FPU)
    - Four vector units and a 32-entry vector register file (VRs)
    - Three-stage load/store unit (LSU)
  - Three issue queues, FIQ, VIQ, and GIQ, can accept as many as one, two, and three instructions, respectively, in a cycle.
  - Rename buffers
  - Dispatch unit
  - Completion unit
  - Two separate 32-Kbyte instruction and data level 1 (L1) caches
  - Integrated 1-Mbyte, eight-way set-associative unified instruction and data level 2 (L2) cache with ECC
  - 36-bit real addressing
  - Separate memory management units (MMUs) for instructions and data
  - Multiprocessing support features
  - Power and thermal management
  - Performance monitor
  - In-system testability and debugging features
  - Reliability and serviceability
- MPX coherency module (MCM)
  - Ten local address windows plus two default windows
  - Optional low-memory offset mode for core 1 to allow for address disambiguation
- Address translation and mapping units (ATMUs)
  - Eight local access windows define mapping within local 36-bit address space
  - Inbound and outbound ATMUs map to larger external address spaces
  - Three inbound windows plus a configuration window on PCI Express
  - Four inbound windows plus a default window on serial RapidIO™ technology
  - Four outbound windows plus default translation for PCI Express
  - Eight outbound windows plus default translation for serial RapidIO technology with segmentation and sub-segmentation support
- DDR memory controllers

  - Dual 64-bit memory controllers (72-bit with ECC)
  - Support of up to a 300-MHz clock rate and a 600-MHz DDR2 SDRAM
  - Support for DDR, DDR2 SDRAM
  - Up to 16 Gbytes per memory controller
  - Cache line and page interleaving between memory controllers.
- Serial RapidIO interface unit
  - Supports *RapidIO Interconnect Specification*, Revision 1.2
  - Both 1x and 4x LP-Serial link interfaces
  - Transmission rates of 1.25-, 2.5-, and 3.125-Gbaud (data rates of 1.0-, 2.0-, and 2.5-Gbps) per lane
  - RapidIO–compliant message unit
  - RapidIO atomic transactions to the memory controller
- PCI Express interface
  - PCI Express 1.0a compatible
  - Supports x1, x2, x4, and x8 link widths
  - 2.5 Gbaud, 2.0 Gbps lane
- Four enhanced three-speed Ethernet controllers (eTSECs)
  - Three-speed support (10/100/1000 Mbps)
  - Four IEEE 802.3, 802.3u, 802.3x, 802.3z, 802.3ac, 802.3ab compliant controllers
  - Support of the following physical interfaces: MII, RMII, GMII, RGMII, TBI, and RTBI
  - Support a full-duplex FIFO mode for high-efficiency ASIC connectivity
  - TCP/IP off-load
  - Header parsing
  - Quality of service support
  - VLAN insertion and deletion
  - MAC address recognition
  - Buffer descriptors are backward compatible with PowerQUICC II and PowerQUICC III programming models
  - RMON statistics support
  - MII management interface for control and status
- 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 and 48 internal interrupts
  - Eight global high resolution timers/counters that can generate interrupts
  - Allows processors to interrupt each other with 32b messages
  - Support for PCI-Express message-shared interrupts (MSIs)

- Local bus controller (LBC)
  - Multiplexed 32-bit address and data operating at up to 133 MHz
  - Eight chip selects support eight external slaves
- Integrated DMA controller
  - Four-channel controller
  - All channels accessible by both the local and the remote masters
  - Supports transfers to or from any local memory or I/O port
  - Ability to start and flow control each DMA channel from external 3-pin interface
- Device 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
- Dual I^2C controllers
  - Two-wire interface
  - Multiple master support
  - Master or slave I^2C mode support
  - On-chip digital filtering rejects spikes on the bus
- Boot sequencer
  - Optionally loads configuration data from serial ROM at reset via the I^2C interface
  - Can be used to initialize configuration registers and/or memory
  - Supports extended I^2C addressing mode
  - Data integrity checked with preamble signature and CRC
- DUART
  - Two 4-wire interfaces (SIN, SOUT, $\overline{\text{RTS}}$, $\overline{\text{CTS}}$)
  - Programming model compatible with the original 16450 UART and the PC16550D
- IEEE 1149.1-compliant, JTAG boundary scan
- Available as 1023 pin Hi-CTE flip chip ceramic land grid (FC-CLGA) and ceramic ball grid array (FC-CBGA)

1.3 MPC8641 Architecture Overview

The following sections describe the major functional units of the MPC8641.

1.3.1 e600 Core

Key features of the e600 core are as follows:

- High-performance, superscalar microprocessor
  - As many as 4 instructions can be fetched from the instruction cache at a time
  - As many as 3 instructions can be dispatched to the issue queues at a time
  - As many as 12 instructions can be in the instruction queue (IQ)
  - As many as 16 instructions can be at some stage of execution simultaneously
  - Single-cycle execution for most instructions
  - One instruction per clock cycle throughput for most instructions
  - Seven-stage pipeline control
- Eleven independent execution units and three register files
  - Branch processing unit (BPU) features static and dynamic branch prediction
    - 128-entry (32-set, four-way set-associative) branch target instruction cache (BTIC), a cache of branch instructions that have been encountered in branch/loop code sequences. If a target instruction is in the BTIC, it is fetched into the instruction queue a cycle sooner than it can be made available from the instruction cache. Typically, a fetch that hits the BTIC provides the first four instructions in the target stream.
    - 2048-entry branch history table (BHT) with two bits per entry for four levels of prediction—not-taken, strongly not-taken, strongly taken
    - Up to three outstanding speculative branches
    - Branch instructions that do not update the count register (CTR) or link register (LR) are often removed from the instruction stream.
    - 8-entry link register stack to predict the target address of Branch Conditional to Link Register (**bclr**) instructions.
  - Four integer units (IUs) that share 32 GPRs for integer operands
    - Three identical low latency IUs (IU1a, IU1b, and IU1c) can execute all integer instructions except multiply, divide, and move to/from special-purpose register instructions.
    - IU2 executes miscellaneous instructions including the CR logical operations, integer multiplication and division instructions, and move to/from special-purpose register instructions.
  - 64-bit floating-point unit (FPU)
    - Five-stage FPU
    - Fully IEEE 754-1985-compliant FPU for both single- and double-precision operations
    - Supports non-IEEE mode for time-critical operations
    - Hardware support for denormalized numbers
    - Thirty-two 64-bit FPRs for single- or double-precision operands

- Four vector units and 32-entry vector register file (VRs) implementing the AltiVec instruction set
  - Vector permute unit (VPU)
  - Vector integer unit 1 (VIU1) handles short-latency AltiVec integer instructions, such as vector add instructions (**vaddsbs**, **vaddshs**, and **vaddsws**, for example)
  - Vector integer unit 2 (VIU2) handles longer-latency AltiVec integer instructions, such as vector multiply add instructions (**vmhaddshs**, **vmhraddshs**, and **vmladduhm**, for example).
  - Vector floating-point unit (VFPU)
- Three-stage load/store unit (LSU)
  - Supports integer, floating-point and vector instruction load/store traffic
  - Four-entry vector touch queue (VTQ) supports all four architected AltiVec data stream operations
  - Three-cycle GPR and AltiVec load latency (byte, half-word, word, vector) with 1 cycle throughput
  - Four-cycle FPR load latency (single, double) with 1 cycle throughput
  - No additional delay for misaligned access within double-word boundary
  - Dedicated adder calculates effective addresses (EAs)
  - Supports store gathering
  - Performs alignment, normalization, and precision conversion for floating-point data
  - Executes cache control and translation lookaside buffer (TLB) instructions
  - Performs alignment, zero padding, and sign extension for integer data
  - Supports hits under misses (multiple outstanding misses)
  - Supports both big- and little-endian modes, including misaligned little-endian accesses

- Three issue queues FIQ, VIQ, and GIQ can accept as many as one, two, and three instructions, respectively, in a cycle. Instruction dispatch requires the following:
  - Instructions can be dispatched only from the three lowest IQ entries—IQ0, IQ1, and IQ2.
  - A maximum of three instructions can be dispatched to the issue queues per clock cycle.
  - Space must be available in the CQ for an instruction to dispatch (this includes instructions that are assigned a space in the CQ but not in an issue queue).
- Rename buffers
  - 16 GPR rename buffers
  - 16 FPR rename buffers
  - 16 VR rename buffers
- Dispatch unit
  - The decode/dispatch stage fully decodes each instruction.

- Completion unit
  - The completion unit retires an instruction from the 16-entry completion queue (CQ) when all instructions ahead of it have been completed, the instruction has finished execution, and no exceptions are pending.
  - Guarantees sequential programming model (precise exception model)
  - Monitors all dispatched instructions and retires them in order
  - Tracks unresolved branches and flushes instructions after a mispredicted branch
  - Retires as many as three instructions per clock cycle
- L1 cache has the following characteristics:
  - Two separate 32-Kbyte instruction and data caches (Harvard architecture)
  - Instruction and data caches are eight-way set-associative.
  - Instruction and data caches have 32-byte cache blocks. A cache block is the block of memory that a coherency state describes—corresponds to a cache line for the L1 data cache.
  - Cache directories are physically addressed. The physical (real) address tag is stored in the cache directory.
  - The caches implement a pseudo least-recently-used (PLRU) replacement algorithm within each way.
  - Cache write-back or write-through operation programmable on a per-page or per-block basis
  - Instruction cache can provide four instructions per clock cycle; data cache can provide four words per clock cycle
    - Two-cycle latency and single-cycle throughput for instruction or data cache accesses.
  - Caches can be disabled in software
  - Caches can be locked in software
  - Supports a four-state modified/exclusive/shared/invalid (MESI) coherency protocol.
    - A single coherency status bit for each instruction cache block allows encoding for the following two possible states:

      Invalid (INV)

      Valid (VAL)
    - Two status bits (MESI[0–1]) for each data cache block allow encoding for coherency, as follows:

      00 = invalid (I)

      01 = shared (S)

      10 = exclusive (E)

      11 = modified (M)
  - Separate copy of data cache tags for efficient snooping
  - Both the L1 caches support parity generation and checking (enabled through bits in the ICTRL register) as follows:
    - Instruction cache—one parity bit per instruction
    - Data cache—one parity bit per byte of data

  - No snooping of instruction cache except for **icbi** instruction
  - Data cache supports AltiVec LRU and transient instructions, as described in Section 1.2.2.2, “AltiVec Instruction Set.”
  - Critical double- and/or quad-word forwarding is performed as needed. Critical quad-word forwarding is used for AltiVec loads and instruction fetches. Other accesses use critical double-word forwarding.
- On-chip level 2 (L2) cache has the following features:
  - Integrated 1-MByte, eight-way set-associative unified instruction and data cache for the e600 core.
  - Fully pipelined every other cycle access to provide 32 bytes per clock cycle to the L1 caches.
  - Total latency of 11.5 processor cycles for L1 data cache miss that hits in the L2 if ECC is disabled. When ECC is enabled, the L2 load access time is 12.5 cycles. The L2 cache is fully pipelined for a 2-cycle throughput.
  - Uses one of two random replacement algorithms (selectable through L2CR).
  - Cache write-back or write-through operation programmable on a per-page or per-block basis
  - Organized as 32 bytes/block and 2 blocks (sectors)/line (a cache block is the block of memory that a coherency state describes).
  - Supports error correction code (ECC) generation and checking for both tags and data (enabled through L2CR).
- Separate memory management units (MMUs) for instructions and data
  - 52-bit virtual address; 32- or 36-bit physical address
  - Address translation for 4-Kbyte pages, variable-sized blocks, and 256-Mbyte segments
  - Memory programmable as write-back/write-through, caching-inhibited/caching-allowed, and memory coherency enforced/memory coherency not enforced on a page or block basis
  - Separate IBATs and DBATs (eight each) also defined as SPRs
  - Separate instruction and data translation lookaside buffers (TLBs)
    - Both TLBs are 128-entry, two-way set-associative, and use LRU replacement algorithm
    - TLBs are hardware- or software-reloadable (that is, on a TLB miss a page table search is performed in hardware or by system software)
- Efficient data flow
  - L1/L2 bus interface allows up to 256 bits.
  - The L1 data cache is fully pipelined to provide 128 bits/cycle to or from the VRs
  - L2 cache is fully pipelined to provide 256 bits per processor clock cycle to the L1 cache.
  - As many as eight outstanding, out-of-order cache misses are allowed between the L1 data cache and L2 bus.
  - As many as 16 out-of-order transactions can be present on the MPX bus
  - Store merging for multiple store misses to the same line. Only coherency action taken (address-only) for store misses merged to all 32 bytes of a cache block (no data tenure needed).

  - Three-entry finished store queue and five-entry completed store queue between the LSU and the L1 data cache
  - Separate additional queues for efficient buffering of outbound data (such as castouts and write-through stores) from the L1 data cache and L2 cache
- Multiprocessing support features include the following:
  - Hardware-enforced, MESI cache coherency protocols for data cache
  - Load/store with reservation instruction pair for atomic memory references, semaphores, and other multiprocessor operations
- Power and thermal management
  - Employs dynamic power management, which automatically minimizes power consumption when idle
  - Three programmable power states are available to the system: full power, nap, and sleep.
  - Instruction cache throttling provides control of instruction fetching to limit device temperature.
  - Temperature diode
    - Can be used in conjunction with external devices to monitor junction temperature
- Performance monitor can be used to help debug system designs and improve software efficiency.
- In-system testability and debugging features through JTAG boundary-scan capability
- Reliability and serviceability

The e600 is the new name for the G4 core. It is a superscalar processor core that can dispatch and complete three instructions simultaneously. The ability to execute several instructions in parallel and the use of simple instructions with rapid execution times yield high efficiency and throughput for e600 core-based systems. Most integer instructions (including VIU1 instructions) have a 1-clock cycle execution latency.

Several execution units feature multiple-stage pipelines; that is, the tasks they perform are broken into subtasks executed in successive stages. Typically, instructions follow one another through the stages, so a four-stage unit can work on four instructions when its pipeline is full. So, although an instruction may have to pass through several stages, the execution unit can achieve a throughput of one instruction per clock cycle.

AltiVec computational instructions are executed in the four independent, pipelined AltiVec execution units. A maximum of two AltiVec instructions can be issued in order to any combination of AltiVec execution units per clock cycle. Moreover, the VIU2, VFPU, and VPU are pipelined, so they can operate on multiple instructions. The VPU has a two-stage pipeline; the VIU2 and VFPU each have four-stage pipelines. As many as 10 AltiVec instructions can be executing concurrently.

Note that for the e600 core, double- and single-precision versions of floating-point instructions have the same latency. For example, a floating-point multiply-add instruction takes five cycles to execute, regardless of whether it is single- (**fmadds**) or double-precision (**fmadd**).

1.3.2 DDR2 SDRAM Controllers

Two memory controllers are provided. The MPC8641 supports a variety of SDRAM configurations that can be configured to support the various memory sizes through software initialization of on-chip configuration registers. SDRAM banks can be built using DIMMs or directly-attached memory devices. Sixteen multiplexed address signals provide for device densities of 64 Mbits to 4 Gbits. Four chip select signals support up to four banks of memory for each memory controller. The MPC8641 supports bank sizes from 64 Mbytes to 4 Gbytes. The theoretical maximum amount of memory supported is achieved using four banks of x8 4-Gbit devices, allowing each memory controller to address up to 16 Gbytes, or 32 Gbytes per MPC8641. Nine input data masks (MDM[0:8]) are used to provide byte selection for memory bank writes. The key features of each memory controller are as follows:

- Programmable timing supporting DDR and DDR2 SDRAM
- 64-bit data interface, up to 300 MHz data rate for DDR and 600 MHz data rate for DDR2
- Support for 4 or 8 sub-banks per chip select
- Interleaving between both memory controllers on either a cache line or page basis
- DRAM chip configurations from 64 Mbits to 4 Gbits organized as x8, x16, and x32 (x4 not supported)
- Full ECC support with eight additional bits
  - Single-error correction, double-error detection within a nibble
- Page mode support
  - Up to 16 simultaneous open pages for 2-bit BA
  - Up to 32 simultaneous open pages for 3-bit BA
- Contiguous or noncontiguous memory mapping
- Read-modify-write support for serial RapidIO atomic increment, decrement, set, and clear transactions
- Stop mode support for self-refresh SDRAM
- On-die termination support when using DDR2
- Supports auto refreshing
- On-the-fly power management using CKE signal
- Registered DIMM support
- 2.5 V SSTL_25 compatible I/O (1.8 V SSTL_1.8 for DDR2)
- Error injection for diagnostic purposes

The MPC8641 can be configured to retain the currently active SDRAM page for pipelined burst accesses. Page mode support of up to 16 or 32 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 MPC8641 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 (power-saving) mode.

Interleaving between memory controllers has two significant advantages. First, it allows the processor to take advantage of the bandwidth of both memory controllers by automatically balancing accesses across

both without requiring the processor to attempt to achieve this using software. Second, it allows twice as many DRAM pages with spatial locality to be open at once. This can increase the likelihood that an access hits an open page, thus reducing latency.

1.3.3 High-Speed I/O Interfaces

The MPC8641 supports the following high-speed I/O interface standards: serial RapidIO, and PCI Express. Due to pin multiplexing, one of the following configurations must be selected during configuration:

- Single x1, x2, x4, or x8 PCI Express and single 1x or 4x serial RapidIO
- Dual x1, x2, x4, or x8 PCI Express
- Single x1, x2, x4, or x8 PCI Express
- Single 1x or 4x serial RapidIO

1.3.3.1 Serial RapidIO Interface

The serial RapidIO interface on the MPC8641 is based on the *RapidIO Interconnect Specification*, Revision 1.2. serial 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 serial 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. Major feature of the serial RapidIO interface are as follows:

- Supports *RapidIO Interconnect Specification*, Revision 1.2
- Both 1x and 4x LP-Serial link interfaces
- Long- and short-haul electricals with selectable pre-compensation
- Transmission rates of 1.25-, 2.5-, and 3.125-Gbaud (data rates of 1.0-, 2.0-, and 2.5-Gbps) per lane
- Auto detection of 1x and 4x mode operation during port initialization
- Link initialization and synchronization
- Large and small size transport information field support selectable at initialization time
- 34-bit addressing
- Up to 256 bytes data payload
- All transaction flows and priorities
- Atomic set/clr/inc/dec for read-modify-write operations
- Generation of IO_READ_HOME and FLUSH with data for accessing cache-coherent data at a remote memory system
- Receiver-controlled flow control
- Error detection, recovery and time-out for packets and control symbols as required by the *RapidIO Interconnect Specification*, Revision 1.2
- Register and register bit extensions as described in Part VIII: Error Management of the *RapidIO Interconnect Specification*, Revision 1.2
- Hardware recovery only

- Register support is not required for software-mediated error recovery.
- Accept-all mode of operation for failover support
- Support for serial RapidIO error injection
- Internal LP-Serial and application interface-level loopback modes
- Memory and PHY BIST for at-speed production test

The serial RapidIO unit on the MPC8641 supports the I/O and message-passing logical specifications, both 8 and 16-bit common transport specification, and the 1x/4x LP-Serial physical layer specification of the *RapidIO Interconnect Specification.* It does not support the globally shared memory logical specification.

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

1.3.3.1.1 Serial RapidIO Message Unit

- 4 Kbytes of payload per message
- Up to sixteen 256-byte segments per message
- Two inbound data message structures within the inbox
- Capable of receiving three letters at any mailbox
- Two outbound data message structures within the outbox
- Capable of sending three letters simultaneously
- Chaining and direct modes in the outbox
- Single inbound doorbell message structure
- Facility to accept port-write messages

The MPC8641 serial RapidIO messaging unit supports two inbox/outbox mailboxes (queues) for data and one doorbell message structure. Both chaining and direct modes are provided for the outbox, and messages can hold up to 16 packets of 256 bytes, or a total of 4 Kbytes.

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

1.3.3.2 PCI Express Interface

- PCI Express 1.0a compatible
- Supports x1, x2, x4 and x8 link widths
- 2.5 Gbaud, 2.0 GHz data rate per lane
- Auto-detection of number of connected lanes
- Selectable operation as root complex or endpoint
- 256-byte maximum payload size
- Virtual channel 0 only
- Traffic class 0 only
- Full 64-bit decode with 32-bit wide windows

The MPC8641 supports up to two x1, x2, x4, or x8 PCI Express interfaces compliant with the *PCI Express Base Specification Revision 1.0a*.

1.3.4 Enhanced Three-Speed Ethernet Controllers

The MPC8641 has four enhanced on-chip three-speed Ethernet controllers called eTSECs. The eTSECs incorporate a media access control sublayer (MAC) that supports 10- and 100-Mbps, and 1-Gbps Ethernet/802.3 networks with MII, RMII, GMII, RGMII, RTBI, and TBI physical interfaces. The eTSECs include 2 Kbyte receive and 10 Kbyte transmit FIFOs, and DMA functions. Major features are as follows:

- Ethernet controllers
  - Three-speed support (10/100/1000 Mbps)
  - Four IEEE 802.3, 802.3u, 802.3x, 802.3z, 802.3ac, 802.3ab compliant controllers
  - Each eTSEC can be independently configured to support one of the following Ethernet physical interfaces:
    - 10/100/1000 Mbps IEEE 802.3 GMII
    - 10/100 Mbps IEEE 802.3 MII and RMII
    - 1000 Mbps full-duplex RGMII
    - 1000 Mbps IEEE 802.3z TBI
  - Each eTSEC can be configured to support a full-duplex FIFO mode for high-efficiency ASIC connectivity
    - Up to four 8-bit wide FIFOs —GMII style and encoded packet
    - Up to two 16-bit wide FIFOs —GMII style and encoded packet
    - Inter-packet and intra-packet flow control
    - Minimal glue logic required to support POS PHY level 3 conversion
    - Optional CRC validation and generation
  - Full and half-duplex Ethernet support (1000 Mbps supports only full-duplex):
    - IEEE 802.3 full-duplex flow control (automatic PAUSE frame generation or software programmed PAUSE frame generation and recognition)
    - Re-transmission following a collision
    - Support CRC generation and verification of inbound/outbound packets
    - Programmable Ethernet preamble insertion and extraction of up to 7 bytes
    - Programmable maximum frame length supports jumbo frames (up to 9.6 Kbytes) and IEEE 802.1 virtual local area network (VLAN) tags and priority
  - VLAN insertion and deletion
    - Per-frame VLAN control word or default VLAN for each eTSEC
    - Extracted VLAN control word passed to software separately
  - MAC address recognition:
    - Exact match on primary and virtual 48-bit unicast addresses
    - VRRP and HSRP support for seamless router fail-over

  - Up to sixteen exact-match MAC addresses supported
  - Broadcast address (accept/reject)
  - Hash table match on up to 512 multicast addresses
  - Promiscuous mode
- TCP/IP off-load
  - IPv4 and IPv6 header recognition on receive
  - IPv4 header checksum verification and generation
  - TCP and UDP checksum verification and generation
  - Per packet configurable off-load
  - Recognition of VLAN, 802.2, PPPoE session, MPLS stacks, and ESP/AH IP-Security headers
  - Supported in Ethernet and FIFO modes
- Quality of service support:
  - Transmission from up to 8 queues with weighted fair queueing
  - Reception to up to 8 physical and 64 virtual queues per eTSEC
  - Supported in Ethernet and FIFO modes
- Buffer descriptors are backward compatible with MPC8260 and MPC860T 10/100 programming models
- RMON statistics support
- 10-Kbyte internal transmit and 2-Kbyte receive FIFOs
- MII management interface for control and status

The MPC8641 eTSECs support programmable CRC generation and checking, RMON statistics, and jumbo frames of up to 9.6 Kbytes. Each eTSEC provides hardware support for accelerating the basic functions of TCP/IP packet transmission and reception. By default, TCP/IP off-load (TOE) functions are not enabled and the eTSECs process frames as pure Ethernet frames.

TOE can be enabled at a number of levels. The eTSEC can parse frames at layer 2 of the stack only (Ethernet headers and switching headers), layers 2 to 3 (including IPv4 or IPv6), or layers 2 to 4 (including TCP and UDP).

On receive, the eTSEC provides protocol header recognition, header verification (IPv4 header checksum verification), and TCP/UDP payload checksum verification including verification of associated pseudo-header checksums. On transmit, TOE provides IPv4 and TCP/UDP header checksum generation. eTSEC does not checksum transmitted packets with IP header options or IP fragments.

To provide for quality of service, transmission from up to 8 queues is supported with priority-based queue selection. Arbitration is a modified weighted round-robin queue selection with fair bandwidth allocation.

On receive, packets may be distributed to up to 8 physical queues with 64 virtual receive queues overlaid on the 8 physical buffer descriptor rings. A table-oriented queue filing strategy is provided based on 16 headers fields or flags. Frame rejection is supported for filtering applications. Filing can be based on

Ethernet, IP, and TCP/UDP properties, including VLAN fields, Ether-type, IP protocol type, IP TOS or differentiated services, IP source and destination addresses, and TCP/UDP port numbers.

Each eTSEC provides a full-duplex packet FIFO interface port that bypasses the Ethernet MAC, but re-uses the PHY interface pins. As a result, the FIFO interface normally does not impose the overhead of Ethernet framing. The FIFO interface operates synchronously, at up to 200 MHz, providing more than OC-48 full-duplex transfer rates. Bare IP packets, with an optional 32-bit CRC check sequence, can be transferred to the eTSECs directly. The eTSEC Tx and Rx FIFOs, TOE functions, and DMA functionality continue to be used in packet FIFO mode.The FIFO interface supports 8 and 16-bit modes. The following configurations are possible:

- Two 16-bit FIFO interfaces
- One 16-bit FIFO interface plus two 8-bit FIFOs
- One 16-bit FIFO interface plus one 8-bit FIFO interface plus one eTSEC with MII/RMII, GMII, RGMII, TBI, or RTBI interface
- One 16-bit FIFO interface plus two eTSECs with MII, RMII, GMII, RGMII, TBI, or RTBI
- Four 8-bit FIFO interfaces
- Three 8-bit FIFO interfaces plus one eTSEC with MII, RMII, GMII, RGMII, TBI, or RTBI
- Two 8-bit FIFO interfaces plus two eTSECs with MII, RMII, GMII, RGMII, TBI, or RTBI
- One 8-bit FIFO interface plus three eTSECs with MII, RMII, GMII, RGMII, TBI, or RTBI
- Four eTSECs with MII, RMII, GMII, RGMII, TBI, or RTBI

1.3.5 MPX Coherency Module (MCM)

The MPX coherency module (MCM) provides coherency management for the device. The MCM is responsible for tracking the coherent state of transactions that are part of the coherent domain. It handles any necessary reordering to resolve all coherent memory references.For the MPC8641D, the MCM is also responsible for managing cache to cache intervention between processor cores. Once addresses are snooped clean they are sent on to the memory controller. Once a transaction has been accepted by the MCM it is considered globally visible.

1.3.6 Low Memory Offset Mode (Core 1)

In certain dual core, dual operating system applications, each processor must see a distinct and private physical address space for a range of addresses starting at address 0x0_0000_0000. For example, in a dual e600 core system, each OS may wish to point to unique exception handler routines so a means must be provided for each core to see these real addresses in separate physical address spaces. Other system considerations may also necessitate creation of a private range of addresses in some range of low memory.

To facilitate this, the device provides the ability to translate (remap) 256 Mbytes of addresses starting at address 0x_0000_0000 for e600 core 1 through a low memory offset mode. (No translation capability is provided for e600 core 0.) A configuration signal sampled at power-on reset enables this capability.

1.3.7 Address Translation and Mapping Units (ATMUs)

When processing across the on-chip fabric, the ATMUs at each fabric port are used to determine the flow of data across the MPC8641. There are ten programmable local access windows (plus two default windows) that define the mapping within the local 36-bit address space. Inbound and outbound ATMUs map to larger external address spaces as follows:

- Three inbound windows plus a configuration window on PCI Express
- Four inbound windows plus a default window on serial RapidIO
- Four outbound windows plus default translation for PCI Express
- Eight outbound windows plus default translation for serial RapidIO with segmentation and sub-segmentation support

1.3.8 Local Bus Controller (LBC)

The MPC8641 local bus controller (LBC) port allows connections with a wide variety of external memories, DSPs, and ASICs. Two 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. Each chip select can be configured so that the associated chip interface can be controlled by the GPCM or UPM controller. Both may exist in the same system. Major features are as follows:

- Multiplexed 32-bit address and data operating at up to 133 MHz
- Up to eight-beat burst transfers
- The 32-, 16-, and 8-bit port sizes are controlled by an on-chip memory controller.
- Two protocol engines available on a per chip select basis:
  - General-purpose chip select machine (GPCM)
  - Three user programmable machines (UPMs)
- Parity support
- Default boot ROM chip select with configurable bus width (8-,16-, or 32-bit)
- Six general purpose output pins

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. Six UPM pins can be configured as general purpose output pins. They are programmed by writing into the UPM array.

1.3.9 Four-Channel DMA Controller

The MPC8641 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. Some features include:

- Integrated DMA 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

DMA transfers can be initiated by either local or remote masters 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 serial RapidIO port. Local attributes, such as whether transactions should be snooped, 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 (with the snoop/no snoop attribute).

1.3.10 Programmable Interrupt Controller (PIC)

The programmable interrupt controller (PIC) is replicated for both e600 cores and implements the necessary functions to provide a flexible solution for a general-purpose interrupt control. The interrupt controller implements the logic and programming structures of the OpenPIC architecture. Inter-processor interrupt (IPI) communication is supported through the external interrupt and core reset signals of the e600 processor cores on this device.

- OpenPIC compatible programming model
- Processor-to-processor interrupts with 32-bit messages
- Support for 12 external and 48 internal interrupt sources.
- 16 programmable interrupt priority levels
- Four interprocessor interrupt channels for the MPC8641D
- Four 32-bit messaging interrupt channels for a total of 256 interrupts that are used with message shared interrupts (MSIs)
- Fully-nested interrupt delivery
- Processor initialization control

- Programmable resetting of the PIC unit through the global configuration register
- Interrupts can be routed to external pin 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, IRQ_OUT
- Recovery from spurious interrupts
- Eight global high resolution timers/counters that can generate interrupts
- Interrupts can be routed to the e600 cores' standard interrupt inputs or the machine check interrupt
- Interrupt summary registers allow fast identification of interrupt source.

1.3.11 I^{2}C Controller

The MPC8641 supports two I^{2}C controllers. The inter-IC (IIC or I^{2}C) bus is a two-wire, bidirectional serial bus that provides a simple, efficient method of data exchange between devices. The synchronous, multi-master bus of the I^{2}C allows the MPC8641 to exchange data with other I^{2}C 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 I^{2}C controller is a true multimaster bus which includes collision detection and arbitration that prevents data corruption if two or more masters attempt to control the bus simultaneously. This feature allows for complex applications with multiprocessor control. The I^{2}C controller consists of a transmitter/receiver unit, a clocking unit, and a control unit. The dual I^{2}C units supports general broadcast mode and on-chip filtering rejects spikes on the bus.

1.3.12 Boot Sequencer

The MPC8641 provides a boot sequencer that uses the I^{2}C interface to access an external serial ROM and loads the data into the MPC8641's configuration registers. The boot sequencer is enabled by a configuration pin sampled at the negation of the MPC8641 hardware reset signal. If enabled, the boot sequencer holds the MPC8641 e600 processor cores in reset until the boot sequence is complete. If the boot sequencer is not enabled, the booting e600 processor core exits reset and fetches boot code with default configurations.

1.3.13 Dual Universal Asynchronous Receiver/Transmitter (DUART)

The MPC8641 includes a DUART intended for use in maintenance, bringing-up, and debugging of systems. The MPC8641 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 MPC8641 DUART supports a 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.

1.3.14 Power Management

The e600 core is designed for low-power operation. It provides both automatic and program-controlled power reduction modes. If an e600 core's functional unit is idle, it automatically goes into a low-power mode. This mode does not affect operational performance.

Dynamic power management automatically supplies or withholds power to execution units individually, based upon the contents of the instruction stream. The e600 core execution units have an independent clock input that is controlled automatically on a clock-by-clock basis; for example, clocking is withheld from the floating-point unit if no floating-point instructions are being executed. The operation of dynamic power management is transparent to software or any external hardware. To save power, the memory controller negates the CKE signal to the DRAM when transactions are not being issued to the DRAM.

The e600 core provides an instruction cache throttling mechanism to effectively reduce the instruction execution rate without the complexity and overhead of dynamic clock control. When used with dynamic power management, instruction cache throttling provides the system designer with a flexible way to control device temperature while allowing the processor to continue operating.

The instruction cache throttling mechanism simply reduces the instruction dispatch rate. Normally, as many as three instructions are dispatched each cycle. For thermal management, the e600 core provides a supervisor-level instruction cache throttling control register (ICTC) that sets a specific number (1–255) of dead cycles between an instruction.

1.3.15 Clocking

The MPC8641 has a system clock input. The e600 core operates at a multiple of this clock.

The device logic, including the DRAM interface, is synchronous to the system clock. The multipliers define the relationship between the core speed and the DRAM speed. For instance, if a 300 MHz clock rate (600 MHz data rate) DDR2 DRAM is used and a multiplier of 2.5 is used, then the core will run at 1.5 GHz (600 MHz × 2.5). The L1 and L2 caches run at the same frequency as the core. The lowest available multiple is 2:1. See Section 4.4.3.3, "e600 Core PLL Ratio," for power-on reset (POR) configuration options.

The DRAM data rate operates at the same rate as the MPX bus. The local bus is also synchronous to the DRAM speed, with multipliers of 2, 4, 8, and 16 between the local bus speed and the DRAM data rate. Two clock outputs are generated. The OCN switch fabric and its peripherals operate at platform frequency if the platform frequency is less than or equal to 400 MHz. Otherwise, if platform frequency is 500 MHz or higher, OCN switch fabric and peripherals operate at one-half of platform frequency. See power-on reset (POR) configuration details in the Reset chapter for clocking configurations.

Each of the four eTSECs can be clocked independently. The high speed interface (serial RapidIO and PCI Express) ports all share two independent PLLs.

The MPC8641 generates six differential pairs of outputs for each DDR memory controller.

1.3.16 Address Map

The MPC8641 supports a flexible 36-bit physical address map. Conceptually, the address map consists of local space and external address space.

The local address map spans 64 Gbytes. The e600 core uses a 52-bit interim virtual address to address 4 Pbytes of virtual memory, and a 36-bit physical address to address 64 Gbytes of physical memory. The MPC8641 can be made part of an even 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 allow the MPC8641 to be part of larger address maps such as the PCI Express 64-bit address environment and the serial RapidIO environment.

1.4 Navigating Transactions Between the eTSECs and I/O Ports

Protocol data units (PDUs) can navigate through the various MPC8641 I/O ports in two ways. With the first example, this is managed by the core. In the second, a remote processor manages this.

1.4.1 Data Traffic is Routed Between eTSEC and the PCI Express Port for Transmission

The following sections describe how data traffic can be routed between the eTSECs and the PCI express port.

1.4.1.1 eTSEC to PCI Express

In this example, a bus master programs the eTSEC to handle reception of an Ethernet frame during initialization. Under user software control, the master builds a receive descriptor table for the eTSEC – the 8B descriptor describes the address where there is room for the data to be accumulated and when to transfer data (data transfer occurs when the buffer is empty). Also during initialization, the MCM is programmed to define the address ranges for memory and the PCI Express windows.

Once the initialization is complete, the eTSEC is ready to transmit and receive traffic.

If the receiver detects the first bytes of a frame, the eTSEC controller begins to perform the frame recognition function through destination address (DA) recognition. Based on this match the frame can be accepted or rejected. The receiver can filter frames based on individual (unicast), group (multicast) and broadcast addresses. Because Ethernet receive frame data is not written to memory until the internal frame recognition algorithm is complete, internal bus bandwidth is not wasted on frames unwanted by this station. The CAM-based filer also classifies the packet based on matching a combination of standard and user-defined fields.

If a frame is accepted, the Ethernet controller fetches the receive buffer descriptor (RxBD) from either queue 0 or the queue determined by the filer. If the RxBD is not being used by software (RxBD[E] is set), the eTSEC’s DMA engine starts transferring the incoming frame to the address in SDRAM specified by the RxBD. RxBD[F] is set for the first RxBD used for any particular receive frame. If the current RxBD is not available for the received frame, a receive busy error condition is flagged in IEVENT[BSY].

After the buffer is filled, the eTSEC clears RxBD[E] and, if RxBD[I] is set, an interrupt is generated. 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.

At this point, the appropriate core (selected by the destination register in the PIC) is interrupted by the eTSEC, indicating that a new frame has arrived. Software will read the descriptor and typically part of the packet header to make a decision about what to do with it. In this example, the data is being forwarded out through the PCI Express port. Software programs the central DMA engine with the location of the packet. The DMA engine can be configured to use the descriptor used by the eTSEC, or a new descriptor can be created.

When the DMA engine is started, it sets up a transfer operation to move the data from memory and send it to the PCI Express port via the OCeaN switch. The DMA transfer uses the source address of the data and the destination address of the PCI Express port that were established by the MCM during initialization. The actual data transfer can be started by the user software or by an external I/O controller.

The OCean switch looks at the target ID and forwards it to the PCI Express interface. This 32-bit address is checked to see if it matches the outbound ranges—it is then put in as a request packet, packetized with the target ID and type of write and forwarded to the PCI Express interface. The PCI Express interface converts the packet to PCI Express format using the ATMU and transmits the data.

Once the packet is transmitted, the processor sets the empty bit (RxBD[E]) so that the buffer is returned to the eTSEC.

1.4.1.2 PCI Express to eTSEC

The MPC8641x PCI Express port can be used to send data through the MPC8641x and out of the eTSEC to a particular destination address.

The first step in this process is initialization. During initialization, a master programs the eTSEC to handle Rx and Tx of an eTSEC frame. A typical initialization sequence is described below:

The master builds an 8B descriptor table for the eTSEC; these 8B descriptors define the address where there is room for the data to be accumulated. During initialization, the MCM defines the address ranges for memory, the PCI Express windows, etc. Once initialization is complete, the eTSEC is ready to transmit and receive traffic.

Data is first transferred through the PCI Express port to the OCeaN switch. The OCeaN switch initiates a DMA transfer of the data, in 32-byte chunks, to the address location in DDR SDRAM that was established for PCI Express data during initialization. This should match buffers pointed to by the eTSEC's buffer descriptors.

The processor updates the TxBD's ready bit, indicating that the buffer is full and ready to be transmitted. The eTSEC polls this bit and when it finds the bit is set, transmission commences.

The eTSEC can perform a read of the first descriptor from the descriptor table to find out the buffer address associated with that descriptor. If data is available, the eTSEC uses its internal DMA to pull a 32-byte burst of data from the associated address in DDR SDRAM. Once that first descriptor is read, an internal pointer increments the descriptor table by 8 bytes so the next descriptor is available. The eTSEC is ready to read the second descriptor as soon as the first 32-byte data DMA operation is complete.

Data that is transferred to the eTSEC during the DMA operation is forwarded to the eTSEC's data processing engine via the Tx FIFO. The Ethernet preamble, start of frame delimiter and PADS and CRC are added and the packet is ready for transmission.

1.5 MPC8641D Application Examples

1.5.1 Dual-Core Device Considerations

There are three main ways that operating systems (OSs) can be mapped to the two cores of the MPC8641D.

- Symmetric multiprocessing (SMP)
- Distributed processing with homogeneous OSs
- Distributed processing with heterogeneous OSs

Figure 1-3 shows a system that exemplifies many of the ways that an integrated dual core device like the MPC8641D can used. A high-end line card uses an ASIC or NPU (network processing unit) for the data path. The ASIC/NPU manages user interfaces on the faceplate on the left as well as the interface to the backplane on the right. The dual-core device is responsible for the control plane. The two cores can operate in an SMP configuration or two separate operating systems may be used for separate control plane tasks.

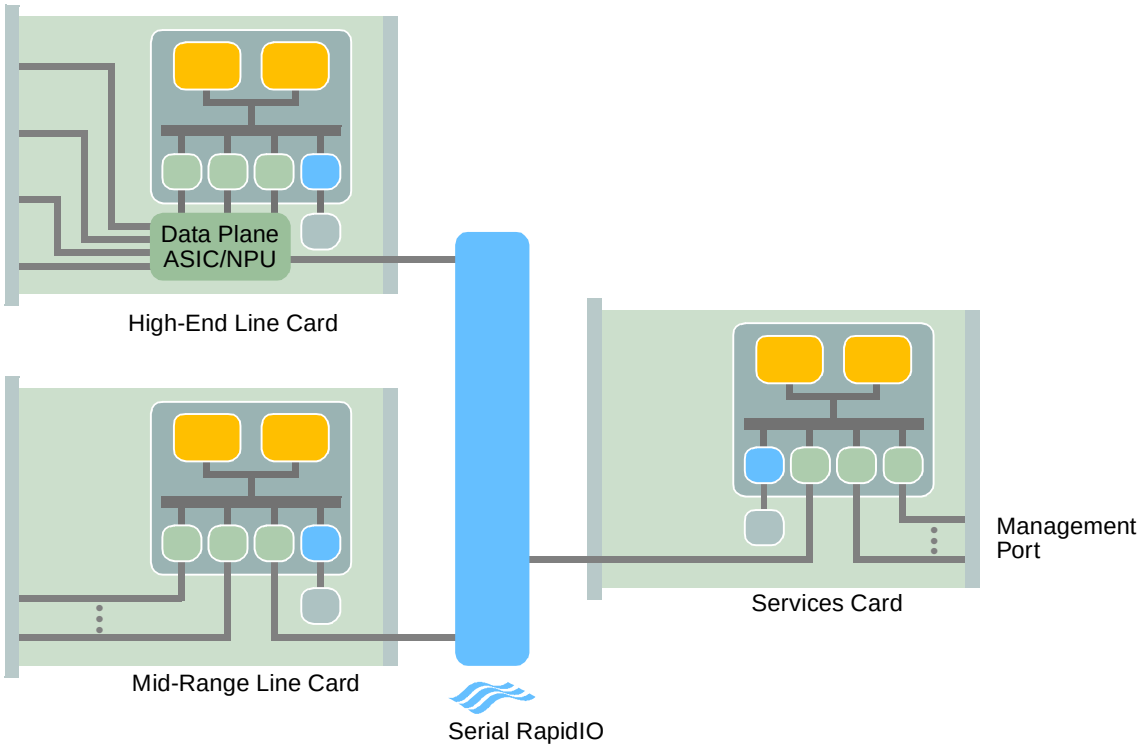


Figure 1-3. Integrated Dual Core Device Application

The lower line card shows a mid-range system. Here, the eTSEC interfaces implement the faceplate interfaces, and the serial RapidIO block connects to the backplane switch fabric. The CPUs implement both the control and data plane, and can be organized in a variety of ways. Two popular examples include

splitting functionality directionally (one core per direction) or splitting functionality vertically (one core handling data plane, one for control plane).

On the right of the switch fabric is an example of a service card. This can be added to a system to easily add new features without needing to replace all the line cards with upgraded ASICs. One service card supports a new feature set in a centralized manner, receiving traffic from all line cards. Because of this, the high performance of a dual-core device is required. The serial RapidIO port connects to the fabric and the eTSECs implement a management interface.

There are two main ways that operating systems can be mapped to the two cores of the MPC8641D.

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

1.5.1.1 Symmetric Multiprocessing Configuration

In a symmetric multiprocessing configuration, both cores operate the same SMP-enabled OS. The OS can share tasks across two cores and determine how to share peripherals on the device, as shown in Figure 1-4. Two cores running an SMP OS may achieve up to 190 percent of the performance of a single core device. SMP would typically be used on the MPC8641 to achieve the highest performance. One common application is to collapse two MPC7447-class products running with a system controller into a single device. SMP is supported with a variety of features:

- MESI (modified exclusive shared invalid) cache coherency protocol
- Intervention, which allows direct cache-to-cache updates.
- Enforcement of ordering on storage operations
- TLB coherency

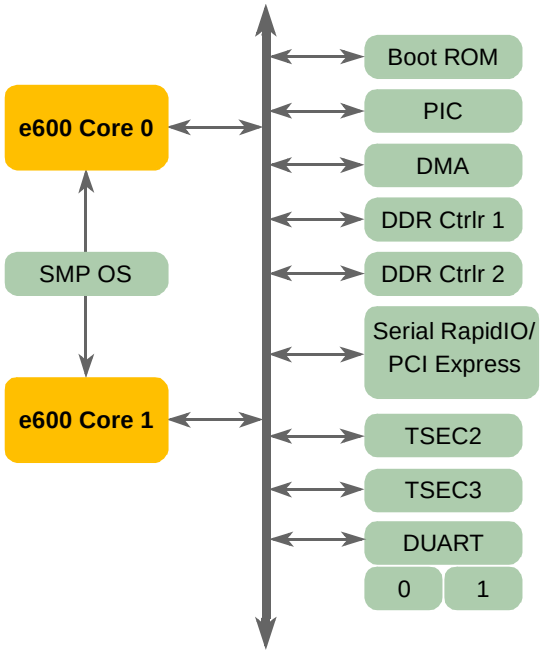


Figure 1-4. Symmetric Multiprocessing Example

Figure 1-5 shows how a memory map can be set up in an SMP system. Each core has a memory management unit (MMU) that is responsible for converting the program's effective address to a physical address (not shown) that is seen within the rest of the MPC8641D, including the MPX bus, the MCM, and the peripherals. The effective address is translated by the memory controller into a physical address that addresses a DRAM. The interrupt vectors must reside at physical address zero because that is where the interrupt service routines are located in the PowerPC ISA.

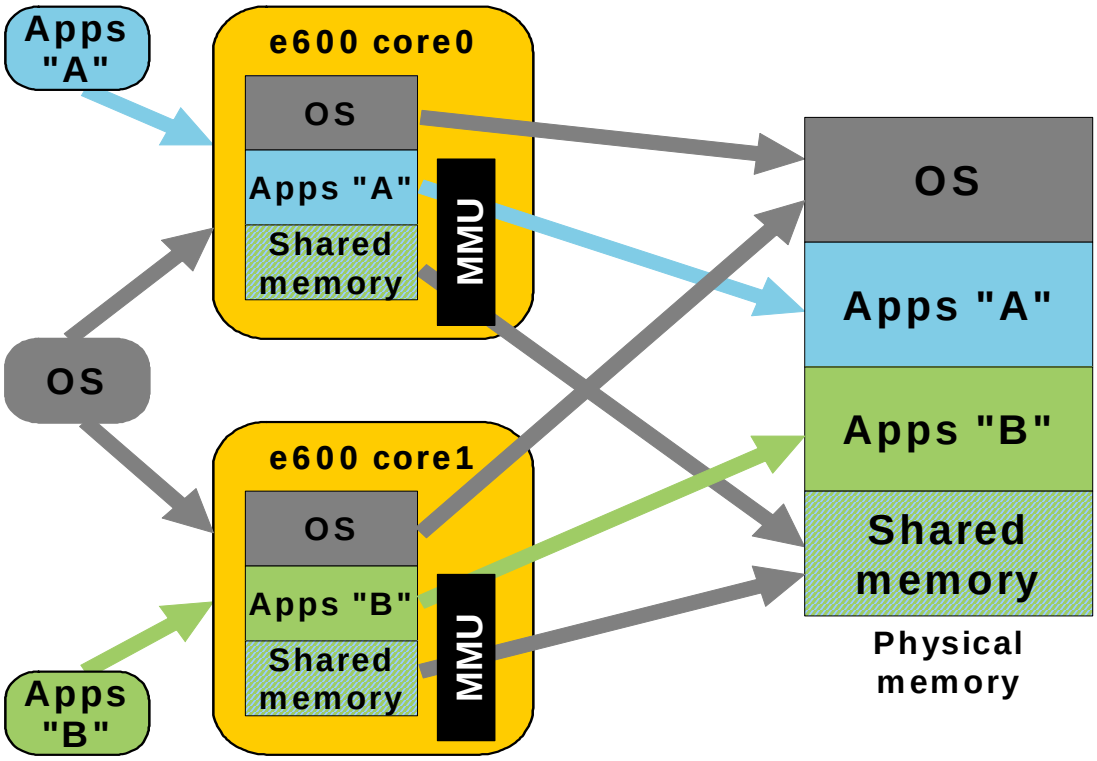


Figure 1-5. Memory Mapping in an SMP System

Each e600 core maps its effective address 0 to physical address 0. Thus, each e600 core accesses the same kernel in main memory. Following the kernel in the effective address space may be application-specific data. This contains information that about the tasks running on each core. Although the effective addresses of the application data is the same in both cores, the MMUs should be programmed differently so that these blocks point to different locations in main memory. Protection can be applied so that only the core that owns the data can access it. Finally, there should be shared memory used for communication mechanisms such as semaphores and message passing. The effective address of the shared memory area is mapped to the same location in physical memory for both cases.

1.5.1.2 Cooperative Multiprocessing

Cooperative asymmetric multiprocessing (CAMP) refers to a configuration in which the two operating systems are not fully aware of each other. Unlike SMP, there are two separate kernels in the main memory. However, some degree of cooperation is required because there are resources with single instances shared across both cores, such as the interrupt controller, boot sequencer, and DMA engines.

There are many possible applications for CAMP implementations, and several are shown in Figure 10, specifically with respect to functionality found on a line card. Mid-range applications often use general purpose processors to manage both the data plane and control plane. The ability to support two distinct operating systems can be valuable in these systems because it gives flexibility in how to partition the control and data plane. It also allows systems that might use two separate single core integrated processors to be combined into a single device solution, saving valuable board space.

Example A of Figure 1-6 shows one OS handling a bi-directional data plane while the other OS handles the control plane. This is useful when it is desirable to have distinct resources applied to the data plane and control plane. For instance, sometimes it is necessary to reset the control plane without impacting the data plane. This is easily accomplished when separate cores that can be reset independently are responsible for each. Databases that are relevant to both directions are easily accessible by the single operating system.

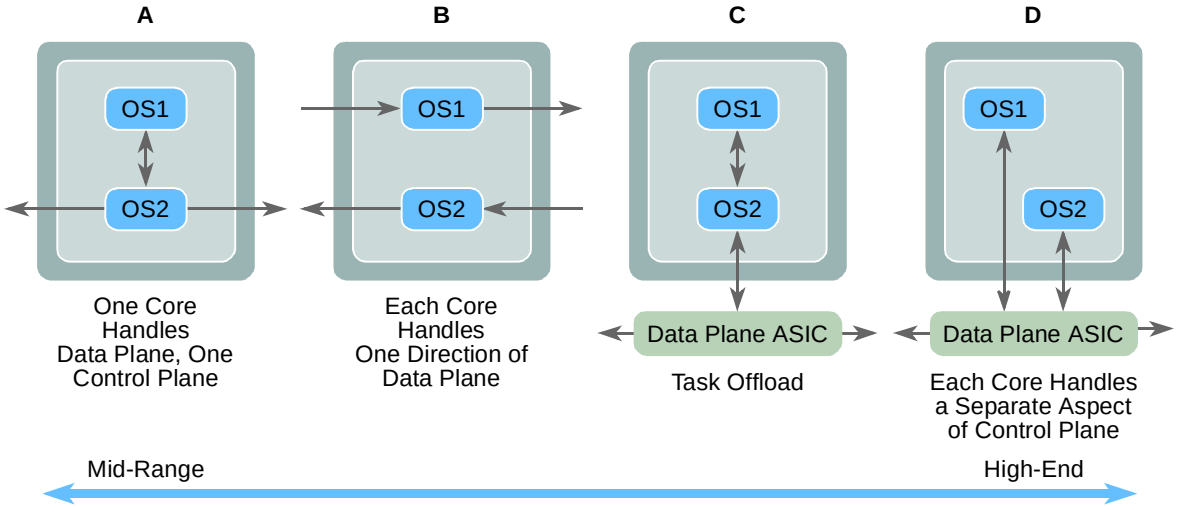


Figure 1-6. OS handling the Data Plane and Control Plane

In example B of Figure 1-6, each OS handles a single direction, both control and data plane. This can be useful when it is important to ensure that processing dedicated to traffic in one direction is not impacted by traffic going in the other direction. This can also be a very high performance solution since inter-processor communication is minimized and optimizations are possible when the same OS performs both packet handling and control plane processing.

Examples C and D of Figure 1-6 show the MPC8641D being used in conjunction with a data plane ASIC or NPU. These configurations are used in the highest performance platforms where hardware-based solutions are needed to keep up with the data rate and the entire capacity of a high performance processor is needed for control plane functions. Example C implements task offload, where one core interacts with the ASIC and delegates particularly intensive functions, such as encryption or TCP session timer management, to the other core. Example D shows an example where the ASIC is able to determine which OS is supposed to be handling a specific task.

There are several possible implementations of CAMP, two of which are shown in Figure 1-7. Implementations will vary on how the two cores are used and how the peripherals are used.

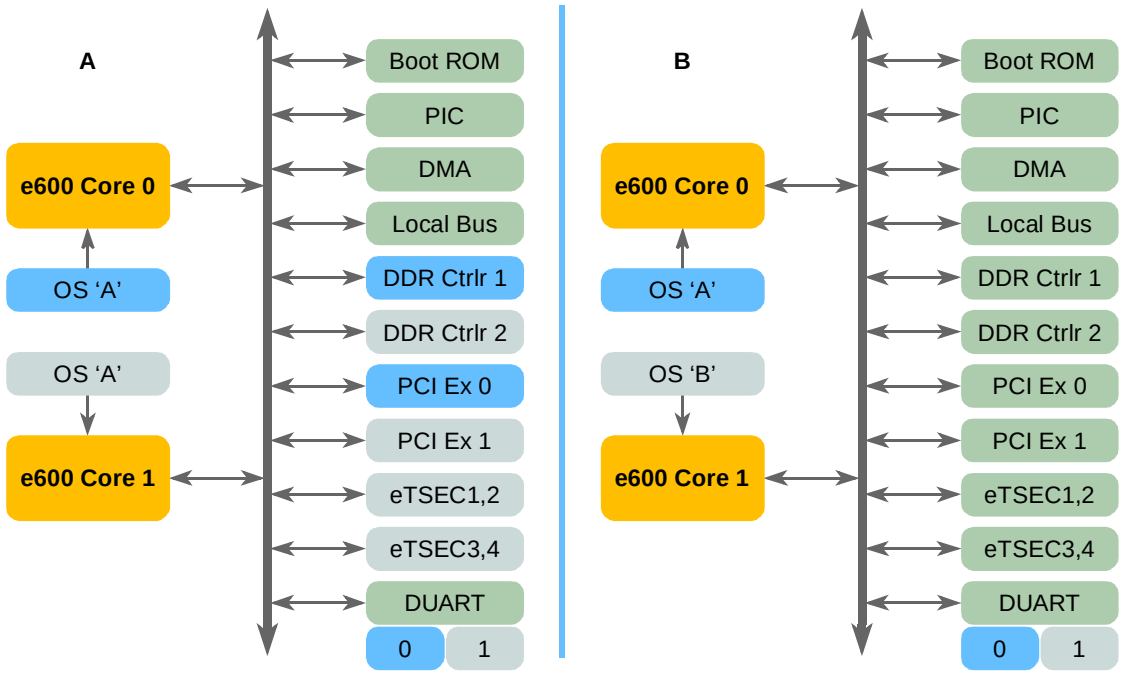


Figure 1-7. CAMP Implementation

Example A in Figure 1-7 shows that two copies of an identical non-SMP OS can run on each core. This shows the minimum amount of cooperation that would be required between two operating systems. Because many of the peripherals, such as the TSECs, memory controllers, DUART, and PCI Express ports, have multiple instances within the MPC8641D, each core can have exclusive ownership of one of these. The two operating systems must cooperate to share the Boot ROM, interrupt controller, DMA engine, and local bus. Example B shows a separate operating system running on each core, but all of the peripherals are shared between them.

Chapter 2
Memory Map

This chapter describes the mechanisms that define the device memory map—the local access windows (LAWs), the address translation and mapping units (ATMUs), and the configuration, control, and status registers (CCSRs).

2.1 Overview

There are several address domains within the MPC8641D, including:

- the logical, virtual, and physical (real) address spaces within the e600 cores
- the internal local address space
- the internal configuration, control, and status register (CCSR) address space
- the external memory, I/O, and configuration address spaces of PCI Express link 1
- the external memory, I/O, and configuration address spaces of PCI Express link 2
- the external memory address space of Serial RapidIO

The MMUs in the e600 cores handle translation of logical (effective) addresses, into virtual addresses, and ultimately to the physical addresses for the local address space. The MMU is described in Section 5.3.5, “Memory Management.” Remapping the real address range of a processor to a distinct physical address space is provided for certain dual operating system applications. It allows each OS running on a core to operate with a unique and private physical address space for a range of real addresses (starting at address 0x0_0000_0000). This capability and its control is described in detail in Section 2.1.1, “Low Memory Offset Mode.”

The local address map refers to the physical 36-bit address space seen by the e600 cores as they access memory and I/O space. The DMA engines also see this same local address map. All memory controlled by the DDR and local bus controllers exists in this address map, as do all memory-mapped configuration, control, and status registers (CCSRs). The local address map is defined by a set of ten local access windows (LAWs). Each of these windows maps a region of the local address space to a specified target interface, such as the DDR controllers, local bus controller, serial RapidIO controller, PCI Express controllers, or other targets. The internal configuration, control, and status registers (CCSRs) for all the functional blocks are located in the local memory space at a specific CCSR window.

If the target mapping performed by the local access windows directs the transaction to one of the external interfaces (as an outbound read or write), the transaction is then mapped into that interface’s external address space by address translation and mapping unit (ATMU) windows associated with the external interface. Outbound ATMUs perform the mapping from the local 36-bit address space to the address space of external interface; inbound ATMU windows perform the address translation from the external address space to the local address space.

2.1.1 Low Memory Offset Mode

In certain dual core, dual operating system applications, each processor must see a distinct and private physical address space for a range of addresses starting at address 0x0_0000_0000. For example, in the e600 core, address translation is disabled for both instruction fetches and data accesses beginning with the first instruction of an exception-handler routine. As the exception vectors are located at fixed addresses (ranging from 0x0_0000_0100 to 0x0_0000_1600 for the e600 core) and as each OS may wish to point to unique exception handler routines, a means must be provided for each core to see these real addresses in separate physical address spaces. Other system considerations may also necessitate creation of a private range of addresses in some range of low memory.

To facilitate this, the device provides the ability to translate (remap) a range of addresses starting at address 0x_0000_0000 for e600 core 1 through a low memory offset feature. (No translation capability is provided for e600 core 0.) A configuration signal sampled at power-on reset enables this capability. (See Section 4.4.3.5, “e600 Core 1 Low Memory Offset Mode,” for a detailed description of this bit.) The range corresponds to a fixed 256-Mbyte window.

When low memory offset mode is enabled, if e600 core 1 performs an access within the specified range, the address is translated such that the system address (address seen by the rest of the device, starting with the MCM) is offset by 256 Mbytes.

Note that to prevent aliasing by the processor, the MMU on e600 core 1 should be programmed such that there is no mapping to physical addresses from 256 Mbytes to 512 Mbytes-1. To keep the address space private to core 1, as the addresses from core 0 are not offset, the MMU on core 0 should also be programmed such that there is a hole in the real address space from 256 Mbytes to 512 Mbytes-1.

The low memory offset function is also performed for snooped addresses such that system addresses in the range 256 Mbytes to 512 Mbytes-1 presented on the MPX bus interface of core 1 are translated to addresses in the range 0 to 256 Mbytes-1 before they are presented to the L1 and L2 caches for snooping.

Note that the translated address is the one seen by all of the downstream decode and translation logic described in other sections of this document. For example, the local access windows must be configured based on this offset address when this mode is enabled. Care must be taken to prevent address configuration conflicts. For example, enabling low memory offset mode and subsequently configuring the CCSR base address (in CCSRBAR) to be within this range would be a programming error, because this would render the CCSR space inaccessible by core 1.

2.2 Local Access Windows

The local address map is defined by a set of ten local access windows (LAWs). Each of these windows maps a programmable 4-Kbyte to 32-Gbyte region of the local 36-bit address space to a specified target interface, such as the DDR controllers, local bus controller, serial RapidIO controller, PCI Express controllers, or other targets. This allows the internal interconnections of the device to route a transaction from its source to the proper target.

Each LAW is defined by a base address register which specifies the starting address for the window, and an attribute register which specifies whether the mapping is enabled, the size of the window, and the target interface for that window. Note that the LAWs do not perform any address translation. and therefore there

are no corresponding translation address registers. The local access window registers exist as part of the local access block in the general utilities registers in CCSR space.

With the exception of configuration space (mapped by CCSRBAR) and the default boot ROM space, all addresses used by the system must be mapped by an LAW. This includes addresses that are mapped by inbound ATMU windows. Thus, target mappings of the local access windows and the inbound ATMU windows must be consistent.

2.2.1 Local Access Window Registers

The local access window registers are accessed by reading and writing to an address comprised of the base address (specified in the CCSRBAR), plus the block base address, plus the offset of the specific register to be accessed. For the LAWs, the block base address is 0x0_0000.

Note that all LAW registers should only be accessed a word (4-bytes) at a time. Table 2-1 shows the memory map for the LAW registers.

Table 2-1. Local Access Window Memory Map

Local Access Windows—Block Base Address 0x0_0000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0xC08	LAWBAR0—Local access window 0 base address register	R/W	0x0000_0000	2.2.1.1/2-4
0xC10	LAWAR0—Local access window 0 attribute register	R/W	0x0000_0000	2.2.1.1/2-4
0xC28	LAWBAR1—Local access window 1 base address register	R/W	0x0000_0000	2.2.1.1/2-4
0xC30	LAWAR1—Local access window 1 attribute register	R/W	0x0000_0000	2.2.1.1/2-4
0xC48	LAWBAR2—Local access window 2 base address register	R/W	0x0000_0000	2.2.1.1/2-4
0xC50	LAWAR2—Local access window 2 attribute register	R/W	0x0000_0000	2.2.1.1/2-4
0xC68	LAWBAR3—Local access window 3 base address register	R/W	0x0000_0000	2.2.1.1/2-4
0xC70	LAWAR3—Local access window 3 attribute register	R/W	0x0000_0000	2.2.1.1/2-4
0xC88	LAWBAR4—Local access window 4 base address register	R/W	0x0000_0000	2.2.1.1/2-4
0xC90	LAWAR4—Local access window 4 attribute register	R/W	0x0000_0000	2.2.1.1/2-4
0xCA8	LAWBAR5—Local access window 5 base address register	R/W	0x0000_0000	2.2.1.1/2-4
0xCB0	LAWAR5—Local access window 5 attribute register	R/W	0x0000_0000	2.2.1.1/2-4
0xCC8	LAWBAR6—Local access window 6 base address register	R/W	0x0000_0000	2.2.1.1/2-4
0xCD0	LAWAR6—Local access window 6 attribute register	R/W	0x0000_0000	2.2.1.1/2-4
0xCE8	LAWBAR7—Local access window 7 base address register	R/W	0x0000_0000	2.2.1.1/2-4
0xCF0	LAWAR7—Local access window 7 attribute register	R/W	0x0000_0000	2.2.1.1/2-4
0xD08	LAWBAR8—Local access window 8 base address register	R/W	0x0000_0000	2.2.1.1/2-4
0xD10	LAWAR8—Local access window 8 attribute register	R/W	0x0000_0000	2.2.1.1/2-4
0xD28	LAWBAR9—Local access window 9 base address register	R/W	0x0000_0000	2.2.1.1/2-4
0xD30	LAWAR9—Local access window 9 attribute register	R/W	0x0000_0000	2.2.1.1/2-4

2.2.1.1 Local Access Window *n* Base Address Registers (LAWBAR0–LAWBAR9)

The LAWBARn registers define the 24 high-order address bits that fixes the location of each window in the local address space. Note that the minimum size of any LAW is 4 Kbytes, so the 12 lowest-order bits of the base address cannot be specified. Figure 2-1 shows the bit fields of the LAWBAR*n* registers.

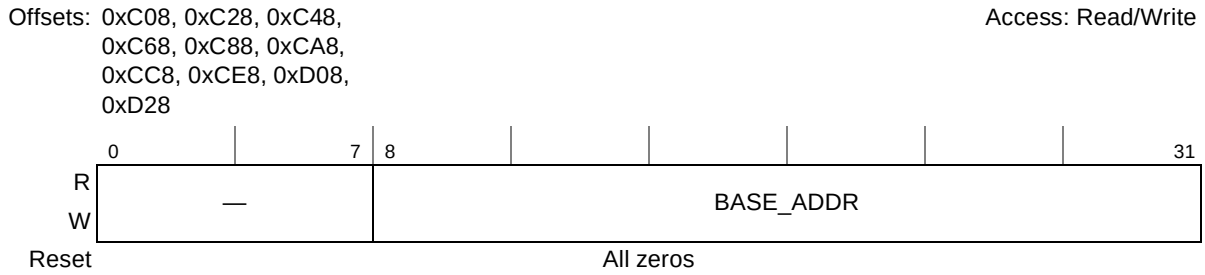


Figure 2-1. Local Access Window Base Address Registers (LAWBAR0 to LAWBAR9)

Table 2-2 describes the LAWBAR*n* fields.

Table 2-2. LAWBAR*n* Field Descriptions

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

2.2.1.2 Local Access Window *n* Attributes Registers (LAWAR0–LAWAR9)

The LAWAR*n* registers are used to enable specific local access windows, define their size and specify the target interface. Figure 2-2 shows the bit fields of the LAWAR*n* registers.

Figure 2-2. Local Access Window Attributes Registers (LAWAR0 to LAWAR9)

Table 2-3 describes the LAWAR*n* fields.

Table 2-3. LAWAR*n* Field Descriptions

Bits	Name	Description
0	EN	0 The local access window *n* (and all other LAWAR*n* and LAWBAR*n* fields) are disabled. 1 The local access window *n* is enabled and other LAWAR*n* and LAWBAR*n* fields combine to identify an address range for this window.
1–6	—	Reserved
7–11	TRGT	Identifies the target interface when a transaction hits in the address range defined by this window. The encodings for TRGT are defined in Table 2-4.
12–25	—	Reserved
26–31	SIZE	Identifies the size of the window from the starting address. Window size is $2^{(SIZE+1)}$ bytes. 000000–001010 Reserved 0010114 Kbytes 0011008 Kbytes 00110116 Kbytes . . . $2^{(SIZE+1)}$ bytes 10001032 Gbytes 100011–111111 Reserved

The target interface for each LAW is specified using the encodings shown in Table 2-4. Note that configuration registers are mapped by the windows defined by CCSRBAR. The CCSR mapping supersedes local access window mappings, so configuration registers do not appear as a target for local access windows.

Table 2-4. Target Interface Encodings

TRGT	Target Interface
00000	PCI Express 1
00001	PCI Express 2
00100	Local Bus (LBC)
01011	Interleaved DDR memory
01100	Serial RapidIO
01111	DDR memory controller 1
10110	DDR memory controller 2
All others	Reserved

2.2.2 Precedence of Local Access Windows

If two or more LAWs overlap, the lower numbered window takes precedence. For instance, consider two LAWs, set up as shown in Table 2-5.

Table 2-5. Overlapping Local Access Windows

LAW	Base Address	Size	Target Interface
1	0x0_7FF0_0000	1 Mbyte	0b00100 (local bus controller—LBC)
2	0x0_0000_0000	2 Gbytes	0b01111 (DDR controller 1)

In this case, LAW 1 governs the mapping of the 1-Mbyte region from 0x0_7FF0_0000 to 0x0_7FFF_FFF, even though the window described in LAW 2 also encompasses that memory region.

NOTE

The CCSR mapping, defined by CCSRBAR, supersedes all local access window mappings.

2.2.3 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 LAW configuration register before enabling any other devices to use the window. For example, if LAWs 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 performed by one of the e600 cores, the read of LAWAR3 should be followed by an **isync** instruction.

2.2.4 Distinguishing Local Access Windows from Other Mapping Functions

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

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

2.2.5 Illegal Interaction Between Local Access Windows and DDR Chip Selects

If a local access window maps an address to an interface other than the DDR controllers, then there should not be a valid chip select configured for the same address in the DDR controllers. Because DDR chip select boundaries are defined by a beginning and ending address, it is easy to define them so that they do not overlap with LAWs that map to other interfaces.

2.2.6 Local Address Map Example

Figure 2-3 shows what a typical local address map might look like.

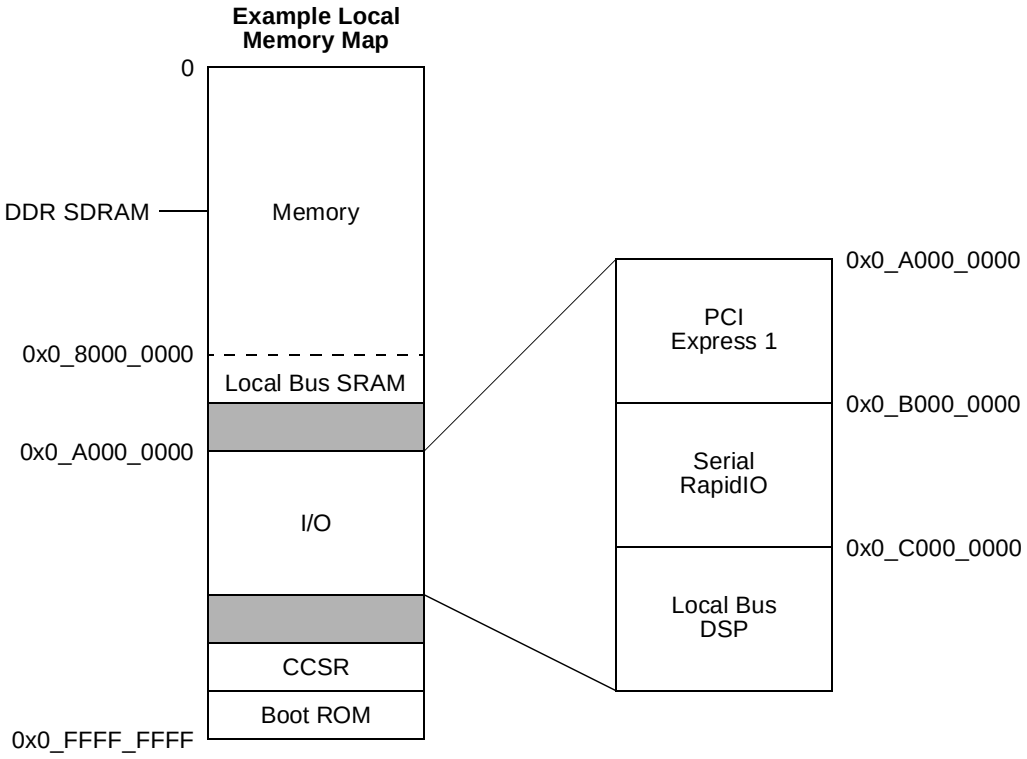


Figure 2-3. Local Address Map Example

Table 2-6 shows the corresponding set of LAW settings for the example shown in Figure 2-3.

Table 2-6. Local Access Window Settings Example

Window	Base Address	SIZE	Target Interface	TRGT
0	0x0_0000_0000	2 Gbytes	DDR controller 1	0b01111
1	0x0_8000_0000	1 Mbyte	Local bus controller (LBC)—SRAM	0b00100
2	0x0_A000_0000	256 Mbytes	PCI Express 1	0b00000
3	0x0_B000_0000	256 Mbytes	Serial RapidIO	0b01100

Table 2-6. Local Access Window Settings Example (continued)

Window	Base Address	SIZE	Target Interface	TRGT
4	0x0_C000_0000	256 Mbytes	Local bus controller (LBC)—DSP	0b00100
5–9	Unused			

In this example, it is not necessary to use a LAW to specify the location of the boot ROM because it is in the default location at the 8 Mbytes of memory from 0x0_FF80_0000 to 0x0_FFFF_FFFF (see Section 4.4.3.7, “Alternate Boot Vector Location”); the e600 cores fetch their reset vectors by performing a burst read from effective address 0x0_FFF0_0100 and begin executing with the instruction at effective address 0x0_FFF0_0100. Neither is it required to define a LAW to describe the range of memory used for memory-mapped configuration, control, and status registers because these occupy a fixed 1-Mbyte space pointed to by CCSRBAR. See Section 4.3.1.1.3, “Configuration, Control, and Status Base Address Register (CCSRBAR).”

2.3 Address Translation and Mapping Units

To facilitate flexibility in defining the address maps for the external interfaces (serial RapidIO, PCI Express 1, and PCI Express 2), the device provides address translation and mapping units (ATMUs).

The following types of translation and mapping operations are performed by the ATMUs:

- Translating the local 36-bit address to an external address space
- Translating external addresses to the local 36-bit address space
- Assigning attributes to transactions
- Mapping a local address to a target interface
- Remapping a real address range from core 1 to a distinct physical address range for that core

Outbound address translation and mapping refers to the translation of addresses from the local 36-bit address space to the external address space and attributes of a particular I/O interface.

Inbound address translation and mapping refers to the translation of an address from the external address space of an I/O interface to the local address space understood by the internal interfaces of this device. It also refers to the mapping of transactions to a particular target interface and the assignment of transaction attributes. Note that in mapping the transaction to the target interface, an inbound ATMU window performs a function similar to the local access windows. The target mappings created by an inbound ATMU must be consistent with those of the LAWs. That is, if an inbound ATMU maps a transaction to a given local address and a given target, a LAW must also map that same local address to the same target.

2.3.1 Address Translation

All of the configuration registers that define an ATMU window's translation and mapping functions follow the same general register format, summarized in Table 2-7.

Table 2-7. Format of ATMU Window Definitions

Register	Function
Translation address (TAR)	High-order address bits defining location of the window in the target address space
Base address (BAR)	High-order address bits defining location of the window in the initiator address space
Window attributes (WAR)	Window enable, window size, target interface, and transaction attributes

The size of the windows must be a power-of-two. 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.

2.3.2 Outbound ATMUs

If the target mapping performed by the local access windows directs the transaction to one of the external interfaces (as an outbound read or write), the transaction is then mapped into that interface's external address space by outbound ATMUs associated with the external interface. The outbound ATMUs perform the mapping from the local 36-bit address space to the address spaces of the PCI Express and RapidIO busses. which may be much larger than the local space. The outbound ATMUs also map attributes such as transaction type and priority level.

The serial RapidIO controller has eight outbound ATMU windows plus a default window. If a transaction's address does not hit any of the eight outbound ATMU windows, the translation attributes defined by the default window are used. The default window is always enabled.

The PCI Express controllers each have four outbound ATMU windows plus a default window. If a transaction's address does not hit any of the four outbound ATMU windows, the translation attributes defined by the default window are used. The default window is always enabled. The PCI Express outbound ATMUs include extended translation address registers so that up to 64 bits of external address space can be supported.

2.3.3 Inbound ATMUs

The inbound ATMUs perform the address translation from the external address spaces to the local address space, attach attributes and transaction types to the transaction, and also map the transaction to its target interface.

The serial RapidIO controller has four inbound ATMU windows plus a default. If the inbound transaction's address does not hit any of the four inbound ATMU windows, the translation attributes defined by the default window are used. The default window is always enabled.

The PCI Express controllers each have three general inbound ATMU windows plus a fixed translation window to the CCSR space.

2.3.3.1 Illegal Interaction Between Inbound ATMUs and LAWs

Since both local access windows and inbound ATMUs map transactions to a target interface, it is essential that they not contradict one another. For example, it is considered a programming error to have an inbound ATMU map a transaction to the DDR memory controller 1 (target interface 0b0_1111) if the resulting translated local address is mapped to the PCI Express 1 interface (target interface 0b0_0000) by a local access window. Such programming errors may result in unpredictable system deadlocks.

2.4 Configuration, Control, and Status Registers

All of the memory-mapped configuration, control, and status registers (CCSRs) in this device are contained within a 1-Mbyte address region. To allow for flexibility, the CCSR block is relocatable in the local address space.

The local address map location of the CCSR block is controlled by the configuration, control, and status base address register (CCSRBAR); see Section 4.3.1.1.3, “Configuration, Control, and Status Base Address Register (CCSRBAR).” The default value for CCSRBAR is 0x0_FF70_0000 (or 4 Gbytes-9 Mbytes). No address translation is performed for CCSR space, so there is no associated translation address register. The CCSR window is always enabled with a fixed size of 1 Mbyte; no other attributes are attached, so there is no associated window attribute register.

NOTE

The CCSR window always takes precedence over all local access windows. However, the CCSR window must not overlap an LAW 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.

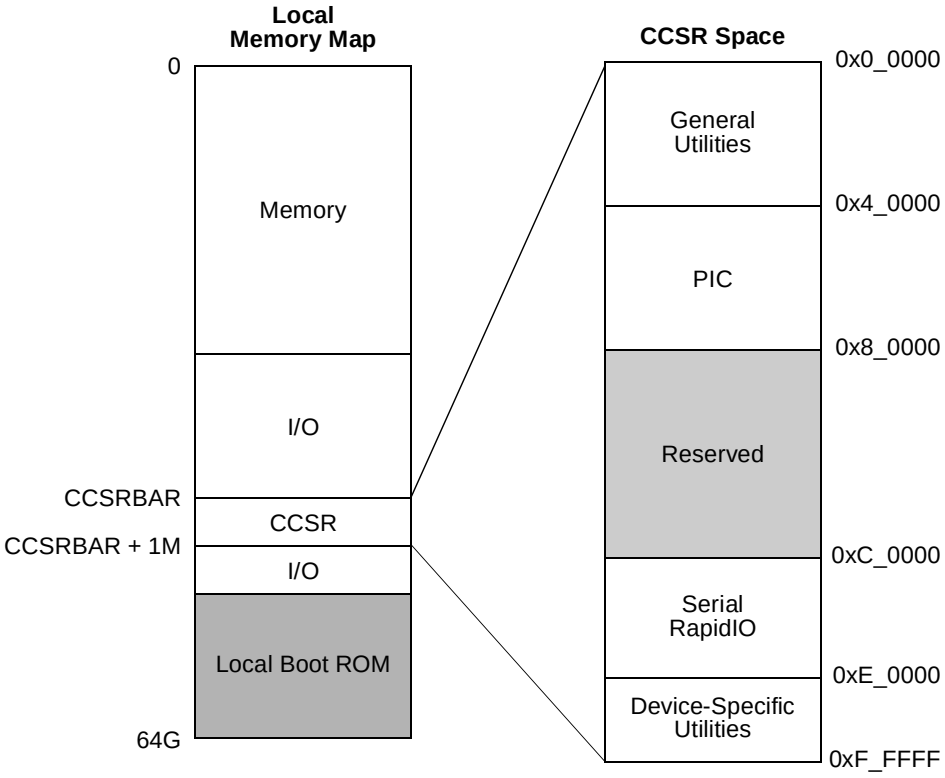


Figure 2-4. CCSR Space

2.4.1 Accessing CCSR Memory from the Local Processor

When a local e600 core is used to configure CCSR space, the CCSR memory space should typically be marked as cache-inhibited and guarded.

In addition, many configuration registers affect accesses to other memory regions or peripherals; therefore writes to these registers must be guaranteed to have taken effect before accesses are made to the associated memory regions or peripherals.

To guarantee that the results of any sequence of writes to configuration registers are in effect, the final configuration register write should be chased by a read of the same register, and that should be followed by a SYNC instruction. Then accesses can safely be made to memory regions affected by the configuration register write.

2.4.2 Accessing CCSR Memory from External Masters

In addition to being accessible by the e600 processors, the CCSRs are accessible from the external PCI Express interfaces. This allows external masters on the I/O ports to configure the device.

External masters do not need to know the location of the CCSR memory in the local address map. Rather, they access the CCSR 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 Express base address for accessing the local CCSR memory is selectable through the PCI Express configuration and status register base address register (PEXCSRBAR), at offset 0x10. An external PCI Express master sets this register by performing a PCI Express configuration cycle to this device. Subsequent memory accesses by a PCI Express master to the PCI Express address range indicated by PEXCSRBAR are translated to the local CCSR address indicated by the current setting of CCSRBAR.

2.4.3 Organization of CCSR Space

As shown in Figure 2-4, the CCSR space is divided into four groups—general utilities, programmable interrupt controller (PIC), serial RapidIO, and device-specific utilities registers.

2.4.3.1 General Utilities Registers

The general utilities registers are the functional block-specific registers that occupy the first 256 Kbytes of CCSR space. Each functional block is allocated a 4-Kbyte address range for its registers within the general utilities space. Figure 2-5 shows the layout of the general utilities registers.

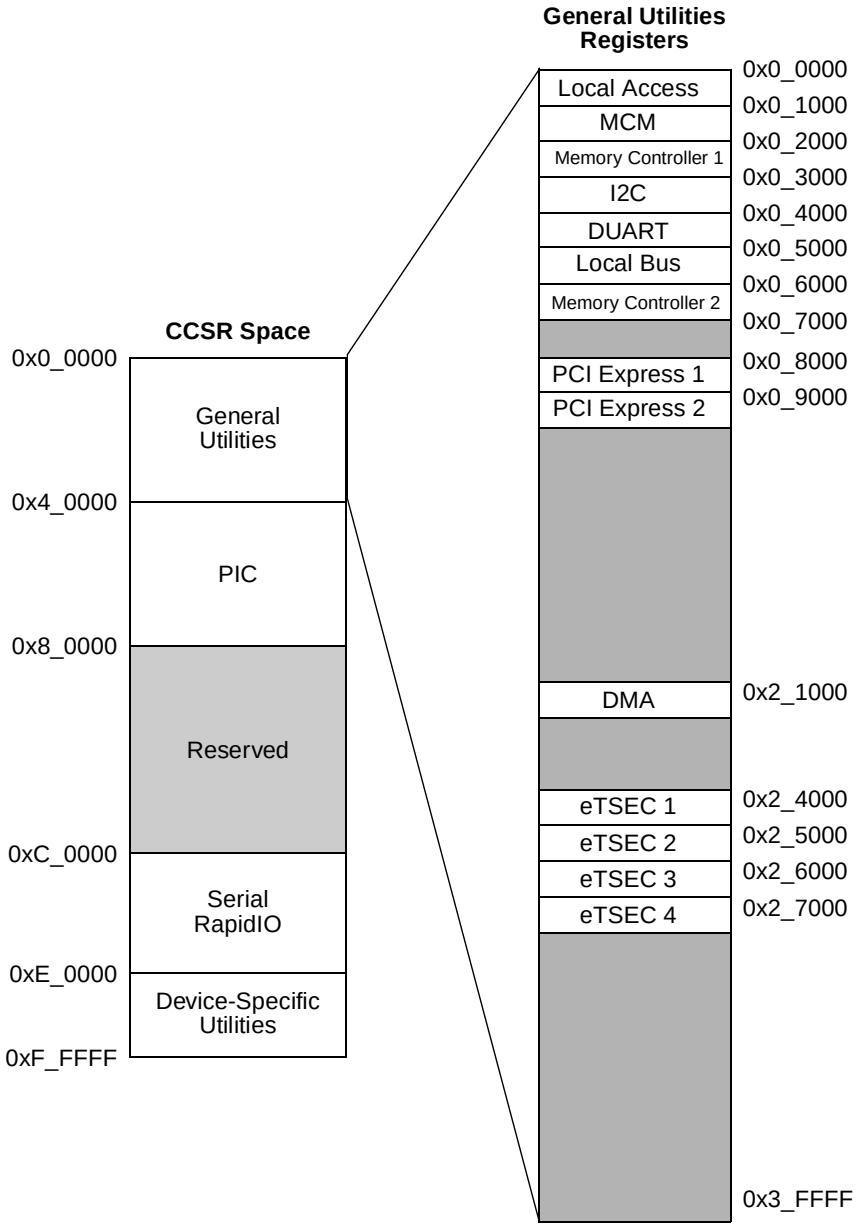


Figure 2-5. General Utilities Register Map within CCSR Space

2.4.3.1.1 General Utilities Register Organization

Figure 2-6 shows the typical organization of registers inside the 4-Kbyte register space allocated to an individual functional block. Starting at the block base address, the first 3 Kbytes are available for general registers. If the functional block has associated ATMUs, the next 512 bytes are dedicated to address translation and mapping registers. If a functional block has error management registers, they are typically placed starting at offset 0xE00 from the block base address, and any debug registers are typically placed in the final 256 bytes of the block's register space starting at offset 0xF00.

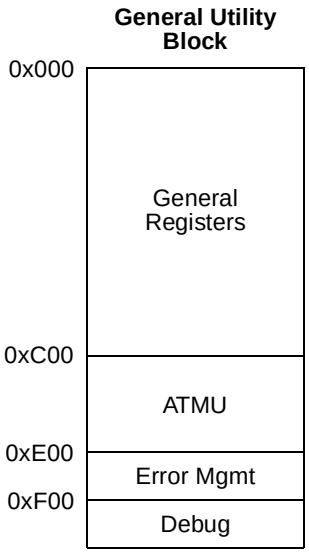


Figure 2-6. General Utility Register Block

NOTE

Refer to detailed register descriptions for each functional block for exact locations, sizes, and access requirements.

2.4.3.2 Programmable Interrupt Controller Registers

The programmable interrupt controller (PIC) follows the OpenPIC programming model which requires a larger register address space than the 4 Kbytes allocated to other blocks within the general utilities space. For this reason, the PIC is allocated the second 256 Kbytes of CCSR space, beginning at offset 0x4_0000 from CCSRBAR. The layout of the PIC register space is shown in Figure 2-7. Note that the PIC registers should only be accessed with 32-bit accesses.

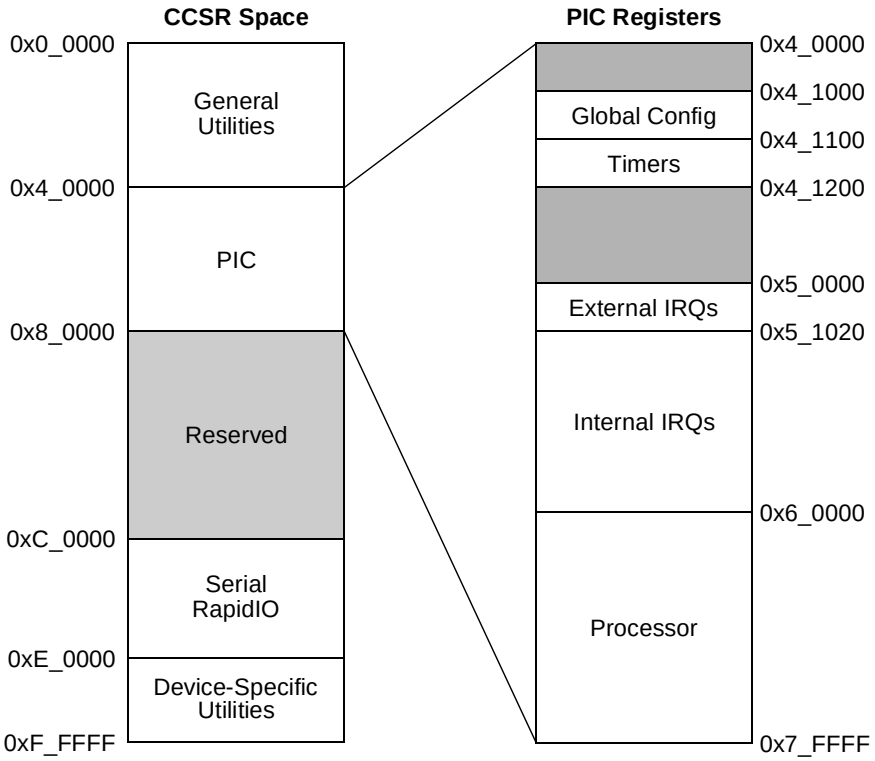


Figure 2-7. PIC Register Map within CCSR Space

2.4.3.3 Serial RapidIO Registers

The serial RapidIO module uses 128 Kbytes of CCSR space, as shown in Figure 2-8. All registers are 32-bits wide and should only be accessed with 32-bit accesses. The 4-Kbyte serial RapidIO implementation block has the same internal organization as that defined for the general utilities described above.

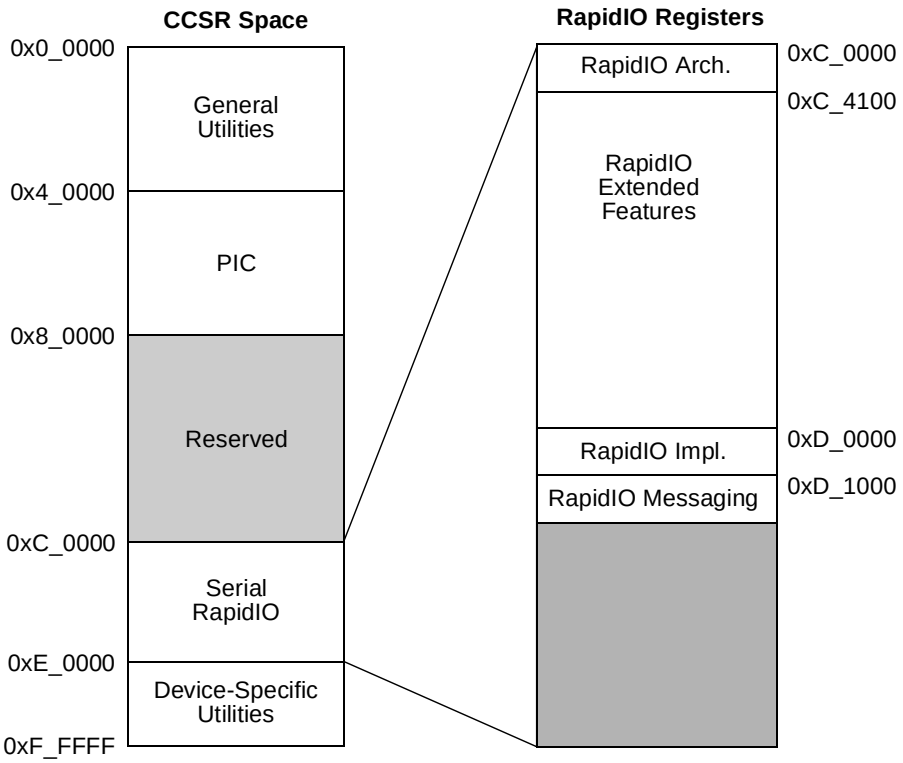


Figure 2-8. Serial RapidIO Register Map within CCSR Space

2.4.3.4 Device-Specific Utilities Registers

The device-specific utilities registers control functions that are not particular to a functional unit but to the device as a whole; they occupy the highest 256 Kbytes of CCSR space. The device-specific utilities registers consist of power management, performance monitors, and device-wide debug utilities. Figure 2-9 shows the layout of the device-specific utilities registers. Note that the device-specific registers are accessible with 32-bit accesses only. Transactions of a size other than 32-bits are considered programming errors and the operation is undefined.

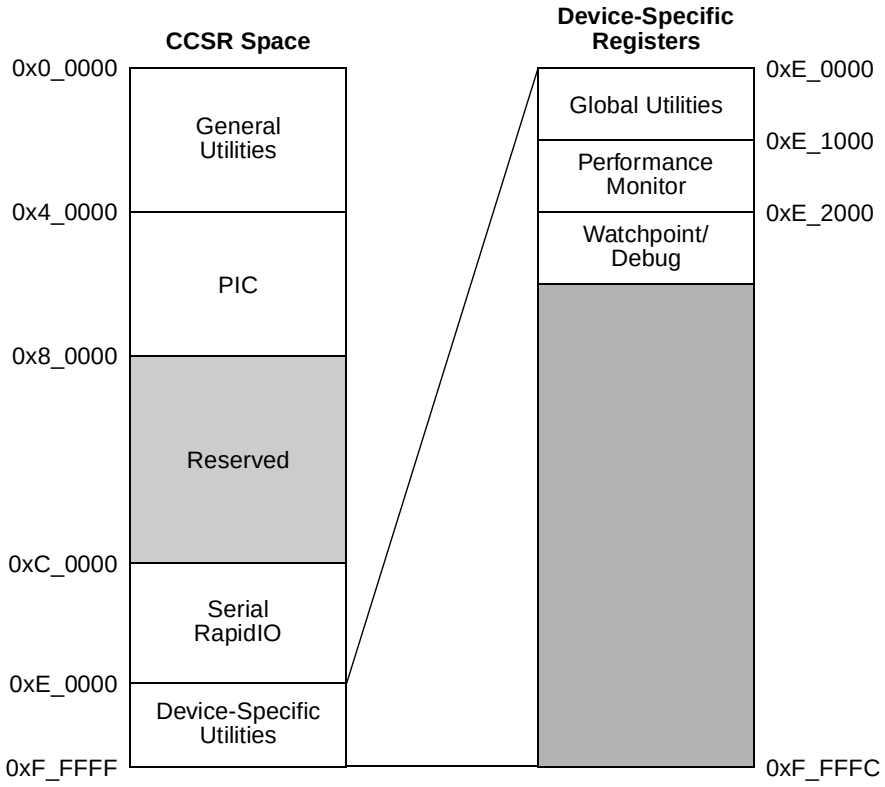


Figure 2-9. Device-Specific Register Map within CCSR Space

2.4.4 CCSR Address Map

The full register address of any CCSR is comprised of the CCSR window base address, specified in CCSRBAR (default address 0x0_FF70_0000), plus the functional block base address, plus the specific register's offset within that block.

Table 2-8 shows the location of the functional block base addresses for the entire CCSR space. Cross-references are provided to the CCSR maps for each individual block. A complete listing of all CCSRs is provided in Appendix A, "Complete List of Configuration, Control, and Status Registers."

Table 2-8. CCSR Block Base Address Map

Block Base Address (Hex)	Block	Section/Page	Comments
General Utilities (0x0_0000–0x3_FFFF)			
0x0_0000	Local Configuration Control	7.4/7-4	—
	Local Access	2.2.1/2-3	—
0x0_1000	MPX coherency module (MCM)	7.4/7-4	—
0x0_2000	DDR memory controller 1	8.4/8-10	DDR memory controller 2 is located at 0x0_6000
0x0_3000	I^2C controllers (2)	10.3/10-4	I^2C controller 1: 0x0_3000 I^2C controller 2: 0x0_3100
0x0_4000	DUART	11.3/11-4	UART0: 0x0_4500 UART1: 0x0_4600
0x0_5000	Local bus controller (LBC)	12.3/12-8	—
0x0_6000	DDR memory controller 2	8.4/8-10	DDR memory controller 1 is located at 0x0_2000
0x0_7000– 0x0_7FFF	Reserved		
0x0_8000	PCI Express controller 1	16.3.1/16-5	PCI Express controller 2 is located at 0x0_9000
0x0_9000	PCI Express controller 2	16.3.1/16-5	PCI Express controller 1 is located at 0x0_8000
0x0_A000– 0x2_0FFF	Reserved		
0x2_1000	DMA controller	14.3/14-6	DMA0: 0x2_1100 DMA1: 0x2_1180 DMA2: 0x2_1200 DMA3: 0x2_1280 General Status: 0x2_1300
0x2_2000– 0x2_3FFF	Reserved		
0x2_4000	eTSEC1	13.5/13-12	—
0x2_5000	eTSEC2	13.5/13-12	—

Table 2-8. CCSR Block Base Address Map (continued)

Block Base Address (Hex)	Block	Section/Page	Comments
0x2_6000	eTSEC3	13.5/13-12	—
0x2_7000	eTSEC4	13.5/13-12	—
0x2_8000–0x3_FFFF	Reserved		
Programmable Interrupt Controller (PIC) (0x4_0000–0x6_FFFF)			
0x4_0000	PIC—Global registers	9.3/9-9	—
0x5_0000	PIC—Interrupt source registers	9.3/9-9	—
0x6_0000	PIC—Processor (per-CPU) registers	9.3/9-9	—
0x7_0000–0xB_FFFF	Reserved		
Serial RapidIO (0xC_0000–0xD_FFFF)			
0xC_0000	RapidIO architectural registers	15.5/15-4	—
0xD_0000	RapidIO implementation registers	15.5/15-4	—
Device-Specific Utilities (0xE_0000–0xF_FFFF)			
0xE_0000	Global utilities	17.4/17-4	—
0xE_1000	Performance monitor	18.3/18-3	—
0xE_2000	Watchpoint monitor and trace buffer	19.3/19-10	—
0xE_3000–0xF_FFFF	Reserved		

Chapter 3
Signal Descriptions

This chapter describes the MPC8641D 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

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.

Power-on reset configuration signals are shown as lower case and begin with the prefix, “cfg_”.

3.1 Signals Overview

The MPC8641D signals are grouped as follows:

- DDR memory interface signals
- High-speed I/O interface signals
- Ethernet 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 and Figure 3-2 illustrate the external signals of the MPC8641, showing how the signals are grouped. Refer to the *MPC8641D Integrated Processor Hardware Specifications* for a pinout diagram showing pin numbers and a listing of all the electrical and mechanical specifications.

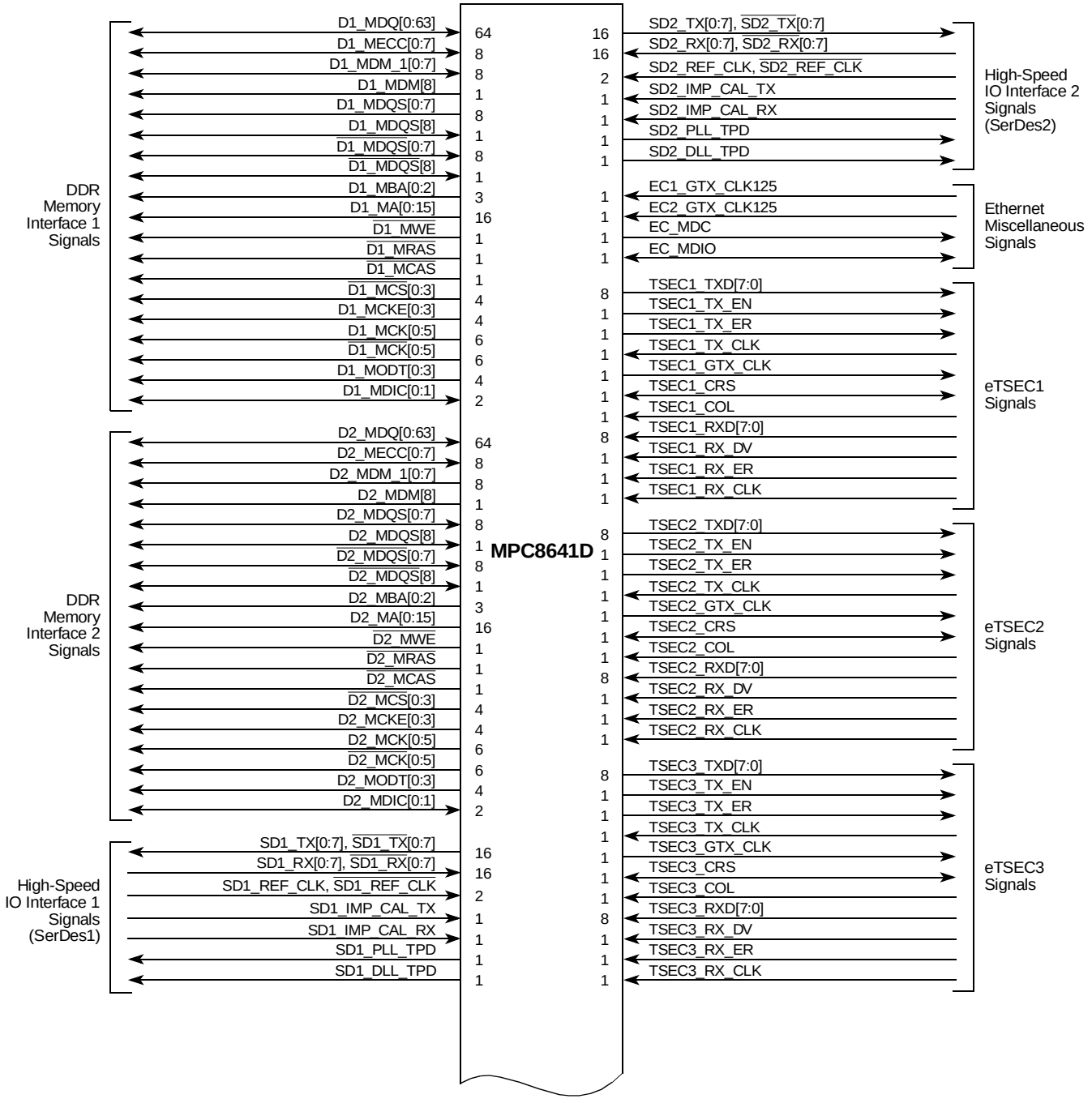


Figure 3-1. MPC8641D Signal Groupings (1 of 2)

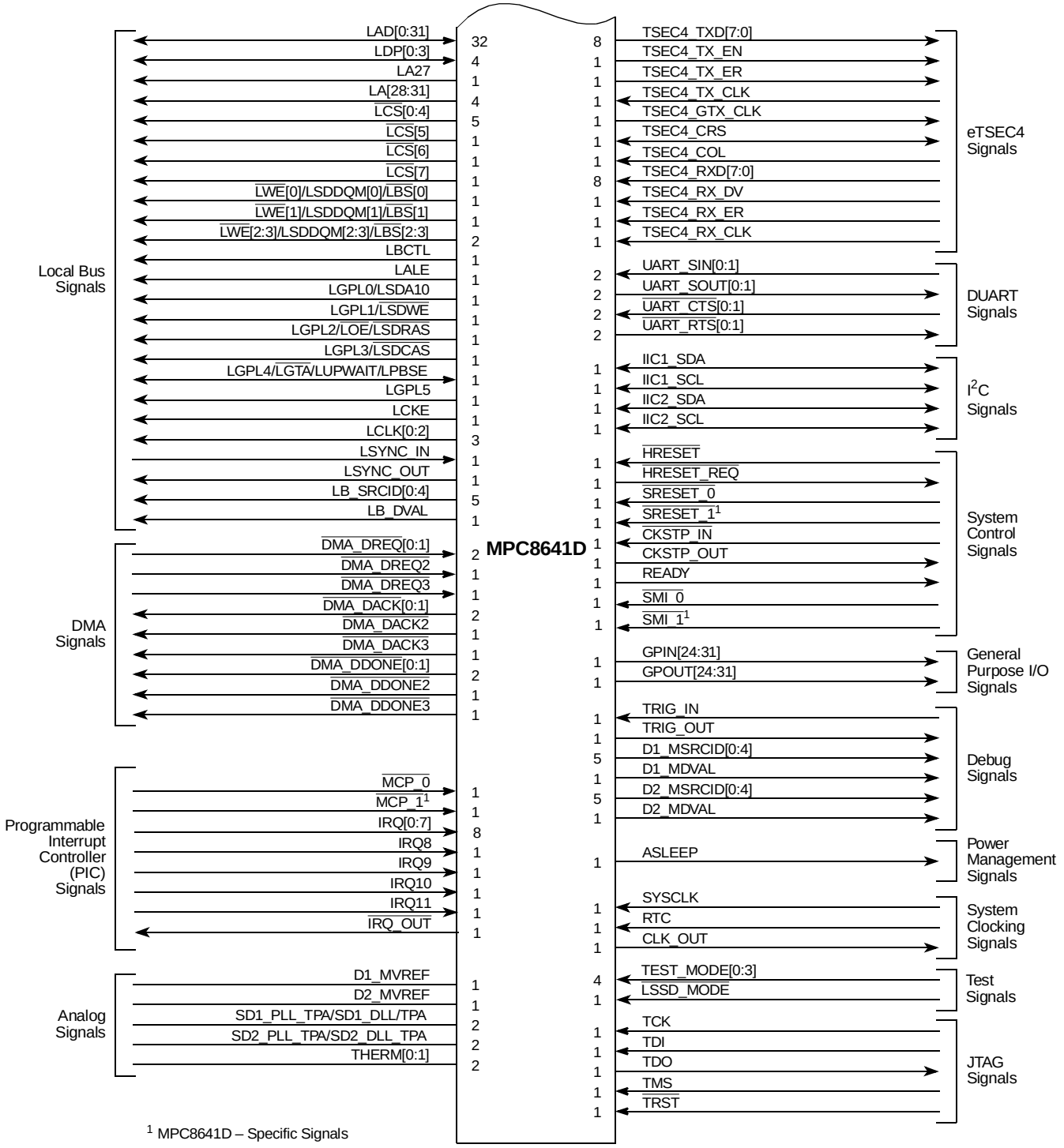


Figure 3-2. MPC8641D Signal Groupings (2 of 2)

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

The following tables provide summaries of signal functionality on the device. 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. MPC8641D Signal Reference by Functional Block

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/Page
DDR Memory Controller 1 Interface Signals						
D1_MDQ[0:63]	DDR Data	DDR Controller 1	—	64	I/O	8-3/8-6
D1_MECC[0:7]	DDR Error Correcting Code	DDR Controller 1	—	8	I/O	8-3/8-6
D1_MDM[0:7]	DDR Data Mask	DDR Controller 1	—	8	O	8-3/8-6
D1_MDM[8]	DDR ECC Data Mask	DDR Controller 1	—	1	O	8-3/8-6
D1_MDQS[0:7]	DDR Data Strobe	DDR Controller 1	—	8	I/O	8-3/8-6
D1_MDQS[8]	DDR ECC Data Strobe	DDR Controller 1	—	1	I/O	8-3/8-6
$\overline{\text{D1\_MDQS}}$[0:7]	DDR Neg. Data Strobe	DDR Controller 1	—	8	I/O	8-3/8-6
$\overline{\text{D1\_MDQS}}$[8]	DDR Neg. ECC Data Strobe	DDR Controller 1	—	1	I/O	8-3/8-6
D1_MBA[2:0]	DDR Bank Select	DDR Controller 1	—	3	O	8-3/8-6
D1_MA[15:0]	DDR Address	DDR Controller 1	—	16	O	8-3/8-6
$\overline{\text{D1\_MWE}}$	DDR Write Enable	DDR Controller 1	—	1	O	8-3/8-6
$\overline{\text{D1\_MRAS}}$	DDR Row Address Strobe	DDR Controller 1	—	1	O	8-3/8-6
$\overline{\text{D1\_MCAS}}$	DDR Column Address Strobe	DDR Controller 1	—	1	O	8-3/8-6
$\overline{\text{D1\_MCS}}$[0:3]	DDR Chip Select (2/DIMM)	DDR Controller 1	—	4	O	8-3/8-6
D1_MCKE[0:3]	DDR Clock Enable	DDR Controller 1	—	4	O	8-4/8-10
D1_MCK[0:5]	DDR Clocks (3 pairs/DIMM)	DDR Controller 1	—	6	O	8-4/8-10
$\overline{\text{D1\_MCK}}$[0:5]	DDR Neg. Clocks (3 pairs/DIMM)	DDR Controller 1	—	6	O	8-4/8-10
D1_MODT[0:3]	DDR On Device Termination	DDR Controller 1	—	4	O	8-4/8-10
D1_MDIC[0:1]	DDR Output Conditioning	DDR Controller 1	—	2	I/O	8-4/8-10

Table 3-1. MPC8641D Signal Reference by Functional Block (continued)

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/Page
DDR Memory Controller 2 Interface Signals						
D2_MDQ[0:63]	DDR Data	DDR Controller 2	—	64	I/O	8-3/8-6
D2_MECC[0:7]	DDR Error Correcting Code	DDR Controller 2	—	8	I/O	8-3/8-6
D2_MDM[0:7]	DDR Data Mask	DDR Controller 2	—	8	O	8-3/8-6
D2_MDM[8]	DDR ECC Data Mask	DDR Controller 2	—	1	O	8-3/8-6
D2_MDQS[0:7]	DDR Data Strobe	DDR Controller 2	—	8	I/O	8-3/8-6
D2_MDQS[8]	DDR ECC Data Strobe	DDR Controller 2	—	1	I/O	8-3/8-6
$\overline{\text{D2\_MDQS}}$[0:7]	DDR Neg. Data Strobe	DDR Controller 2	—	8	I/O	8-3/8-6
$\overline{\text{D2\_MDQS}}$[8]	DDR Neg. ECC Data Strobe	DDR Controller 2	—	1	I/O	8-3/8-6
D2_MBA[2:0]	DDR Bank Select	DDR Controller 2	—	3	O	8-3/8-6
D2_MA[15:0]	DDR Address	DDR Controller 2	—	16	O	8-3/8-6
$\overline{\text{D2\_MWE}}$	DDR Write Enable	DDR Controller 2	—	1	O	8-3/8-6
$\overline{\text{D2\_MRAS}}$	DDR Row Address Strobe	DDR Controller 2	—	1	O	8-3/8-6
$\overline{\text{D2\_MCAS}}$	DDR Column Address Strobe	DDR Controller 2	—	1	O	8-3/8-6
$\overline{\text{D2\_MCS}}$[0:3]	DDR Chip Select (2/DIMM)	DDR Controller 2	—	4	O	8-3/8-6
D2_MCKE[0:3]	DDR Clock Enable	DDR Controller 2	—	4	O	8-4/8-10
D2_MCK[0:5]	DDR Clocks (3 pairs/DIMM)	DDR Controller 2	—	6	O	8-4/8-10
$\overline{\text{D2\_MCK}}$[0:5]	DDR Neg. Clocks (3 pairs/DIMM)	DDR Controller 2	—	6	O	8-4/8-10
D2_MODT[0:3]	DDR On Device Termination	DDR Controller 2	—	4	O	8-4/8-10
D2_MDIC[0:1]	DDR Output Conditioning	DDR Controller 2	—	2	IO	8-4/8-10
High Speed IO Interface 1 Signals (SerDes1)						
SD1_TX[0:7], $\overline{\text{SD1\_TX}}$[0:7]	Transmit Data	High Speed IO 1	—	16	O	16.2/16-4
SD1_RX[0:7], $\overline{\text{SD1\_RX}}$[0:7]	Receive Data	High Speed IO 1	—	16	I	16.2/16-4
SD1_REF_CLK, $\overline{\text{SD1\_REF\_CLK}}$	PLL Reference Clock	High Speed IO 1	—	2	I	MPC8641DEC

Table 3-1. MPC8641D Signal Reference by Functional Block (continued)

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/Page
SD1_DLL_TPD	Test Point (Digital) SerDes1 DLL	High Speed IO 1	—	1	O	MPC8641DEC
SD1_PLL_TPD	Test Point for (Digital) SerDes1 PLL	High Speed IO 1	—	1	O	MPC8641DEC
High Speed IO Interface 2 Signals (SerDes 2)						
SD2_TX[0:7], $\overline{\text{SD2\_TX[0:7]}}$	Transmit Data	High Speed IO 2	—	16	O	15.4/15-3 (SRIO) 16.2/16-4 (PEX2)
SD2_RX[0:7], $\overline{\text{SD2\_RX[0:7]}}$	Receive Data	High Speed IO 2	—	16	I	15.4/15-3 (SRIO) 16.2/16-4 (PEX2)
SD2_REF_CLK, $\overline{\text{SD2\_REF\_CLK}}$	PLL Reference Clock	High Speed IO 2	—	2	I	MPC8641DEC
SD2_DLL_TPD	Test Point (Digital) SerDes2 DLL	High Speed IO 2	—	1	O	MPC8641DEC
SD2_PLL_TPD	Test Point (Digital) SerDes2 PLL	High Speed IO 2	—	1	O	MPC8641DEC
Ethernet Miscellaneous Signals						
EC1_GTX_CLK125	Gigabit Reference Clock 1	Gigabit Clock	—	1	I	13-2/13-9
EC2_GTX_CLK125	Gigabit Reference Clock 2	Gigabit Clock	—	1	I	13-2/13-9
EC_MDC	Ethernet Management Data Clock	Ethernet Management	—	1	O	13-2/13-9
EC_MDIO	Ethernet Management Data In/Out	Ethernet Management	—	1	I/O	13-2/13-9
eTSEC1 Signals						
TSEC1_TXD[7]	TSEC1 Transmit Data	TSEC1	GPOUT[7]cfg_tsec1_prtcl[1]	1	O	13-1/13-7
TSEC1_TXD[6]	TSEC1 Transmit Data	TSEC1	GPOUT[6]cfg_tsec1_prtcl[0]	1	O	13-1/13-7
TSEC1_TXD[5]	TSEC1 Transmit Data	TSEC1	GPOUT[5]cfg_tsec1_reduce	1	O	13-1/13-7
TSEC1_TXD[4]	TSEC1 Transmit Data	TSEC1	GPOUT[4]cfg_device_id[7]	1	O	13-1/13-7
TSEC1_TXD[3]	TSEC1 Transmit Data	TSEC1	GPOUT[3]cfg_device_id[6]	1	O	13-1/13-7

Table 3-1. MPC8641D Signal Reference by Functional Block (continued)

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/Page
TSEC1_TXD[2]	TSEC1 Transmit Data	TSEC1	GPOUT[2] / cfg_device_id[5]	1	O	13-1/13-7
TSEC1_TXD[1]	TSEC1 Transmit Data	TSEC1	GPOUT[1] / cfg_platform_freq	1	O	13-1/13-7
TSEC1_TXD[0]	TSEC1 Transmit Data	TSEC1	GPOUT[0]	1	O	13-1/13-7
TSEC1_TX_EN	TSEC1 Transmit Enable	TSEC1	—	1	O	13-1/13-7
TSEC1_TX_ER	TSEC1 Transmit Error	TSEC1	—	1	O	13-1/13-7
TSEC1_TX_CLK	TSEC1 Transmit Clock In	TSEC1	—	1	I	13-1/13-7
TSEC1_GTX_CLK	TSEC1 Transmit Clock Out	TSEC1	—	1	O	13-1/13-7
TSEC1_CRS	TSEC1 Carrier Sense	TSEC1	—	1	I/O	13-1/13-7
TSEC1_COL	TSEC1 Collision Detect	TSEC1	—	1	I	13-1/13-7
TSEC1_RXD[7:0]	TSEC1 Receive Data	TSEC1	GPIN[7:0]	8	I	13-1/13-7
TSEC1_RX_DV	TSEC1 Receive Data Valid	TSEC1	—	1	I	13-1/13-7
TSEC1_RX_ER	TSEC1 Receiver Error	TSEC1	—	1	I	13-1/13-7
TSEC1_RX_CLK	TSEC1 Receive Clock	TSEC1	—	1	I	13-1/13-7
eTSEC2 Signals						
TSEC2_TXD[7]	TSEC2 Transmit Data	TSEC2	GPOUT15]cfg_tsec2_prtcl[1]	1	O	13-1/13-7
TSEC2_TXD[6]	TSEC2 Transmit Data	TSEC2	GPOUT[14]cfg_tsec2_prtcl[0]	1	O	13-1/13-7
TSEC2_TXD[5]	TSEC2 Transmit Data	TSEC2	GPOUT[13]cfg_tsec2_reduce	1	O	13-1/13-7
TSEC2_TXD[4]	TSEC2 Transmit Data	TSEC2	GPOUT[12]cfg_dram_type[0]	1	O	13-1/13-7
TSEC2_TXD[3]	TSEC2 Transmit Data	TSEC2	GPOUT[11]cfg_rom_loc[3]	1	O	13-1/13-7
TSEC2_TXD[2]	TSEC2 Transmit Data	TSEC2	GPOUT[10]cfg_rom_loc[2]	1	O	13-1/13-7
TSEC2_TXD[1]	TSEC2 Transmit Data	TSEC2	GPOUT[9]cfg_rom_loc[1]	1	O	13-1/13-7
TSEC2_TXD[0]	TSEC2 Transmit Data	TSEC2	GPOUT[8]cfg_rom_loc[0]	1	O	13-1/13-7
TSEC2_TX_EN	TSEC2 Transmit Enable	TSEC2	—	1	O	13-1/13-7

Table 3-1. MPC8641D Signal Reference by Functional Block (continued)

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/Page
TSEC2_TX_ER	TSEC2 Transmit Error	TSEC2	cfg_dram_type[1]	1	O	13-1/13-7
TSEC2_TX_CLK	TSEC2 Transmit Clock In	TSEC2	—	1	I	13-1/13-7
TSEC2_GTX_CLK	TSEC2 Transmit Clock Out	TSEC2	—	1	O	13-1/13-7
TSEC2_CRS	TSEC2 Carrier Sense	TSEC2	—	1	I/O	13-1/13-7
TSEC2_COL	TSEC2 Collision Detect	TSEC2	—	1	I	13-1/13-7
TSEC2_RXD[7:0]	TSEC2 Receive Data	TSEC2	GPIN[15:8]	4	I	13-1/13-7
TSEC2_RX_DV	TSEC2 Receive Data Valid	TSEC2	—	1	I	13-1/13-7
TSEC2_RX_ER	TSEC2 Receiver Error	TSEC2	—	1	I	13-1/13-7
TSEC2_RX_CLK	TSEC2 Receive Clock	TSEC2	—	1	I	13-1/13-7
eTSEC3 Signals						
TSEC3_TXD[7]	TSEC3 Transmit Data	TSEC3	cfg_tsec3_prtc[1]	1	O	13-1/13-7
TSEC3_TXD[6]	TSEC3 Transmit Data	TSEC3	cfg_tsec3_prtcl[0]	1	O	13-1/13-7
TSEC3_TXD[5]	TSEC3 Transmit Data	TSEC3	cfg_tsec3_reduce	1	O	13-1/13-7
TSEC3_TXD[4]	TSEC3 Transmit Data	TSEC3	—	1	O	13-1/13-7
TSEC3_TXD[3]	TSEC3 Transmit Data	TSEC3	$\overline{\text{cfg\_core1\_lm\_offset}}$	1	O	13-1/13-7
TSEC3_TXD[2]	TSEC3 Transmit Data	TSEC3	cfg_core1_enable	1	O	13-1/13-7
TSEC3_TXD[1]	TSEC3 Transmit Data	TSEC3		1	O	13-1/13-7
TSEC3_TXD[0]	TSEC3 Transmit Data	TSEC3		1	O	13-1/13-7
TSEC3_TX_EN	TSEC3 Transmit Enable	TSEC3	—	1	O	13-1/13-7
TSEC3_TX_ER	TSEC3 Transmit Error	TSEC3	—	1	O	13-1/13-7
TSEC3_TX_CLK	TSEC3 Transmit Clock In	TSEC3	—	1	I	13-1/13-7
TSEC3_GTX_CLK	TSEC3 Transmit Clock Out	TSEC3	—	1	O	13-1/13-7
TSEC3_CRS	TSEC3 Carrier Sense	TSEC3	—	1	I/O	13-1/13-7
TSEC3_COL	TSEC3 Collision Detect	TSEC3	—	1	I	13-1/13-7
TSEC3_RXD[7:0]	TSEC3 Receive Data	TSEC3	—	8	I	13-1/13-7
TSEC3_RX_DV	TSEC3 Receive Data Valid	TSEC3	—	1	I	13-1/13-7

Table 3-1. MPC8641D Signal Reference by Functional Block (continued)

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/Page
TSEC3_RX_ER	TSEC3 Receiver Error	TSEC3	—	1	I	13-1/13-7
TSEC3_RX_CLK	TSEC3 Receive Clock	TSEC3	—	1	I	13-1/13-7
eTSEC4 Signals						
TSEC4_TXD[7]	TSEC4 Transmit Data	TSEC4	cfg_tsec4_prtcl[1]	1	O	13-1/13-7
TSEC4_TXD[6]	TSEC4 Transmit Data	TSEC4	cfg_tsec4_prtcl[0]	1	O	13-1/13-7
TSEC4_TXD[5]	TSEC4 Transmit Data	TSEC4	cfg_tsec4_reduce	1	O	13-1/13-7
TSEC4_TXD[4]	TSEC4 Transmit Data	TSEC4	—	1	O	13-1/13-7
TSEC4_TXD[3]	TSEC4 Transmit Data	TSEC4	cfg_io_ports[3]	1	O	13-1/13-7
TSEC4_TXD[2]	TSEC4 Transmit Data	TSEC4	cfg_io_ports[2]	1	O	13-1/13-7
TSEC4_TXD[1]	TSEC4 Transmit Data	TSEC4	cfg_io_ports[1]	1	O	13-1/13-7
TSEC4_TXD[0]	TSEC4 Transmit Data	TSEC4	cfg_io_ports[0]	1	O	13-1/13-7
TSEC4_TX_EN	TSEC4 Transmit Enable	TSEC4	—	1	O	13-1/13-7
TSEC4_TX_ER	TSEC4 Transmit Error	TSEC4	—	1	O	13-1/13-7
TSEC4_TX_CLK	TSEC4 Transmit Clock In	TSEC4	—	1	I	13-1/13-7
TSEC4_GTX_CLK	TSEC4 Transmit Clock Out	TSEC4	—	1	O	13-1/13-7
TSEC4_CRS	TSEC4 Carrier Sense	TSEC4	—	1	I/O	13-1/13-7
TSEC4_COL	TSEC4 Collision Detect	TSEC4	—	1	I	13-1/13-7
TSEC4_RXD[7:0]	TSEC4 Receive Data	TSEC4	—	8	I	13-1/13-7
TSEC4_RX_DV	TSEC4 Receive Data Valid	TSEC4	—	1	I	13-1/13-7
TSEC4_RX_ER	TSEC4 Receiver Error	TSEC4	—	1	I	13-1/13-7
TSEC4_RX_CLK	TSEC4 Receive Clock	TSEC4	—	1	I	13-1/13-7
Local Bus Signals						
LAD[0:31]	LBC Address/Data	LBC	cfg_gpporcr[0:31]	32	I/O	12-1/12-4
LDP[0:3]	LBC Data Parity	LBC	cfg_core_pll[0:3]	4	I/O	12-1/12-4
LA[27]	LBC Burst Address	LBC	cfg_core_pll[4]	1	O	12-1/12-4
LA[28:31]	LBC Port Address	LBC	cfg_sys_pll[0:3]	4	O	12-1/12-4
$\overline{\text{LCS}}$[0:4]	LBC Chip Select	LBC	—	5	O	12-1/12-4
$\overline{\text{LCS}}$[5]	LBC Chip Select	LBC	$\overline{\text{DMA\_DREQ}}$[2]	1	O	12-1/12-4
$\overline{\text{LCS}}$[6]	LBC Chip Select	LBC	$\overline{\text{DMA\_DACK}}$[2]	1	O	12-1/12-4

Table 3-1. MPC8641D Signal Reference by Functional Block (continued)

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/Page
$\overline{\text{LCS}}$[7]	LBC Chip Select	LBC	$\overline{\text{DMA\_DDONE}}$[2]	1	O	12-1/12-4
$\overline{\text{LWE}}$[0]/LSDDQM[0]/ $\overline{\text{LBS}}$[0]	LBC Write Enable/Data Mask/ Byte Select 0	LBC	cfg_cpu_boot	1	O	12-1/12-4
$\overline{\text{LWE}}$[1]/LSDDQM[1]/ $\overline{\text{LBS}}$[1]	LBC Write Enable/Data Mask/ Byte Select 1	LBC	cfg_rio_sys_size	1	O	12-1/12-4
$\overline{\text{LWE}}$[2]/LSDDQM[2]/ $\overline{\text{LBS}}$[2]	LBC Write Enable/Data Mask/ Byte Select 2	LBC	cfg_host_agt[0]	1	O	12-1/12-4
$\overline{\text{LWE}}$[3]/LSDDQM[3]/ $\overline{\text{LBS}}$[3]	LBC Write Enable/Data Mask/ Byte Select 3	LBC	cfg_host_agt[1]	1	O	12-1/12-4
LBCTL	LBC Data Buffer Control	LBC	—	1	O	12-1/12-4
LALE	LBC Address Latch Enable	LBC	—	1	O	12-1/12-4
LGPL0/LSDA10	LBC UPM General Purpose Line 0/SDRAM Address bit 10	LBC	—	1	O	12-1/12-4
LGPL1/$\overline{\text{LSDWE}}$	LBC GP Line 1/SDRAM Write Enable	LBC	—	1	O	12-1/12-4
LGPL2/$\overline{\text{LOE}}$/ $\overline{\text{LSDRAS}}$	LBC GP Line 2/Output Enable/SDRAM RAS	LBC	—	1	O	12-1/12-4
LGPL3/$\overline{\text{LSDCAS}}$	LBC GP Line 3/SDRAM CAS	LBC	cfg_boot_seq[0]l	1	O	12-1/12-4
LGPL4/$\overline{\text{LGTA}}$/ LUPWAIT/LPBSE	LBC GP Line 4/GPCM Terminate Access/UPM Wait/SDRAM Parity Byte Select	LBC	—	1	I/O	12-1/12-4
LGPL5	LBC GP Line 5 Address	LBC	cfg_boot_seq[1]l	1	O	12-1/12-4
LCKE	LBC Clock Enable	LBC	—	1	O	12-1/12-4
LCLK[0:2]	LBC Clock	LBC	—	3	O	12-1/12-4
LSYNC_IN	LBC DLL Synchronization	LBC	—	1	I	12-1/12-4
LSYNC_OUT	LBC DLL Synchronization	LBC	—	1	O	12-1/12-4

Table 3-1. MPC8641D Signal Reference by Functional Block (continued)

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/Page
LB_SRCID[0:4]	Memory Debug Source Port ID 0–4, Interface1	LBC	D1_MSRCID[0:4]	5	O	8-6/8-12
LB_DVAL	Memory Debug Data Valid, Interface1	LBC	D1_MDVAL	1	O	19-2/19-6
DMA Signals						
$\overline{\text{DMA\_DREQ}}$[0:1]	DMA Request 0–1	DMA	—	2	I	14-3/14-6
$\overline{\text{DMA\_DREQ2}}$	DMA Request 2	DMA	$\overline{\text{LCS5}}$	1	I	14-3/14-6
$\overline{\text{DMA\_DREQ3}}$	DMA Request 3	DMA	IRQ[9]	1	I	14-3/14-6
$\overline{\text{DMA\_DACK}}$[0:1]	DMA Acknowledge 0–1	DMA	—	2	O	14-3/14-6
$\overline{\text{DMA\_DACK2}}$	DMA Acknowledge 2	DMA	$\overline{\text{LCS6}}$	1	O	14-3/14-6
$\overline{\text{DMA\_DACK3}}$	DMA Acknowledge 3	DMA	IRQ[10]	1	O	14-3/14-6
$\overline{\text{DMA\_DDONE}}$[0:1]	DMA Done 0–1	DMA	—	2	O	14-3/14-6
$\overline{\text{DMA\_DDONE2}}$	DMA Done 2	DMA	$\overline{\text{LCS7}}$	1	O	14-3/14-6
$\overline{\text{DMA\_DDONE3}}$	DMA Done 3	DMA	IRQ[11]	1	O	14-3/14-6
Programmable Interrupt Controller (PIC) Signals						
$\overline{\text{MCP\_0}}$	Machine Check Processor 0	PIC	—	1	I	9-3/9-9
$\overline{\text{MCP\_1}}$ (MPC8641D–Specific)	Machine Check Processor 1 on the MPC8641D	PIC	—	1	I	9-3/9-9
IRQ[0:7]	External Interrupt 0–7	PIC	—	8	I	9-3/9-9
IRQ[8]	External Interrupt 8	PIC		1	I	9-3/9-9
IRQ[9]	External Interrupt 9	PIC	$\overline{\text{DMA\_DREQ[3]}}$	1	I	9-3/9-9
IRQ[10]	External Interrupt 10	PIC	$\overline{\text{DMA\_DACK[3]}}$	1	I	9-3/9-9
IRQ[11]	External Interrupt 11	PIC	$\overline{\text{DMA\_DDONE[3]}}$	1	I	9-3/9-9
$\overline{\text{IRQ\_OUT}}$	Interrupt Output	PIC	—	1	O	9-3/9-9
DUART Signals						
UART_SIN[0:1]	DUART Serial Data In	Dual UART	—	2	I	11-1/11-3
UART_SOUT[0:1]	DUART Serial Data Out	Dual UART	—	2	O	11-1/11-3
$\overline{\text{UART\_CTS}}$[0:1]	DUART Clear To Send	Dual UART	—	2	I	11-1/11-3

Table 3-1. MPC8641D Signal Reference by Functional Block (continued)

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/Page
$\overline{\text{UART\_RTS}}$[0:1]	DUART Ready to Send	Dual UART	—	2	O	11-1/11-3
I^2C Signals						
IIC1_SDA	I^2C Serial Data port 1	I^2C	—	1	I/O	10-2/10-4
IIC1_SCL	I^2C Serial Clock port 1	I^2C	—	1	I/O	10-2/10-4
IIC2_SDA	I^2C Serial Data port 2	I^2C	—	1	I/O	10-2/10-4
IIC2_SCL	I^2C Serial Clock port 2	I^2C	—	1	I/O	10-2/10-4
System Control Signals						
$\overline{\text{CKSTP\_IN}}$	Checkstop In	System Control	—	1	I	17-2/17-3
$\overline{\text{CKSTP\_OUT}}$	Checkstop Out	System Control	—	1	O	17-2/17-3
$\overline{\text{HRESET}}$	Hard Reset	System Control	—	1	I	4-2/4-2
$\overline{\text{HRESET\_REQ}}$	Hard Reset Request	System Control	—	1	O	4-2/4-2
$\overline{\text{SRESET\_0}}$	Soft Reset for core 0	System Control	—	1	I	4-2/4-2
$\overline{\text{SRESET\_1}}$ (MPC8641D–Specific)	Soft Reset for core 1 on the MPC8641D	System Control	—	1	I	4-2/4-2
READY	Device Ready	System Control	TRIG_OUT	1	O	4-2/4-2
$\overline{\text{SMI\_0}}$	System Management Interrupt for core 0	System Control	—	1	I	17-2/17-3
$\overline{\text{SMI\_1}}$ (MPC8641D–Specific)	System Management Interrupt for core 1 on the MPC8641D	System Control	—	1	I	17-2/17-3
General Purpose I/O Signals						
GPIN[7:0]	General-purpose input	General-purpose I/O	TSEC1_RXD[7:0]	8	I	17-2/17-3 17.4.1.7.3/17-15
GPIN[15:8]	General-purpose input	General-purpose I/O	TSEC2_RXD[7:0]	8	I	17-2/17-3 17.4.1.7.3/17-15
GPOUT[7:0]	General-purpose output	General-purpose I/O	TSEC1_TXD[7:0]	8	O	17-2/17-3 17.4.1.7.2/17-15
GPOUT[15:8]	General-purpose output	General-purpose I/O	TSEC2_TXD[7:0]	8	O	17-2/17-3 17.4.1.7.2/17-15

Table 3-1. MPC8641D Signal Reference by Functional Block (continued)

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/Page
Debug Signals						
TRIG_IN	Watchpoint Trigger In	Debug	—	1	I	19-2/19-6
TRIG_OUT	Watchpoint Trigger Out	Debug	READY	1	O	19-2/19-6
D1_MSRCID[0]	Debug Interface 1 Memory Debug Source Port ID 0	Debug	LB_SRCID[0] cfg_mem_debug	1	O	8-6/8-12
D1_MSRCID[1]	Debug Interface 1 Memory Debug Source Port ID 1	Debug	LB_SRCID[1] cfg_ddr_debugl	1	O	8-6/8-12
D1_MSRCID[2]	Debug Interface 1 Memory Debug Source Port ID 2	Debug	LB_SRCID[2]	1	O	8-6/8-12
D1_MSRCID[3:4]	Debug Interface 1 Memory Debug Source Port ID 4	Debug	LB_SRCID[3:4]	2	O	8-6/8-12
D1_MDVAL	Debug Interface 1 Memory Debug Data Valid	Debug	LB_DVAL	1	O	19-2/19-6
D2_MSRCID[0:4]	Debug Interface 2 Memory Debug Source Port ID 0–4	Debug	—	5	O	8-6/8-12
D2_MDVAL	Debug Interface 2 Memory Debug Data Valid	Debug	—	1	O	19-2/19-6
Power Management Signals						
ASLEEP	Asleep	Power Mgmt	—	1	O	17-2/17-3
System Clocking Signals						
SYSCLK	System Clock	Clock	—	1	I	4-3/4-3
RTC	Real Time Clock	Clock	—	1	I	4-3/4-3
CLK_OUT	Clock Out	Clock	—	1	O	17-2/17-3
Analog Signals						
$D1\_MV_{REF}$	DDR SSTL2 Reference Voltage	DDR Memory Controller 1	—	1	Analog	8-4/8-10
$D2\_MV_{REF}$	DDR SSTL2 Reference Voltage	DDR Memory Controller 2	—	1	Analog	8-4/8-10
SD1_DLL_TPA	Test Point (Analog) for SerDes1 DLL	High Speed IO 1	—	1	Analog	17-2/17-3

Table 3-1. MPC8641D Signal Reference by Functional Block (continued)

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/Page
SD1_PLL_TPA	Test Point (Analog) SerDes1 PLL	High Speed IO 1	—	1	Analog	17-2/17-3
SD2_DLL_TPA	Test Point (Analog) for SerDes2 DLL	High Speed IO 2	—	1	Analog	17-2/17-3
SD2_PLL_TPA	Test Point (Analog) for SerDes2 PLL	High Speed IO 2	—	1	Analog	17-2/17-3
TEMP_ANODE	Thermal Diode	Test	—	1	Analog	19-2/19-6
TEMP_CATHODE	Thermal Diode	Test	—	1	Analog	19-2/19-6
Test Signals						
TEST_MODE[0:3]	Test Mode	Test	—	4	I	19-2/19-6
$\overline{\text{LSSD\_MODE}}$	LSSD Mode	Test	—	1	I	19-2/19-6
JTAG Signals						
TCK	Test Clock	JTAG	—	1	I	19-2/19-6
TDI	Test Data In	JTAG	—	1	I	19-2/19-6
TDO	Test Data Out	JTAG	—	1	O	19-2/19-6
TMS	Test Mode Select	JTAG	—	1	I	19-2/19-6
$\overline{\text{TRST}}$	Test Reset	JTAG	—	1	I	19-2/19-6

Table 3-2 provides the summary list of the signals grouped alphabetically.

Table 3-2. MPC8641D Signal Names Alphabetical Reference

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/Page
ASLEEP	Asleep	Power Mgmt	—	1	O	17-2/17-3
$\overline{\text{CKSTP\_IN}}$	Checkstop In	System Control	—	1	I	17-2/17-3
$\overline{\text{CKSTP\_OUT}}$	Checkstop Out	System Control	—	1	O	17-2/17-3
CLK_OUT	Clock Out	Clock	—	1	O	17-2/17-3
D1_MA[15:0]	DDR Address	DDR Controller 1	—	16	O	8-3/8-6
D1_MBA[2:0]	DDR Bank Select	DDR Controller 1	—	3	O	8-3/8-6
$\overline{\text{D1\_MCAS}}$	DDR Column Address Strobe	DDR Controller 1	—	1	O	8-3/8-6
D1_MCK[0:5]	DDR Clocks (3 pairs/DIMM)	DDR Controller 1	—	6	O	8-4/8-10

Table 3-2. MPC8641D Signal Names Alphabetical Reference (continued)

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/Page
$\overline{\text{D1\_MCK}}$[0:5]	DDR Neg. Clocks (3 pairs/DIMM)	DDR Controller 1	—	6	O	8-4/8-10
D1_MCKE[0:3]	DDR Clock Enable	DDR Controller 1	—	4	O	8-4/8-10
$\overline{\text{D1\_MCS}}$[0:3]	DDR Chip Select (2/DIMM)	DDR Controller 1	—	4	O	8-3/8-6
D1_MDIC[0:1]	DDR Output Conditioning	DDR Controller 1	—	2	I/O	8-4/8-10
D1_MDM[0:7]	DDR Data Mask	DDR Controller 1	—	8	O	8-3/8-6
D1_MDM[8]	DDR ECC Data Mask	DDR Controller 1	—	1	O	8-3/8-6
D1_MDQ[0:63]	DDR Data	DDR Controller 1	—	64	I/O	8-3/8-6
D1_MDQS[0:7]	DDR Data Strobe	DDR Controller 1	—	8	I/O	8-3/8-6
$\overline{\text{D1\_MDQS}}$[0:7]	DDR Neg. Data Strobe	DDR Controller 1	—	8	I/O	8-3/8-6
D1_MDQS[8]	DDR ECC Data Strobe	DDR Controller 1	—	1	I/O	8-3/8-6
$\overline{\text{D1\_MDQS}}$[8]	DDR Neg. ECC Data Strobe	DDR Controller 1	—	1	I/O	8-3/8-6
D1_MDVAL	Debug Interface 1 Memory Debug Data Valid	Debug	LB_DVAL	1	O	19-2/19-6
D1_MECC[0:7]	DDR Error Correcting Code	DDR Controller 1	—	8	I/O	8-3/8-6
D1_MODT[0:3]	DDR On Device Termination	DDR Controller 1	—	4	O	8-4/8-10
$\overline{\text{D1\_MRAS}}$	DDR Row Address Strobe	DDR Controller 1	—	1	O	8-3/8-6
D1_MSRCID[0]	Debug Interface 1 Memory Debug Source Port ID 0	Debug	LB_SRCID[0] cfg_mem_debug	1	O	8-6/8-12
D1_MSRCID[1]	Debug Interface 1 Memory Debug Source Port ID 1	Debug	LB_SRCID[1] cfg_ddr_debugl	1	O	8-6/8-12
D1_MSRCID[2]	Debug Interface 1 Memory Debug Source Port ID 2	Debug	LB_SRCID[2]	1	O	8-6/8-12
D1_MSRCID[3:4]	Debug Interface 1 Memory Debug Source Port ID 4	Debug	LB_SRCID[3:4]	2	O	8-6/8-12
$D1\_MV_{REF}$	DDR SSTL2 Reference Voltage	DDR Controller 1	—	1	Analog	8-4/8-10
$\overline{\text{D1\_MWE}}$	DDR Write Enable	DDR Controller 1	—	1	O	8-3/8-6

Table 3-2. MPC8641D Signal Names Alphabetical Reference (continued)

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/Page
D2_MA[15:0]	DDR Address	DDR Controller 2	—	16	O	8-3/8-6
D2_MBA[2:0]	DDR Bank Select	DDR Controller 2	—	3	O	8-3/8-6
$\overline{\text{D2\_MCAS}}$	DDR Column Address Strobe	DDR Controller 2	—	1	O	8-3/8-6
D2_MCK[0:5]	DDR Clocks (3 pairs/DIMM)	DDR Controller 2	—	6	O	8-4/8-10
$\overline{\text{D2\_MCK}}$[0:5]	DDR Neg. Clocks (3 pairs/DIMM)	DDR Controller 2	—	6	O	8-4/8-10
D2_MCKE[0:3]	DDR Clock Enable	DDR Controller 2	—	4	O	8-4/8-10
$\overline{\text{D2\_MCS}}$[0:3]	DDR Chip Select (2/DIMM)	DDR Controller 2	—	4	O	8-3/8-6
D2_MDIC[0:1]	DDR Output Conditioning	DDR Controller 2	—	2	IO	8-4/8-10
D2_MDM[0:7]	DDR Data Mask	DDR Controller 2	—	8	O	8-3/8-6
D2_MDM[8]	DDR ECC Data Mask	DDR Controller 2	—	1	O	8-3/8-6
D2_MDQ[0:63]	DDR Data	DDR Controller 2	—	64	I/O	8-3/8-6
D2_MDQS[0:7]	DDR Data Strobe	DDR Controller 2	—	8	I/O	8-3/8-6
$\overline{\text{D2\_MDQS}}$[0:7]	DDR Neg. Data Strobe	DDR Controller 2	—	8	I/O	8-3/8-6
D2_MDQS[8]	DDR ECC Data Strobe	DDR Controller 2	—	1	I/O	8-3/8-6
$\overline{\text{D2\_MDQS}}$[8]	DDR Neg. ECC Data Strobe	DDR Controller 2	—	1	I/O	8-3/8-6
D2_MDVAL	Debug Interface 2 Memory Debug Data Valid	Debug	—	1	O	19-2/19-6
D2_MECC[0:7]	DDR Error Correcting Code	DDR Controller 2	—	8	I/O	8-3/8-6
D2_MODT[0:3]	DDR On Device Termination	DDR Controller 2	—	4	O	8-4/8-10
$\overline{\text{D2\_MRAS}}$	DDR Row Address Strobe	DDR Controller 2	—	1	O	8-3/8-6
D2_MSRCID[0:4]	Debug Interface 2 Memory Debug Source Port ID 0–4	Debug	—	5	O	8-6/8-12
$\text{D2\_MV}_{\text{REF}}$	DDR SSTL2 Reference Voltage	DDR Controller 2	—	1	Analog	8-4/8-10
$\overline{\text{D2\_MWE}}$	DDR Write Enable	DDR Controller 2	—	1	O	8-3/8-6
$\overline{\text{DMA\_DACK}}$[0:1]	DMA Acknowledge 0–1	DMA	—	2	O	14-3/14-6

Table 3-2. MPC8641D Signal Names Alphabetical 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	14-3/14-6
$\overline{\text{DMA\_DACK3}}$	DMA Acknowledge 3	DMA	IRQ[10]	1	O	14-3/14-6
$\overline{\text{DMA\_DDONE}}$[0:1]	DMA Done 0–1	DMA	—	2	O	14-3/14-6
$\overline{\text{DMA\_DDONE2}}$	DMA Done 2	DMA	$\overline{\text{LCS7}}$	1	O	14-3/14-6
$\overline{\text{DMA\_DDONE3}}$	DMA Done 3	DMA	IRQ[11]	1	O	14-3/14-6
$\overline{\text{DMA\_DREQ}}$[0:1]	DMA Request 0–1	DMA	—	2	I	14-3/14-6
$\overline{\text{DMA\_DREQ2}}$	DMA Request 2	DMA	$\overline{\text{LCS5}}$	1	I	14-3/14-6
$\overline{\text{DMA\_DREQ3}}$	DMA Request 3	DMA	IRQ[9]	1	I	14-3/14-6
EC_MDC	Ethernet Management Data Clock	Ethernet Management	—	1	O	13-2/13-9
EC_MDIO	Ethernet Management Data In/Out	Ethernet Management	—	1	I/O	13-2/13-9
EC1_GTX_CLK125	Gigabit Reference Clock 1	Gigabit Clock	—	1	I	13-2/13-9
EC2_GTX_CLK125	Gigabit Reference Clock 2	Gigabit Clock	—	1	I	13-2/13-9
GPIN[15:8]	General-purpose input	General-purpose I/O	TSEC2_RXD[7:0]	8	I	17-2/17-3 17.4.1.7.3/17-15
GPIN[7:0]	General-purpose input	General-purpose I/O	TSEC1_RXD[7:0]	8	I	17-2/17-3 17.4.1.7.3/17-15
GPOUT[15:8]	General-purpose output	General-purpose I/O	TSEC2_TXD[7:0]	8	O	17-2/17-3 17.4.1.7.2/17-15
GPOUT[7:0]	General-purpose output	General-purpose I/O	TSEC1_TXD[7:0]	8	O	17-2/17-3 17.4.1.7.2/17-15
$\overline{\text{HRESET}}$	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
IIC1_SCL	I^2C Serial Clock port 1	I^2C	—	1	I/O	10-2/10-4
IIC1_SDA	I^2C Serial Data port 1	I^2C	—	1	I/O	10-2/10-4
IIC2_SCL	I^2C Serial Clock port 2	I^2C	—	1	I/O	10-2/10-4
IIC2_SDA	I^2C Serial Data port 2	I^2C	—	1	I/O	10-2/10-4

Table 3-2. MPC8641D Signal Names Alphabetical Reference (continued)

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/Page
IRQ[0:7]	External Interrupt 0–7	PIC	—	8	I	9-3/9-9
IRQ[10]	External Interrupt 10	PIC	DMA_DACK[3]	1	I	9-3/9-9
IRQ[11]	External Interrupt 11	PIC	DMA_DDONE[3]	1	I	9-3/9-9
IRQ[8]	External Interrupt 8	PIC		1	I	9-3/9-9
IRQ[9]	External Interrupt 9	PIC	DMA_DREQ[3]	1	I	9-3/9-9
IRQ_OUT	Interrupt Output	PIC	—	1	O	9-3/9-9
LA[27]	LBC Burst Address	LBC	cfg_core_pll[4]	1	O	12-1/12-4
LA[28:31]	LBC Port Address	LBC	cfg_sys_pll[0:3]	4	O	12-1/12-4
LAD[0:31]	LBC Address/Data	LBC	cfg_gpporcr[0:31]	32	I/O	12-1/12-4
LALE	LBC Address Latch Enable	LBC	—	1	O	12-1/12-4
LB_DVAL	Memory Debug Data Valid, Interface1	LBC	D1_MDVAL	1	O	19-2/19-6
LB_SRCID[0:4]	Memory Debug Source Port ID 0–4, Interface1	LBC	D1_MSRCID[0:4]	5	O	8-6/8-12
LBCTL	LBC Data Buffer Control	LBC	—	1	O	12-1/12-4
LCKE	LBC Clock Enable	LBC	—	1	O	12-1/12-4
LCLK[0:2]	LBC Clock	LBC	—	3	O	12-1/12-4
LCS[0:4]	LBC Chip Select	LBC	—	5	O	12-1/12-4
LCS[5]	LBC Chip Select	LBC	DMA_DREQ[2]	1	O	12-1/12-4
LCS[6]	LBC Chip Select	LBC	DMA_DACK[2]	1	O	12-1/12-4
LCS[7]	LBC Chip Select	LBC	DMA_DDONE[2]	1	O	12-1/12-4
LDP[0:3]	LBC Data Parity	LBC	cfg_core_pll[0:3]	4	I/O	12-1/12-4
LGPL0/LSDA10	LBC UPM General Purpose Line 0/SDRAM Address bit 10	LBC	—	1	O	12-1/12-4
LGPL1/LSDWE	LBC GP Line 1/SDRAM Write Enable	LBC	—	1	O	12-1/12-4
LGPL2/LOE/ LSDRAS	LBC GP Line 2/Output Enable/SDRAM RAS	LBC	—	1	O	12-1/12-4
LGPL3/LSDCAS	LBC GP Line 3/SDRAM CAS	LBC	cfg_boot_seq[0]l	1	O	12-1/12-4

Table 3-2. MPC8641D Signal Names Alphabetical Reference (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/SDRAM Parity Byte Select	LBC	—	1	I/O	12-1/12-4
LGPL5	LBC GP Line 5 Address	LBC	cfg_boot_seq[1]l	1	O	12-1/12-4
LSSD_MODE	LSSD Mode	Test	—	1	I	19-2/19-6
LSYNC_IN	LBC DLL Synchronization	LBC	—	1	I	12-1/12-4
LSYNC_OUT	LBC DLL Synchronization	LBC	—	1	O	12-1/12-4
LWE[0]/LSDDQM[0]/ LBS[0]	LBC Write Enable/Data Mask/ Byte Select 0	LBC	cfg_cpu_boot	1	O	12-1/12-4
LWE[1]/LSDDQM[1]/ LBS[1]	LBC Write Enable/Data Mask/ Byte Select 1	LBC	cfg_rio_sys_size	1	O	12-1/12-4
LWE[2]/LSDDQM[2]/ LBS[2]	LBC Write Enable/Data Mask/ Byte Select 2	LBC	cfg_host_agt[0]	1	O	12-1/12-4
LWE[3]/LSDDQM[3]/ LBS[3]	LBC Write Enable/Data Mask/ Byte Select 3	LBC	cfg_host_agt[1]	1	O	12-1/12-4
MCP_0	Machine Check Processor 0	PIC	—	1	I	9-3/9-9
MCP_1 (MPC8641D–Specific)	Machine Check Processor 1 on the MPC8641D	PIC	—	1	I	9-3/9-9
READY	Device Ready	System Control	TRIG_OUT	1	O	4-2/4-2
RTC	Real Time Clock	Clock	—	1	I	4-3/4-3
SD1_DLL_TPA	Test Point (Analog) for SerDes1 DLL	High Speed IO 1	—	1	Analog	MPC8641DEC
SD1_DLL_TPD	Test Point (Digital) SerDes1 DLL	High Speed IO 1	—	1	O	MPC8641DEC
SD1_PLL_TPA	Test Point (Analog) SerDes1 PLL	High Speed IO 1	—	1	Analog	MPC8641DEC
SD1_PLL_TPD	Test Point for (Digital) SerDes1 PLL	High Speed IO 1	—	1	O	MPC8641DEC
SD1_REF_CLK, SD1_REF_CLK	PLL Reference Clock	High Speed IO 1	—	2	I	MPC8641DEC

Table 3-2. MPC8641D Signal Names Alphabetical Reference (continued)

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/Page
SD1_RX[0:7], $\overline{\text{SD1\_RX[0:7]}}$	Receive Data	High Speed IO 1	—	16	I	16.2/16-4
SD1_TX[0:7], $\overline{\text{SD1\_TX[0:7]}}$	Transmit Data	High Speed IO 1	—	16	O	16.2/16-4
SD2_DLL_TPA	Test Point (Analog) for SerDes2 DLL	High Speed IO 2	—	1	Analog	MPC8641DEC
SD2_DLL_TPD	Test Point (Digital) SerDes2 DLL	High Speed IO 2	—	1	O	MPC8641DEC
SD2_PLL_TPA	Test Point (Analog) for SerDes2 PLL	High Speed IO 2	—	1	Analog	MPC8641DEC
SD2_PLL_TPD	Test Point (Digital) SerDes2 PLL	High Speed IO 2	—	1	O	MPC8641DEC
SD2_REF_CLK, $\overline{\text{SD2\_REF\_CLK}}$	PLL Reference Clock	High Speed IO 2	—	2	I	MPC8641DEC
SD2_RX[0:7], $\overline{\text{SD2\_RX[0:7]}}$	Receive Data	High Speed IO 2	—	16	I	15.4/15-3 (SRIO) 16.2/16-4 (PEX2)
SD2_TX[0:7], $\overline{\text{SD2\_TX[0:7]}}$	Transmit Data	High Speed IO 2	—	16	O	15.4/15-3 (SRIO) 16.2/16-4 (PEX2)
$\overline{\text{SMI\_0}}$	System Management Interrupt for core 0	System Control	—	1	I	17-2/17-3
$\overline{\text{SMI\_1}}$ (MPC8641D–Specific)	System Management Interrupt for core 1 on the MPC8641D	System Control	—	1	I	17-2/17-3
$\overline{\text{SRESET\_0}}$	Soft Reset for core 0	System Control	—	1	I	4-2/4-2
$\overline{\text{SRESET\_1}}$ (MPC8641D–Specific)	Soft Reset for core 1 on the MPC8641D	System Control	—	1	I	4-2/4-2
SYSCLK	System Clock	Clock	—	1	I	4-3/4-3
TCK	Test Clock	JTAG	—	1	I	19-2/19-6
TDI	Test Data In	JTAG	—	1	I	19-2/19-6
TDO	Test Data Out	JTAG	—	1	O	19-2/19-6
TEMP_ANODE	Thermal Diode	Test	—	1	Analog	19-2/19-6
TEMP_CATHODE	Thermal Diode	Test	—	1	Analog	19-2/19-6
TEST_MODE[0:3]	Test Mode	Test	—	4	I	19-2/19-6

Table 3-2. MPC8641D Signal Names Alphabetical Reference (continued)

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/Page
TMS	Test Mode Select	JTAG	—	1	I	19-2/19-6
TRIG_IN	Watchpoint Trigger In	Debug	—	1	I	19-2/19-6
TRIG_OUT	Watchpoint Trigger Out	Debug	READY	1	O	19-2/19-6
$\overline{\text{TRST}}$	Test Reset	JTAG	—	1	I	19-2/19-6
TSEC1_COL	TSEC1 Collision Detect	TSEC1	—	1	I	13-1/13-7
TSEC1_CRS	TSEC1 Carrier Sense	TSEC1	—	1	I/O	13-1/13-7
TSEC1_GTX_CLK	TSEC1 Transmit Clock Out	TSEC1	—	1	O	13-1/13-7
TSEC1_RX_CLK	TSEC1 Receive Clock	TSEC1	—	1	I	13-1/13-7
TSEC1_RX_DV	TSEC1 Receive Data Valid	TSEC1	—	1	I	13-1/13-7
TSEC1_RX_ER	TSEC1 Receiver Error	TSEC1	—	1	I	13-1/13-7
TSEC1_RXD[7:0]	TSEC1 Receive Data	TSEC1	GPIN[7:0]	8	I	13-1/13-7
TSEC1_TX_CLK	TSEC1 Transmit Clock In	TSEC1	—	1	I	13-1/13-7
TSEC1_TX_EN	TSEC1 Transmit Enable	TSEC1	—	1	O	13-1/13-7
TSEC1_TX_ER	TSEC1 Transmit Error	TSEC1	—	1	O	13-1/13-7
TSEC1_TXD[0]	TSEC1 Transmit Data	TSEC1	GPOUT[0]	1	O	13-1/13-7
TSEC1_TXD[1]	TSEC1 Transmit Data	TSEC1	GPOUT[1] / cfg_platform_freq	1	O	13-1/13-7
TSEC1_TXD[2]	TSEC1 Transmit Data	TSEC1	GPOUT[2] / cfg_device_id[5]	1	O	13-1/13-7
TSEC1_TXD[3]	TSEC1 Transmit Data	TSEC1	GPOUT[3]cfg_device_id[6]	1	O	13-1/13-7
TSEC1_TXD[4]	TSEC1 Transmit Data	TSEC1	GPOUT[4]cfg_device_id[7]	1	O	13-1/13-7
TSEC1_TXD[5]	TSEC1 Transmit Data	TSEC1	GPOUT[5]cfg_tsec1_reduce	1	O	13-1/13-7
TSEC1_TXD[6]	TSEC1 Transmit Data	TSEC1	GPOUT[6]cfg_tsec1_prtcl[0]	1	O	13-1/13-7
TSEC1_TXD[7]	TSEC1 Transmit Data	TSEC1	GPOUT[7]cfg_tsec1_prtcl[1]	1	O	13-1/13-7
TSEC2_COL	TSEC2 Collision Detect	TSEC2	—	1	I	13-1/13-7
TSEC2_CRS	TSEC2 Carrier Sense	TSEC2	—	1	I/O	13-1/13-7

Table 3-2. MPC8641D Signal Names Alphabetical Reference (continued)

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/Page
TSEC2_GTX_CLK	TSEC2 Transmit Clock Out	TSEC2	—	1	O	13-1/13-7
TSEC2_RX_CLK	TSEC2 Receive Clock	TSEC2	—	1	I	13-1/13-7
TSEC2_RX_DV	TSEC2 Receive Data Valid	TSEC2	—	1	I	13-1/13-7
TSEC2_RX_ER	TSEC2 Receiver Error	TSEC2	—	1	I	13-1/13-7
TSEC2_RXD[7:0]	TSEC2 Receive Data	TSEC2	GPIN[15:8]	4	I	13-1/13-7
TSEC2_TX_CLK	TSEC2 Transmit Clock In	TSEC2	—	1	I	13-1/13-7
TSEC2_TX_EN	TSEC2 Transmit Enable	TSEC2	—	1	O	13-1/13-7
TSEC2_TX_ER	TSEC2 Transmit Error	TSEC2	cfg_dram_type[1]	1	O	13-1/13-7
TSEC2_TXD[0]	TSEC2 Transmit Data	TSEC2	GPOUT[8]cfg_rom_loc[0]	1	O	13-1/13-7
TSEC2_TXD[1]	TSEC2 Transmit Data	TSEC2	GPOUT[9]cfg_rom_loc[1]	1	O	13-1/13-7
TSEC2_TXD[2]	TSEC2 Transmit Data	TSEC2	GPOUT[10]cfg_rom_loc[2]	1	O	13-1/13-7
TSEC2_TXD[3]	TSEC2 Transmit Data	TSEC2	GPOUT[11]cfg_rom_loc[3]	1	O	13-1/13-7
TSEC2_TXD[4]	TSEC2 Transmit Data	TSEC2	GPOUT[12]cfg_dram_type[0]	1	O	13-1/13-7
TSEC2_TXD[5]	TSEC2 Transmit Data	TSEC2	GPOUT[13]cfg_tsec2_reduce	1	O	13-1/13-7
TSEC2_TXD[6]	TSEC2 Transmit Data	TSEC2	GPOUT[14]cfg_tsec2_prtcl[0]	1	O	13-1/13-7
TSEC2_TXD[7]	TSEC2 Transmit Data	TSEC2	GPOUT15]cfg_tsec2_prtcl[1]	1	O	13-1/13-7
TSEC3_COL	TSEC3 Collision Detect	TSEC3	—	1	I	13-1/13-7
TSEC3_CRS	TSEC3 Carrier Sense	TSEC3	—	1	I/O	13-1/13-7
TSEC3_GTX_CLK	TSEC3 Transmit Clock Out	TSEC3	—	1	O	13-1/13-7
TSEC3_RX_CLK	TSEC3 Receive Clock	TSEC3	—	1	I	13-1/13-7
TSEC3_RX_DV	TSEC3 Receive Data Valid	TSEC3	—	1	I	13-1/13-7
TSEC3_RX_ER	TSEC3 Receiver Error	TSEC3	—	1	I	13-1/13-7
TSEC3_RXD[7:0]	TSEC3 Receive Data	TSEC3	—	8	I	13-1/13-7
TSEC3_TX_CLK	TSEC3 Transmit Clock In	TSEC3	—	1	I	13-1/13-7

Table 3-2. MPC8641D Signal Names Alphabetical Reference (continued)

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/Page
TSEC3_TX_EN	TSEC3 Transmit Enable	TSEC3	—	1	O	13-1/13-7
TSEC3_TX_ER	TSEC3 Transmit Error	TSEC3	—	1	O	13-1/13-7
TSEC3_TXD[0]	TSEC3 Transmit Data	TSEC3	—	1	O	13-1/13-7
TSEC3_TXD[1]	TSEC3 Transmit Data	TSEC3	—	1	O	13-1/13-7
TSEC3_TXD[2]	TSEC3 Transmit Data	TSEC3	cfg_core1_enable	1	O	13-1/13-7
TSEC3_TXD[3]	TSEC3 Transmit Data	TSEC3	$\overline{\text{cfg\_core1\_lm\_offset}}$	1	O	13-1/13-7
TSEC3_TXD[4]	TSEC3 Transmit Data	TSEC3	—	1	O	13-1/13-7
TSEC3_TXD[5]	TSEC3 Transmit Data	TSEC3	cfg_tsec3_reduce	1	O	13-1/13-7
TSEC3_TXD[6]	TSEC3 Transmit Data	TSEC3	cfg_tsec3_prtcl[0]	1	O	13-1/13-7
TSEC3_TXD[7]	TSEC3 Transmit Data	TSEC3	cfg_tsec3_prtc[1]	1	O	13-1/13-7
TSEC4_COL	TSEC4 Collision Detect	TSEC4	—	1	I	13-1/13-7
TSEC4_CRS	TSEC4 Carrier Sense	TSEC4	—	1	I/O	13-1/13-7
TSEC4_GTX_CLK	TSEC4 Transmit Clock Out	TSEC4	—	1	O	13-1/13-7
TSEC4_RX_CLK	TSEC4 Receive Clock	TSEC4	—	1	I	13-1/13-7
TSEC4_RX_DV	TSEC4 Receive Data Valid	TSEC4	—	1	I	13-1/13-7
TSEC4_RX_ER	TSEC4 Receiver Error	TSEC4	—	1	I	13-1/13-7
TSEC4_RXD[7:0]	TSEC4 Receive Data	TSEC4	—	8	I	13-1/13-7
TSEC4_TX_CLK	TSEC4 Transmit Clock In	TSEC4	—	1	I	13-1/13-7
TSEC4_TX_EN	TSEC4 Transmit Enable	TSEC4	—	1	O	13-1/13-7
TSEC4_TX_ER	TSEC4 Transmit Error	TSEC4	—	1	O	13-1/13-7
TSEC4_TXD[0]	TSEC4 Transmit Data	TSEC4	cfg_io_ports[0]	1	O	13-1/13-7
TSEC4_TXD[1]	TSEC4 Transmit Data	TSEC4	cfg_io_ports[1]	1	O	13-1/13-7
TSEC4_TXD[2]	TSEC4 Transmit Data	TSEC4	cfg_io_ports[2]	1	O	13-1/13-7
TSEC4_TXD[3]	TSEC4 Transmit Data	TSEC4	cfg_io_ports[3]	1	O	13-1/13-7
TSEC4_TXD[4]	TSEC4 Transmit Data	TSEC4	—	1	O	13-1/13-7
TSEC4_TXD[5]	TSEC4 Transmit Data	TSEC4	cfg_tsec4_reduce	1	O	13-1/13-7
TSEC4_TXD[6]	TSEC4 Transmit Data	TSEC4	cfg_tsec4_prtcl[0]	1	O	13-1/13-7
TSEC4_TXD[7]	TSEC4 Transmit Data	TSEC4	cfg_tsec4_prtcl[1]	1	O	13-1/13-7

Table 3-2. MPC8641D Signal Names Alphabetical Reference (continued)

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/Page
$\overline{\text{UART\_CTS}}$[0:1]	DUART Clear To Send	Dual UART	—	2	I	11-1/11-3
$\overline{\text{UART\_RTS}}$[0:1]	DUART Ready to Send	Dual UART	—	2	O	11-1/11-3
UART_SIN[0:1]	DUART Serial Data In	Dual UART	—	2	I	11-1/11-3
UART_SOUT[0:1]	DUART Serial Data Out	Dual UART	—	2	O	11-1/11-3

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 *MPC8641D 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 *MPC8641D Integrated Processor Hardware Specifications*.

Note that the multiplexing of various signals on the MPC8641D is controlled by the PMUXCR register described in Chapter 17, “Global Utilities.”

Table 3-3. MPC8641D Reset Configuration Signals

Functional Interface	Functional Signal Name	Reset Configuration Name	Default
TSEC1	TSEC1_TXD[0]	$\overline{\text{cfg\_alt\_boot\_vec}}$	1
	TSEC1_TXD[1]	cfg_platform_freq	1
	TSEC1_TXD[2:4]	cfg_device_id[5:7]	111
	TSEC1_TXD[5]	cfg_tsec1_reduce	1
	TSEC1_TXD[6:7]	cfg_tsec1_prtcl[0:1]	11

Table 3-3. MPC8641D Reset Configuration Signals (continued)

Functional Interface	Functional Signal Name	Reset Configuration Name	Default
TSEC2	TSEC2_TXD[0:3]	cfg_rom_loc[0:3]	1111
	TSEC2_TXD[4]	cfg_dram_type[0]	1
	TSEC2_TX_ER	cfg_dram_type[1]	1
	TSEC2_TXD[5]	cfg_tsec2_reduce	1
	TSEC2_TXD[6:7]	cfg_tsec2_prtcl[0:1]	11
TSEC3	TSEC3_TXD[2]	cfg_core1_enable	1
	TSEC3_TXD[3]	$\overline{\text{cfg\_core1\_lm\_offset}}$	1
	TSEC3_TXD[5]	cfg_tsec3_reduce	1
	TSEC3_TXD[6:7]	cfg_tsec3_prtcl[0:1]	11
TSEC4	TSEC4_TXD[0:3]	cfg_io_ports[0:3]	1111
	TSEC4_TXD[5]	cfg_tsec4_reduce	1
	TSEC4_TXD[6:7]	cfg_tsec4_prtcl[0:1]	11
LBC	LAD[0:31]	cfg_gpporcr[0:31]	all 1's
	LA[28:31]	cfg_sys_pll[0:3]	1111
	LDP[0:3]	cfg_core_pll[0:3]	1111
	LA[27]	cfg_core_pll[4]	1
	$\overline{\text{LWE}}$[0]	cfg_cpu_boot	1
	$\overline{\text{LWE}}$[1]	cfg_rio_sys_size	1
	$\overline{\text{LWE}}$[2:3]	cfg_host_agt[0:1]	11
	LGPL[3]	cfg_boot_seq[0]	1
	LGPL[5]	cfg_boot_seq[1]	1
Debug	D1_MSRCID[0]	cfg_mem_debug	1
	D1_MSRCID[1]	cfg_ddr_debug	1

Chapter 4
Reset, Clocking, and Initialization

4.1 Overview

The reset, clocking, and control signals provide many options for the operation of the device. 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	—
$\overline{\text{SRESET\_0}}$	I	Soft reset input for processor 0. Causes core 0 to service a soft reset exception.	4.4.1.1/4-9
$\overline{\text{SRESET\_1}}$ (MPC8641D-specific signals)	I	Soft reset input for processor 1. Causes core 1 to service a soft reset exception.	4.4.1.1/4-9
READY (/TRIG_OUT)	O	The READY signal is functional when TOSR[SEL] = 0b000; this signal indicates that the device has completed the reset operation and is not in a power-down (nap, doze or sleep) or debug state. See Chapter 19, “Debug Features and Watchpoint Facilities,” for more information on TOSR and TRIG_OUT.	4.4.2/4-9
SYSCLK	I	Primary clock input to the device.	4.4.4.1/4-23
RTC	I	Real time clock input.	—

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 device. Section 4.4.3, "Power-On Reset Configuration," describes the signals that also function as reset configuration signals. Note that the CKSTP_IN and CKSTP_OUT signals are described in Chapter 17, "Global Utilities."

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

Signal	I/O	Description	
HRESET	I	Hard reset. Causes the device 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 device signals during reset.
		Timing	Assertion/Negation—The *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 device is embedded) that a condition requiring the assertion of HRESET has been detected.	
		State Meaning	Asserted—A serial RapidIO command or a boot sequencer failure (see Section 10.4.5, "Boot Sequencer Mode,") has triggered a request for hard reset. Negated—Indicates no reset request.
		Timing	Assertion/Negation—May occur anytime, synchronous to the MPX bus clock. Once asserted, HRESET_REQ does not negate until HRESET is asserted.
SRESET_0	I	Soft reset. An edge sensitive signal that causes a soft reset exception on e600 core 0.	
		State Meaning	Asserted—Asserting SRESET_0 causes a soft reset interrupt (edge sensitive) to e600 core. SRESET_0 has no effect while HRESET is asserted. However, the POR sequence is paused if SRESET_0 is asserted during POR.
		Timing	Assertion—May occur at any time, asynchronous to any clock. Negation—Must be asserted for at least two *MPX_clk* cycles.
SRESET_1 (MPC8641D-specific signals)	I	Soft reset. An edge sensitive signal that causes a soft reset exception on e600 core 1.	
		State Meaning	Asserted—Asserting SRESET_1 causes a soft reset interrupt (edge sensitive) to e600 core 1.
		Timing	Assertion—May occur at any time, asynchronous to any clock. Negation—Must be asserted for at least two *MPX_clk* cycles.
READY (/TRIG_OUT)	O	Ready. Reset status signal.	
		State Meaning	Asserted—Indicates that the device 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 has been synchronized 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 device. Note that some clock signals are specific to blocks within the device, 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 device; the signal driven on the CLK_OUT pin is selectable and described in Section 17.4.1.18, "Clock Out Control Register (CLKOCR)."

Table 4-3. Clock Signals—Detailed Signal Descriptions

Signal	I/O	Description	
SYSCLK	I	System clock. SYSCLK is the primary clock input to the device. It is multiplied up with a platform phased-lock loop (PLL) based on LA[28"31] at reset to create the MPX bus clock (also called the platform clock) which is used by virtually all of the synchronous system logic, including the DDR SDRAM and local bus memory controllers, and other internal blocks such as the DMA and interrupt controllers. The MPX bus clock, in turn, feeds the PLL based on LDP[0:3] and LA[27] at reset for the e600 cores.	
		Timing	Assertion/Negation—See the *Integrated Processor Hardware Specifications* for specific timing information for this signal.
RTC	I	Real time clock. This signal can be used to clock the global timers in the programmable interrupt controller (PIC).	
		Timing	Assertion/Negation—See the *Integrated Processor Hardware Specifications* for specific timing information for this signal. Note that this signal may be asynchronous to the platform or reference clock. It is sampled using the platform/MPX clock.

4.3 Memory Map/Register Definition

There are no unique registers in the reset and clocking blocks of the device. However, this section describes the configuration, control, and status registers of the overall device; it also contains a brief description of the boot sequencer.

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 Configuration Control—Block Base Address 0x0_0000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0x000	CCSRBAR—Configuration, control, and status registers base address register	R/W	0x000F_F700	4.3.1.1.3/4-5
0x008	ALTCBAR—Alternate configuration base address register	R/W	0x0000_0000	4.3.1.2.1/4-6
0x010	ALTCAR—Alternate configuration attribute register	R/W	0x0000_0000	4.3.1.2.2/4-6
0x020	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 0x0_FF70_0000 (CCSRBAR = 0x0_000F_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.

4.3.1.1.1 POR I/O Impedance Control Register (PORIMPCR)

This register needs to get set early in the configuration data process. See Section 17.4.1.3, "POR I/O Impedance Control Register (PORIMPCR)," for details. If booting from local bus and the controls in PORIMPCR are not correct for local bus accesses, then PORIMPCR must be updated via boot sequencer or some other mechanism before allowing the part to start the boot process.

4.3.1.1.2 Updating CCSRBAR

Updates to CCSRBAR that relocate the entire 1-Mbyte region of configuration, control, and status registers require special treatment. The effect of the update must be guaranteed to be visible by the mapping logic before an access to the new location is seen. To make sure this happens, these guidelines should be followed:

- CCSRBAR should be updated during initial configuration of the device when only one host or controller has access to the device.
  - If the boot sequencer is being used to initialize the device, it is recommended that the boot sequencer set CCSRBAR to its desired final location.
  - If an external host on PCI Express or serial RapidIO is configuring the device, it should set CCSRBAR to the desired final location before appropriate e600 core is released to boot.
  - If one of the cores is initializing the device, it should set CCSRBAR to the desired final location before enabling other I/O devices to access the device.
- When either of the cores is writing to CCSRBAR, it should use the following sequence:
  - The application program updating the core should make sure that the other core will not access the CCSRBAR register or configuration space pointed to by that register until the update is complete.
  - 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 but that has an address mapping already in effect (for example, boot ROM). Follow this load with an **isync**.
  - Read the contents of CCSRBAR from its new location, followed by another **isync**.

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

Figure 4-1 shows the fields of CCSRBAR.

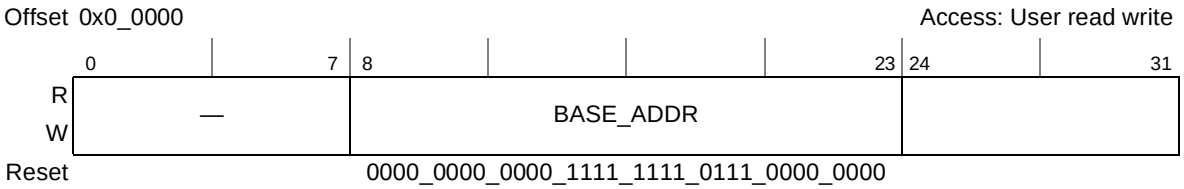


Figure 4-1. Configuration Control and Status Register Base Address Register (CCSRBAR)

Table 4-5 defines the bit fields of CCSRBAR.

Table 4-5. CCSRBAR Bit Settings

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

4.3.1.2 Accessing Alternate Configuration Space

An alternate configuration space can be accessed by configuring the ALTCBAR and ALTCAR registers located in the MPX coherency module (MCM). These are intended to be used with the boot sequencer to allow the boot sequencer to access an alternate 1-Mbyte region of configuration space. By loading the proper boot sequencer command in the serial ROM, the base address in the ALTCBAR can be combined with the 20 bits of address offset supplied from the serial ROM to generate a 36-bit address that is mapped to the target specified in ALTCAR. Thus, by configuring these registers, the boot sequencer has access to the entire memory map, one 1-Mbyte block at a time. See Section 10.4.5, “Boot Sequencer Mode,” for more information.

NOTE

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

4.3.1.2.1 Alternate Configuration Base Address Register (ALTCBAR)

Figure 4-2 shows the fields of ALTCBAR.

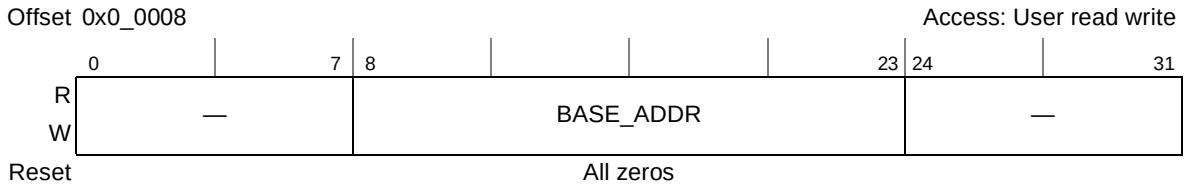


Figure 4-2. Alternate Configuration Base Address Register (ALTCBAR)

Table 4-6 defines the bit fields of ALTCBAR.

Table 4-6. ALTCBAR Bit Settings

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

4.3.1.2.2 Alternate Configuration Attribute Register (ALTCAR)

Figure 4-3 shows the fields of ALTCAR.

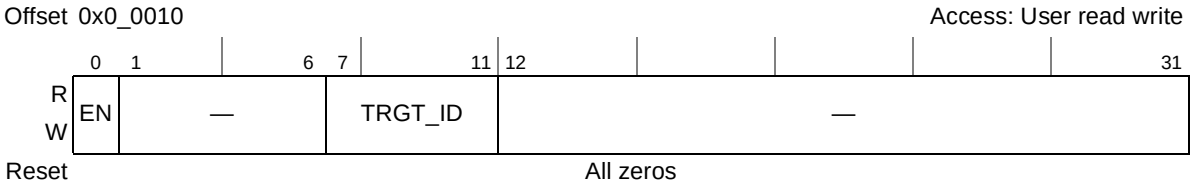


Figure 4-3. Alternate Configuration Attribute Register (ALTCAR)

Table 4-7 defines ALTCAR fields.

Table 4-7. ALTCAR Bit Settings

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

Table 4-7. ALTCAR Bit Settings (continued)

Bits	Name	Description
7–11	TRGT_ID	Identifies the device ID to target when a transaction hits in the 1-Mbyte address range defined by the second configuration window. 00000 PCI Express interface 1 00001 PCI Express interface 2 00010 Reserved00011Reserved 00100 LBC 00101–01010 Reserved 01011 Interleaved DDR memory 01100 Serial RapidIO 01101–01110 Reserved 01111 DDR controller 1 10000–10101 Reserved 10110 DDR controller 2 10111–11111 Reserved
12–31	—	Write reserved, read = 0

4.3.1.3 Boot Page Translation

When an e600 core comes out of reset, it begins execution with the instruction at effective address 0x0_FFF0_0100.To get this instruction, the core's first instruction fetch is a burst read of boot code from effective address 0x0_FFF0_0100. For systems in which the boot code resides at a different address, the device provides boot page translation capability. Boot page translation is controlled by the boot page translation register (BPTR).

By default, the MCM always reserves a special boot window that covers the 8-Mbyte address range from 0x0_FF80_0000 to 0x0_FFFF_FFFF. This allows the cores to issue their first instruction fetch without needing a local access window to be set up in the MCM. An address hit in the special boot window (the default window) is routed to the target defined by the values of the cfg_rom_loc[0:3] configuration signals defined in Section 4.4.3.6, "Boot ROM Location."

As an alternative, the BPTR can be used to identify the upper 16 most-significant address bits. These bits are used to translate an MPX bus initiated access to the 4 Kbytes starting at 0x0_FFF0_0000. The translation is from an address at 0x0_FFF0_0xxx (the 4KB boot page) to an address at 0xy_yyy0_0xxx, where yyyy is BPTR[8:23]. If a boot controller device is used, the boot controller may write to BPTR (with the enable bit set) prior to the cores issuing their first instruction fetches, thereby forcing the cores to get the data from the translated address. This of course may require the boot sequencer to set up a local access window in the MCM that will route the translated boot address to the proper target device. See Section 10.4.5, "Boot Sequencer Mode," for more information.

The boot sequencer can enable boot page translation, or the boot page translation can be set up by an external host when the device is configured to be in boot holdoff mode. If translation is to be performed to a page outside the default boot ROM address range defined (8 Mbytes at 0x0_FF80_0000 to 0x0_FFFF_FFFF as defined in Section 4.4.3.6, "Boot ROM Location"), the external host or boot sequencer must then also set up a local access window to define the routing of the boot code fetch to the target interface that contains the boot code, because the BPTR defines only the address translation, not the target interface. See Section 10.4.5, "Boot Sequencer Mode," for more information.

4.3.1.3.1 Boot Page Translation Register (BPTR)

Figure 4-4 shows the fields of BPTR.

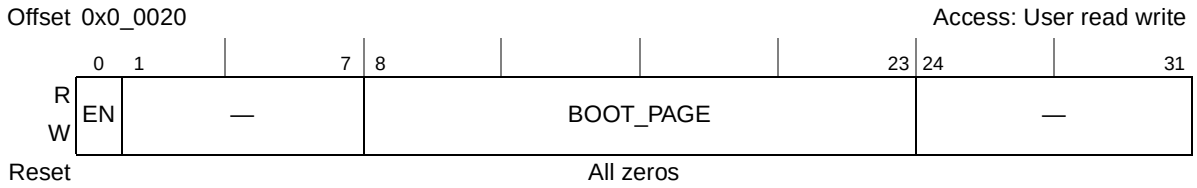


Figure 4-4. Boot Page Translation Register (BPTR)

Table 4-8 describes BPTR bit settings.

Table 4-8. BPTR Bit Settings

Bits	Name	Description
0	EN	Boot page translation enable. If set, the upper 16 address bits of any MPX bus initiated access to local address space from 4 Gb-1Mb to 16 Gbytes (0x0_FFxx_xxxx) is translated to the value in BOOT_PAGE. Note that the lower 8 bits of the vector are not translated and bits 16–31 are zero. 0 Boot page is not translated. 1 Boot page is translated as defined in the BPTR[BOOT_PAGE] parameter.
1–7	—	Write reserved, read = 0
8–23	BOOT_PAGE	Translation for boot page. If enabled, the high order 16 bits of address for accesses to 0x0_FFxx_xxxx are replaced with this value.
23–31	—	Write reserved, read = 0

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 4.3.1.1, “Accessing Configuration, Control, and Status Registers,” and Section 4.3.1.2, “Accessing Alternate Configuration Space.” The boot sequencer is enabled by reset configuration pins as described in Section 4.4.3.11, “Boot Sequencer Configuration.” If the boot sequencer is enabled, the e600 cores are held in reset until the boot sequencer has completed its operation. For more details, see Section 10.4.5, “Boot Sequencer Mode,” in the I^2C chapter. Note that the boot sequencer only works in conjunction with I^2C port 0.

4.4 Functional Description

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

4.4.1 Reset Operations

This section describes the reset input signals for hard and soft reset operation.

4.4.1.1 Soft Reset

Assertion of the SRESET_0 or SRESET_1 signals causes a soft reset interrupt to the corresponding e600 core. This signal may be asserted asynchronously. It is a non-maskable exception and is recoverable. A soft reset interrupt is third in priority behind HRESET and a machine check.

4.4.1.2 Hard Reset

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

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

The hard reset request output signal (HRESET_REQ) indicates to external logic that a hard reset is being requested by one of the following:

- A boot sequencer failure (see Section 10.4.5, “Boot Sequencer Mode,” and Section 10.4.5.2, “EEPROM Data Format”).
- A serial 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.
- Software can request a hard reset as described in Section 17.4.1.15, “Reset Control Register (RSTCR).”

See Section 17.4.1.12, “Reset Request Status and Control Register (RSTRSCR),” for details of these hard reset request sources.

4.4.2 Power-On Reset Sequence

The POR sequence is as follows:

1. Power is applied to meet the specifications in the *MPC8641 and MPC8641D Integrated Host Processor Hardware Specifications*.
2. System asserts HRESET and TRST causing all registers to be initialized to their default states and and releases all bidirectional I/O signals to a high-impedance state.
3. System applies a stable SYSCLK signal and stable PLL configuration inputs (cfg_sys_pll[0:3]), and the platform PLL begins locking to SYSCLK.
4. HRESET negates after a minimum of 100 µs. Note that POR configuration inputs (except those used during HRESET assertion) should be valid at least 4 SYSCLK cycles prior to HRESET negation and held valid at least 2 SYSCLK cycles after HRESET has been negated.

NOTE

The common on-chip processor (COP) requires the ability to independently assert HRESET and TRST to fully control the processor. If a JTAG/COP port is used, follow the JTAG/COP interface connection recommendations given in the *MPC8641 and MPC8641D Integrated Host Processor Hardware Specifications*. If the JTAG interface and COP header are not being used, Freescale recommends that TRST be tied to HRESET so that TRST is asserted when the HRESET is asserted, ensuring that the JTAG scan chain is initialized during the power-on reset flow. See the JTAG configuration signals section in the hardware specifications document for more information.

There is no need to assert the SRESET_0 signal when HRESET is asserted. If SRESET_0 is asserted upon negation of HRESET, the POR sequence will be paused after the e600 core PLL is locked and before the e600 reset is negated. The POR sequence will be resumed when SRESET is negated.

5. The device enables the I/O drivers.
6. The e600 PLL configuration inputs are applied, allowing the e600 PLLs to begin locking to the device clock (MPX_clk).
7. Once the MPX_clk and cfg_core_pll[0:4] are stable, the time required to lock e600 PLLs is approximately 100 µs plus 255 MPX_clk cycles. The core_clk is then generated.
8. The internal hard reset to both the e600 cores are negated and soft resets are negated to the DLLs and other remaining I/O blocks. The DLLs begin to lock.
9. When DLL locking is completed, loading of configuration data can begin and complete from the I^2C if boot sequencing is enabled or directly from the boot ROM location determined by the cfg_rom_loc[0:3] (local bus, and others). See Section 4.4.3.11, “Boot Sequencer Configuration,” for details. Boot sequencing is disabled by default.
10. Booting the e600 cores is allowed to proceed from the default boot vector fetch location unless boot hold off mode is enabled. If boot page translation is enabled in software, the fetch vector is based on the address programmed in the Boot Page Translation register, BPTR[Boot_Page] field, if BPTR[EN] is set; this could be done either in boot hold off mode or during boot sequencing. See Section 4.3.1.3.1, “Boot Page Translation Register (BPTR),” for details. Alternatively, an alternate boot vector can use used if cfg_alt_boot_vec is enabled during POR. See Section 4.4.3.7, “Alternate Boot Vector Location,” for details. The device is now in its ready state.
11. The ASLEEP signal negates synchronized to a rising edge of SYSCLK, indicating the ready state. The ready state is also indicated by the assertion of READY/TRIG_OUT, if TOSR[SEL] = 000. In this case, READY is asserted with the same rising edge of SYSCLK, to indicate that the device has reached its ready state. See Section 19.3.4.1, “Trigger Out Source Register (TOSR),” for more information on this register.

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

For more information about power management states, see Section 17.4.1, “Register Descriptions.”

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

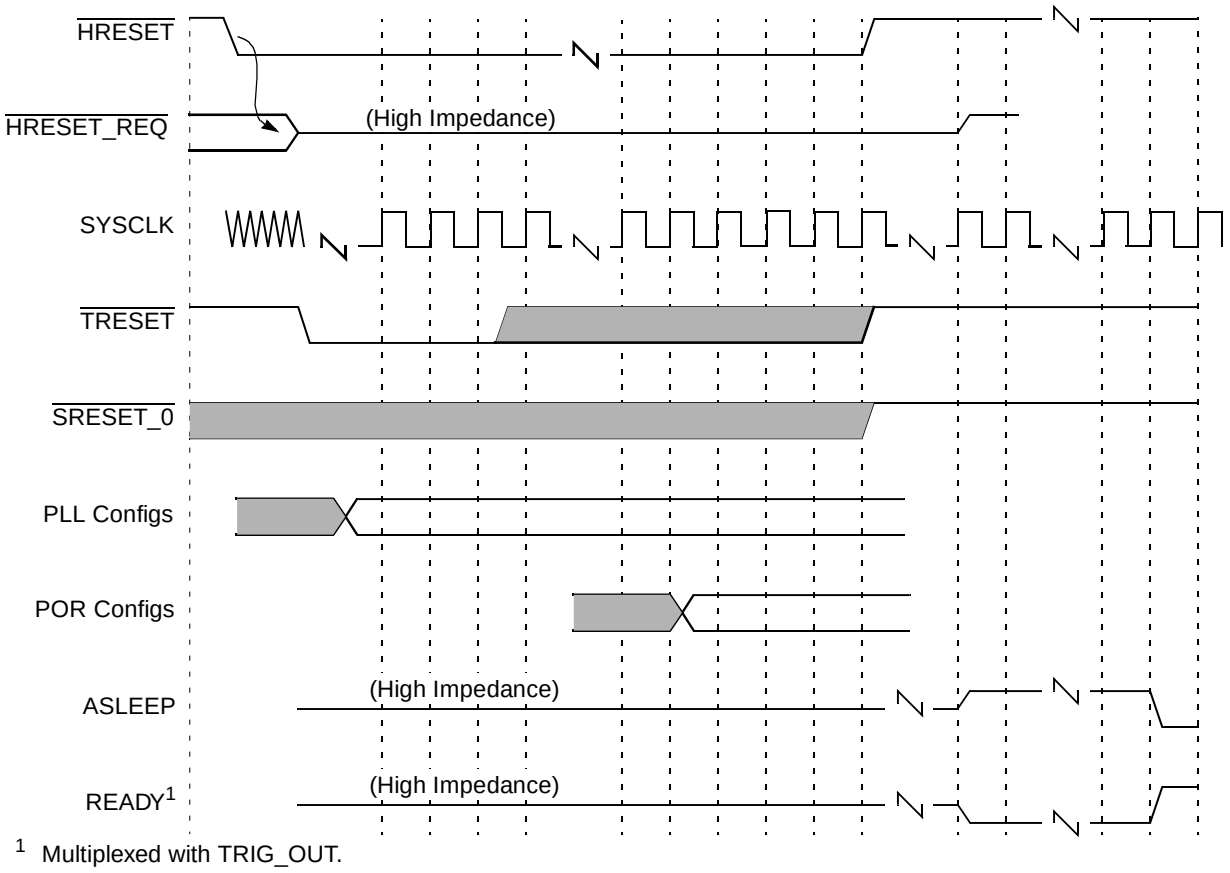


Figure 4-5. Power-On Reset Sequence

4.4.3 Power-On Reset Configuration

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

All POR configuration signals have internal pull-up resistors. Refer to the *Integrated Processor Hardware Specifications* for proper resistor values to be used for pulling POR configuration signals high or low.

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

- POR PLL status register (PORPLLSR)
- POR boot mode status register (PORBMSR)
- POR 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 V_{DD}, regardless of the sense of the functional signal name on the signal.

4.4.3.1 System PLL Ratio

The system PLL inputs, shown in Table 4-9, establish the clock ratio between the SYSCLK input and the MPX clock used by the device. The MPX clock, also called the platform clock, drives the DDR SDRAM data rate and the e600 core MPX bus interface. The default value is a reserved 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 17.4.1.1, “POR PLL Status Register (PORPLLSR).”

Table 4-9. MPX Clock PLL Ratio

Functional Signals	Reset Configuration Name	Value (Binary)	MPX Clock : SYSCLK Ratio
LA[28:31] Default (1111)	cfg_sys_pll[0:3]	0000	Reserved
		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	Reserved
		1011	Reserved
		1100	Reserved
		1101	Reserved
		1110	Reserved
		1111	Reserved

4.4.3.2 Platform Frequency

The platform frequency POR input, shown in Table 4-10, properly configures internal logic appropriate to the platform frequency defined in Section 4.4.3.1, "System PLL Ratio." Note that platform frequencies greater than 400 MHz and less than 500 MHz are not allowed.

Table 4-10. Platform Frequency

Functional Signals	Reset Configuration Name	Value (Binary)	Logic Configuration
TSEC1_TXD[1] Default (1)	cfg_platform_freq	0	Logic configured properly for a platform frequency of 400 MHz
		1	Logic configured properly for platform frequencies of 500 MHz or greater (default)

4.4.3.3 e600 Core PLL Ratio

Table 4-11 describes the e600 core clock PLL inputs that program the core PLL and establish the ratio between the e600 core clock and the e600 core MPX clock. The default value is a reserved 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 17.4.1.1, "POR PLL Status Register (PORPLLSR)," and also in the e600 core HID1 register, as described in Section 6.1.6.2, "Hardware Implementation-Dependent Register 1 (HID1)."

Table 4-11. e600 Core Clock PLL Ratios

Functional Signals	Reset Configuration Name	Value (Binary)	e600 Core: MPX ClockRatio
LDP[0:3], LA[27] Default (11111)	cfg_core_pll[0:4]	0_1000	2:1
		0_1100	2.5:1
		1_0000	3:1
		1_1100	3.5:1
		1_0100	4:1
		0_1110	4.5:1

4.4.3.4 Core 1 Enable

The core 1 enable input, shown in Table 4-12, allows the user to enable or disable the second e600 core during power-on reset. This includes all core1 related logic and functionality. The state of this POR configuration signal is reflected in the SVR, as described in Section 17.4.1.14, "System Version Register (SVR)."

Table 4-12. Core 1 Enable

Functional Signals	Reset Configuration Name	Value (Binary)	Core 1 Status
TSEC3_TXD[2]	cfg_core1_enable	0	Core 1 disabled
Default (1)		1	Core 1 enabled (default)

4.4.3.5 e600 Core 1 Low Memory Offset Mode

As described in Section 2.1.1, "Low Memory Offset Mode," the device provides the capability for accesses to the lowest 256 Mbytes of real addresses from e600 core 1 to be offset in order to provide each core with a unique and private address space in actual memory. This feature is enabled with a power-on reset configuration signal as described in Table 4-13.

Note that the value latched on this signal during POR is reflected in the memory-mapped PORDEVSR (POR device status register) described in Section 17.4.1.4, "POR Device Status Register (PORDEVSR)."

Table 4-13. Core 1 Low Memory Offset Mode

Functional Signal	Reset Configuration Name	Value (Binary)	Meaning
TSEC3_TXD[3]	$\overline{\text{cfg\_core1\_lm\_offset}}$	0	Real address A in range 0 to 256 Mbyte-1 translated to address A + 256 Mbytes.
Default (1)		1	No extra address translation performed for e600 core 1. System address == real address. (default)

4.4.3.6 Boot ROM Location

This device defines the default boot ROM address range to be 8 Mbytes at address 0x0_FF80_0000 to 0x0_FFFF_FFFF. However, which on-chip peripheral handles these boot ROM accesses can be selected at power up.

The boot ROM location reset configuration inputs, shown in Table 4-14, establish the location of boot ROM. Accesses to the boot vector and the default boot ROM region of the local address map are directed to the interface specified by these inputs.

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

Table 4-14. Boot ROM Location

Functional Signals	Reset Configuration Name	Value (Binary)	Meaning
TSEC2_TXD[0:3] Default (1111)	cfg_rom_loc[0:3]	0000	PCI Express 1
		0001	PCI Express 2
		0010	Serial RapidIO
		0100	DDR Memory Controller 1
		0101	DDR Memory Controller 2
		0110–1100	Reserved
		1101	Local bus GPCM—8-bit ROM
		1110	Local bus GPCM—16-bit ROM
		1111	Local Bus GPCM—32-bit ROM (default)

See Section 2.2.6, “Local Address Map 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 the translation options for the boot page.

4.4.3.7 Alternate Boot Vector Location

The alternate boot vector input which exists on external signal TSEC1_TXD[0], shown in Table 4-15, allows the processor to boot from PCI Express 1 outbound translation window 3 by automatically enabling the window and programming the base address to 0x0_FFF0_xxxx and the translated address to 0x0_FFFF_xxxx. This POR configuration option only affects the outbound ATMU window 3 of PCI Express controller 1. As such, this option only affects booting when PCI Express 1 is selected as the boot ROM location by the POR option described in Section 4.4.3.6, “Boot ROM Location.” Note that the value latched on this signal during POR is accessible through PORBMSR[ABV] described in Section 17.4.1.2, “POR Boot Mode Status Register (PORBMSR).”

Table 4-15. Alternate Boot Vector Location

Functional Signal	Reset Configuration Name	Value (Binary)	Meaning
TSEC1_TXD[0] Default (1)	$\overline{\text{cfg\_alt\_boot\_vec}}$	0	PCI Express 1 outbound ATMU window 3 is enabled, base address is set to 0x0_FFF0_xxxx and translation address is set to 0x0_FFFF_xxxx.
		1	Boot vector fetched from default boot ROM location as described in Section 4.4.3.6, “Boot ROM Location.” (default)

4.4.3.8 SerDes Port Selection

The device can be configured with different SerDes ports active. Table 4-16 shows the configuration of I/O ports and bit rates (and required reference clocks) that are possible for the serial RapidIO and PCI Express interfaces.

Table 4-16. I/O Port Selection

Functional Signals	Reset Configuration Name	Value (Binary)	Meaning
TSEC4_TXD[0:3] Default (1111)	cfg_io_ports[0:3]	0000	SerDes1 and SerDes2 disabled; no reference clocks required.
		0001	Reserved
		0010	SerDes1: x1/x2/x4/x8 PCI Express, 100 MHz reference clock to SerDes1 SerDes2: disabled
		0011	SerDes1: x1/x2/x4/x8 PCI Express, 100 MHz reference clock to SerDes1 SerDes2: x1/x2/x4/x8 PCI Express, 100 MHz reference clock to SerDes2 (Same configuration as the default, 1111 below.)
		0100	Reserved
		0101	SerDes1: x1/x2/x4/x8 PCI Express, 100 MHz reference clock to SerDes1 SerDes2: x4 Serial RapidIO, 3.125 Gb interface, 125 MHz reference clock to SerDes2
		0110	SerDes1: x1/x2/x4/x8 PCI Express, 100 MHz reference clock to SerDes1 SerDes2: x4 Serial RapidIO, 2.5 Gb interface, 100 MHz reference clock to SerDes2
		0111	SerDes1: x1/x2/x4/x8 PCI Express, 100 MHz reference clock to SerDes1 SerDes2: x4 Serial RapidIO, 1.25 Gb interface, 100 MHz reference clock to SerDes2
		1000	Reserved
		1001	SerDes1: disabled SerDes2: x4 Serial RapidIO, 3.125 Gb interface, 125 MHz reference clock to SerDes2
		1010	SerDes1: disabled SerDes2: x4 Serial RapidIO, 2.5 Gb interface, 100 MHz reference clock to SerDes2
		1011	SerDes1: disabled SerDes2: x4 Serial RapidIO, 1.25 Gb interface, 100 MHz reference clock to SerDes2
		1100	Reserved
		1101	Reserved
		1110	SerDes1: disabled SerDes2: x1/x2/x4/x8 PCI Express, 100 MHz reference clock to SerDes2
		1111	SerDes1: x1/x2/x4/x8 PCI Express, 100 MHz reference clock to SerDes1 SerDes2: x1/x2/x4/x8 PCI Express, 100 MHz reference clock to SerDes2 (Same configuration as 0011 above.) (default)

4.4.3.9 SerDes Host/Agent Configuration

The SerDes host/agent reset configuration inputs, shown in Table 4-17, configure the device to act as a host or as an agent of a master on another interface. The meaning of the inputs depends on the SerDes port chosen as described in Section 4.4.3.8, "SerDes Port Selection." In host mode, the device is immediately enabled to master transactions to the serial RapidIO and PCI Express interfaces. If the device is an agent on the serial RapidIO or the PCI Express interfaces, then the device 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 device's interfaces appropriately. See details in the PCI Express and serial RapidIO programming models described in Chapter 16, "PCI Express Interface Controller," and Chapter 15, "Serial RapidIO Interface," respectively.

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

Table 4-17. Host/Agent Configuration

Functional Signals	Reset Configuration Name	Value (Binary)	Meaning
$\overline{\text{LWE}}$[2:3] Default (11)	cfg_host_agt[0:1]	00	If the SerDes*n* port is PCI Express, then it is an endpoint. If the SerDes2 port is SRIO, then it is an agent.
		01	If the SerDes1 is PCI Express, then it is a root complex. If the SerDes2 is PCI Express, then it is an endpoint. If the SerDes2 port is SRIO, then it is a host.
		10	If the SerDes1 is PCI Express, then it is an endpoint. If the SerDes2 is PCI Express, then it is a root complex. If the SerDes2 is SRIO, then it is an agent.
		11	If the SerDes*n* is PCI Express, then it is a root complex. If the SerDes2 is SRIO, then it is a host. (default)

NOTE

If the device is an agent on a particular interface, and the CPU is not in boot holdoff mode (as described in Section 4.4.3.10, "CPU Boot Configuration") then the boot ROM should not be located on the interface where the external host exists, because the device is not enabled to master reads onto that interface.

4.4.3.10 CPU Boot Configuration

The CPU boot configuration input, shown in Table 4-18, specifies the boot configuration mode. The value sampled at reset determines if core 0 is prevented from fetching boot code after reset. If core 0 is prevented from fetching boot code, it is expected that the device is being configured by an external master. In that case, the final step that the master must take is to free the cores to proceed with boot access by setting the PORT0_EN bit in the MCM port configuration register (PCR) register. On the MPC8641D, core 1 is always prevented from booting after reset. It is expected that core 0 or an external master will release core 1 to be able to boot. See Section 7.4.1.3, "Port Configuration Register (PCR)" for more information.

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

Note also that the value latched on this signal during POR affects the PCI Express Configuration Ready Register (See Section 16.5.1, “Boot Mode and Inbound Configuration Transactions,” and Section 16.3.10.18, “Configuration Ready Register—0x4B0.”).

Table 4-18. CPU Boot Configuration

Functional Signal	Reset Configuration Name	Value (Binary)	Meaning
$\overline{\text{LWE}}$[0] Default (1)	cfg_cpu_boot	0	CPU boot holdoff mode. The e600 core 0 is prevented from booting until configured by an external master.
		1	The core 0 is allowed to boot without waiting for configuration by an external master. (default)

4.4.3.11 Boot Sequencer Configuration

The boot sequencer configuration options, shown in Table 4-19, allow the boot sequencer to load configuration data from the serial ROM located on I^2C port 1 before the host tries to configure the device. These options also specify normal or extended I^2C addressing modes for both I^2C interfaces. See Section 10.4.5, “Boot Sequencer Mode,” for more information on the boot sequencer.

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

Table 4-19. 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 1. 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 1. 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, core 0 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 4.4.3.10, “CPU Boot Configuration.”

4.4.3.12 DDR SDRAM Type

Low-power DDR SDRAM (or mobile DDR DRAM) requires different initialization from other DDR SDRAM types. DDR1 memories require different voltage levels from DDR2. Table 4-20 describes the configuration of the DDR SDRAM type. Note that the value latched on these signals during POR is accessible through the memory-mapped PORDEVSR (POR device status register) described in Section 17.4.1.4, "POR Device Status Register (PORDEVSR)."

Table 4-20. DDR SDRAM Type

Functional Signal	Reset Configuration Name	Value (Binary)	Meaning
TSEC2_TXD[4] TSEC2_TX_ER Default (11)	cfg_dram_type[0:1]	00	Reserved
		01	DDR1 2.5V, CKE low at reset
		10	Reserved
		11	DDR2(default) 1.8V, CKE low at reset

4.4.3.13 eTSEC*n* Width

The eTSEC width inputs, shown in Table 4-21, select standard versus reduced widths for each of the three-speed Ethernet controller interfaces. Note that the values latched on these signals during POR are accessible through the memory-mapped PORDEVSR (POR device status register) described in Section 17.4.1.4, "POR Device Status Register (PORDEVSR)."

NOTE

A full-width (16-bit) FIFO interface requires signals from two controllers; therefore, eTSEC2 is unavailable if eTSEC1 is in 16-bit FIFO mode and eTSEC4 is unavailable if eTSEC3 is in 16-bit FIFO mode. If eTSEC1 is in 16-bit FIFO mode, cfg_tsec2_reduce must be pulled low; if eTSEC3 is in 16-bit FIFO mode, cfg_tsec4_reduce must be pulled low.

Also note that cfg_tsec2_reduce must be pulled low if the 8-bit FIFO interface is selected for eTSEC2; cfg_tsec4_reduce must be pulled low if the 8-bit FIFO interface is selected for eTSEC4.

Table 4-21. eTSEC Width Configuration

Functional Signal	Reset Configuration Name	Value (Binary)	Meaning
TSEC*n*_TXD5 Default (1)	cfg_tsec*n*_reduce	0	Ethernet interface operates in reduced mode, either RTBI or RGMII, using only four transmit data signals and four receive data signals. **Note:** If eTSEC1 is in 16-bit FIFO mode, cfg_tsec2_reduce must be pulled low; if eTSEC3 is in 16-bit FIFO mode, cfg_tsec4_reduce must be pulled low.
		1	Ethernet interface operates in standard TBI or GMII modes, including 16-bit FIFO mode, using eight transmit data signals (ten for TBI) and eight receive data signals (ten for TBI) (default).

4.4.3.14 eTSEC*n* Protocol

The eTSEC protocol reset configuration inputs, shown in Table 4-22, select the protocol (FIFO, MII, GMII, or TBI) used by the eTSEC controllers (where N = 1 to 4). If eTSEC1 is in 16 bit FIFO mode, eTSEC2 should not be configured to be in FIFO mode.The same applies to the eTSEC3 and eTSEC4 pair. Note that the value latched on this signal during POR is accessible through the memory-mapped PORDEVSR (POR device status register) described in Section 17.4.1.4, “POR Device Status Register (PORDEVSR).”

Table 4-22. eTSEC*n* Protocol Configuration

Functional Signal	Reset Configuration Name	Value (Binary)	Meaning
TSEC*n*_TXD[6:7], Default (11)	cfg_tsec*n*_prtcl[0:1]	00	The eTSEC*n* controller operates using the FIFO protocol.
		01	The eTSEC*n* controller operates using the MII protocol (or RMII if configured in reduced mode).
		10	The eTSEC*n* controller operates using the GMII protocol (or RGMII if configured in reduced mode).
		11	The eTSEC*n* controller operates using the TBI protocol (or RTBI if configured in reduced mode) (default).

4.4.3.15 Serial RapidIO Device ID

The serial RapidIO device ID reset configuration inputs, shown in Table 4-23, are used to determine the device ID of the RapidIO device as reflected in the BDIDCSR (See Section 15.6.1.12, “Base Device ID Command and Status Register (BDIDCSR)”).

Note that the high-order bits of the serial RapidIO device ID cannot be set via POR configuration inputs. They can be initialized via the boot sequencer, by the processor from boot ROM, or by the serial RapidIO discovery process.

If configured as a serial RapidIO host, then the device ID is 0b0000_0||cfg_device_id[5:7].

If configured as a serial RapidIO agent:

- if cfg_dev_ID[7] input = 0, then the device ID is 0xFE and
- if cfg_dev_ID[7] input = 1, then the device is a generic, undiscovered endpoint with a device ID of 0xFF.

Unconnected cfg_dev_ID*n* inputs default to 1s regardless of the host/agent mode configuration.

Note that the value latched on these signals at POR is accessible through the memory-mapped PORDEVSR, described in Section 17.4.1.4, "POR Device Status Register (PORDEVSR)."

Table 4-23. RapidIO Device ID

Functional Signals	Reset Configuration Name	Meaning
TSEC1_TXD[2:4] Default (111)]	cfg_dev_id[5:7]	Device ID used for serial RapidIO hosts

4.4.3.16 Serial RapidIO System Size

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

The serial RapidIO system size configuration shown in Table 4-24 specifies which system size is to be used by the RapidIO controller.

Note that the system size (as determined by the value latched on this signal at POR) is accessible through the memory-mapped PEFCAR[CTLS], described in 15.6.1.5, "Processing Element Features Capability Register (PEFCAR)."

Table 4-24. Serial RapidIO System Size

Functional Signals	Reset Configuration Name	Value (Binary)	Meaning
$\overline{\text{LWE1}}$ Default (1)	cfg_rio_sys_size	0	Large system size (up to 65,536 devices)
		1	Small system size (up to 256 devices) (default)

4.4.3.17 Memory Debug Configuration

The memory debug configuration input, shown in Table 4-25, selects which debug outputs (DDR controller 1 or LBC memory controller) are driven onto the D1_MSRCID*n* and D1_MDVAL debug signals. Note that the value latched on this signal during POR is accessible through the memory-mapped PORDBGMSR (POR debug mode register) described in Section 17.4.1.5, "POR Debug Mode Status Register (PORDBGMSR)."

Table 4-25. Memory Debug Configuration

Functional Signal	Reset Configuration Name	Value (Binary)	Meaning
D1_MSRCID[0] Default (1)	cfg_mem_debug	0	Debug information from the local bus controller (LBC) is driven on the D1_MSRCID*n* and D1_MDVAL signals
		1	Debug information from the DDR SDRAM controller is driven on the D1_MSRCID and D1_MDVAL signals (default).

4.4.3.18 DDR Debug Configuration

The DDR debug reset 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 this device in this mode. Note that the value latched on this signal during POR is accessible through the memory-mapped PORDBGMSR (POR debug mode register), described in Section 17.4.1.4, "POR Device Status Register (PORDEVSR)."

Note that when the DDR memory controller debug mode is set both DDR controllers' debug information is driven on their ECC pins.

Table 4-26. DDR Debug Configuration

Functional Signal	Reset Configuration Name	Value (Binary)	Meaning
D1_MSRCID[1] Default (1)	cfg_ddr_debug	0	DDR debug information is driven on the ECC pins instead of normal ECC I/O. ECC signals from memory devices must be disconnected.
		1	DDR debug information is not driven on ECC pins. ECC pins function in their normal mode (default).

4.4.3.19 General-Purpose POR Configuration

The LBC address/data bus inputs, shown in Table 4-27, configure the value of the general-purpose POR configuration register defined in Section 17.4.1.4, "POR Device Status Register (PORDEVSR)." 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 4-27. General-Purpose POR Configuration

Functional Signals	Reset Configuration Name	Value (Binary)	Meaning
LAD[0:31] Default (all 1's)	cfg_gpporcr[0:31]	xx	32 bit general-purpose POR configuration vector to be placed in GPPORCR

4.4.4 Clocking

The following paragraphs describe the clocking within the device.

4.4.4.1 System Clock

The device takes a single input clock, SYSCLK, as its primary clock source for the e600 cores and all of the devices and interfaces that operate synchronously with the cores. As shown in Figure 4-6, the SYSCLK input (frequency) is multiplied up using a phase-locked loop (PLL) to create the MPX bus clock (also called the platform clock). The MPX bus clock is used by virtually all of the synchronous system logic and other internal blocks such as the DMA and interrupt controller. The MPX bus clock also feeds the PLL in the e600 cores, the DLL that creates clocks for the DDR SDRAM and the PLL for the local bus memory controller. Note that the divide-by-two MPX clock divider and the divide-by-*n* MPX clock divider, shown in Figure 4-6, are located in the DDR and local bus blocks, respectively.

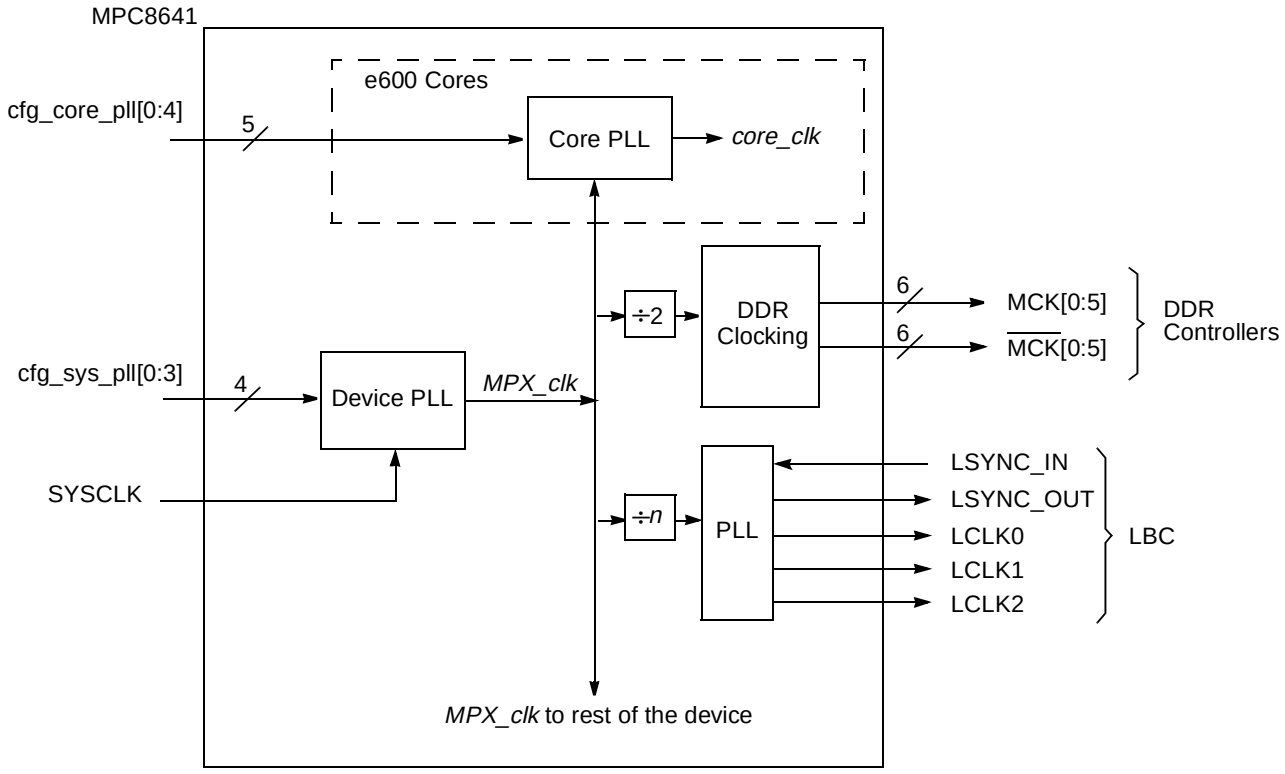


Figure 4-6. Clock Subsystem Block Diagram

4.4.4.2 RapidIO and PCI Express Clocks

Clocks for these high speed interfaces on the MPC8641 are derived from a PLL in the SerDes block. This PLL is driven by a reference clock(SD_REF_CLK/$\overline{\text{SD\_REF\_CLK}}$) whose input frequency is a function of the protocol and bit rate being used as shown in Table 4-28.

Table 4-28. High Speed Interface Clocking

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

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 MPX clock internally.

Part II
e600 Core

This part describes the many features of the MPC8641 core processor at an overview level. The following chapters are included:

- Chapter 5, “e600 Core Overview,” provides an overview of the e600 core processor and the L1 caches and MMU that, together with the core, comprise the core complex.
- Chapter 6, “e600 Core Registers and Instruction Set Summary,” provides a listing of the e600 registers in reference form.

Chapter 5
e600 Core Overview

This chapter provides an overview of the e600 core features, including a block diagram showing the major functional components.

This document also provides information about how the e600 implementation complies with the PowerPC™ instruction set architecture (ISA) definitions including AltiVec™ extensions.

5.1 e600 Core Overview

This section describes the features and general operation of the e600 core and provides a block diagram showing the major functional units. The e600 implements the PowerPC ISA and is a reduced instruction set computer (RISC) core. The e600 core includes separate 32-Kbyte L1 instruction and data caches and a 1-Mbyte L2 cache. The core is a high-performance superscalar design supporting multiple execution units, including four independent units that execute AltiVec instructions.

The e600 core implements the 32-bit portion of the PowerPC ISA, which provides 32-bit effective addresses, integer data types of 8, 16, and 32 bits, and floating-point data types of 32 (single precision) and 64 (double precision) bits. The core supports up to 4 Petabytes (2^{52}) of virtual memory and up to 64 Gigabytes (2^{36}) of physical memory.

The e600 core also implements the AltiVec instruction set architectural extension. The e600 core can dispatch and complete three instructions simultaneously. It incorporates the following execution units:

- 64-bit floating-point unit (FPU)
- Branch processing unit (BPU)
- Load/store unit (LSU)
- Four integer units (IUs):
  - Three shorter latency IUs (IU1a–IU1c)—execute all integer instructions except multiply, divide, and move to/from special-purpose register (SPR) instructions.
  - Longer latency IU (IU2)—executes miscellaneous instructions including condition register (CR) logical operations, integer multiplication and division instructions, and move to/from SPR instructions.
- Four vector units that support AltiVec instructions:
  - Vector permute unit (VPU)
  - Vector integer unit 1 (VIU1)—performs shorter latency integer calculations
  - Vector integer unit 2 (VIU2)—performs longer latency integer calculations
  - Vector floating-point unit (VFPU)

The ability to execute several instructions in parallel and the use of simple instructions with rapid execution times yield high efficiency and throughput for e600-based systems. Most integer instructions (including VIU1 instructions) have a one-clock cycle execution latency.

Several execution units feature multiple-stage pipelines; that is, the tasks they perform are broken into subtasks executed in successive stages. Typically, instructions follow one another through the stages, so a four-stage unit can work on four instructions when its pipeline is full. So, although an instruction may have to pass through several stages, the execution unit can achieve a throughput of one instruction per clock cycle.

AltiVec computational instructions are executed in four independent, pipelined AltiVec execution units. A maximum of two AltiVec instructions can be issued out-of-order to any combination of AltiVec execution units per clock cycle from the bottom two VIQ entries (VIQ1–VIQ0). This means an instruction in VIQ1 does not have to wait for an instruction in VIQ0 that is waiting for operand availability. Moreover, the VIU2, VFPU, and VPU are pipelined, so they can operate on multiple instructions. Because VPU has a two-stage pipeline; the VIU2 and VFPU each have four-stage pipelines, as many as ten AltiVec instructions can be executing concurrently.

Note that for the e600 core, double- and single-precision versions of floating-point instructions have the same latency. For example, a floating-point multiply-add instruction takes 5 cycles to execute, regardless of whether it is single (**fmadds**) or double precision (**fmadd**).

The e600 core has independent 32-Kbyte, eight-way set-associative, physically addressed L1 (level one) caches for instructions and data, and independent instruction and data memory management units (MMUs). Each MMU has a 128-entry, two-way set-associative translation lookaside buffer (DTLB and ITLB) that saves recently used page address translations. Block address translation is implemented with the eight-entry instruction and data block address translation (IBAT and DBAT) arrays defined by the PowerPC ISA. During block translation, effective addresses are compared simultaneously with all BAT entries, as described in the “Memory Management” chapter of the *e600 PowerPC Core Reference Manual*. For information about the L1 caches, see the “L1 and L2 Cache Operation” chapter of the *e600 PowerPC Core Reference Manual*.

The L2 cache is implemented with 1-Mbyte, eight-way set-associative physically addressed memory within the core available for storing data, instructions, or both. The L2 cache supports parity generation and checking for both tags and data. If error checking and correction (ECC) is disabled, it responds with an 11.5-cycle load latency for an L1 miss that hits in the L2; if ECC is enabled, the L2 load access time is 12.5 cycles. The L2 cache is fully pipelined for two-cycle throughput. For information about the L2 cache implementation, see the “L1 and L2 Cache Operation” chapter of the *e600 PowerPC Core Reference Manual*.

The two power-saving modes, nap and sleep, progressively reduce power dissipation. When functional units are idle, a dynamic power management mode causes those units to enter a low-power mode automatically without affecting operational performance, software execution, or hardware external to the core. Section 5.2.7, “Power and Thermal Management,” describes how power management can be used to reduce power consumption when the core, or portions of it, are idle. It also describes how the instruction cache throttling mechanism reduces the instruction dispatch rate. Power management states are described more fully in the “Power and Thermal Management” chapter of the *e600 PowerPC Core Reference Manual*.

The performance monitor facility provides the ability to monitor and count predefined events such as core clocks, misses in the instruction cache, data cache, or L2 cache, types of instructions dispatched, mispredicted branches, and other occurrences. The count of such events (which may be an approximation) can be used to trigger the performance monitor interrupt. Section 5.2.8, “Core Performance Monitor,” describes the operation of the performance monitor diagnostic tool. This functionality is fully described in the “Performance Monitor” chapter of the *e600 PowerPC Core Reference Manual*.

Figure 5-1 shows the parallel organization of the execution units (shaded in the diagram). Note that this is a conceptual model showing basic features, rather than an attempt at showing how features are implemented physically.

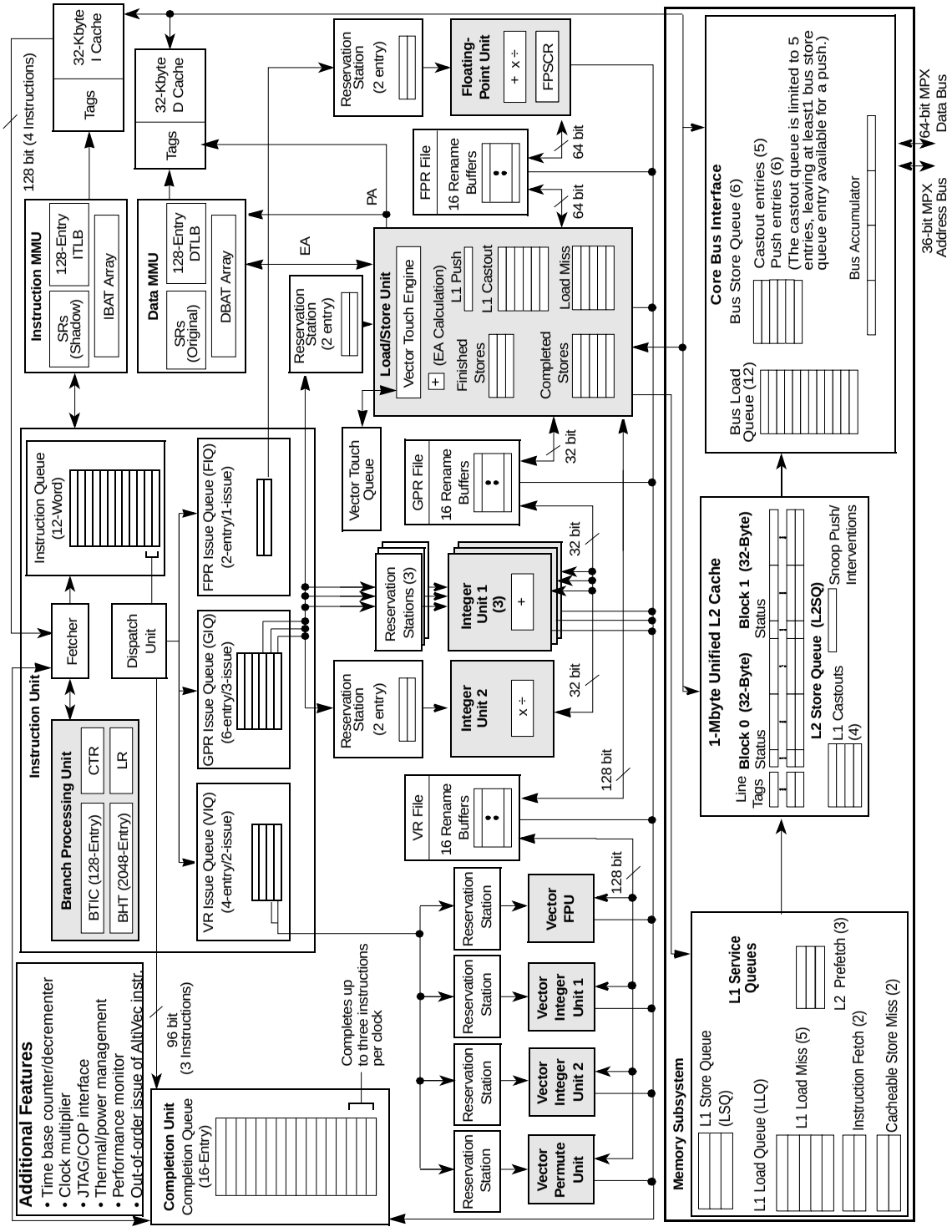


Figure 5-1. e600 Core Block Diagram

5.2 e600 Core Features

This section describes the features of the e600 core. The interrelationships of these features are shown in Figure 5-1.

Major features of the e600 core are as follows:

- High-performance superscalar e600 core
  - As many as 4 instructions can be fetched from the instruction cache at a time.
  - As many as 3 instructions can be dispatched to the issue queues at a time.
  - As many as 12 instructions can be in the instruction queue (IQ).
  - As many as 16 instructions can be at some stage of execution simultaneously.
  - Single-cycle execution for most instructions
  - One-instruction throughput per clock cycle for most instructions
  - Seven-stage pipeline control
- Eleven independent execution units and three register files
  - Branch processing unit (BPU) features static and dynamic branch prediction
    - 128-entry (32-set, four-way set-associative) branch target instruction cache (BTIC), a cache of branch instructions that have been encountered in branch/loop code sequences. If a target instruction is in the BTIC, it is fetched into the instruction queue a cycle sooner than it can be made available from the instruction cache. Typically, a fetch that hits the BTIC provides the first 4 instructions in the target stream.
    - 2048-entry branch history table (BHT) with 2 bits per entry for four levels of prediction—strongly not-taken, not-taken, taken, strongly taken
    - Up to three outstanding speculative branches
    - Branch instructions that do not update the count register (CTR) or link register (LR) are often removed from the instruction stream.
    - Eight-entry link register stack to predict the target address of Branch Conditional to Link Register (**bclr**) instructions
  - Four integer units (IUs) that share 32 GPRs for integer operands
    - Three identical IUs (IU1a, IU1b, and IU1c) can execute all integer instructions except multiply, divide, and move to/from SPR instructions.
    - IU2 executes miscellaneous instructions including the CR logical operations, integer multiplication and division instructions, and move to/from special-purpose register instructions.
  - 64-bit floating-point unit (FPU)
    - Five-stage FPU
    - Designed to comply with IEEE Std. 754™-1985 FPU for both single- and double-precision operations
    - Supports non-IEEE Std. 754 mode for time-critical operations
    - Hardware support for denormalized numbers
    - Thirty-two 64-bit FPRs for single- or double-precision operands

  - Four vector units and 32-entry vector register file (VRs)
    - Vector permute unit (VPU)
    - Vector integer unit 1 (VIU1) handles short-latency AltiVec integer instructions, such as vector add instructions (for example, **vaddsbs**, **vaddshs**, and **vaddsws**)
    - Vector integer unit 2 (VIU2) handles longer-latency AltiVec integer instructions, such as vector multiply add instructions (for example, **vmhaddshs**, **vmhraddshs**, and **vmladduhm**).
    - Vector floating-point unit (VFPU)
  - Three-stage load/store unit (LSU)
    - Supports integer, floating-point and vector instruction load/store traffic
    - Four-entry vector touch queue (VTQ) supports all four architected AltiVec data stream operations
    - Three-cycle GPR and AltiVec load latency (byte, half word, word, vector) with single-cycle throughput
    - Four-cycle FPR load latency (single, double) with single-cycle throughput
    - No additional delay for misaligned access within double-word boundary
    - Dedicated adder calculates effective addresses (EAs)
    - Supports store gathering
    - Performs alignment, normalization, and precision conversion for floating-point data
    - Executes cache control and TLB instructions
    - Performs alignment, zero padding, and sign extension for integer data
    - Supports hits under misses (multiple outstanding misses)
    - Supports both big- and little-endian modes, including misaligned little-endian accesses
- Dispatch unit—The decode/dispatch stage fully decodes each instruction.
- The FIQ (floating-point issue queue), VIQ (vector issue queue), and GIQ (general purpose issue queue) can accept as many as one, two, and three instructions, respectively, in a cycle. Instruction dispatch requires the following:
  - Instructions can be dispatched only from the three lowest IQ entries—IQ0, IQ1, and IQ2.
  - A maximum of three instructions can be dispatched to the issue queues per clock cycle.
  - Space must be available in the completion queue (CQ) for an instruction to dispatch (this includes instructions that are assigned a space in the CQ but not in an issue queue).
- Rename buffers
  - 16 GPR (general purpose register) rename buffers
  - 16 FPR (floating-point register) rename buffers
  - 16 VR (vector register) rename buffers
- Completion unit
  - Retires an instruction from the 16-entry CQ when all instructions ahead of it have been completed, the instruction has finished execution, and no interrupts are pending
  - Guarantees sequential programming model (precise interrupt model)

  - Monitors all dispatched instructions and retires them in order
  - Tracks unresolved branches and flushes instructions after a mispredicted branch
  - Retires as many as three instructions per clock cycle
- L1 cache has the following characteristics:
  - Two separate 32-Kbyte instruction and data caches (Harvard architecture)
  - Instruction and data caches are eight-way set-associative
  - Instruction and data caches have 32-byte cache blocks. A cache block is the block of memory that a coherency state describes—it corresponds to a cache line for the L1 data cache.
  - Cache directories are physically addressed. The physical (real) address tag is stored in the cache directory.
  - The caches implement a pseudo least-recently-used (PLRU) replacement algorithm within each way.
  - Cache write-back or write-through operation is programmable on a per-page or per-block basis.
  - Instruction cache can provide four instructions per clock cycle; data cache can provide four words per clock cycle
    - Two-cycle latency and single-cycle throughput for instruction or data cache accesses
  - Caches can be disabled in software
  - Caches can be locked in software
  - Supports a four-state modified/exclusive/shared/invalid (MESI) coherency protocol
    - A single coherency status bit for each instruction cache block allows encoding for the following two possible states:

      Invalid (INV)

      Valid (VAL)
    - Three status bits for each data cache block allow encoding for coherency, as follows:

      0xx = invalid (I)

      101 = shared (S)

      100 = exclusive (E)

      110 = modified (M)
  - Separate copy of data cache tags for efficient snooping
  - Both L1 caches support parity generation and checking (enabled through bits in the instruction cache and interrupt control (ICTRL) register) as follows:
    - Instruction cache—one parity bit per instruction
    - Data cache—one parity bit per byte of data
  - No snooping of instruction cache except for **icbi** instruction
  - Caches implement a pseudo least-recently-used (PLRU) replacement algorithm within each way
  - Data cache supports AltiVec LRU and transient instructions, as described in Section 5.3.2.2, “AltiVec Instruction Set”

  - Critical double- and/or quad-word forwarding is performed as needed. Critical quad-word forwarding is used for AltiVec loads and instruction fetches. Other accesses use critical double-word forwarding.
- Level 2 (L2) cache within the core has the following features:
  - Integrated 1-Mbyte, eight-way set-associative unified instruction and data cache
  - Pipelined to provide 32 bytes every other clock cycle to the L1 caches
  - Total latency of 11.5 processor cycles for L1 data cache miss that hits in the L2 with ECC disabled, 12.5 cycles when ECC is enabled
  - Uses one of two random replacement algorithms (selectable through L2CR)
  - Cache write-back or write-through operation programmable on a per-page or per-block basis
  - Organized as 32 bytes/block and 2 blocks (sectors)/line (a cache block is the block of memory that a coherency state describes).
  - Supports error correction and detection using a SECDED (single-error correction, double-error detection) protocol. Every 64 bits of data comes with 8 bits of error detection/correction, which can be programmed as ECC across the 64 bits of data, byte parity, or no error detection/correction.
  - Supports parity generation and checking for both tags and data (enabled through L2CR). Tag parity is enabled separately in the L2ERRDIS register, and data parity can be enabled through L2CR only when ECC is disabled.
  - Error injection modes provided for testing
- Separate memory management units (MMUs) for instructions and data
  - 52-bit virtual address; 32- or 36-bit physical address
  - Address translation for 4-Kbyte pages, variable-sized blocks, and 256-Mbyte segments
  - Memory programmable as write-back/write-through, caching-inhibited/caching-allowed, and memory coherency enforced/memory coherency not enforced on a page or block basis
  - Separate IBATs and DBATs (eight each) also defined as SPRs
  - Separate instruction and data translation lookaside buffers (TLBs)
    - Both TLBs are 128-entry, two-way set-associative, and use LRU replacement algorithm
    - TLBs are hardware or software reloadable (that is, on a TLB miss a page table search is performed in hardware or by system software).
- Efficient data flow
  - Although the VR/LSU interface is 128 bits, the L1/L2 bus interface allows up to 256 bits.
  - The L1 data cache is pipelined to provide 128 bits/cycle to or from the VRs.
  - L2 cache is fully pipelined to provide 32 bytes every other processor clock cycle to the L1 cache
  - As many as nine outstanding, out-of-order cache misses are allowed between the L1 data cache and L2 bus: up to 5 load or touch instructions, 2 cacheable store instructions, and/or 2 instruction fetches.
  - As many as 8 out-of-order transactions can be present on the MPX bus.

  - Store merging for multiple store misses to the same line. Only coherency action taken (address-only) for store misses merged to all 32 bytes of a cache block (no data tenure needed)
  - Support for a second cacheable store miss
  - Three-entry finished store queue and five-entry completed store queue between the LSU and the L1 data cache
  - Separate additional queues for efficient buffering of outbound data (such as castouts and write-through stores) from the L1 data cache and L2 cache
- Multiprocessing support features include the following:
  - Hardware-enforced, MESI cache coherency protocols for data cache
  - Load/store with reservation instruction pair for atomic memory references, semaphores, and other multiprocessor operations
- Power and thermal management
  - The following two power-saving modes are available to the system:
    - Nap—Instruction fetching is halted. Only those clocks for the time base, decrementer, and JTAG logic remain running. The core goes into the doze state to snoop memory operations on the MPX bus and then back to nap using a *qreq*/*qack* processor-system handshake protocol.
    - Sleep—Power consumption is further reduced by disabling bus snooping, leaving only the PLL in a locked and running state. All internal functional units are disabled.
  - Instruction cache throttling provides control of instruction fetching to lower device temperature.
- Performance monitor helps to debug system designs and improve software efficiency
- In-system testability and debugging features through JTAG boundary-scan capability
- Reliability and serviceability
  - Parity checking on L1 and L2 cache arrays

5.2.1 Instruction Flow

As shown in Figure 5-1, the e600 core instruction unit provides centralized control of instruction flow to the execution units. The instruction unit contains a sequential fetcher, 12-entry instruction queue (IQ), dispatch unit, and branch processing unit (BPU). It determines the address of the next instruction to be fetched based on information from the sequential fetcher and from the BPU.

See the “Instruction Timing” chapter of the *e600 PowerPC Core Reference Manual* for a detailed discussion of instruction timing.

The sequential fetcher loads instructions from the instruction cache into the instruction queue. The BPU extracts branch instructions from the sequential fetcher. Branch instructions that cannot be resolved immediately are predicted using either e600-specific dynamic branch prediction or architecture-defined static branch prediction.

Branch instructions that do not affect the LR or CTR are often removed from the instruction stream. The “Instruction Timing” chapter of the *e600 PowerPC Core Reference Manual* describes when a branch can be removed from the instruction stream.

Instructions dispatched beyond a predicted branch do not complete execution until the branch is resolved, preserving the programming model of sequential execution. If branch prediction is incorrect, the instruction unit flushes all predicted path instructions, and instructions are fetched from the correct path.

5.2.1.1 Instruction Queue and Dispatch Unit

The instruction queue (IQ), shown in Figure 5-1 holds as many as 12 instructions and loads as many as 4 instructions from the instruction cache during a single processor clock cycle.

The fetcher attempts to initiate a new fetch every cycle. The two fetch stages are pipelined, so as many as four instructions can arrive to the IQ every cycle. All instructions except branch (**b***x*), Return from Interrupt (**rfi**), System Call (**sc**), Instruction Synchronize (**isync**), and no-op instructions are dispatched to their respective issue queues from the dispatch entries in the instruction queue (IQ0–IQ2) at a maximum rate of three instructions per clock cycle. Reservation stations are provided for the three IU1s, IU2, FPU, LSU, VPU, VIU2, VIU1, and VFPU. The dispatch unit checks for source and destination register dependencies, determines whether a position is available in the CQ, and inhibits subsequent instruction dispatching as required.

Branch instruction can be detected, decoded, and predicted from entries IQ0–IQ7. See the “Instruction Timing” chapter of the *e600 PowerPC Core Reference Manual.*

5.2.1.2 Branch Processing Unit (BPU)

The BPU receives branch instructions from the IQ and executes them early in the pipeline, achieving the effect of a zero-cycle branch in some cases.

Branches with no outstanding dependencies (CR, LR, or CTR unresolved) can be processed and resolved immediately. For branches in which only the direction is unresolved due to a CR or CTR dependency, the branch path is predicted using either architecture-defined static branch prediction or e600-specific dynamic branch prediction. Dynamic branch prediction is enabled if HID0[BHT] is set. For **bclr** branches

where the target address is unresolved due to an LR dependency, the branch target can be predicted using the hardware link stack. Link stack prediction is enabled if HID0[LRSTK] is set.

When a prediction is made, instruction fetching, dispatching, and execution continue from the predicted path, but instructions cannot complete and write back results to architected registers until the prediction is determined to be correct (resolved). When a prediction is incorrect, the instructions from the incorrect path are flushed from the core and processing begins from the correct path.

Dynamic prediction is implemented using a 2048-entry branch history table (BHT), a cache that provides two bits per entry that together indicate four levels of prediction for a branch instruction—not-taken, strongly not-taken, taken, strongly taken. When dynamic branch prediction is disabled, the BPU uses a bit in the instruction encoding to predict the direction of the conditional branch. Therefore, when an unresolved conditional branch instruction is encountered, the e600 core executes instructions from the predicted target stream although the results are not committed to architected registers until the conditional branch is resolved. Unresolved branches are held in a three-entry branch queue. When the branch queue is full, no further conditional branches can be processed until one of the conditions in the branch queue is resolved.

When a branch is taken or predicted as taken, instructions from the untaken path must be flushed and the target instruction stream must be fetched into the IQ. The BTIC is a 128-entry, four-way set associative cache that contains the most recently used branch target instructions (up to four instructions per entry) for **b** and **bc** branches. When a taken branch instruction of this type hits in the BTIC, the instructions arrive in the IQ two clock cycles later, a clock cycle sooner than they would arrive from the instruction cache. Additional instructions arrive from the instruction cache in the next clock cycle. The BTIC reduces the number of missed opportunities to dispatch instructions and gives the core a 1-cycle head start on processing the target stream.

The BPU contains an adder to compute branch target addresses and three user-accessible registers—the link register (LR), the count register (CTR), and the condition register (CR). The BPU calculates the return pointer for subroutine calls and saves it in the LR for certain types of branch instructions. The LR also contains the branch target address for Branch Conditional to Link Register (**bclr***x*) instructions. The CTR contains the branch target address for Branch Conditional to Count Register (**bcctr***x*) instructions. Because the LR and CTR are SPRs, their contents can be copied to or from any GPR. Also, because the BPU uses dedicated registers rather than GPRs or FPRs, execution of branch instructions is largely independent from execution of integer and floating-point instructions.

5.2.1.3 Completion Unit

The completion unit operates closely with the instruction unit. Instructions are fetched and dispatched in program order. At the point of dispatch, the program order is maintained by assigning each dispatched instruction a successive entry in the 16-entry CQ. The completion unit tracks instructions from dispatch through execution and retires them in program order from the three bottom CQ entries (CQ0–CQ2).

Instructions cannot be dispatched to an execution unit unless there is a CQ vacancy.

Branch instructions that do not update the CTR or LR are often removed from the instruction stream. Those that are removed do not take a CQ entry. Branches that are not removed from the instruction stream

follow the same dispatch and completion procedures as non-branch instructions but are not dispatched to an issue queue.

Completing an instruction commits execution results to architected registers (GPRs, FPRs, VRs, LR, and CTR). In-order completion ensures the correct architectural state when the e600 core must recover from a mispredicted branch or any interrupt. An instruction is retired as it is removed from the CQ.

For a more detailed discussion of instruction completion, see the “Instruction Timing” chapter of the *e600 PowerPC Core Reference Manual.*

5.2.1.4 Independent Execution Units

In addition to the BPU, the e600 core provides the ten execution units described in the following sections.

5.2.1.4.1 AltiVec Vector Permute Unit (VPU)

The VPU executes permutation instructions such as pack, unpack, merge, splat, and permute on vector operands.

5.2.1.4.2 AltiVec Vector Integer Unit 1 (VIU1)

The VIU1 executes simple vector integer computational instructions, such as addition, subtraction, maximum and minimum comparisons, averaging, rotation, shifting, comparisons, and Boolean operations.

5.2.1.4.3 AltiVec Vector Integer Unit 2 (VIU2)

The VIU2 executes longer-latency vector integer instructions, such as multiplication, multiplication/addition, and sum-across with saturation.

5.2.1.4.4 AltiVec Vector Floating-Point Unit (VFPU)

The VFPU executes all vector floating-point instructions.

A maximum of two AltiVec instructions can be issued out-of-order to any combination of AltiVec execution units per clock cycle from the bottom two VIQ entries (VIQ1–VIQ0). An instruction in VIQ1 does not have to wait for an instruction in VIQ0 that is waiting for operand availability. Moreover, the VIU2, VFPU, and VPU are pipelined, so they can operate on multiple instructions.

5.2.1.4.5 Integer Units (IUs)

The integer units (three IU1s and IU2) are shown in Figure 5-1. The IU1s execute shorter latency integer instructions, that is, all integer instructions except multiply, divide, and move to/from special-purpose register instructions. IU2 executes integer instructions with latencies of three cycles or more.

IU2 has a 32-bit integer multiplier/divider and a unit for executing CR logical operations and move to/from SPR instructions. The multiplier supports early exit for operations that do not require full 32×32-bit multiplication.

5.2.1.4.6 Floating-Point Unit (FPU)

The FPU, shown in Figure 5-1, is designed such that double-precision operations require only a single pass, with a latency of 5 cycles. As instructions are dispatched to the FPUs reservation station, source operand data can be accessed from the FPRs or from the FPR rename buffers. Results in turn are written to the rename buffers and made available to subsequent instructions. Instructions start execution from the bottom reservation station only and execute in program order.

The FPU contains a single-precision multiply-add array and the floating-point status and control register (FPSCR). The multiply-add array allows the e600 core to implement multiply and multiply-add operations efficiently. The FPU is pipelined so that one single- or double-precision instruction can be issued per clock cycle.

Note that an execution bubble occurs after four consecutive, independent floating-point arithmetic instructions execute to allow for a normalization special case. Thirty-two 64-bit floating-point registers are provided to support floating-point operations. Stalls due to contention for FPRs are minimized by automatic allocation of the 16 floating-point rename registers. The e600 core writes the contents of the rename registers to the appropriate FPR when floating-point instructions are retired by the completion unit.

The e600 core supports all IEEE Std. 754 floating-point data types (normalized, denormalized, NaN, zero, and infinity) in hardware, eliminating the latency incurred by software interrupt routines.

5.2.1.4.7 Load/Store Unit (LSU)

The LSU executes all load and store instructions as well as the AltiVec LRU and transient instructions and provides the data transfer interface between the GPRs, FPRs, VRs, and the cache/core memory subsystem (MSS). The LSU also calculates effective addresses and aligns data.

Load and store instructions are issued and translated in program order; however, some memory accesses can occur out of order. Synchronizing instructions can be used to enforce strict ordering. When there are no data dependencies and the guarded bit for the page or block is cleared, a maximum of one out-of-order cacheable load operation can execute per clock cycle from the perspective of the LSU. Loads to FPRs require a 4-cycle total latency. Data returned from the cache is held in a rename register until the completion logic commits the value to a GPR, FPR, or VR. Stores cannot be executed out of order and are held in the store queue until the completion logic signals that the store operation is to be completed to memory. The e600 core executes store instructions with a maximum throughput of one every three clock cycles and a 3-cycle total latency to the data cache. The time required to perform the load or store operation depends on the processor: bus clock ratio and whether the operation involves the caches, system memory, or an I/O device.

5.2.2 Memory Management Units (MMUs)

The e600 core MMUs support up to 4 Petabytes (2^{52}) of virtual memory and 64 Gigabytes (2^{36}) of physical memory for instructions and data. The MMUs control access privileges for these spaces on block and page granularities. Referenced and changed status is maintained by the core for each page to support demand-paged virtual memory systems. The MMUs are contained within the LSU.

The MMU translates the effective address calculated by the IU and LSU to determine the physical address for the memory access.

The e600 core supports the following types of memory translation:

- Real addressing mode—In this mode, translation is disabled by clearing bits in the machine state register (MSR): MSR[IR] for instruction fetching or MSR[DR] for data accesses. When address translation is disabled, the physical address is identical to the effective address. When extended addressing is disabled (HID0[XAEN] = 0) a 32-bit physical address is used, PA[4–35]. For more details, see the “Memory Management” chapter of the *e600 PowerPC Core Reference Manual*.
- Page address translation—translates the page frame address for a 4-Kbyte page size
- Block address translation—translates the base address for blocks (4 Gbytes)

If translation is enabled, the appropriate MMU translates the higher-order bits of the effective address into physical address bits. Lower-order address bits are untranslated and are the same for both logical and physical addresses. These bits are directed to the caches where they form the index into the eight-way set-associative tag array. After translating the address, the MMU passes the higher-order physical address bits to the cache and the cache lookup completes. For caching-inhibited accesses or accesses that miss in the cache, the untranslated lower-order address bits are concatenated with the translated higher-order address bits; the resulting 32- or 36-bit physical address is used by the core memory subsystem (MSS) and the bus interface unit (BIU) to access memory external to the core.

The TLBs store page address translations for recent memory accesses. For each access, an effective address is presented for page and block translation simultaneously. If a translation is found in both the TLB and the BAT array, the block address translation in the BAT array is used. Usually the translation is in a TLB and the physical address is readily available to the cache. When a page address translation is not in a TLB, hardware or system software searches for one in the page table following the model defined by the PowerPC ISA.

Instruction and data TLBs provide address translation in parallel with the cache access, incurring no additional time penalty in the event of a TLB hit. The e600 core instruction and data TLBs are 128-entry, two-way set-associative caches that contain address translations. The core can initiate a hardware or system software search of the page tables in memory on a TLB miss.

5.2.3 L1 Instruction and Data Caches Within the Core

The e600 core implements separate L1 instruction and data caches. Each cache is 32 Kbytes and eight-way set-associative. As defined by the PowerPC ISA, they are physically indexed. Each cache block contains eight contiguous words from memory that are loaded from an eight-word boundary (that is, bits EA[27–31] are zeros); thus, a cache block never crosses a page boundary. An entire cache block can be updated by a four-beat burst load across a 64-bit MPX bus. Misaligned accesses across a page boundary can incur a performance penalty. The data cache is a non-blocking, write-back cache with hardware support for reloading on cache misses. The critical double word is transferred on the first beat and is forwarded to the requesting unit, minimizing stalls due to load delays. For vector loads, the critical quad word is handled similarly but is transferred on the second beat. The cache being loaded is not blocked to internal accesses while the load completes.

The e600 core L1 cache organization is shown in Figure 5-2.

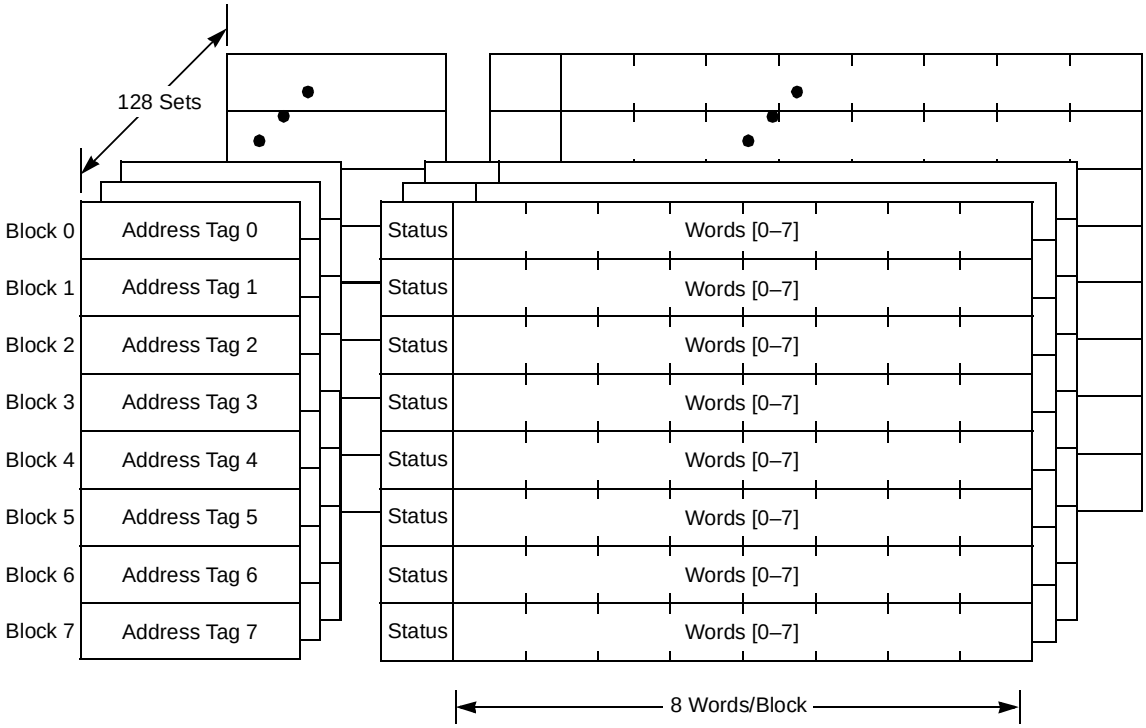


Figure 5-2. L1 Cache Organization

The instruction cache provides up to four instructions per clock cycle to the instruction queue. The instruction cache can be invalidated entirely or on a cache-block basis. It is invalidated and disabled by setting HID0[ICFI] and then clearing HID0[ICE]. The instruction cache can be locked by setting HID0[ILOCK]. The instruction cache supports only the valid/invalid states.

The data cache provides four words per clock cycle to the LSU. Like the instruction cache, the data cache can be invalidated all at once or on a per-cache-block basis. The data cache can be invalidated and disabled by setting HID0[DCFI] and then clearing HID0[DCE]. The data cache can be locked by setting HID0[DLOCK]. The data cache tags are dual-ported, so a load or store can occur simultaneously with a snoop.

The e600 core also implements a 128-entry (32-set, four-way set-associative) branch target instruction cache (BTIC). The BTIC is a cache of branch instructions that have been encountered in branch/loop code sequences. If the target instruction is in the BTIC, it is fetched into the instruction queue a cycle sooner than it can be made available from the instruction cache. Typically, the BTIC contains the first four instructions in the target stream.

The BTIC can be disabled and invalidated through software. As with other aspects of e600 core instruction timing, BTIC operation is optimized for cache-line alignment. If the first target instruction is one of the first five instructions in the cache block, the BTIC entry holds four instructions. If the first target instruction is the last instruction before the cache block boundary, it is the only instruction in the corresponding BTIC entry. If the next-to-last instruction in a cache block is the target, the BTIC entry holds two valid target instructions, as shown in Figure 5-3.

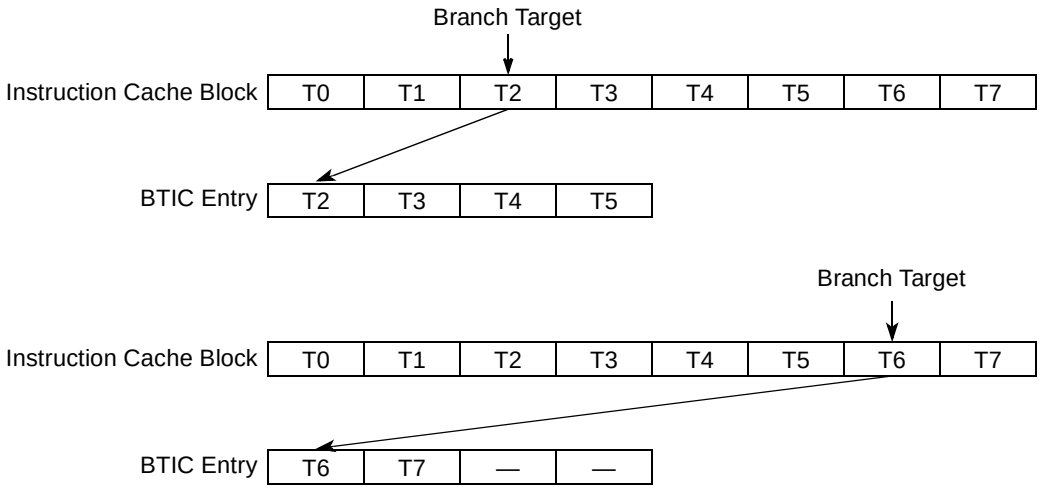


Figure 5-3. Alignment of Target Instructions in the BTIC

BTIC ways are updated using a FIFO algorithm.

For more information and timing examples showing cache hit and cache miss latencies, see the "Instruction Timing" chapter of the *e600 PowerPC Core Reference Manual.*

5.2.4 L2 Cache Implementation

The integrated L2 cache is a unified 1-Mbyte cache that receives memory requests from both the L1 instruction and data caches independently. It is eight-way set-associative and organized with 32-byte blocks and two blocks/line.

Each line consists of 64 bytes of data organized as two blocks (also called sectors). Although all 16 words in a cache line share the same address tag, each block maintains the three separate status bits for the 8 words of the cache block, the unit of memory at which coherency is maintained. Thus, each cache line can contain 16 contiguous words from memory that are read or written as 8-word operations.

The integrated L2 cache organization of the e600 core is shown in Figure 5-4.

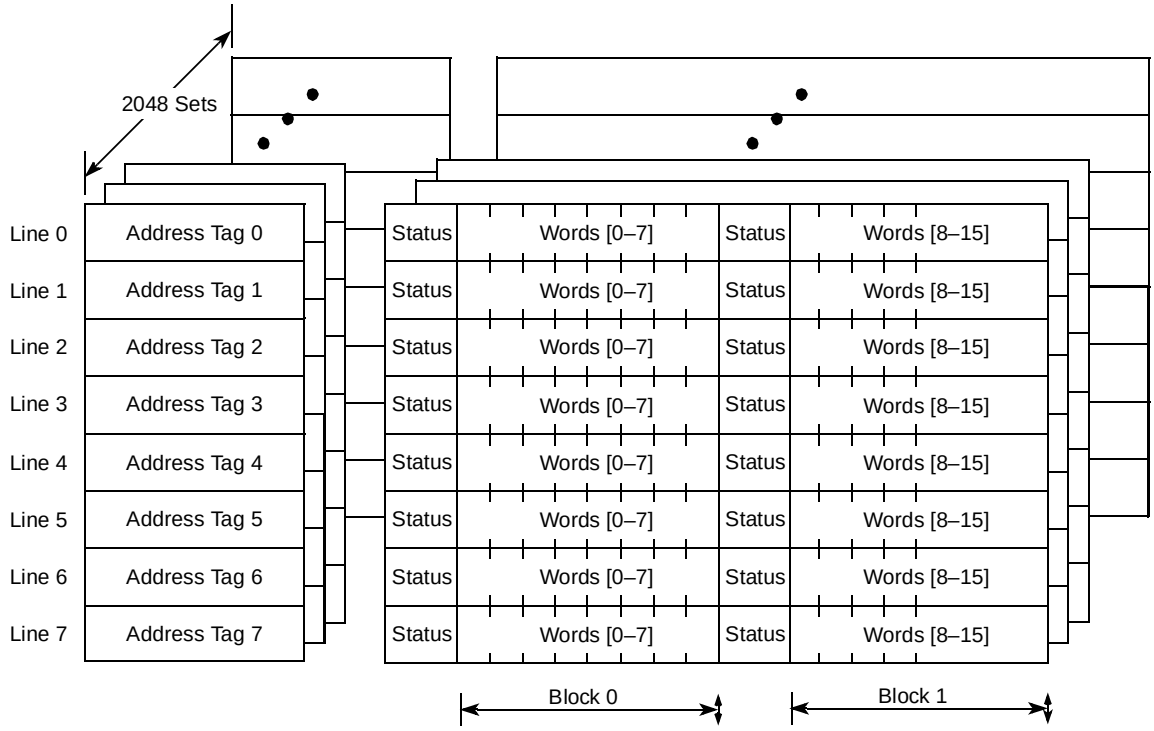


Figure 5-4. L2 Cache Organization

The L2 cache controller contains the L2 cache control register (L2CR), which does the following:

- Includes bits for enabling parity checking on the L2
- Provides for instruction-only and data-only modes
- Provides hardware flushing for the L2
- Selects between two available replacement algorithms for the L2 cache

The L2 implements the MESI cache coherency protocol using three status bits per sector.

Requests from the L1 cache generally result from instruction misses, data load or store misses, write-through operations, or cache management instructions. Requests from the L1 cache are compared against the L2 tags and serviced by the L2 cache if they hit; if they miss in the L2 cache, they go to main memory.

The L2 cache tags are fully pipelined and non-blocking for efficient operation. Thus the L2 cache can be accessed internally while a load for a miss is pending (allowing hits under misses). A reload for a cache miss is treated as a normal access and blocks other accesses for only 1 cycle.

For more information, see the “L1 and L2 Cache Operation” chapter of the *e600 PowerPC Core Reference Manual*.”

5.2.5 Core Interface

The e600 core has an advanced bus interface, the MPX bus interface. The MPX bus includes a 64 data bus and a 36-bit address bus along with sufficient control signals to allow for unique system level optimizations. The MPX bus has the following features:

- Extended 36-bit address bus
- 64-bit data bus; a 32-bit data bus mode is not supported
- Support for a four-state (MESI) cache coherence protocol
- Snooping within the core to maintain L1 data cache, and L2 cache coherency for multiprocessing applications and DMA environments
- Support for address-only transfers (useful for a variety of broadcast operations in multiprocessor applications)
- Address pipelining
- Support for up to 8 out-of-order transactions
- Full data streaming
- Support for data intervention in multiprocessor systems

5.2.6 Overview of Core Interface Accesses

The core interface includes address register queues, prioritization logic, and a bus control unit. The core interface latches snoop addresses for snooping in the L1 data cache and the L2 cache, the memory hierarchy address register queues, and the reservation controlled by the Load Word and Reserve Indexed (**lwarx**) and Store Word Conditional Indexed (**stwcx.**) instructions. Accesses are prioritized with load operations preceding store operations.

Instructions are automatically fetched from the memory system into the instruction unit where they are issued to the execution units at a peak rate of three instructions per clock cycle. Conversely, load and store instructions explicitly specify the movement of operands to and from the integer, floating-point, and AltiVec register files and the memory system.

When the e600 core encounters an instruction or data access, it calculates the effective address and uses the lower-order address bits to check for a hit in the 32-Kbyte L1 instruction and data caches within the core. During L1 cache lookup, the instruction and data memory management units (MMUs) use the higher-order address bits to calculate the virtual address, from which they calculate the physical (real) address. The physical address bits are then compared with the corresponding cache tag bits to determine if a cache hit occurred in the L1 instruction or data cache. If the access misses in the corresponding cache, the transaction is sent to L1 load miss queue or the L1 store miss queue. L1 load miss queue transactions are sent to the internal 1-Mbyte L2 cache. Store miss queue transactions are queued up in the L2 cache controller. If no match is found in the L2 tags, the physical address is used to access system memory.

In addition to loads, stores, and instruction fetches, the e600 core performs hardware table search operations following TLB misses, L1 and L2 cache castout operations, and cache-line snoop push operations when a modified cache line detects a snoop hit from another bus master.

5.2.6.1 Signal Groupings

A subset of the selected internal e600 core signals is grouped as follows:

- Interrupts/resets—These signals include the external interrupt signal, checkstop signals, and both soft reset and hard reset signals. They are used to interrupt and, under various conditions, to reset the core.
- Core status and control—These signals indicate the state of the core. They include the time-base enable, machine quiesce control, and power management signals.
- Clock control—These signals determine the system clock frequency and provide a flexible clocking scheme that allows the processor to operate at an integer multiple of the system clock frequency. They are also used to synchronize multiprocessor systems.
- Test interface—The JTAG (IEEE Std. 1149.1a™-1993) interface and the common on-chip processor (COP) unit provide a serial interface to the core for performing board-level boundary-scan interconnect tests. The test data input (*tdi*) and test data output (*tdo*) scan ports are used to scan instructions as well as data into the various scan registers for JTAG operations. The scan operation is controlled by the test access port (TAP) controller which in turn is controlled by the test mode select (*tms*) input sequence. The scan data is latched in at the rising edge of test clock (*tck*).
- Master address bus—These signals provide information about the type of transfer, such as whether the transaction is bursted, global, write-through, or cache-inhibited. A signal is also provided that indicates if a transfer error occurred.

NOTE

Active-low signals are shown with overbars. For example, $\overline{int}$ (interrupt) and $\overline{sreset}$ (soft reset). 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 *sysclk* (system clock) and *tdo* (JTAG test data output) are referred to as asserted when they are high and negated when they are low.

Figure 5-5 shows the subset of internal core interface signals. Signal functionality is described in detail in the “Core Interface” chapter of the *e600 PowerPC Core Reference Manual*.

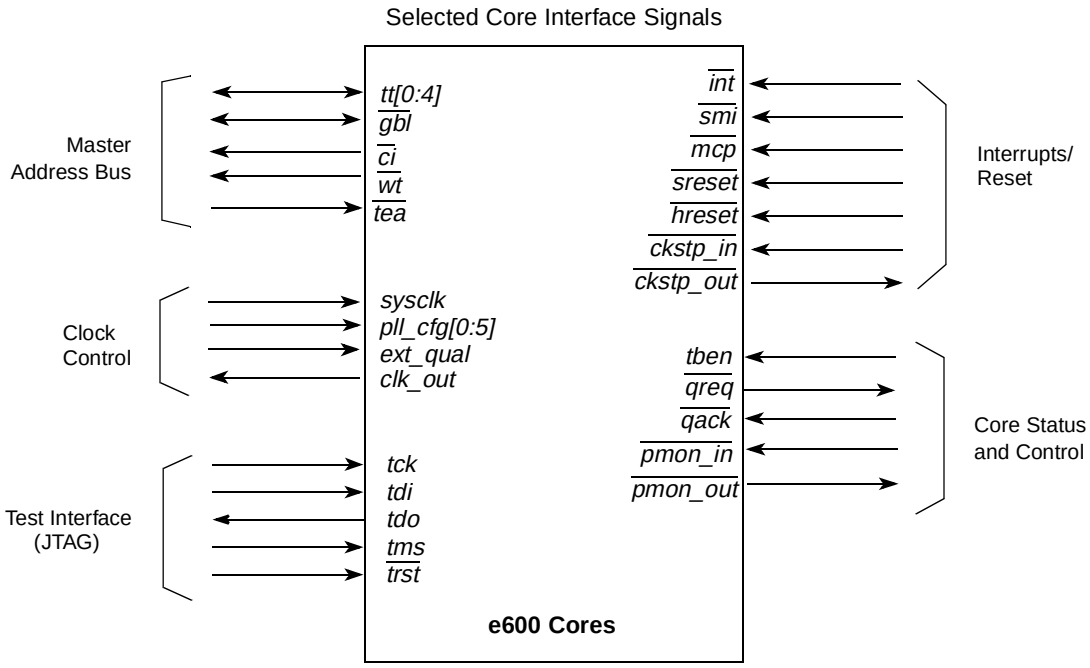


Figure 5-5. Core Interface Signals

5.2.6.2 Clocking

For functional operation, the e600 core uses a single clock input signal, *sysclk*, from which clocking is derived for the processor core and the MPX bus interface. Additionally, internal clock information is made available at the periphery of the core to support debug and development.

The e600 core clocking structure supports a wide range of core-to-MPX bus clock ratios. The internal core clock is synchronized to *sysclk* with the aid of a VCO-based PLL. The *pll_cfg*[0:5] signals are used to program the internal clock rate to a multiple of *sysclk*. The bus clock is maintained at the same frequency as *sysclk*. *sysclk* does not need to be a 50% duty-cycle signal.

5.2.7 Power and Thermal Management

The e600 core is designed for low-power operation. It provides both automatic and program-controlled power reduction modes. If an e600 core functional unit is idle, it automatically goes into a low-power mode. This mode does not affect operational performance. Dynamic power management automatically supplies or withholds power to execution units individually, based upon the contents of the instruction stream. The operation of dynamic power management is transparent to software or any hardware external to the core.

The following two programmable power modes are available to the system:

- Nap—Instruction fetching is halted. Only those clocks for time base, decrementer, and JTAG logic remain running. The e600 core goes into the doze state to snoop memory operations on the MPX bus and then back to nap using a $\overline{qreq}$/$\overline{qack}$ processor-system handshake protocol.
- Sleep—Power consumption is further reduced by disabling bus snooping, leaving only the PLL in a locked and running state. All internal functional units are disabled.

The e600 core also provides an instruction cache throttling mechanism to reduce the instruction execution rate. For thermal management, the e600 core provides a supervisor-level instruction cache throttling control register (ICTC). The “Power and Thermal Management” chapter of the *e600 PowerPC Core Reference Manual* explains how to configure the ICTC register for the e600 core.

5.2.8 Core Performance Monitor

The e600 core incorporates a performance monitor facility that system designers can use to help bring up, debug, and optimize software performance. The performance monitor counts events during execution of instructions related to dispatch, execution, completion, and memory accesses.

The performance monitor incorporates several registers that can be read and written to by supervisor-level software. User-level versions of these registers provide read-only access for user-level applications. These registers are described in Section 5.3.1, “PowerPC ISA Registers and Programming Model.” Performance monitor control registers, MMCR0, MMCR1, and MMCR2 can be used to specify which events are to be counted and the conditions for which a performance monitoring interrupt is taken. Additionally, the sampled instruction address register, SIAR (USIAR), holds the address of the first instruction to complete after the counter overflowed.

Attempting to write to a user-level read-only performance monitor register causes a program interrupt, regardless of the MSR[PR] setting.

When a performance monitor interrupt occurs, program execution continues from vector offset 0x00F00.

The Performance Monitor chapter of the *e600 PowerPC Core Reference Manual* describes the operation of the performance monitor diagnostic tool incorporated in the e600 core.

5.3 e600 Core Architectural Implementation

The PowerPC ISA consists of three layers. Adherence to the PowerPC ISA can be described in terms of which of the following levels of the architecture is implemented:

- User instruction set architecture (UISA)—Defines the base user-level instruction set, user-level registers, data types, floating-point exception model, memory models for a uniprocessor environment, and programming model for a uniprocessor environment
- Virtual environment architecture (VEA)—Describes the memory model for a multiprocessor environment, defines cache control instructions, and describes other aspects of virtual environments. Implementations that conform to the VEA also adhere to the UISA, but may not necessarily adhere to the OEA.
- Operating environment architecture (OEA)—Defines the memory management model, supervisor-level registers, synchronization requirements, and the interrupt model. Implementations that conform to the OEA also adhere to the UISA and the VEA.

The e600 core implementation supports the three levels of the architecture described above. For more information about the PowerPC ISA, see the *Programming Environments Manual for 32-Bit Implementations of the PowerPC Architecture.* Features specific to the e600 core are listed in Section 5.2, "e600 Core Features."

This section describes the PowerPC ISA in general, and specific details about the implementation of the e600 core as a low-power, 32-bit device that implements this architecture. The structure of this section follows the reference manual organization; each subsection provides an overview of that chapter.

- Registers and programming model—Section 5.3.1, "PowerPC ISA Registers and Programming Model," describes the registers for the operating environment architecture common to the e600 core and describes the programming model. It also describes the registers that are unique to the e600 core.

  Instruction set and addressing modes—Section 5.3.2, "Instruction Set," describes the instruction set and addressing modes for the operating environment architecture, and defines and describes the instructions implemented in the e600 core. The information in this section is described more fully in Chapter 6, "e600 Core Registers and Instruction Set Summary."
- Cache implementation—Section 5.3.3, "Cache Implementation within the Core," describes the cache model that is defined generally by the virtual environment architecture. It also provides specific details about the e600 core cache implementation. The information in this section is described more fully in the "L1 and L2 Cache Operation" chapter of the *e600 PowerPC Core Reference Manual*.
- Interrupt model—Section 5.3.4, "Interrupt Model," describes the interrupt model of the operating environment architecture and the differences in the e600 core interrupt model. The information in this section is described more fully in the "Interrupts" chapter of the *e600 PowerPC Core Reference Manual.*
- Memory management—Section 5.3.5, "Memory Management," describes generally the conventions for memory management. This section also describes the e600 core implementation of the 32-bit memory management specification. The information in this section is described more fully in the "Memory Management" chapter of the *e600 PowerPC Core Reference Manual.*
- Instruction timing—Section 5.3.6, "Instruction Timing," provides a general description of the instruction timing provided by the parallel execution supported by the PowerPC ISA and the e600 core. The information in this section is described more fully in the "Instruction Timing" chapter of the *e600 PowerPC Core Reference Manual.*
- AltiVec implementation—Section 5.3.7, "AltiVec Implementation," points out that the e600 core implements AltiVec registers, instructions, and interrupts as described in the *AltiVec Technology Programming Environments Manual.* "AltiVec Technology Implementation" chapter of the *e600 PowerPC Core Reference Manual* provides complete details.

5.3.1 PowerPC ISA Registers and Programming Model

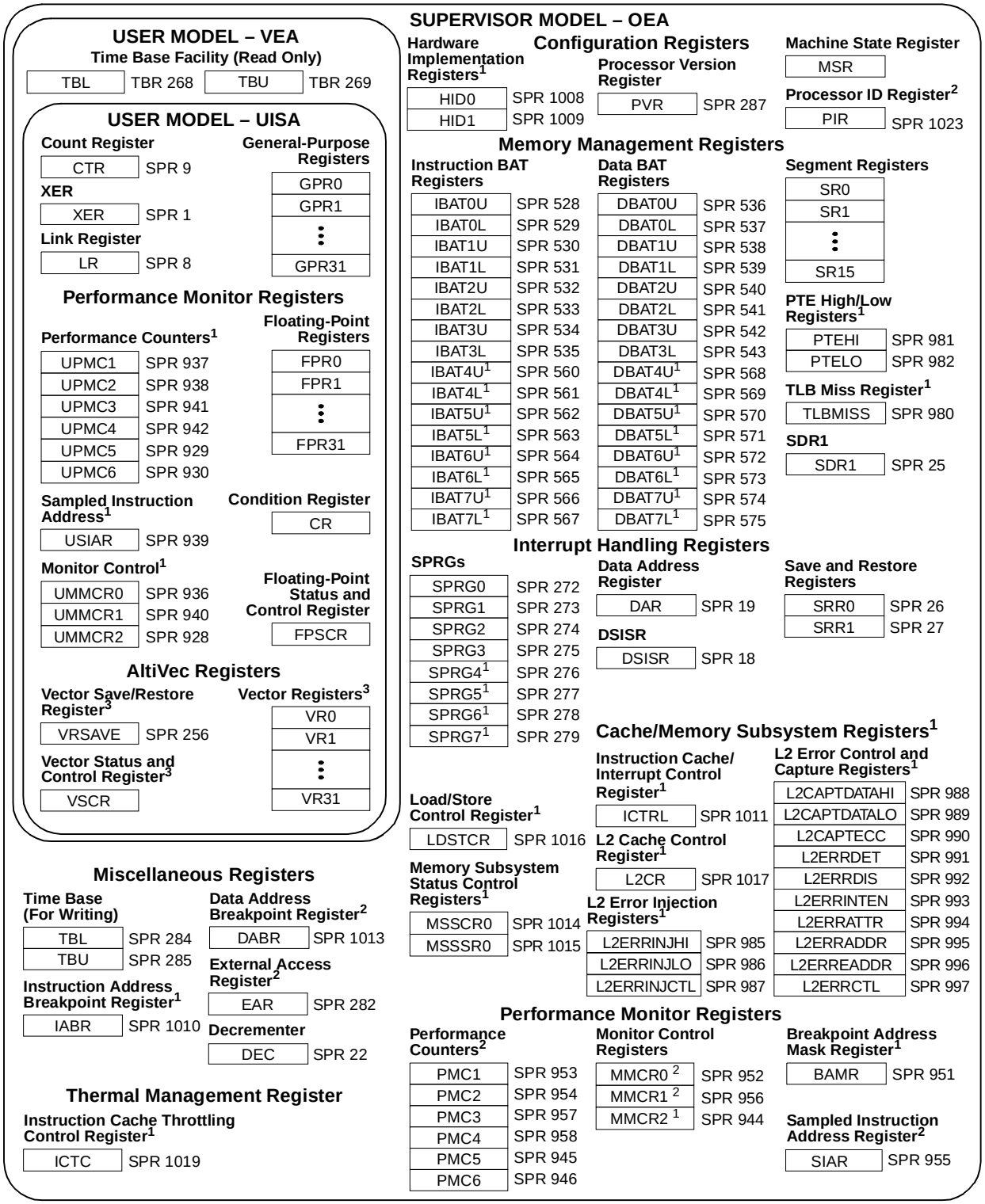
The PowerPC architecture defines register-to-register operations for most computational instructions. Source operands for these instructions are accessed from the registers or are provided as immediate values embedded in the instruction opcode. The three-register instruction format allows specification of a target register distinct from the two source operands. Load and store instructions transfer data between registers and memory.

The PowerPC architecture also defines two levels of privilege—supervisor mode of operation (typically used by the operating system) and user mode of operation (used by the application software). The programming models incorporate 32 GPRs, 32 FPRs, SPRs, and several miscellaneous registers. The AltiVec extensions to the PowerPC architecture augment the programming model with 32 VRs, one status and control register, and one save and restore register. Each processor that implements the PowerPC architecture also has a unique set of implementation-specific registers to support functionality that may not be defined by the PowerPC architecture.

Having access to privileged instructions, registers, and other resources allows the operating system to control the application environment (providing virtual memory and protecting operating-system and critical machine resources). Instructions that control the state of the processor, the address translation mechanism, and supervisor registers can be executed only when the processor is operating in supervisor mode.

Figure 5-6 shows all of the e600 core registers, indicating which are available at the user and supervisor levels. The numbers to the right of the SPRs indicate the number that is used in the syntax of the instruction operands to access the register. For more information, see Chapter 6, “e600 Core Registers and Instruction Set Summary.”

The OEA defines numerous SPRs that serve a variety of functions, such as providing controls, indicating status, configuring the processor, and performing special operations. During normal execution, a program can access the registers shown in Figure 5-6, depending on the program’s access privilege (supervisor or user, determined by the privilege-level bit, MSR[PR]). GPRs, FPRs, and VRs are accessed through operands that are part of the instructions. Access to registers can be explicit (that is, through the use of specific instructions for that purpose such as Move to Special-Purpose Register (**mtspr**) and Move from Special-Purpose Register (**mfspr**) instructions) or implicit, as the part of the execution of an instruction.



1 e600-specific register that may not be supported on other processors or cores that implement the PowerPC architecture.
2 Register defined as optional in the PowerPC architecture.
3 Register defined by the AltiVec technology.

Figure 5-6. Programming Model—e600 Core Registers

Some registers can be accessed both explicitly and implicitly. In the e600 core, all SPRs are 32 bits wide. Table 6-1 describes the registers implemented by the e600 core.

5.3.2 Instruction Set

All PowerPC instructions are encoded as single-word (32-bit) opcodes. Instruction formats are consistent among all instruction types, permitting efficient decoding to occur in parallel with operand accesses. This fixed instruction length and consistent format greatly simplifies instruction pipelining.

For more information, see Chapter 6, "e600 Core Registers and Instruction Set Summary."

5.3.2.1 PowerPC Instruction Set

The PowerPC instructions are divided into the following categories:

- Integer instructions—These include computational and logical instructions.
  - Integer arithmetic instructions
  - Integer compare instructions
  - Integer logical instructions
  - Integer rotate and shift instructions
- Floating-point instructions—These include floating-point computational instructions, as well as instructions that affect the FPSCR.
  - Floating-point arithmetic instructions
  - Floating-point multiply/add instructions
  - Floating-point rounding and conversion instructions
  - Floating-point compare instructions
  - Floating-point status and control instructions
- Load and store instructions—These include integer and floating-point load and store instructions.
  - Integer load and store instructions
  - Integer load and store multiple instructions
  - Floating-point load and store instructions
  - Primitives used to construct atomic memory operations (**lwarx** and **stwcx.** instructions)
- Flow control instructions—These include branching instructions, condition register logical instructions, trap instructions, and other instructions that affect the instruction flow.
  - Branch and trap instructions
  - Condition register logical instructions
- Processor control instructions—These instructions are used for synchronizing memory accesses and management of caches, TLBs, and the segment registers.
  - Move to/from SPR instructions (**mtspr**, **mfspr**)
  - Move to/from MSR (**mtmsr**, **mfmsr)**
  - Synchronize (**sync**)
  - Instruction synchronize (**isync**)

  - Order loads and stores
- Memory control instructions—These instructions provide control of caches, TLBs, and SRs.
  - Supervisor-level cache management instructions
  - User-level cache instructions
  - Segment register manipulation instructions
  - Translation lookaside buffer management instructions

This grouping does not indicate the execution unit that executes a particular instruction or group of instructions.

Integer instructions operate on byte, half-word, and word operands. Floating-point instructions operate on single-precision (one word) and double-precision (one double word) floating-point operands. The PowerPC architecture uses instructions that are four bytes long and word-aligned. It provides for byte, half-word, and word operand loads and stores between memory and a set of 32 GPRs. It also provides for word and double-word operand loads and stores between memory and a set of 32 floating-point registers (FPRs).

Computational instructions do not modify memory. To use a memory operand in a computation and then modify the same or another memory location, the memory contents must be loaded into a register, modified, and then written back to the target location with distinct instructions.

Processors that implement the PowerPC architecture follow the program flow when they are in the normal execution state. However, the flow of instructions can be interrupted directly by the execution of an instruction or by an asynchronous event. Either kind of interrupt may cause one of several components of the system software to be invoked.

Effective address computations for both data and instruction accesses use 32-bit unsigned binary arithmetic. A carry from bit 0 is ignored in 32-bit implementations.

5.3.2.2 AltiVec Instruction Set

The AltiVec instructions are divided into the following categories:

- Vector integer arithmetic instructions—These include arithmetic, logical, compare, rotate, and shift instructions.
- Vector floating-point arithmetic instructions—These include floating-point arithmetic instructions, as well as a discussion on floating-point modes.
- Vector load and store instructions—These include load and store instructions for vector registers. The AltiVec technology defines LRU and transient type instructions that can be used to optimize memory accesses.
  - LRU instructions. The AltiVec architecture specifies that the **lvxl** and **stvxl** instructions differ from other AltiVec load and store instructions in that they leave cache entries in a least-recently-used (LRU) state instead of a most-recently-used state.
  - Transient instructions. The AltiVec architecture describes a difference between static and transient memory accesses. A static memory access should have some reasonable degree of locality and be referenced several times or reused over some reasonably long period of time. A

transient memory reference has poor locality and is likely to be referenced a very few times or over a very short period of time.

The following instructions are interpreted to be transient:

– **dstt** and **dststt** (transient forms of the two data stream touch instructions)
– **lvxl** and **stvxl**

- Vector permutation and formatting instructions—These include pack, unpack, merge, splat, permute, select, and shift instructions.
- Processor control instructions—These instructions are used to read and write from the AltiVec status and control register.
- Memory control instructions—These instructions are used for managing of caches (user level and supervisor level).

5.3.2.3 e600 Core Instruction Set

The e600 core instruction set is defined as follows:

- The e600 core provides hardware support for all 32-bit PowerPC instructions.
- The e600 core implements the following instructions optional to the PowerPC architecture:
  — External Control In Word Indexed (**eciwx**)
  — External Control Out Word Indexed (**ecowx**)
  — Data Cache Block Allocate (**dcba**)
  — Floating Select (**fsel**)
  — Floating Reciprocal Estimate Single-Precision (**fres**)
  — Floating Reciprocal Square Root Estimate (**frsqrte**)
  — Store Floating-Point as Integer Word (**stfiwx**)
  — Load Data TLB Entry (**tlbld**)
  — Load Instruction TLB Entry (**tlbli**)

5.3.3 Cache Implementation within the Core

The following subsections describe the PowerPC architecture's treatment of cache in general, and the e600-specific implementation, respectively. A detailed description of the e600 core cache implementation is provided in the "L1 and L2 Cache Operation" chapter of the *e600 PowerPC Core Reference Manual*.

5.3.3.1 PowerPC Cache Model

The PowerPC architecture does not define hardware aspects of cache implementations. For example, devices that implement the PowerPC architecture can have unified caches, separate L1 instruction and data caches (Harvard architecture), or no cache at all. These devices control the following memory access modes on a page or block basis:

- Write-back/write-through mode
- Caching-inhibited/caching-allowed mode
- Memory coherency required/memory coherency not required mode

The caches are physically addressed, and the data cache can operate in either write-back or write-through mode as specified by the PowerPC architecture.

The PowerPC architecture defines the term ‘cache block’ as the cacheable unit. The VEA and OEA define cache management instructions a programmer can use to affect cache contents.

5.3.3.2 e600 Core Cache Implementation

The BPU contains a 128-entry BTIC that provides immediate access to cached target instructions.

5.3.4 Interrupt Model

The following sections describe the PowerPC interrupt model and the e600 core implementation. A detailed description of the e600 core interrupt model is provided in the “Interrupts” chapter of the *e600 PowerPC Core Reference Manual.*

5.3.4.1 PowerPC Interrupt Model

The OEA portion of the PowerPC architecture defines the mechanism by which processors that implement the PowerPC architecture invoke interrupts. Exception conditions may be defined at other levels of the architecture. For example, the UISA defines floating-point exception conditions; the OEA defines the mechanism by which the resulting interrupt is taken.

The PowerPC interrupt mechanism allows the processor to change to supervisor state as a result of unusual conditions arising in the execution of instructions and from signals external to the core, bus errors, or various internal conditions. When interrupts occur, information about the state of the processor is saved to certain registers and the processor begins execution at an address (interrupt vector) predetermined for each interrupt. Processing of interrupts begins in supervisor mode.

Although multiple exception conditions can map to a single interrupt vector, often a more specific condition may be determined by examining a register associated with the interrupt—for example, the DSISR and the floating-point status and control register (FPSCR). Also, software can explicitly enable or disable some exception conditions.

The PowerPC architecture requires that interrupts be taken in program order; therefore, although a particular implementation may recognize exception conditions out of order, they are handled strictly in order with respect to the instruction stream. When an instruction-caused interrupt is recognized, any unexecuted instructions that appear earlier in the instruction stream, including any that have not yet entered the execute state, are required to complete before the interrupt is taken. In addition, if a single instruction encounters multiple exception conditions, the resulting interrupts are taken and handled sequentially. Likewise, asynchronous, precise interrupts are recognized when they occur, but are not handled until all instructions currently in the execute stage successfully complete execution and report their results.

To prevent loss of state information, interrupt handlers must save the information stored in the machine status save/restore registers, SRR0 and SRR1, soon after the interrupt is taken to prevent this information from being lost due to another interrupt event. Because exceptions can occur while an interrupt handler routine is executing, multiple interrupts can become nested. It is the interrupt handler’s responsibility to save the necessary state information if control is to return to the interrupted program.

In many cases, after the interrupt handler handles an interrupt, there is an attempt to execute the instruction that caused the interrupt. Instruction execution continues until the next interrupt is encountered. Recognizing and handling interrupt conditions sequentially guarantees that the machine state is recoverable and processing can resume without losing instruction results.

The following terms are used to describe the stages of interrupt processing: recognition, taken, and handling.

- Recognition—Exception recognition occurs when the exception condition that can cause an interrupt is identified by the processor.
- Taken—An interrupt is said to be taken when control of instruction execution is passed to the interrupt handler; that is, the context is saved and the instruction at the appropriate vector offset is fetched and the interrupt handler routine begins executing in supervisor mode.
- Handling—Interrupt handling is performed by the software at the appropriate vector offset. Interrupt handling is begun in supervisor mode.

The occurrence of IEEE Std.754 floating-point exceptions may not cause an exception to be taken. These IEEE-defined exceptions are referred to as IEEE floating-point exceptions or floating-point exceptions.

5.3.4.2 e600 Core Interrupts

As specified by the PowerPC architecture, interrupts can be either precise or imprecise and either synchronous or asynchronous. Asynchronous interrupts are caused by events external to the core's execution; synchronous interrupts are caused by instructions.

The types of interrupts are shown in Table 5-1. Note that all interrupts except for the performance monitor, AltiVec unavailable, instruction address breakpoint, system management, AltiVec assist, and the three software table search interrupts are described in Chapter 6, "Interrupts," in *The Programming Environments Manual*.

Table 5-1. e600 Core Interrupt Classifications

Synchronous/Asynchronous	Precise/Imprecise	Interrupt Types
Asynchronous, nonmaskable	Imprecise	System reset, machine check
Asynchronous, maskable	Precise	External interrupt, system management interrupt, decrementer interrupt, performance monitor interrupt
Synchronous	Precise	Instruction-caused interrupts

The interrupt classifications are discussed in greater detail in the "Interrupts" chapter of the *e600 PowerPC Core Reference Manual* For a better understanding of how the e600 core implements precise interrupts, see the "Instruction Timing" chapter of the *e600 PowerPC Core Reference Manual.* Table 5-2 lists the interrupts implemented in the e600 core and conditions that cause them. Table 5-2 also notes the e600-specific interrupts.

The three software table search interrupts support software page table searching and are enabled by setting HID0[STEN]. See the "Interrupts" and "Memory Management" chapters of the *e600 PowerPC Core Reference Manual.*

Table 5-2. Interrupts and Exception Conditions

Interrupt Type	Vector Offset	Causing Conditions
Reserved	0x00000	—
System reset	0x00100	Assertion of either *hreset* or *sreset* or at power-on reset
Machine check	0x00200	Assertion of *tea* during a data bus transaction (see Section 5.3.4.2.1, "Sources of tea_assertion," for more information), assertion of *mcp*, an address bus parity error on the MPX bus, a data bus parity error on the MPX bus, an L1 instruction cache error, an L1 data cache error, and a core memory subsystem detected error including the following: • L2 data parity error • L2 tag parity error • Single-bit and multiple-bit L2 ECC errors MSR[ME] must be set.
DSI	0x00300	As specified in the PowerPC architecture. Also includes the following: • A hardware table search due to a TLB miss on load, store, or cache operations results in a page fault • Any load or store to a direct-store segment (SR[T] = 1) • A **lwarx** or **stwcx.** instruction to memory with cache-inhibited or write-through memory/cache access attributes.
ISI	0x00400	As specified in the PowerPC architecture
External interrupt	0x00500	MSR[EE] = 1 and *int* is asserted
Alignment	0x00600	• A floating-point load/store, **stmw**, **stwcx.**, **lmw**, **lwarx**, **eciwx**, or **ecowx** instruction operand is not word-aligned. • A multiple/string load/store operation is attempted in little-endian mode • An operand of a **dcbz** instruction is on a page that is write-through or cache-inhibited for a virtual mode access. • An attempt to execute a **dcbz** instruction occurs when the cache is disabled or locked.
Program	0x00700	As specified in the PowerPC architecture
Floating-point unavailable	0x00800	As specified in the PowerPC architecture
Decrementer	0x00900	As defined by the PowerPC architecture, when the msb of the DEC register changes from 0 to 1 and MSR[EE] = 1
Reserved	0x00A00–00BFF	—
System call	0x00C00	Execution of the System Call (**sc**) instruction
Trace	0x00D00	MSR[SE] =1 or a branch instruction is completing and MSR[BE] =1. The e600 core operates as specified in the OEA by taking this interrupt on an **isync**.

Table 5-2. Interrupts and Exception Conditions (continued)

Interrupt Type	Vector Offset	Causing Conditions
Reserved	0x00E00	The e600 core does not generate an interrupt to this vector. Other processors may use this vector for floating-point assist interrupts.
Reserved	0x00E10–00EFF	—
Performance monitor	0x00F00	The limit specified in PMC*n* is met and MMCR0[ENINT] = 1 (e600-specific)
AltiVec unavailable	0x00F20	Occurs due to an attempt to execute any non-streaming AltiVec instruction when MSR[VEC] = 0. This interrupt is not taken for data streaming instructions (**dst*x***, **dss,** or **dssall**). (e600-specific)
ITLB miss	0x01000	Caused when HID0[STEN] = 1 and the effective address for an instruction fetch cannot be translated by the ITLB (e600-specific).
DTLB miss-on-load	0x01100	Caused when HID0[STEN] = 1 and the effective address for a data load operation cannot be translated by the DTLB (e600-specific).
DTLB miss-on-store	0x01200	Caused when HID0[STEN] = 1, and either the effective address for a data store operation cannot be translated by the DTLB, or when a DTLB hit occurs, and the changed bit in the PTE must be set due to a data store operation (e600-specific).
Instruction address breakpoint	0x01300	IABR[0–29] matches EA[0–29] of the next instruction to complete and IABR[BE] = 1 (e600-specific).
System management interrupt	0x01400	MSR[EE] = 1 and $\overline{smi\_0}$ and $\overline{smi\_1}$ are asserted (e600-specific). See section, “System Management Interrupt (0x01400)” of the “Interrupts” chapter of the e600 PowerPC Core Reference Manual.
Reserved	0x01500–015FF	—
AltiVec assist	0x01600	This e600-specific interrupt supports denormalization detection in Java™ mode as specified in the *AltiVec Technology Programming Environments Manual* in Chapter 3, “Operand Conventions.”
Reserved	0x01700–02FFF	—

5.3.4.2.1 Sources of *tea_* assertion

When the core performs a read transaction to a target on one of the external interfaces, a bus error may occur, which in turn causes the assertion of *tea_* to the core. Table 5-3 summarizes the potential sources of *tea_* assertion and the associated error reporting register fields.

Table 5-3. *tea_* Sources

Read Target	Error Register	Error Register Fields
MCM	EDR	LAE
DDR	ERR_DETECT	MBE MSE
LBIU	LTESR	DM PAR
SRIO	LTLEDCSR	IER PRT

Table 5-3. *tea_* Sources (continued)

Read Target	Error Register	Error Register Fields
PCI Express	PEX_ERR_DR	PCT PCAC CDNSC CRSNC IACA CRST IOIS CIS CIEP IOIEP OAC IOIA
	PEX_PME_MES_DR	HRD LDD

5.3.5 Memory Management

The following subsections describe the memory management features of the PowerPC architecture, and the e600 core implementation, respectively.

5.3.5.1 PowerPC Memory Management Model

The primary function of the MMU in a processor or core that implements the PowerPC architecture is the translation of logical (effective) addresses to physical addresses (referred to as real addresses in the architecture specification) for memory accesses and I/O accesses (I/O accesses are assumed to be memory-mapped). In addition, the MMU provides access protection on a segment, block, or page basis. Note that the e600 core does not implement the optional direct-store facility.

Two general types of memory accesses generated by processors that implement the PowerPC architecture require address translation—instruction accesses and data accesses generated by load and store instructions. In addition, the addresses specified by cache instructions and the optional external control instructions also require translation. Generally, the address translation mechanism is defined in terms of the segment descriptors and page tables that the processors use to locate the effective-to-physical address mapping for memory accesses. The segment information translates the effective address to an interim virtual address, and the page table information translates the virtual address to a physical address.

The segment descriptors, used to generate the interim virtual addresses, are stored as segment registers on 32-bit implementations (such as the e600 core). In addition, two translation lookaside buffers (TLBs) are implemented on the e600 core to keep recently used page address translations within the core. Although the PowerPC OEA describes one MMU (conceptually), the e600 core hardware maintains separate TLBs and table search resources for instruction and data accesses that can be performed independently (and simultaneously). Therefore, the e600 core is described as having two MMUs, one for instruction accesses (IMMU) and one for data accesses (DMMU).

The block address translation (BAT) mechanism is a software-controlled array that stores the available block address translations within the core. BAT array entries are implemented as pairs of BAT registers

that are accessible as supervisor special-purpose registers (SPRs). There are separate instruction and data BAT mechanisms. In the e600 core, they reside in the instruction and data MMUs, respectively.

The MMUs, together with the interrupt processing mechanism, provide the necessary support for the operating system to implement a paged virtual memory environment and for enforcing protection of designated memory areas. The “Interrupts” chapter of the *e600 PowerPC Core Reference Manual* describes how the MSR controls critical MMU functionality.

5.3.5.2 e600 Core Memory Management Implementation

The e600 core implements separate MMUs for instructions and data. It maintains a copy of the segment registers in the instruction MMU; however, read and write accesses to the segment registers (**mfsr** and **mtsr**) are handled through the segment registers in the data MMU. The e600 core MMU is described in the “Memory Management” chapter of the *e600 PowerPC Core Reference Manual.*

The e600 core implements the memory management specification of the PowerPC OEA for 32-bit implementations but adds capability for supporting 36-bit physical addressing. Thus, it provides 4 Gbytes of physical address space accessible to supervisor and user programs, with a 4-Kbyte page size and 256-Mbyte segment size. In addition, the e600 core MMUs use an interim virtual address (52 bits) and hashed page tables in the generation of 32- or 36-bit physical addresses (depending on the setting of HID0[XAEN]). Processors that implement the PowerPC architecture also have a BAT mechanism for mapping large blocks of memory. Block range from 128 Kbytes to 256 Mbytes and are software programmable.

The e600 core provides table search operations performed in hardware. The 52-bit virtual address is formed and the MMU attempts to fetch the PTE that contains the physical address from the appropriate TLB within the core. If the translation is not found in either the BAT array or in a TLB (that is, a TLB miss occurs), the hardware performs a table search operation (using a hashing function) to search for the PTE. Hardware table searching is the default mode for the e600 core; however, if HID0[STEN] = 1, a software table search is performed.

The e600 core also provides support for table search operations performed in software (if HID0[STEN] is set). In this case, the TLBMISS register saves the effective address of the access that requires a software table search. The PTEHI and PTELO registers and the **tlbli** and **tlbld** instructions are used in reloading the TLBs during a software table search operation.

The following interrupts support software table searching if HID0[STEN] is set and a TLB miss occurs:

- For an instruction fetch, an ITLB miss interrupt
- For a data load, a DTLB miss-on-load interrupt
- For a data store, a DTLB miss-on-store interrupt

The e600 core implements the optional TLB invalidate entry (**tlbie**) and TLB synchronize (**tlbsync**) instructions that can be used to invalidate TLB entries.

5.3.6 Instruction Timing

This section describes how the e600 core performs operations defined by instructions and reports the results of instruction execution. The e600 core design minimizes average instruction execution latency,

which is the number of clock cycles it takes to fetch, decode, dispatch, issue, and execute instructions and make results available for subsequent instructions. Some instructions, such as loads and stores, access memory and require additional clock cycles between the execute phase and the write-back phase. Latencies depend on whether an access is to cacheable or noncacheable memory, whether it hits in the L1 or L2 cache, whether a cache access generates a write back to memory, whether the access causes a snoop hit from another device that generates additional activity, and other conditions that affect memory accesses.

To improve throughput, the e600 core implements pipelining, superscalar instruction issue, branch folding, removal of fall-through branches, three-level speculative branch handling, and multiple execution units that operate independently and in parallel.

As an instruction passes from stage to stage, the subsequent instruction can follow through the stages as the preceding instruction vacates them, allowing several instructions to be processed simultaneously. Although it may take several cycles for an instruction to pass through all the stages, when the pipeline is full, one instruction can complete its work on every clock cycle. Figure 5-7 represents a generic four-stage pipelined execution unit, which when filled has a throughput of one instruction per clock cycle.

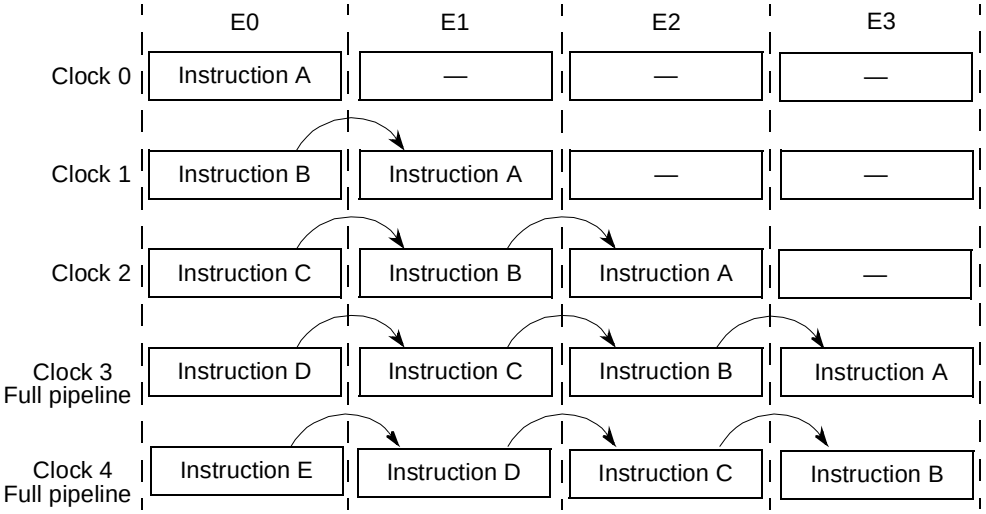


Figure 5-7. Pipelined Execution Unit

Figure 5-8 shows the entire path that instructions take through the fetch1, fetch2, decode/dispatch, execute, issue, complete, and write-back stages, which is considered the master pipeline of the e600 core. The FPU, LSU, IU2, VIU2, VFPU, and VPU are multiple-stage pipelines.

The e600 core contains the following execution units:

- Branch processing unit (BPU)
- Three integer unit 1s (IU1a, IU1b, and IU1c)—execute all integer instructions except multiply, divide, and move to/from SPR instructions.
- Integer unit 2 (IU2)—executes miscellaneous instructions including the CR logical operations, integer multiplication and division instructions, and move to/from special-purpose register instructions
- 64-bit floating-point unit (FPU)

- Load/store unit (LSU)
- The AltiVec unit contains the following four independent execution units for vector computations:
  - AltiVec permute unit (VPU)
  - AltiVec integer unit 1 (VIU1)
  - Vector integer unit 2 (VIU2)
  - Vector floating-point unit (VFPU)

  A maximum of two AltiVec instructions can be issued out-of-order to any combination of AltiVec execution units per clock cycle from the bottom two VIQ entries (VIQ1–VIQ0). An instruction in VIQ1 does not have to wait for an instruction in VIQ0 that is waiting for operand availability. Moreover, the VIU2, VFPU, and VPU are pipelined, so they can operate on multiple instructions.

The e600 core can complete as many as three instructions on each clock cycle. In general, the e600 core processes instructions in seven stages—fetch1, fetch2, decode/dispatch, issue, execute, complete, and write-back, as shown in Figure 5-8.

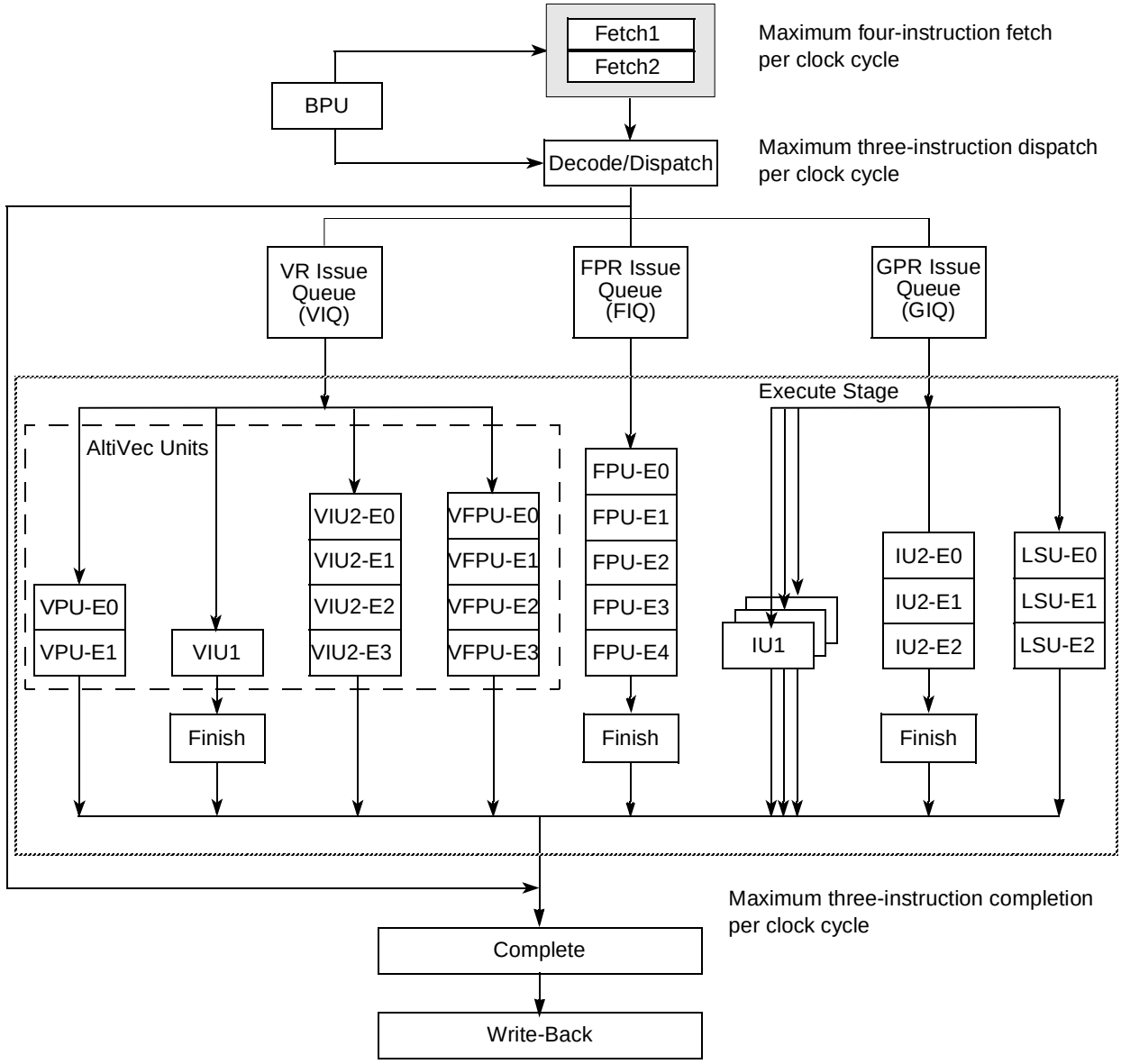


Figure 5-8. Superscalar/Pipeline Diagram

The instruction pipeline stages are described 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 branch target instruction cache (BTIC), the instruction cache, or the L2 cache. Those factors increase when it is necessary to fetch instructions from system memory and include the core-to-MPX bus clock ratio, the amount of bus traffic, and whether any cache coherency operations are required.

- The decode/dispatch stage fully decodes each instruction; most instructions are dispatched to the issue queues (branch, **isync**, **rfi**, and **sc** instructions do not go to issue queues).
- The three issue queues, FIQ, VIQ, and GIQ, can accept as many as one, two, and three instructions, respectively, in a cycle. Instruction dispatch requires the following:
  - Instructions are dispatched only from the three lowest IQ entries—IQ0, IQ1, and IQ2.
  - A maximum of three instructions can be dispatched to the issue queues per clock cycle.
  - Space must be available in the CQ for an instruction to dispatch (this includes instructions that are assigned a space in the CQ but not an issue queue).
- The issue stage reads source operands from rename registers and register files and determines when instructions are latched into the execution unit reservation stations. The GIQ, FIQ, and VIQ (AltiVec) issue queues have the following similarities:
  - Operand lookup in the GPRs, FPRs, and VRs, and their rename registers.
  - Issue queues issue instructions to the proper execution units.
  - Each issue queue holds twice as many instructions as can be dispatched to it in one cycle; the GIQ has six entries, the VIQ has four, and the FIQ has two.

  The three issue queues are described as follows:
  - The GIQ accepts as many as three instructions from the dispatch unit each cycle. IU1, IU2, and all LSU instructions (including floating-point and AltiVec loads and stores) are dispatched to the GIQ.
  - Instructions can be issued out-of-order from the bottom three GIQ entries (GIQ2–GIQ0). An instruction in GIQ1 destined for an IU1 does not have to wait for an instruction in GIQ0 that is stalled behind a long-latency integer divide instruction in the IU2.
  - The VIQ accepts as many as two instructions from the dispatch unit each cycle. All AltiVec instructions (other than load, store, and vector touch instructions) are dispatched to the VIQ. A maximum of two AltiVec instructions can be issued out-of-order to any combination of AltiVec execution units per clock cycle from the bottom two VIQ entries (VIQ1–VIQ0). This means an instruction in VIQ1 does not have to wait for an instruction in VIQ0 that is waiting for operand availability.
  - The FIQ can accept one instruction from the dispatch unit per clock cycle. It looks at the first instruction in its queue and determines if the instruction can be issued to the FPU in this cycle.
- 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 interrupts to the completion stage. Instruction-generated interrupts 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 FPU, LSU, IU2, VIU2, VFPU, and VPU units are pipelined, as shown in the “AltiVec Technology Implementation” chapter of the *e600 PowerPC Core Reference Manual*.

Note that AltiVec computational instructions are executed in the four independent, pipelined AltiVec execution units. The VPU has a two-stage pipeline, the VIU1 has a one-stage pipeline, and the VIU2 and VFPU have four-stage pipelines. As many as 10 AltiVec instructions can be executing concurrently.

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

  The complete stage ends when the instruction is retired. Three instructions can be retired per clock cycle. If no dependencies exist, as many as three instructions are retired in program order. The "Instruction Timing" chapter of the *e600 PowerPC Core Reference Manual* describes completion dependencies.

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

5.3.7 AltiVec Implementation

The e600 core implements the AltiVec registers and instruction set as they are described in the *AltiVec Technology Programming Environments Manual* in Chapter 2, "AltiVec Register Set," and in Chapter 6, "AltiVec Instructions." One additional implementation-specific interrupt, the AltiVec assist interrupt, is used in handling denormalized numbers in Java mode.

Both interrupts are described fully in the "Interrupts" chapter of the *e600 PowerPC Core Reference Manual.* Also, the default setting for the VSCR[NJ] bit is Java–compliant (VSCR[NJ] = 0) in the e600 core. The AltiVec implementation is described fully in the "AltiVec Technology Implementation" chapter of the *e600 PowerPC Core Reference Manual*.

5.4 MPC8641D Implementation Details

Table 5-4 summarizes e600 core functionality specifics for the MPC8641D.

Table 5-4. MPC8641D Implementation

Feature	MPC8641D Implementation
MSSCR0 implementation	DTQ: Set to 0b000 (8 entries) in each core.
	L2PFE: Set to 0b00 (L2 prefetching disabled, no prefetch engine)
HID1 implementation	EBA (address bus parity checking): Read-only and set to 0
	EBD (data bus parity checking): Read-only and set to 0
Hard reset setting for MSSCR0	0x0000_8000 for core 0, 0x0000_8020 for core 1
Hard reset setting for L2CR	0x3000_00n0, where n = xxx and xxx is set by fuses on production parts
PIR value	The PIR value is all zeros.
PVR value	See Section 6.1.4.1, “Processor Version Register (PVR),” for the specific values of the PVR on the MPC8641 and MPC8641D.
SVR value	See Section 17.4.1.14, “System Version Register (SVR),” for the specific values of the SVR on the MPC8641 and MPC8641D.

Chapter 6
e600 Core Registers and Instruction Set Summary

This chapter describes the PowerPC ISA registers on the e600 core, emphasizing those features specific to the e600 core and summarizing those that are common to processors implementing the PowerPC ISA. It consists of three major sections, which describe the following:

- Registers implemented in the e600 core
- Operand conventions
- The e600 core instruction set

For detailed information about architecture-defined features, see the *Programming Environments Manual* and the *AltiVec Technology Programming Environments Manual*.

AltiVec Technology and the Programming Model

AltiVec programming model features are described as follows:

- Thirty-four additional registers—32 VRs, VRSAVE, and VSCR. See the “AltiVec Technology Implementation” chapter of the *e600 PowerPC Core Reference Manual*.

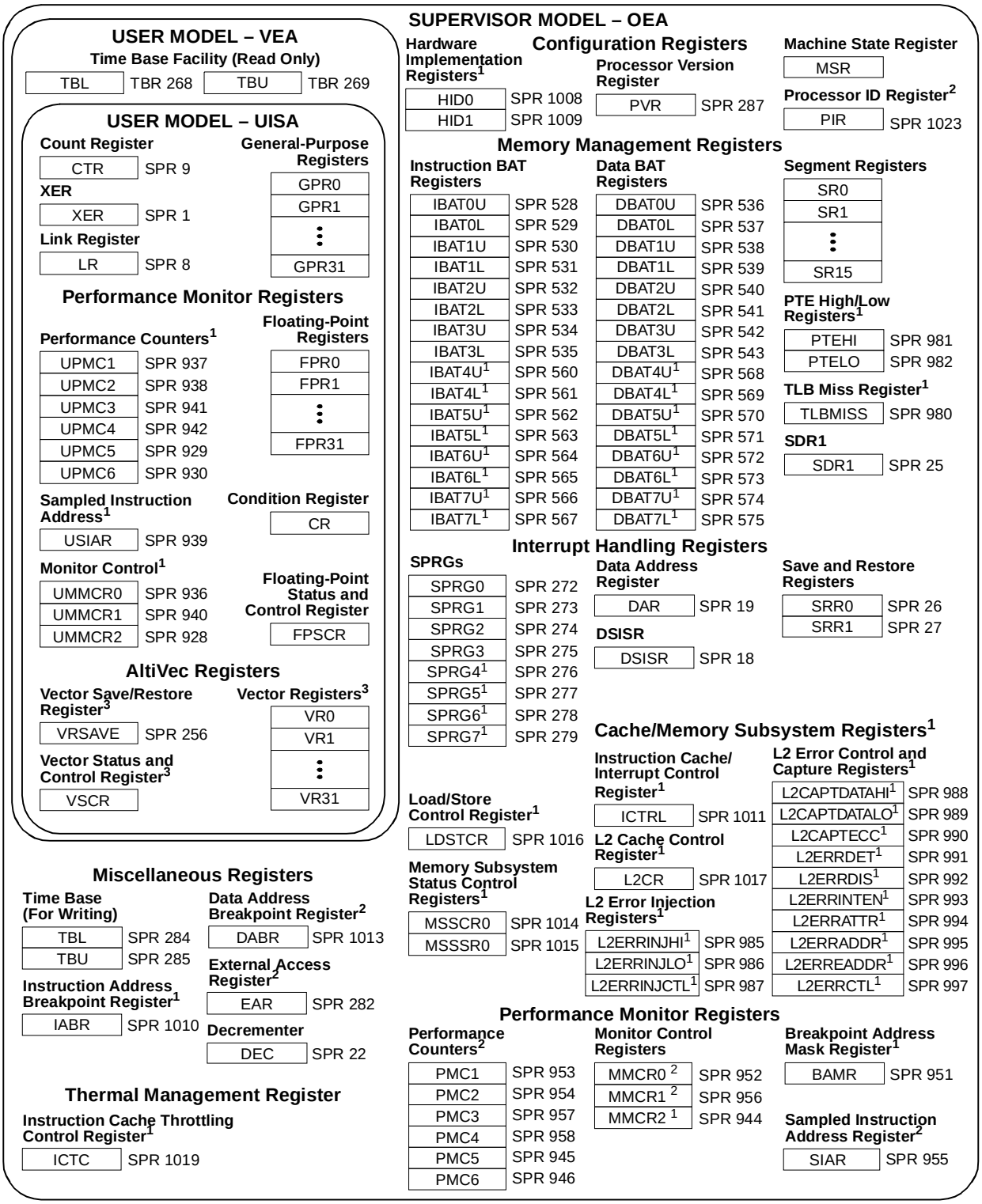
6.1 e600 Core Register Set

This section describes the registers implemented in the e600 core. It includes an overview of registers defined by the PowerPC ISA and the AltiVec technology, highlighting differences in how these registers are implemented in the e600 core, and a detailed description of e600-core-specific registers. Full descriptions of the architecture-defined register set are provided in Chapter 2, “Register Set,” in the *Programming Environments Manual*, Chapter 2, “AltiVec Register Set,” in the *AltiVec Technology Programming Environments Manual* (PEM), and in the *e600 PowerPC Core Reference Manual* (e600RM).

Registers are defined at all three levels of the PowerPC ISA—user instruction set architecture (UISA), virtual environment architecture (VEA), and operating environment architecture (OEA). The PowerPC ISA defines register-to-register operations for all computational instructions. Source data for these instructions is accessed from the registers within the core or provided as immediate values embedded in the opcode. The three-register instruction format allows specification of a target register distinct from the two source registers, thus preserving the original data for use by other instructions and reducing the number of instructions required for certain operations. Data is transferred between memory and registers with explicit load and store instructions only.

6.1.1 Register Set Overview

Figure 6-1 shows the e600 core register set.



1 e600-specific register that may not be supported on other processors that implement the PowerPC architecture.
2 Register defined as optional in the PowerPC architecture.
3 Register defined by the AltiVec technology.

Figure 6-1. Programming Model—e600 Core Registers

The number to the right of the special-purpose registers (SPRs) is the number used in the syntax of the instruction operands to access the register (for example, the number used to access the XER register is SPR 1). These registers can be accessed using **mtspr** and **mfspr**. Note that not all registers in Figure 6-1 are SPRs; for example, VSCR and VRs are AltiVec registers and do not have an SPR number.

6.1.2 e600 Core Register Set

Table 6-1 summarizes the registers implemented in the e600 core.

Table 6-1. e600 Core Register Summary

Name	SPR	Description	Reference/ Section
UISA Registers			
CR	—	Condition register. The 32-bit CR consists of eight 4-bit fields, CR0–CR7, that reflect results of certain arithmetic operations and provide a mechanism for testing and branching.	PEM
CTR	9	Count register. Holds a loop count that can be decremented during execution of appropriately coded branch instructions. The CTR can also provide the branch target address for the Branch Conditional to Count Register (**bcctr***x*) instruction.	PEM
FPR0–FPR31	—	Floating-point registers (FPR*n*). The 32 FPRs serve as the data source or destination for all floating-point instructions.	PEM
FPSCR	—	Floating-point status and control register. Contains floating-point exception signal bits, exception summary bits, exception enable bits, and rounding control bits for compliance with the IEEE 754 standard.	PEM
GPR0–GPR31	—	General-purpose registers (GPR*n*). The thirty-two GPRs serve as data source or destination registers for integer instructions and provide data for generating addresses.	PEM
LR	8	Link register. Provides the branch target address for the Branch Conditional to Link Register (**bclr***x*) instruction, and can be used to hold the logical address of the instruction that follows a branch and link instruction, typically used for linking to subroutines.	PEM
UMMCR0 [1] UMMCR1 [1] UMMCR2 [1]	936 940 928	User monitor mode control registers (UMMCR*n*). Used to enable various performance monitor interrupt functions. UMMCRs provide user-level read access to MMCR registers.	6.1.6.9.2, e600RM, 6.1.6.9.4, 6.1.6.9.6,
UPMC1–UPMC6 [1]	937, 938 941, 942 929, 930	User performance monitor counter registers (UPMC*n*). Used to record the number of times a certain event has occurred. UPMCs provide user-level read access to PMC registers.	6.1.6.9.9, e600RM
USIAR [1]	939	User sampled instruction address register. Contains the effective address of an instruction executing at or around the time that the core signals the performance monitor interrupt. USIAR provides user-level read access to the SIAR.	6.1.6.9.11 e600RM
VR0–VR31 [2]	—	Vector registers (VR*n*). Data source and destination registers for all AltiVec instructions.	e600RM
VRSAVE [2]	256	Vector save/restore register. Defined by the AltiVec technology to assist application and operating system software in saving and restoring the architectural state across process context-switched events. The register is maintained only by software to track live or dead information on each AltiVec register.	e600RM
VSCR [2]	—	Vector status and control register. A 32-bit vector register that is read and written in a manner similar to the FPSCR.	e600RM

Table 6-1. e600 Core Register Summary (continued)

Name	SPR	Description	Reference/ Section
XER	1	Indicates overflows and carries for integer operations. **Implementation Note**—To emulate the **lscbx** instruction, XER[16–23] are be read with **mfspr**[XER] and written with **mtspr**[XER].	PEM
VEA			
TBL, TBU (For reading)	TBR 268 TBR 269	Time base facility. Consists of two 32-bit registers, time base lower and upper registers (TBL/TBU). TBL (TBR 268) and TBU (TBR 269) can only be read from and not written to.TBU and TBL can be read with the move from time base register (**mftb**) instruction. **Implementation Note**—Reading from SPR 284 or 285 using the **mftb** instruction causes an illegal instruction interrupt.	PEM, 6.1.5.1, 6.3.5.1
OEA			
BAMR [1]	951	Breakpoint address mask register. Used in conjunction with the events that monitor IABR hits. **Note:** See Table 6-37 for specific synchronization requirements on this register.	6.1.6.9.7 e600RM
DABR [3]	1013	Data address breakpoint register. Optional register implemented in the e600 core and is used to cause a breakpoint interrupt if a specified data address is encountered. See Table 6-37 for specific synchronization requirements on this register.	PEM
DAR	19	Data address register. After a DSI or alignment interrupt, DAR is set to the effective address (EA) generated by the faulting instruction.	PEM
DEC	22	Decrementer register. A 32-bit decrementer counter used with the decrementer interrupt. **Implementation Note**—In the e600 core, DEC is decremented and the time base increments at 1/4 of the MPX bus clock frequency.	PEM
DSISR	18	DSI source register. Defines the cause of DSI and alignment interrupts.	PEM
EAR	282	External access register. Used with **eciwx** and **ecowx**. Note that the EAR and the **eciwx** and **ecowx** instructions are optional in the PowerPC ISA. Since **eciwx** and **ecowx** are not implemented in the e600 core, the EAR must not be enabled. Otherwise, attempted execution of **eciwx**/**ecowx** causes boundedly undefined results. See Table 6-37 for specific synchronization requirements on this register.	PEM
HID0 [1] HID1 [1]	1008, 1009	Hardware implementation-dependent registers. Control various functions, such as the power management features, and locking, enabling, and invalidating the instruction and data caches. The HID1 includes bits that reflects the state of *pll_cfg*[0:5] clock signals and control other bus-related functions. See Table 6-37 for specific synchronization requirements on this register.	6.1.6.1, 6.1.6.2
IABR [1]	1010	Instruction address breakpoint register. Used to cause a breakpoint interrupt if a specified instruction address is encountered. See Table 6-37 for specific synchronization requirements on this register.	6.1.6.6

Table 6-1. e600 Core Register Summary (continued)

Name	SPR	Description	Reference/ Section
IBAT0U/L IBAT1U/L IBAT2U/L IBAT3U/L IBAT4U/L [1] IBAT5U/L [1] IBAT6U/L [1] IBAT7U/L [1] DBAT0U/L DBAT1U/L DBAT2U/L DBAT3U/L DBAT4U/L [1] DBAT5U/L [1] DBAT6U/L [1] DBAT7U/L [1]	528, 529 530, 531 532, 533 534, 535 560, 561 562, 563 564, 565 566, 567 536, 537 538, 539 540, 541 542, 543 568, 569 570, 571 572, 573 574, 575	Block-address translation (BAT) registers. The OEA includes an array of block address translation registers that can be used to specify four blocks of instruction space and four blocks of data space. The BAT registers are implemented in pairs: four pairs of instruction BATs (IBAT0U–IBAT3U and IBAT0L–IBAT3L) and four pairs of data BATs (DBAT0U–DBAT3U and DBAT0L–DBAT3L). Sixteen additional BAT registers are available for the e600 core. These registers are enabled by setting HID0[HIGH_BAT_EN]. When HID0[HIGH_BAT_EN] = 1, the 16 additional BAT registers, organized as 4 pairs of instruction BAT registers (IBAT4U–IBAT7U paired with IBAT4L–IBAT7L) and 4 pairs of data BAT registers (DBAT4U–DBAT7U paired with DBAT4L–DBAT7L) are available. Thus, the e600 can define a total of 16 blocks implemented as 32 BAT registers. Because BAT upper and lower words are loaded separately, software must ensure that BAT translations are correct during the time that both BAT entries are being loaded. The e600 core implements IBAT[G]; however, attempting to execute code from an IBAT area with G = 1 causes an ISI interrupt. See Table 6-37 for specific synchronization requirements on this register.	PEM e600RM
ICTC [1]	1019	Instruction cache throttling control register. Has bits for enabling instruction cache throttling and for controlling the interval at which instructions are fetched. This controls overall junction temperature.	6.1.6.8 e600RM
ICTRL [1]	1011	Instruction cache and interrupt control register. Used in configuring interrupts and error reporting for the instruction and data caches. See Table 6-37 for specific synchronization requirements on this register.	6.1.6.5.15
L2CR [1]	1017	L2 cache control register. Includes bits for enabling parity checking, setting the L2 cache size, and flushing and invalidating the L2 cache.	6.1.6.5.1
L2ERRINJHI [1] L2ERRINJLO [1] L2ERRINJCTL [1] L2CAPTDATAHI [1] L2CAPTDATALO [1] L2CAPTDATAECC [1] L2ERRDET [1] L2ERRDIS [1] L2ERRINTEN [1] L2ERRATTR [1] L2ERRADDR [1] L2ERREADDR [1] L2ERRCTL [1]	985 986 987 988 989 990 991 992 993 994 995 996 997	L2 error registers. The L2 cache supports error injection into the L2 data, data ECC or tag, which can be used to test error recovery software by deterministically creating error scenarios. L2ERRINJHI, L2ERRINJLO, and L2ERRINJCTL are error injection registers. The error control and capture registers control the detection and reporting of tag parity and ECC errors.	6.1.6.5.2 6.1.6.5.3 6.1.6.5.4 6.1.6.5.5 6.1.6.5.6 6.1.6.5.7 6.1.6.5.8 6.1.6.5.9 6.1.6.5.10 6.1.6.5.11 6.1.6.5.12 6.1.6.5.13 6.1.6.5.14
LDSTCR [1]	1016	Load/store control register. Controls data L1 cache way-locking. See Table 6-37 for specific synchronization requirements on this register.	6.1.6.5.16
MMCR0 [3] MMCR1 [3] MMCR2 [1]	952 956 944	Monitor mode control registers (MMCR*n*). Enable various performance monitor interrupt functions. UMMCR0–UMMCR2 provide user-level read access to these registers.	6.1.6.9.1 6.1.6.9.3 6.1.6.9.5

Table 6-1. e600 Core Register Summary (continued)

<table>
<tr><th>Name</th><th>SPR</th><th>Description</th><th>Reference/ Section</th></tr>
<tr><td>MSR</td><td>—</td><td>Machine state register. Defines the processor state. The MSR can be modified by the mtmsr, sc, and rfi instructions. It can be read by the mfmsr instruction. When an interrupt is taken, MSR contents are saved to SRR1. See the “Interrupts” chapter of the e600 PowerPC Core Reference Manual. The following bits are optional in the PowerPC ISA.
Note that setting MSR[EE] masks decrementer and external interrupts and e600-core-specific system management and performance monitor interrupts.
See Table 6-37 for specific synchronization requirements on this register.
<table>
<tr><th>Bit</th><th>Name</th><th>Description</th></tr>
<tr><td>6</td><td>VEC</td><td>AltiVec available. e600 core and AltiVec technology specific; optional to the PowerPC ISA.
0 AltiVec technology is disabled.
1 AltiVec technology is enabled.
Note: When a non-stream AltiVec instruction accesses VRs or the VSCR when VEC = 0 an AltiVec unavailable interrupt is generated. This does not occur for data streaming instructions (dst(t), dstst(t), and dss); the VRs and the VSCR are available to data streaming instructions even if VEC = 0. VRSAVE can be accessed even if VEC = 0.</td></tr>
<tr><td>13</td><td>POW</td><td>Power management enable. e600-core-specific and optional to the PowerPC ISA.
0 Power management is disabled.
1 Power management is enabled. The processor can enter a power-saving mode determined by HID0[NAP,SLEEP] when additional conditions are met. See Table 6-7.</td></tr>
<tr><td>29</td><td>PMM</td><td>Performance monitor marked mode. e600-core-specific and optional to the PowerPC ISA. See the “Performance Monitor” chapter of the e600 PowerPC Core Reference Manual.
0 Process is not a marked process.
1 Process is a marked process.</td></tr>
</table></td><td>PEM, 6.1.4.4, e600RM</td></tr>
<tr><td>MSSCR0 [1]</td><td>1014</td><td>Memory subsystem control register. Used to configure and operate many aspects of the core memory subsystem.</td><td>6.1.6.3</td></tr>
<tr><td>MSSSR0 [1]</td><td>1015</td><td>Memory subsystem status register. Used to configure and operate the parity functions in the L2 cache for the e600 core.
See Table 6-37 for specific synchronization requirements on this register.</td><td>6.1.6.4</td></tr>
<tr><td>PIR</td><td>1023</td><td>Processor identification register. Provided for system use. All 32 bits of the PIR can be written to with the mtspr instruction.</td><td>PEM, 6.1.4.3</td></tr>
<tr><td>PMC1–PMC6 [3]</td><td>953, 954
957, 958
945, 946</td><td>Performance monitor counter registers (PMCn). Used to record the number of times a certain event has occurred. UPMCs provide user-level read access to these registers.</td><td>6.1.6.9.8</td></tr>
<tr><td>PTEHI, PTELO</td><td>981, 982</td><td>The PTEHI and PTELO registers are used by the tlbld and tlbli instructions to create a TLB entry. When software table searching is enabled (HID0[STEN] = 1), and a TLB miss interrupt occurs, the bits of the page table entry (PTE) for this access are located by software and saved in the PTE registers.</td><td>6.1.6.7.2 e600RM</td></tr>
<tr><td>PVR</td><td>287</td><td>Processor version register. Read-only register that identifies the version (model) and revision level of the processor.</td><td>PEM 6.1.4.1</td></tr>
</table>

Table 6-1. e600 Core Register Summary (continued)

Name	SPR	Description	Reference/ Section
SDAR, USDAR	—	Sampled data address register. The e600 core does not implement the optional registers (SDAR or the user-level, read-only USDAR register) defined by the PowerPC ISA. Note that in previous processors the SDA and USDA registers could be written to by boot code without causing an interrupt. This is not the case in the e600 core. A **mtspr** or **mfspr** SDAR or USDAR instruction causes a program interrupt**.**	6.1.6.9.12
SDR1	25	Sample data register. Specifies the base address of the page table entry group (PTEG) address used in virtual-to-physical address translation. **Implementation Note**—The SDR1 register has been modified (with the SDR1[HTABEXT] and SDR1[HTMEXT] fields) for the e600 core to support the extended 36-bit physical address (when HID0[XAEN] = 1]). See Table 6-37 for specific synchronization requirements on this register.	PEM, 6.1.4.6 e600RM
SIAR [3]	955	Sampled instruction address register. Contains the effective address of an instruction executing at or around the time that the processor signals the performance monitor exception condition. USIAR provides user-level read access to the SIAR.	6.1.6.9.10 e600RM
SPRG0–SPRG3 SPRG4–SPRG7 [1]	272–275 276–279	SPRG*n*. Provided for operating system use. The SPRG4–7 provide additional registers to be used by system software for software table searching.	PEM e600RM
SR0–SR15	—	Segment registers (SR*n*). Note that the e600 core implements separate instruction and data MMUs. It associates architecture-defined SRs with the data MMU. It reflects SRs values in separate, shadow SRs in the instruction MMU. See Table 6-37 for specific synchronization requirements on this register.	PEM
SRR0 SRR1	26 27	Machine status save/restore registers (SRR*n*). Used to save the address of the instruction at which execution continues when **rfi** executes at the end of an interrupt handler routine. SRR1 is used to save machine status on interrupts and to restore machine status when **rfi** executes. **Implementation Note**—When a machine check interrupt occurs, the e600 core sets one or more error bits in SRR1. Refer to the individual interrupts for individual SRR1 bit settings.	PEM, 6.1.4.5 e600RM
SVR	286	System version register. Read-only register provided for future product compatibility.	6.1.4.2
TBL TBU (For writing)	284 285	Time base. A 64-bit structure (two 32-bit registers) that maintains the time of day and operating interval timers. The TB consists of two registers—time base upper (TBU) and time base lower (TBL). The time base registers can be written to only by supervisor-level software. TBL (SPR 284) and TBU (SPR 285) can only be written to and not read from. TBL and TBU can be written to, with the Move to Special Purpose Register (**mtspr**) instruction. **Implementation Note**—Reading from SPR 284 or 285 causes an illegal instruction interrupt. In the e600 core, DEC is decremented and the time base increments at 1/4 of the MPX bus clock frequency.	PEM, 6.1.5.1, 6.3.5.1
TLBMISS [1]	980	The TLBMISS register is automatically loaded when software searching is enabled (HID0[STEN] = 1) and a TLB miss interrupt occurs. Its contents are used by the TLB miss interrupt handlers (the software table search routines) to start the search process.	6.1.6.7.1 e600RM

1 e600-core-specific register that may not be supported on other cores or processors.

2 Register is defined by the AltiVec technology.

3 Defined as optional register in the PowerPC ISA.

6.1.3 User-Level Registers (UISA)

The UISA registers are user-level. General-purpose registers (GPRs), floating-point registers (FPRs) and vector registers (VRs) are accessed through instruction operands. Access to registers can be explicit (by using instructions for that purpose such as Move to Special-Purpose Register (**mtspr**) and Move from Special-Purpose Register (**mfspr**) instructions, or implicit as part of the execution of an instruction. Some registers are accessed both explicitly and implicitly.

Implementation Note—The e600 core fully decodes the SPR field of the instruction. If the SPR specified is undefined, an illegal instruction program interrupt occurs.

6.1.4 Supervisor-Level Registers (OEA)

The OEA defines the registers and operating system uses for memory management, configuration, interrupt handling, and other operating system functions as summarized in Table 6-1. The following supervisor-level register defined by the PowerPC ISA contains additional implementation-specific information for the e600 core.

6.1.4.1 Processor Version Register (PVR)

The PVR identifies the version (model) and revision level of the processor. For more information, see "Processor Version Register (PVR)," in Chapter 2, "PowerPC Register Set," of the *Programming Environments Manual*. Table 6-2 shows the PVR settings. The revision level is updated for each silicon revision.

Table 6-2. PVR Settings

Device Name	Version No.	Starting Processor Revision Level
MPC7448 1.0	0x8004	0100
MPC8641	0x8004	0010
MPC8641D	0x8004	0010

Table 6-3 describes the e600 core PVR bits that are not required by the PowerPC ISA.

Table 6-3. Additional PVR Bits

Bits	Name	Description
0–15	Type	Processor type
16–19	Tech	Processor technology
20–23	Major	Major revision number
24–31	Minor	Minor revision number

6.1.4.2 System Version Register (SVR)

See Section 17.4.1.14, "System Version Register (SVR)," for the specific values of the SVR on the MPC8641 and MPC8641D.

6.1.4.3 Processor Identification Register (PIR)

For more information, see "Processor Identification Register (PIR)," in Chapter 2, "PowerPC Register Set," of the *Programming Environments Manual*.

Implementation Note—The e600 core provides write access to the PIR with **mtspr** using SPR 1023.

6.1.4.4 Machine State Register (MSR)

The MSR defines the state of the processor. When an interrupt occurs, MSR bits, as described in Table 6-4 are altered as determined by the interrupts. The MSR can also be modified by the **mtmsr**, **sc**, and **rfi** instructions. It can be read by the **mfmsr** instruction.

The MSR is shown in Figure 6-2.

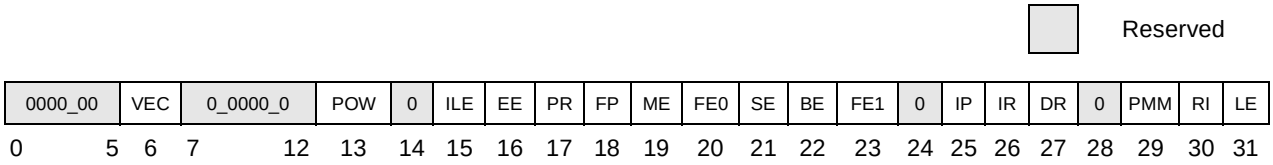


Figure 6-2. Machine State Register (MSR)

The MSR bits are defined in Table 6-4.

Table 6-4. MSR Bit Settings

Bits	Name	Description
0–5	—	Reserved
6	VEC [1, 2]	AltiVec vector unit available 0 The processor prevents dispatch of AltiVec instructions (excluding the data streaming instructions—**dst**, **dstt**, **dstst**, **dststt**, **dss**, and **dssall**). The processor also prevents access to the vector register file (VRF) and the vector status and control register (VSCR). Any attempt to execute an AltiVec instruction that accesses the VRF or VSCR, excluding the data streaming instructions generates the AltiVec unavailable interrupt. The data streaming instructions are not affected by this bit; the VRF and VSCR registers are available to the data streaming instructions even when the MSR[VEC] is cleared. 1 The processor can execute AltiVec instructions and the VRF and VSCR registers are accessible to all AltiVec instructions. Note that the VRSAVE register is not protected by MSR[VEC].
7–12	—	Reserved
13	POW [1, 3]	Power management enable 0 Power management disabled (normal operation mode) 1 Power management enabled (reduced power mode) Power management functions are implementation-dependent. See the "Power and Thermal Management" chapter of the *e600 PowerPC Core Reference Manual*.

Table 6-4. MSR Bit Settings (continued)

Bits	Name	Description
14	—	Reserved. Implementation-specific
15	ILE	Interrupt little-endian mode. When an interrupt occurs, this bit is copied into MSR[LE] to select the endian mode for the context established by the interrupt.
16	EE	External interrupt enable 0 The processor delays recognition of interrupts external to the core and decrementer interrupt conditions. 1 The processor is enabled to take an interrupt external to the core or the decrementer interrupt.
17	PR [4]	Privilege level 0 The processor can execute both user- and supervisor-level instructions. 1 The processor can only execute user-level instructions.
18	FP [2]	Floating-point available 0 The processor prevents dispatch of floating-point instructions, including floating-point loads, stores, and moves. 1 The processor can execute floating-point instructions and can take floating-point enabled program interrupts.
19	ME	Machine check enable 0 Machine check interrupts are disabled. 1 Machine check interrupts are enabled.
20	FE0 [2]	IEEE Std. 754 floating-point exception mode 0 (see Table 6-5)
21	SE	Single-step trace enable 0 The processor executes instructions normally. 1 The processor generates a single-step trace interrupt upon the successful execution of every instruction except **rfi** and **sc**. Successful execution means that the instruction caused no other interrupt.
22	BE	Branch trace enable 0 The processor executes branch instructions normally. 1 The processor generates a branch type trace interrupt when a branch instruction executes successfully.
23	FE1 [2]	IEEE Std. 754 floating-point exception mode 1 (see Table 6-5)
24	—	Reserved. This bit corresponds to the AL bit of the architecture.
25	IP	Interrupt prefix. The setting of this bit specifies whether an interrupt vector offset is prepended with Fs or 0s. In the following description, *nnnnn* is the offset of the interrupt. 0 Interrupts are vectored to the physical address 0x000*n_nnnn*. 1 Interrupts are vectored to the physical address 0xFFF*n_nnnn*.
26	IR [5]	Instruction address translation 0 Instruction address translation is disabled. 1 Instruction address translation is enabled. For more information see the “Memory Management” chapter of the *e600 PowerPC Core Reference Manual*.
27	DR [4]	Data address translation 0 Data address translation is disabled. 1 Data address translation is enabled. For more information see the “Memory Management” chapter of the *e600 PowerPC Core Reference Manual*.
28	—	Reserved

Table 6-4. MSR Bit Settings (continued)

Bits	Name	Description
29	PMM [1]	Performance monitor marked mode 0 Process is not a marked process. 1 Process is a marked process. This bit can be set when statistics need to be gathered on a specific (marked) process. The statistics will only be gathered when the marked process is executing. e600-core-specific; defined as optional by the PowerPC architecture. For more information about the performance monitor marked mode bit, see the “Performance Monitor” chapter of the *e600 PowerPC Core Reference Manual*.
30	RI	Indicates whether system reset or machine check interrupt is recoverable. indicates whether from the perspective of the processor, it is safe to continue (that is, processor state data such as that saved to SRR0 is valid), but it does not guarantee that the interrupted process is recoverable. 0 Interrupt is not recoverable. 1 Interrupt is recoverable.
31	LE [6]	Little-endian mode enable 0 The processor runs in big-endian mode. 1 The processor runs in little-endian mode.

1 Optional to the PowerPC architecture.

2 A context synchronizing instruction must follow an mtmsr instruction.

3 A dssall and sync must precede an mtmsr instruction and then a context synchronizing instruction must follow.

4 A dssall and sync must precede an mtmsr and then a sync and context synchronizing instruction must follow. Note that if a user is not using the AltiVec data streaming instructions, then a dssall is not necessary prior to accessing the MSR[DR] or MSR[PR] bit.

5 A context synchronizing instruction must follow an mtmsr. When changing the MSR[IR] bit the context synchronizing instruction must reside at both the untranslated and the translated address following the mtmsr.

6 A dssall and sync must precede an rfi to guarantee a solid context boundary. Note that if a user is not using AltiVec data streaming instructions, a dssall is not necessary before accessing MSR[LE].

Note that setting MSR[EE] masks not only the architecture-defined interrupt and decrementer interrupts external to the core but also the e600-specific system management, and performance monitor interrupts.

The IEEE Std. 754 floating-point exception mode bits (FE0 and FE1) together define whether floating-point exceptions are handled precisely, imprecisely, or whether they are taken at all. As shown in Table 6-5, if either FE0 or FE1 is set, the e600 core treats interrupts as precise. MSR bits are guaranteed to be written to SRR1 when the first instruction of the interrupt handler is encountered. For further details, see Chapter 2, “Register Set” and Chapter 6, “Interrupts,” of the *Programming Environments Manual*.

Table 6-5. IEEE Std. 754 Floating-Point Exception Mode Bits

FE0	FE1	Mode
0	0	Floating-point exceptions disabled
0	1	Imprecise nonrecoverable. For this setting, the e600 core operates in floating-point precise mode.
1	0	Imprecise recoverable. For this setting, the e600 core operates in floating-point precise mode.
1	1	Floating-point precise mode

6.1.4.5 Machine Status Save/Restore Registers (SRR0, SRR1)

When an interrupt is taken, the processor uses SRR0 and SRR1 to save the contents of the MSR for the current context and to identify where instruction execution should resume after the interrupt is handled.

When an interrupt occurs, the address saved in SRR0 helps determine where instruction processing should resume when the interrupt handler returns control to the interrupted process. Depending on the interrupt, this may be the address in SRR0 or at the next address in the program flow. All instructions in the program flow preceding this one will have completed execution and no subsequent instruction will have begun execution. This may be the address of the instruction that caused the interrupt or the next one (as in the case of a system call or trace interrupt). The SRR0 register is shown in Figure 6-3.

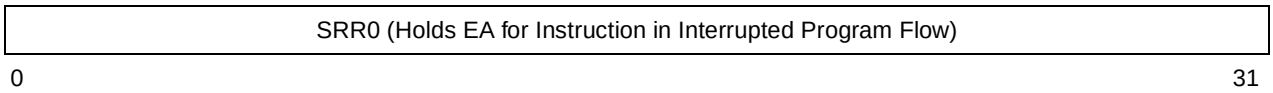


Figure 6-3. Machine Status Save/Restore Register 0 (SRR0)

SRR1 is used to save machine status (selected MSR bits and possibly other status bits) on interrupts and to restore those values when an **rfi** instruction is executed. SRR1 is shown in Figure 6-4.

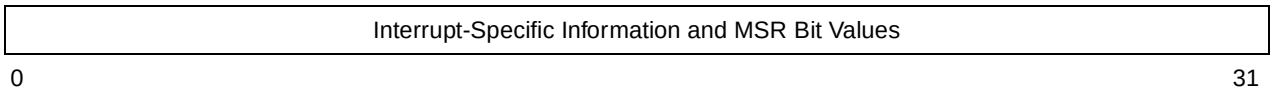


Figure 6-4. Machine Status Save/Restore Register 1 (SRR1)

Typically, when an interrupt occurs, SRR1[0–15] are loaded with interrupt-specific information and MSR[16–31] are placed into the corresponding bit positions of SRR1. For most interrupts, SRR1[0–5] and SRR1[7–15] are cleared, and MSR[6, 16–31] are placed into the corresponding bit positions of SRR1. Table 6-4 provides a summary of the SRR1 bit settings when a machine check interrupt occurs. For a specific interrupt's SRR1 bit settings, see the "Interrupts" chapter of the *e600 PowerPC Core Reference Manual*.

6.1.4.6 SDR1 Register

The SDR1 register specifies the page table entry group (PTEG) address used in virtual-to-physical address translation. See "SDR1," in Chapter 2, "PowerPC Register Set," of the *Programming Environments Manual* for the description with a 32-bit physical address. The SDR1 register has been modified for the e600 core to support the extended 36-bit physical address (when HID0[XAEN] = 1]). See the "Memory Management" chapter of the *e600 PowerPC Core Reference Manual* for details on how SDR1 is modified to support a 36-bit physical address.

Implementation Note—SDR1[HTABEXT] and SDR1[HTMEXT] fields have been added to support extended addressing. The "Memory Management" chapter of the *e600 PowerPC Core Reference Manual* describes in detail the differences when generating a 36-bit PTEG address. Figure 6-5 shows the format of the modified SDR1.

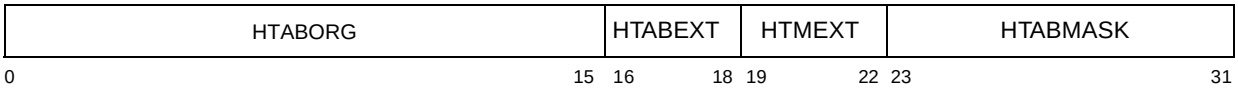


Figure 6-5. SDR1 Register Format—Extended Addressing

Bit settings for the SDR1 register are described in Table 6-6.

Table 6-6. SDR1 Register Bit Settings—Extended Addressing

Bits	Name	Description
0–15	HTABORG	Physical base address of page table If HID0[XAEN] = 1, field contains physical address [4–19] If HID0[XAEN] = 0, field contains physical address [0–15]
16–18	HTABEXT	Extension bits for physical base address of page table If HID0[XAEN] = 1, field contains physical address [1–3] If HID0[XAEN] = 0, field is reserved
19–22	HTMEXT	Hash table mask extension bits If HID0[XAEN] = 1, field contains hash table mask [0–3] If HID0[XAEN] = 0, field is reserved
23–31	HTABMASK	Mask for page table address If HID0[XAEN] = 1, field contains hash table mask [4–12] If HID0[XAEN] = 0, field contains hash table mask [0–7]

SDR1 can be accessed with **mtspr** and **mfspr** using SPR 25. For synchronization requirements on accesses to the register see Section 6.3.2.4, “Synchronization.”

6.1.5 User-Level Registers (VEA)

The VEA defines the time base facility (TB), which consists of two 32-bit registers—time base upper (TBU) and time base lower (TBL).

6.1.5.1 Time Base Registers (TBL, TBU)

The time base registers can be written only by supervisor-level instructions but can be read by both user- and supervisor-level software. The time base registers have two different addresses. TBU and TBL can be read from the TBR 268 and 269, respectively, with the move from time base register (**mftb**) instruction. TBU and TBL can be written to TBR 284 and 285, respectively, with the move to special purpose register (**mtspr**) instruction. Reading from SPR 284 or 285 causes an illegal instruction interrupt. For more information, see “PowerPC VEA Register Set—Time Base,” in Chapter 2, “PowerPC Register Set,” of the *Programming Environments Manual*. Note that DEC is decremented and the time base increments at 1/4 of the MPX bus clock frequency.

6.1.6 e600-Core-Specific Register Descriptions

The PowerPC ISA allows for implementation-specific SPRs. This section describes registers that are defined for the e600 core but are not included in the PowerPC ISA. Note that in the e600 core, these

registers are all supervisor-level registers. All the registers described in the *AltiVec Technology Programming Environments Manual* are implemented in e600 core. See Chapter 2, “AltiVec Register Set,” in the *AltiVec Technology Programming Environments Manual* for details about these registers.

Note that while it is not guaranteed that the implementation of e600-core-specific registers is consistent among processors that implement the PowerPC ISA, other processors can implement similar or identical registers.

The registers in the following subsections are presented in the order of the chapters in this book. First, the processor control registers are described followed by the cache control registers. Next, the implementation-specific registers for interrupt processing and memory management are presented, followed by the thermal management register. Finally, the performance monitor registers are presented.

6.1.6.1 Hardware Implementation-Dependent Register 0 (HID0)

The hardware implementation-dependent register 0 (HID0) controls the state of several functions within the e600 core. The HID0 register is shown in Figure 6-6.

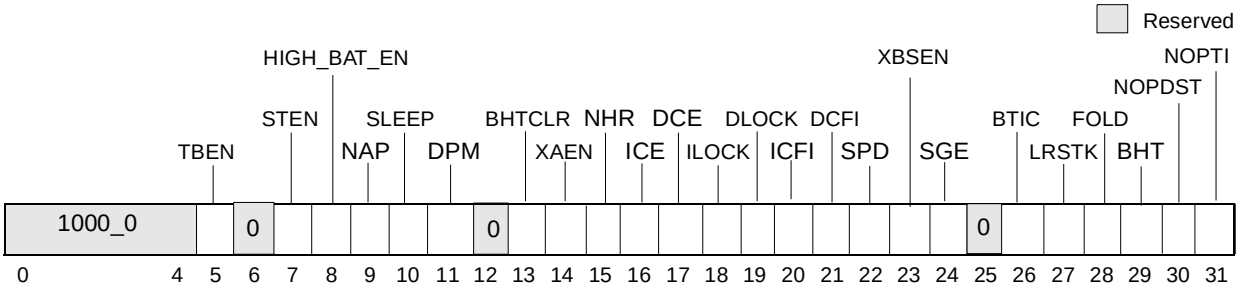


Figure 6-6. Hardware Implementation-Dependent Register 0 (HID0)

The HID0 bits are described in Table 6-7.

Table 6-7. HID0 Field Descriptions

Bits	Name	Description
0–4	—	Reserved. Read as 0b1000_0.
5	TBEN [1]	Time base enable. Note that this bit must be set and the *tben* signal must be asserted to enable the time base and decrementer.
6	—	Reserved.
7	STEN[2]	Software table search enable. When a TLB miss occurs, the e600 core takes one of three TLB miss interrupts so that software can search the page tables for the desired PTE. See the “Interrupts” chapter of the *e600 PowerPC Core Reference Manual* for details on the e600 core facilities for software table searching. 0 Hardware table search enabled 1 Software tables search enabled
8	HIGH_BAT_EN	Additional BATs enabled for the e600 core 0 Additional 4 IBATs (4–7) and 4 DBATs (4–7) disabled 1 Additional 4 IBATs (4–7) and 4 DBATs (4–7) enabled The additional BATs provide for more mapping of memory with the block address translation method.

Table 6-7. HID0 Field Descriptions (continued)

Bits	Name	Description
9	NAP [1]	Nap mode enable. Operates in conjunction with MSR[POW]. 0 Nap mode disabled. 1 Nap mode enabled. Nap mode is invoked by setting MSR[POW] while this bit is set. In nap mode, the PLL and the time base remain active. Note that if both NAP and SLEEP are set, the e600 core ignores the SLEEP bit.
10	SLEEP [1]	Sleep mode enable. Operates in conjunction with MSR[POW]. 0 Sleep mode disabled. 1 Sleep mode enabled. Sleep mode is invoked by setting MSR[POW] while this bit is set. $\overline{qreq}$ is asserted to indicate that the processor is ready to enter sleep mode. If the system logic determines that the core can enter sleep mode, the quiesce acknowledge signal, $\overline{qack}$, is asserted back to the core. When the $\overline{qack}$ signal assertion is detected, the core enters sleep mode after several core clocks. At this point, the system logic can turn off the PLL by first configuring *pll_cfg*[0:5] to PLL bypass mode, and then disabling *sysclk*.
11	DPM [1]	Dynamic power management enable 0 Dynamic power management is disabled. 1 Functional units enter a low-power mode automatically if the unit is idle. This does not affect operational performance and is transparent to software or any hardware external to the core.
12	—	Reserved. For test use; software should not set this bit.
13	BHTCLR [3]	Clear branch history table 0 The e600 core clears this bit one cycle after it is set. 1 Setting BHTCLR bit initializes all entries in BHT to weakly, not taken whether or not the BHT is enabled by HID0[BHT]. However, for correct results, the BHT should be disabled (HID0[BHT] = 0) before setting BHTCLR. Setting BHTCLR causes the branch unit to be busy for 64 cycles while the initialization process is completed.
14	XAEN [4]	Extended addressing enabled 0 Extended addressing is disabled; the 4 most significant bits of the 36-bit physical address are cleared and a 32-bit physical address is used. 1 Extended addressing is enabled; the 32-bit effective address is translated to a 36-bit physical address. If HID0[XAEN] is changed (cleared or set), the BATs and TLBs must be invalidated first.
15	NHR [1]	Not hard reset (software-use only). Helps software distinguish a hard reset from a soft reset. 0 A hard reset occurred if software had previously set this bit. 1 A hard reset has not occurred. If software sets this bit after a hard reset, when a reset occurs and this bit remains set, software knows it was a soft reset. The e600 core never writes this bit unless executing an **mtspr**(HID0).
16	ICE [5]	Instruction cache enable 0 The instruction cache is neither accessed nor updated. All pages are accessed as if they were marked cache-inhibited (WIMG = x1xx). Potential cache accesses from the MPX bus (snoop and cache operations) are ignored. In the disabled state for the L1 caches, the cache tag state bits are ignored and all accesses are propagated to the MPX bus as burst transactions. For those transactions, $\overline{ci}$ is asserted regardless of address translation. ICE is zero at power-up. 1 The instruction cache is enabled. Note that HID0[ICFI] must be set at the same time that this bit is set.

Table 6-7. HID0 Field Descriptions (continued)

Bits	Name	Description
17	DCE [2]	Data cache enable 0 The data cache is neither accessed nor updated. All pages are accessed as if they were marked cache-inhibited (WIMG = x1xx). Potential cache accesses from the MPX bus (snoop and cache operations) are ignored. In the disabled state for the L1 caches, the cache tag state bits are ignored and all accesses are propagated to the L2 cache or MPX bus as cache-inhibited. For those transactions, $\overline{ci}$ is asserted regardless of address translation. DCE is zero at power-up. 1 The data cache is enabled. Note that HID0[DCFI] must be set at the same time that this bit is set.
18	ILOCK [6]	Instruction cache lock 0 Normal operation 1 All of the ways of the instruction cache are locked. A locked cache supplies data normally on a read hit. On a miss, the access is treated the same as if the instruction cache was disabled.Thus, the MPX bus request is a 32-byte burst read, but the cache is not loaded with data. The data is reloaded into the L2, unless the L2CR[L2DO] bit is set. Note that setting this bit has the same effect as setting ICTRL[ICWL] to all ones. However, when this bit is set, ICTRL[ICWL] is ignored. the “L1 and L2 Cache Operation” chapter of the *e600 PowerPC Core Reference Manual* gives further details.
19	DLOCK [2]	Data cache lock 0 Normal operation 1 All the ways of the data cache are locked. A locked cache supplies data normally on a read hit but is treated as a cache-inhibited transaction on a miss. On a miss, a load transaction still reads a full cache line from the L2 or MPX bus but does not reload that line into the L1. Any store miss is treated like a write-through store and the transaction occurs on the MPX bus with the $\overline{wt}$ signal asserted. A snoop hit to a locked L1 data cache operates as if the cache were not locked. A cache block invalidated by a snoop remains invalid until the cache is unlocked. Note that setting this bit has the same effect as setting LDSTCR[DCWL] to all ones. However, when this bit is set, LDSTCR[DCWL] is ignored. Refer to the “L1 and L2 Cache Operation” chapter of the *e600 PowerPC Core Reference Manual* for further details. To prevent locking during a cache access, a **sync** instruction must precede the setting of DLOCK and a **sync** must follow.
20	ICFI [5]	Instruction cache flash invalidate 0 The instruction cache is not invalidated. The bit is cleared when the invalidation operation begins (the next cycle after the write operation to the register). The instruction cache must be enabled for the invalidation to occur. 1 An invalidate operation is issued that marks the state of each instruction cache block as invalid. Cache access is blocked during this time. Setting ICFI clears all the valid bits of the blocks and sets the PLRU bits to point to way L0 of each set. When the L1 flash invalidate bits are set through an **mtspr** operation, the hardware automatically clears these bits in the next cycle (provided that the corresponding cache enable bits are set in HID0). Note that in the MPC603e processors, the proper use of the ICFI and DCFI bits was to set and clear them in two consecutive **mtspr** operations. Software that already has this sequence of operations does not need to be changed to run on the e600 core.

Table 6-7. HID0 Field Descriptions (continued)

Bits	Name	Description
21	DCFI [2]	Data cache flash invalidate 0 The data cache is not invalidated. The bit is cleared when the invalidation operation begins (the next cycle after the write operation to the register). 1 An invalidate operation is issued that marks the state of each data cache block as invalid without writing back modified cache blocks to memory. Cache access is blocked during this time. MPX bus accesses to the cache are signaled as a miss during invalidate-all operations. Setting DCFI clears all the valid bits of the blocks and the PLRU bits to point to way L0 of each set. When the L1 flash invalidate bits are set through an **mtspr** operation, the hardware automatically clears these bits in the next cycle. Note that setting DCFI invalidates the data cache regardless of whether it is enabled. Note that in the MPC603e processors, the proper use of the ICFI and DCFI bits was to set them and clear them in two consecutive **mtspr** operations. Software that already has this sequence of operations does not need to be changed to run on the e600 core.
22	SPD [1]	Speculative data cache and instruction cache access disable 0 Speculative bus accesses to nonguarded space (G = 0) from both the instruction and data caches is enabled. 1 Speculative bus accesses to nonguarded space in both caches is disabled. Thus, setting this bit prevents L1 data cache misses from going to the core memory subsystem until the instruction that caused the miss is next to complete. The HID0[SPD] bit also prevents instruction cache misses from going to the core memory subsystem until there are no unresolved branches. For more information on this bit and its effect on re-ordering of loads and stores, see the “L1 and L2 Cache Operation” chapter of the *e600 PowerPC Core Reference Manual.*
23	XBSEN	Extended BAT block size enable. 0 Disables IBAT*n*U[XBL] and DBAT*n*U[XBL] bits and clears these bits to zero. 1 Enables IBAT*n*U[XBL] and DBAT*n*U[XBL] bits. BATnU[15–18] become the 4 MSBs of the extended 15-bit BL field (BATnU[15–29]). This allows for extended BAT block sizes of 512 Mbytes, 1 Gbyte, 2 Gbytes, and 4 Gbytes. If HID0[XBSEN] is set at startup and then cleared after startup, the XBL bits will not clear but stay the same as they were set at startup. HID0[XBSEN] should be set once at startup and once set should not be cleared. When HID0[XBSEN] is set at startup, and then HID0[XBSEN] is cleared, the IBAT*n*U[XBL] and DBAT*n*U[XBL] bits are not cleared but stay the same as what was set at startup. If backwards compatibility with previous processors is a concern, then HID0[XBSEN] should stay cleared so that the XBL bits are treated as zeros. This allows the BAT translation to have a maximum block length of 256 Mbytes.
24	SGE [7]	Store gathering enable 0 Store gathering is disabled. 1 Integer store gathering is performed as described in the “L1 and L2 Cache Operation” and “Instruction Timing” chapters of the *e600 PowerPC Core Reference Manual*.
25	—	Reserved. Defined as DCFA on some earlier processors.
26	BTIC [1]	Branch target instruction cache enable. Used to enable use of the 128-entry branch instruction cache. 0 The BTIC contents are invalidated and the BTIC behaves as if it were empty. New entries cannot be added until the BTIC is enabled. 1 The BTIC is enabled and new entries can be added. The BTIC is flushed by context synchronization, which is required after a move to HID0. Thus, if the synchronization rules are followed, modifying this BTIC bit implicitly flushes the BTIC. See the “Instruction Timing” chapter of the *e600 PowerPC Core Reference Manual* for further details.

Table 6-7. HID0 Field Descriptions (continued)

Bits	Name	Description
27	LRSTK [1]	Link register stack enable 0 Link register prediction is disabled. 1 Allows **bclr** and **bclrl** instructions to predict the branch target address using the link register stack which can accelerate returns from subroutines. See the “Instruction Timing” chapter of the *e600 PowerPC Core Reference Manual* for further details.
28	FOLD [1]	Branch folding enable 0 Branch folding is disabled. All branches are dispatched to the completion queue. 1 Branch folding is enabled, allowing branches to be folded out of the instruction prefetch stream before dispatch. The e600 core attempts to fold branches that do not modify the link and or count register. Note that a branch that is taken or predicted as taken is folded regardless of where it is in the IQ; however, a branch that is predicted as not taken must be dispatched (cannot be fall-through folded) if it is in one of the dispatch entries (IQ0–IQ2) the cycle after it is decoded. See the “Instruction Timing” chapter of the *e600 PowerPC Core Reference Manual* for further details.
29	BHT [1]	Branch history table enable 0 BHT disabled. The e600 core uses static branch prediction as defined by the PowerPC architecture (UISA) for those branch instructions the BHT would have otherwise used to predict (that is, those that use the CR or CTR mechanism to determine direction). For more information on static branch prediction, see “Conditional Branch Control,” in Chapter 4 of the *Programming Environments Manual*. 1 Allows the use of dynamic prediction in the 2048-entry branch history table (BHT). The BHT is disabled at power-on reset. All entries are set to weakly, not-taken.
30	NOPDST [2]	No-op **dst**, **dstt**, **dstst**, and **dststt** instructions 0 The **dst**, **dstt**, **dstst**, and **dststt** instructions are enabled. 1 The **dst**, **dstt**, **dstst**, and **dststt** instructions are no-oped globally, and all previously executed **dst** streams are cancelled.
31	NOPTI [7]	No-op the data cache touch instructions 0 The **dcbt** and **dcbtst** instructions are enabled. 1 The **dcbt** and **dcbtst** instructions are no-oped globally.

1 A context synchronizing instruction must follow the mtspr.

2 A dssall and sync must precede an mtspr and then a sync and context synchronizing instruction must follow. Note that if a user is not using the AltiVec data streaming instructions, then a dssall is not necessary prior to accessing the HID0[DCE] or HID0[DCFI] bit.

3 A context synchronizing instruction must precede an mtspr and a branch instruction should follow. The branch instruction may be either conditional or unconditional. It ensures that all subsequent branch instructions see the newly initialized BHT values. For correct results, the BHT should be disabled (HID0[BHT] = 0) before setting BHTCLR.

4 A dssall and sync must precede an mtspr and then a sync and a context-synchronizing instruction must follow. Alteration of HID0[XAEN] must be done with caches and translation disabled. The caches and TLBs must be flushed before they are re-enabled after the XAEN bit is altered. Note that if a user is not using the AltiVec data streaming instructions, then a dssall is not necessary prior to accessing the HID0[XAEN] bit.

5 A context synchronizing instruction must immediately follow an mtspr. An mtspr instruction for HID0 should not modify either of these bits at the same time it modifies another bit that requires additional synchronization.

6 A context synchronizing instruction must precede and follow an mtspr.

7 An mtspr must follow a sync and a context synchronizing instruction.

HID0 can be accessed with **mtspr** and **mfspr** using SPR 1008. All **mtspr** instructions should be followed by a context synchronization instruction such as **isync**; for specific details see Section 6.3.2.4, “Synchronization.”

6.1.6.2 Hardware Implementation-Dependent Register 1 (HID1)

The hardware implementation-dependent register 1 (HID1) reflects the state of the *pll_cfg*[0:5] signals and controls other functions. At reset, PC bits 0–5 reflect the values on the *pll_cfg*[0:5] pins. The HID1 bits are shown in Figure 6-7.

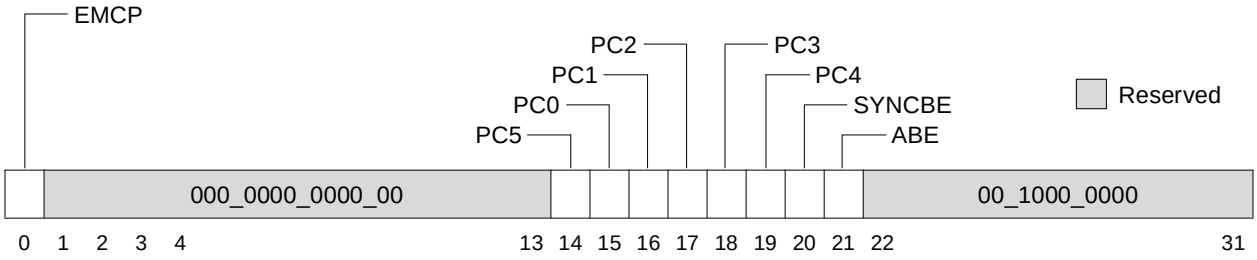


Figure 6-7. Hardware Implementation-Dependent Register 1 (HID1)

The HID1 bits are described in Table 6-8.

Table 6-8. HID1 Field Descriptions

Bits[1]	Name	Description
0	EMCP	Machine check signal enable 0 Machine check is disabled. 1 Machine check input signal (*mcp*) is enabled to cause machine check errors or checkstops.
1–13	—	Reserved
14	PC5	PLL configuration bit 5 (read-only). Reflects the state of the PLL multiplier.
15	PC0	PLL configuration bit 0 (read-only). Reflects the state of the PLL multiplier.
16	PC1	PLL configuration bit 1 (read-only). Reflects the state of the PLL multiplier.
17	PC2	PLL configuration bit 2 (read-only). Reflects the state of the PLL multiplier.
18	PC3	PLL configuration bit 3 (read-only). Reflects the state of the PLL multiplier.
19	PC4	PLL configuration bit 4 (read-only). Reflects the state of the PLL multiplier.
20	SYNCBE	Address broadcast enable for **sync**, **eieio** 0 Address broadcasting of **sync**, and **eieio** is disabled. 1 Address broadcasting of **sync**, and **eieio** is enabled. Note this bit must be set in MP systems and systems that reorder stores.
21	ABE	Address broadcast enable for **dcbf**, **dcbst**, **dcbi**, **icbi**, **tlbie**, and **tlbsync**. 0 Address broadcasting of **dcbf**, **dcbst**, **dcbi**, **icbi**, **tlbie**, and **tlbsync** is disabled. Note that when HID1[ABE] is cleared this does not exclude all cache operations from the MPX bus—just **icbi**, **tlbie**, and **tlbsync**. 1 Address broadcasting for cache control operations (**dcbf**, **dcbst**, **dcbi**, **icbi**) and TLB control operations (**tlbie** and **tlbsync**) is enabled. Note that whether the broadcast occurs depends on the setting of the M bit of WIMG and whether the access causes a hit to modified memory. See the “L1 and L2 Cache Operation” chapter of the *e600 PowerPC Core Reference Manual* for more information on broadcast operations. The ABE bit must be set for MP systems.
22–31	—	Reserved. Read as 0b00_1000_0000.

[1] A sync and context synchronizing instruction must follow an mtspr.

HID1 can be accessed with **mtspr** and **mfspr** using SPR 1009. All **mtspr** instructions should be followed by a **sync** and context synchronization instruction; for specific details, see Section 6.3.2.4, "Synchronization."

6.1.6.3 Memory Subsystem Control Register (MSSCR0)

The memory subsystem control register (MSSCR0), shown in Figure 6-8, is used to configure and operate the memory subsystem for the e600 core. It is accessed as SPR 1014. The MSSCR0 is initialized to all zeros except for the read-only bits.

Because MSSCR0 alters how the e600 responds to snoop requests, it is important that changes to the values of the fields in MSSCR0 are handled correctly.

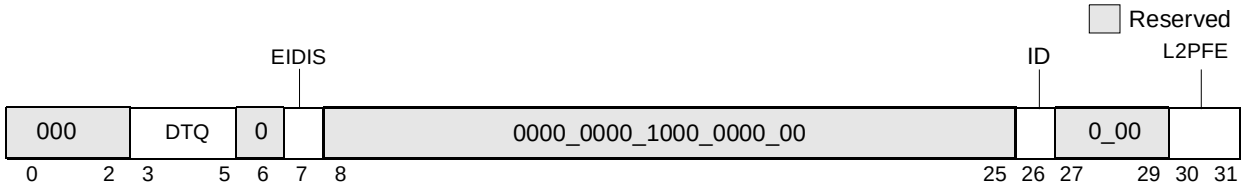


Figure 6-8. Memory Subsystem Control Register (MSSCR0)

Table 6-9 describes the MSSCR0 fields.

Table 6-9. MSSCR0 Field Descriptions

Bits	Name	Function
0–2	—	Reserved
3–5	DTQ	DTQ size. Determines the maximum number of outstanding data bus transactions that the e600 core can support. The DTQ bit values are as follows: 000 8 entries 001 16 entries 010 2 entries 011 3 entries 100 4 entries 101 5 entries 110 6 entries 111 7 entries The value of this field must match what the integrated device supports.
6	—	Reserved
7	EIDIS	Disable external intervention 0 Interventions external to the core occur. 1 The e600 core performs external pushes instead of external interventions. External interventions are disabled.
8–15	—	Reserved

Table 6-9. MSSCR0 Field Descriptions (continued)

Bits	Name	Function
16–17	BMODE	Bus mode (read-only). Indicates whether the core interface uses the 60x or MPX bus protocol as described in the “Core Interface” chapter of the *e600 PowerPC Core Reference Manual*. 00 Reserved 01 Reserved 10 MPX bus mode 11 Reserved
18–25	—	Reserved. Normally cleared. Used in debug. Writing non-zero values may cause boundedly undefined results.
26	ID	Processor identification. Sets the processor ID to either processor 0 or 1. Determined by the integration logic. Software can then find processor 0 and use it to re-identify the other processors by writing unique values to the PIR of the other CPUs.
27–29	—	Reserved. Read as zeros.
30–31	L2PFE	L2 prefetching enabled. The following values determine the number of L2 prefetch engines enabled as follows: 00 L2 prefetching disabled, no prefetch engines 01 One prefetch engine enabled 10 Two prefetch engines enabled 11 Three prefetch engines enabled These bits enable alternate sector prefetching in the 2-sectored L2 cache; up to 3 outstanding prefetch engines may be active. See the integrated device reference manual for specific recommendations.

6.1.6.4 Memory Subsystem Status Register (MSSSR0)

The memory subsystem status register (MSSSR0), shown in Figure 6-9, is used to report parity in the L2 cache and MSS enabled error status. It is accessed as SPR 1015. The MSSSR0 is initialized to all 0s except for the read-only bits.

Note that tag and data parity and data ECC errors are reported in the error detect (L2ERRDET) register whether or not error reporting is enabled. The corresponding bit in MSSSR0 is set only if error reporting is enabled.

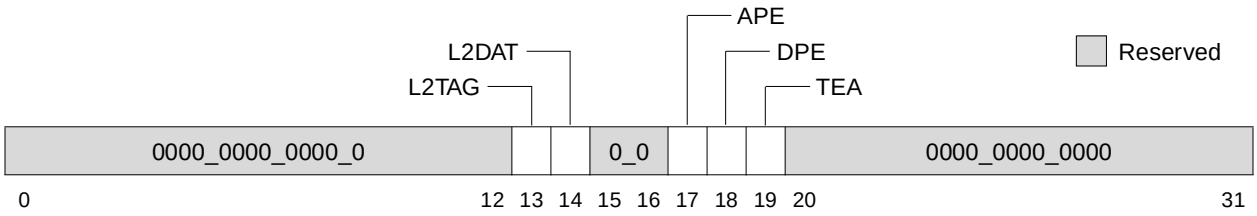


Figure 6-9. MSS Status Register (MSSSR0)

Table 6-10 describes MSSSR0 fields.

Table 6-10. MSSSR0 Field Descriptions

Bits	Name	Description
0–12	—	Reserved. Normally cleared. Used in debug. Writing nonzero values may cause boundedly undefined results.
13	L2TAG	L2 tag parity error 0 L2 tag parity error not detected. 1 L2 tag parity error detected.
14	L2DAT	L2 data parity error 0 L2 data parity error not detected. 1 L2 data parity error detected.
15	—	Reserved. Normally cleared. Used in debug. Writing nonzero values may cause boundedly undefined results.
16	—	Reserved. Normally cleared. Used in debug. Writing nonzero values may cause boundedly undefined results.
17–18	—	Reserved
19	TEA	Bus transfer error acknowledge 0 $\overline{tea}$ not detected as asserted. 1 $\overline{tea}$ detected as asserted.
20–31	—	Reserved

6.1.6.5 Instruction and Data Cache Registers

Several registers are used for configuring and controlling the various L1 and L2 caches. Along with the cache registers (L2CR, ICTRL, and LDSTCR), HID0 is used in configuring the caches. Details of how the various cache registers are used is discussed below. See the “L1 and L2 Cache Operation” chapter of the *e600 PowerPC Core Reference Manual* for further details on configuring the cache.

6.1.6.5.1 L2 Cache Control Register (L2CR)

The L2 cache control register (L2CR), shown in Figure 6-10, is a supervisor-level, implementation-specific SPR used to configure and operate the L2 cache. It is cleared by a hard reset or power-on reset.

With the addition of ECC support, tag parity is controlled separately through the TPARDIS bit in the L2ERRDIS register. Data parity can only be enabled through L2CR[L2PE] if ECC is disabled in the L2ERRDIS register.

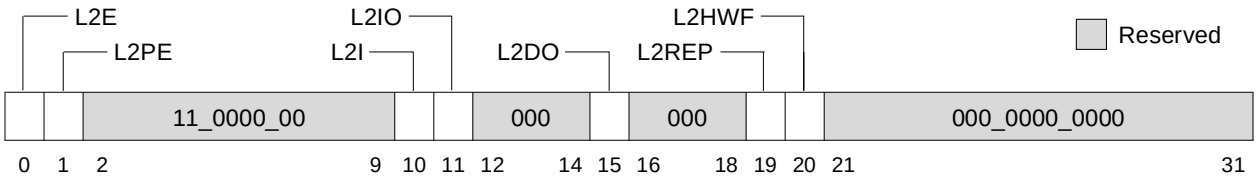


Figure 6-10. L2 Control Register (L2CR)

The L2 cache interface is described in the “L1 and L2 Cache Operation” chapter of the *e600 PowerPC Core Reference Manual*. The bits of the L2CR are described in Table 6-11.

Table 6-11. L2CR Field Descriptions

Bits	Name	Description
0	L2E	L2 enable. Used to enable the L2 cache. 0 The L2 cache is disabled and is not accessed for reads, snoops, or writes. 1 The L2 cache is enabled.
1	L2PE	L2 data parity checking enable 0 L2 data parity checking disabled 1 L2 data parity checking enabled if L2ERRDIS[MBECCDIS] = 1 and L2ERRDIS[SBECCDIS] = 1. If ECC is enabled (L2ERRDIS[MBECCDIS] = 0 or L2ERRDIS[SBECCDIS] = 0), setting L2PE has no effect; ECC checking will still be performed. Note: The L2ERRDIS register includes bits to enable/disable tag parity checking. Data parity can only be enabled with L2CR[L2PE] if ECC is disabled in the L2ERRDIS register. By default, tag parity and data ECC checking are enabled on the e600 core.
2–3	—	Reserved, will read as 0b11.
4–9	—	Reserved
10	L2I	L2 global invalidate 0 Do not perform global invalidate operation on the L2 cache. 1 L2 cache invalidate globally Invalidates the L2 cache globally by clearing the L2 status bits. This bit must not be set while the L2 cache is enabled. Note that L2I is automatically cleared when the global invalidate operation completes.
11	L2IO	L2 instruction-only. Causes the L2 cache to allocate lines for instruction cache transactions only. 0 The L2 cache allocates entries for data accesses that miss. 1 The L2 cache does not allocate entries for data accesses that miss in the L2. Data accesses that hit, instruction accesses, and system accesses are unaffected. If L2DO and L2IO are both set, no new lines are allocated in the L2 cache, effectively locking the entire cache.
12–14	—	Reserved
15	L2DO	L2 data only. L2 cache lines are allocated for data cache transactions only. 0 The L2 cache allocates entries for instruction accesses that miss. 1 The L2 cache does not allocate entries for instruction accesses that miss in the L2. Instruction accesses that hit in the L2, data accesses, and system accesses are unaffected. If both L2DO and L2IO are set, no new lines are allocated in the L2 cache, effectively locking the entire cache.
16–18	—	Reserved
19	L2REP	L2 replacement algorithm selection on a miss 0 Pseudo-random replacement algorithm is used (default) 1 3-bit counter replacement algorithm is used See the “L1 and L2 Cache Operation” chapter of the *e600 PowerPC Core Reference Manual* for more information.
20	L2HWF	L2 hardware flush 0 The L2 cache is not being flushed. 1 A 0->1 transition on this bit triggers a global flush of the entire L2 cache. All modified lines will be cast out to main memory. The cache must be locked by setting L2CR[L2DO] = 1 and L2CR[L2IO] = 1 before setting this bit. See the “L1 and L2 Cache Operation” chapter of the *e600 PowerPC Core Reference Manual* for more information.

Table 6-11. L2CR Field Descriptions (continued)

Bits	Name	Description
21–23	—	Reserved
24–27	—	Reserved. Writing these bits will result in undefined L2 cache behavior.
28–31	—	Reserved

The L2CR register can be accessed with the **mtspr** and **mfspr** instructions using SPR 1017.

6.1.6.5.2 L2 Error Injection Mask High Register (L2ERRINJHI)

The L2 error injection mask high register (L2ERRINJHI), shown in Figure 6-11, is a supervisor-level SPR used for error injection of the high word of the data path.

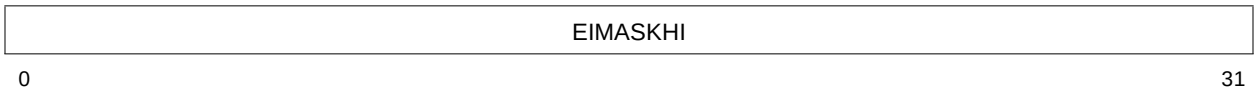


Figure 6-11. L2 Error Injection Mask High Register (L2ERRINJHI)

Table 6-12 describes L2ERRINJHI[EIMASKHI].

Table 6-12. L2ERRINJHI Field Description

Bits	Name	Description
0–31	EIMASKHI	Error injection mask for the high word of the data path. A set bit corresponding to a data path bit causes that bit on the data path to be inverted on cache writes if date array error injection is enabled by setting L2ERRINJCTL[DERRIEN] = 1.

6.1.6.5.3 L2 Error Injection Mask High Register (L2ERRINJLO)

The L2 error injection mask low register (L2ERRINJLO), shown in Figure 6-12, is a supervisor-level SPR used for error injection of the low word of the data path.

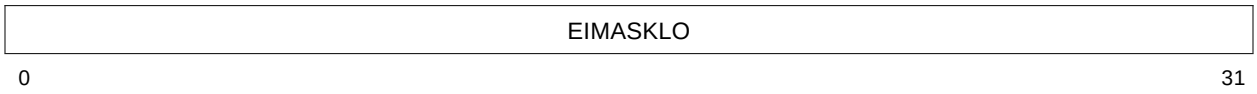


Figure 6-12. L2 Error Injection Mask Low Register (L2ERRINJLO)

Table 6-13 describes L2ERRINJLO[EIMASKLO].

Table 6-13. L2ERRINJLO Field Description

Bits	Name	Description
0–31	EIMASKLO	Error injection mask for the low word of the data path. A set bit corresponding to a data path bit causes that bit on the data path to be inverted on cache writes if data array error injection is enabled by setting L2ERRINJCTL[DERRIEN] = 1.

6.1.6.5.4 L2 Error Injection Mask Control Register (L2ERRINJCTL)

The L2 error injection mask control register (L2ERRINJCTL), shown in Figure 6-13, is a supervisor-level SPR used to configure error injection.

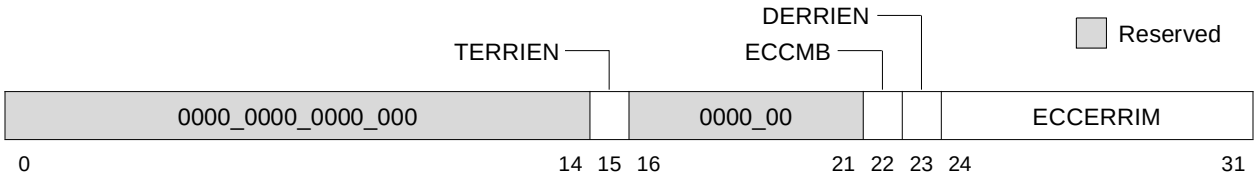


Figure 6-13. L2 Error Injection Mask Control Register (L2ERRINJCTL)

Table 6-14 describes L2ERRINJCTL fields.

Table 6-14. L2ERRINJCTL Field Descriptions

Bits	Name	Description
0–14	—	Reserved
15	TERRIEN	L2 tag array error injection enable 0 No tag errors are injected. 1 All subsequent entries written to the L2 tag array have the parity bit inverted.
16–21	—	Reserved
22	ECCMB	ECC mirror byte enable 0 ECC byte mirroring is disabled. 1 The most significant data path byte is mirrored onto the ECC byte if DERRIEN = 1.
23	DERRIEN	L2 data array error injection enable 0 No data errors are injected. 1 All subsequent entries written to the L2 data array have data or ECC bits inverted as specified in the data error injection masks and ECC error injection mask and/or data path byte mirrored onto ECC as specified by the ECC mirror byte enable bit, ECCMB. 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 cache writes if DERRIEN = 1.

6.1.6.5.5 L2 Error Capture Data High Register (L2CAPTDATAHI)

The L2 error capture data high register (L2CAPTDATAHI), shown in Figure 6-15, holds the high word of the L2 data that contains the detected error.

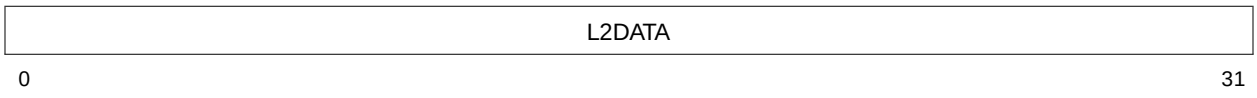


Figure 6-14. L2 Error Capture Data High Register (L2CAPTDATAHI)

Table 6-15 describes L2CAPTDATAHI[L2DATA].

Table 6-15. L2CAPTDATAHI Field Description

Bits	Name	Description
0–31	L2DATA	L2 data high word (read only)

6.1.6.5.6 L2 Error Capture Data Low Register (L2CAPTDATALO)

The L2 error capture data low register (L2CAPTDATALO), shown in Figure 6-15, holds the low word of the L2 data that contains the detected error.

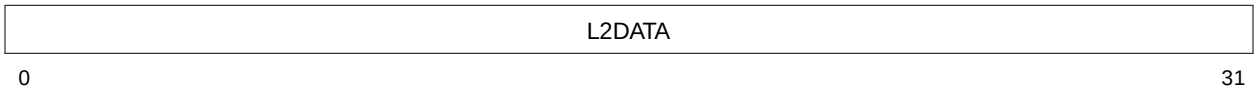


Figure 6-15. L2 Error Capture Data Low Register (L2CAPTDATALO)

Table 6-16 describes L2CAPTDATALO[L2DATA].

Table 6-16. L2CAPTDATALO Field Description

Bits	Name	Description
0–31	L2DATA	L2 data low word (read only)

6.1.6.5.7 L2 Error Syndrome Register (L2CAPTECC)

The L2 error syndrome register (L2CAPTECC), shown in Figure 6-16, is a supervisor-level SPR that contains the ECC syndrome and datapath ECC of the failing double word.

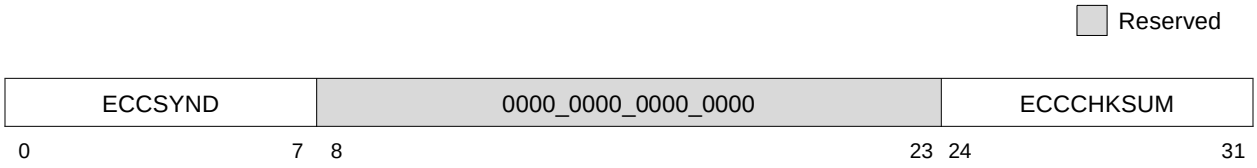


Figure 6-16. L2 Error Syndrome Register (L2CAPTECC)

Table 6-17 describes L2CAPTECC fields.

Table 6-17. L2CAPTECC Field Descriptions

Bits	Name	Description
0–7	ECCSYND	The calculated ECC syndrome of the failing double word (read only)
8–23	—	Reserved
24–31	ECCCHKSUM	The datapath ECC of the failing double word (read only)

6.1.6.5.8 L2 Error Detect Register (L2ERRDET)

The L2 error detect register (L2ERRDET), shown in Figure 6-17, is a supervisor-level SPR that shows the errors detected.

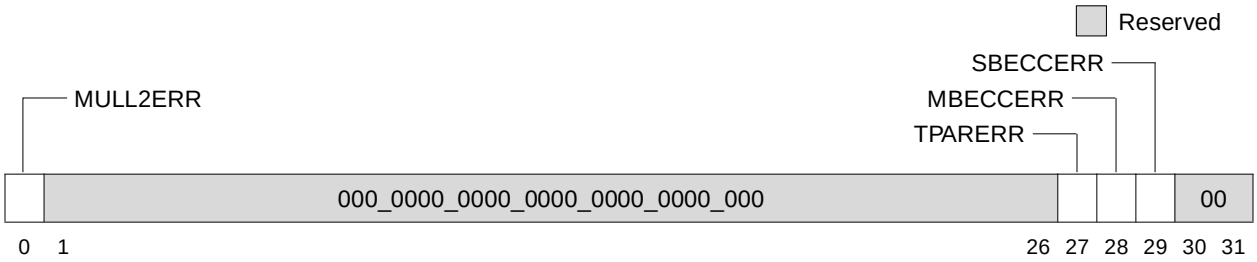


Figure 6-17. L2 Error Detect Register (L2ERRDET)

Table 6-18 describes L2ERRDET fields.

Table 6-18. 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 Note that setting this bit to 1 clears it to a value of 0.
1–26	—	Reserved
27	TPARERR	Tag parity error (Bit reset, write-1-to-clear) 0 Tag parity error was not detected 1 Tag parity error was detected Note that setting this bit to 1 clears it to a value of 0.
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 Note that setting this bit to 1 clears it to a value of 0.
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 Note that setting this bit to 1 clears it to a value of 0.
30–31	—	Reserved

6.1.6.5.9 L2 Error Disable Register (L2ERRDIS)

The L2 error disable register (L2ERRDIS), shown in Figure 6-18, is a supervisor-level SPR that disables and enables error detection. Note that the L2 cache must be disabled and flushed before enabling or disabling ECC to ensure that no errors occur. See the “L1 and L2 Cache Operation” chapter of the *e600 PowerPC Core Reference Manual* and Table 6-37 for the synchronization requirements required to enable or disable ECC.

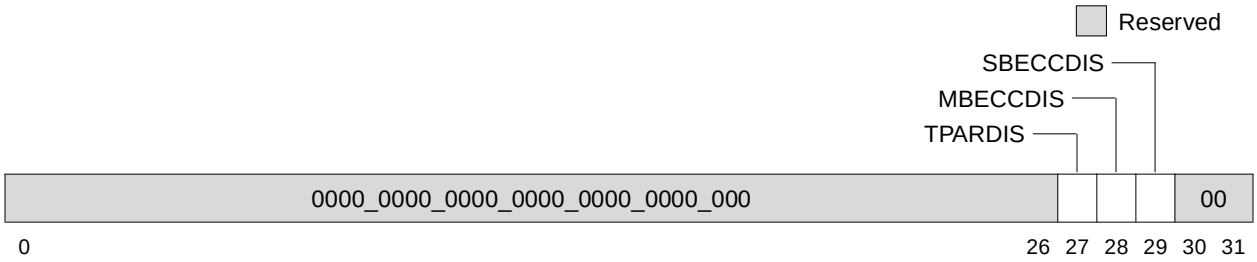


Figure 6-18. L2 Error Disable Register (L2ERRDIS)

Table 6-19 describes L2ERRDIS fields.

Table 6-19. L2ERRDIS Field Descriptions

Bits	Name	Description
0–26	—	Reserved
27	TPARDIS	Tag parity error disable 0 Tag parity error detection enabled 1 Tag parity error detection disabled
28	MBECCDIS	Multiple-bit ECC error disable 0 Multiple-bit ECC error detection enabled 1 Multiple-bit ECC error detection disabled
29	SBECCDIS	Single-bit ECC error disable 0 Single-bit ECC error detection enabled 1 Single-bit ECC error detection disabled
30–31	—	Reserved

6.1.6.5.10 L2 Error Interrupt Enable Register (L2ERRINTEN)

The L2 error interrupt enable register (L2ERRINTEN), shown in Figure 6-19, is a supervisor-level SPR used to enable L2 error interrupts. When any of these error conditions exist and the corresponding bit in the L2ERRINTEN register is enabled, a machine check interrupt is generated.

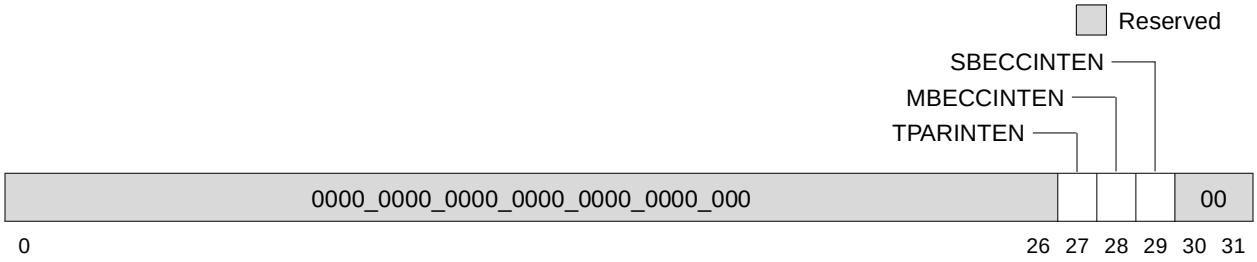


Figure 6-19. L2 Error Interrupt Enable Register (L2ERRINTEN)

Table 6-20 describes L2ERRINTEN fields.

Table 6-20. L2ERRINTEN Field Descriptions

Bits	Name	Description
0–26	—	Reserved
27	TPARINTEN	Tag parity error reporting enable 0 Tag parity error reporting disabled 1 Tag parity error reporting enabled.
28	MBECCINTEN	Multiple-bit ECC error reporting enable 0 Multiple-bit ECC error reporting disabled 1 Multiple-bit ECC error reporting enabled
29	SBECCINTEN	Single-bit ECC error reporting enable 0 Single-bit ECC error reporting disabled 1 Single-bit ECC error reporting enabled
30–31	—	Reserved

6.1.6.5.11 L2 Error Attributes Capture Register (L2ERRATTR)

The L2 error attributes capture register (L2ERRATTR), shown in Figure 6-20, is a supervisor-level SPR that describes the L2 error attributes. All the fields of the L2ERRATTR are read-only except for bit 31, VALINFO.

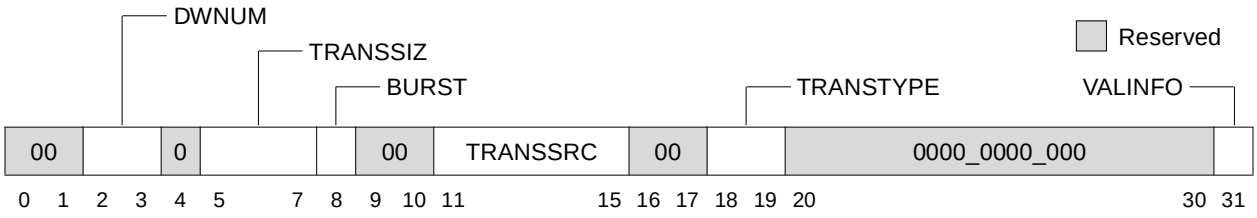


Figure 6-20. L2 Error Attributes Capture Register (L2ERRATTR)

Table 6-21 describes L2ERRATTR fields.

Table 6-21. L2ERRATTR Field Descriptions

Bits	Name	Description
0–1	—	Reserved
2–3	DWNUM	Double-word number of the detected error (data ECC errors only). Read only.
4	—	Reserved
5–7	TRANSSIZ	Transaction size for detected error (read only). Reserved for read transactions (TRANSTYPE = 10). The transaction size for a read to the L2 will always be a 32-byte burst. 000 8 bytes (single-beat) or reserved (burst) 001 1 byte (single-beat) or 16 bytes (burst) 010 2 bytes (single-beat) or 32 bytes (burst) 011 3 bytes (single beat) or reserved (burst) 100 4 bytes (single-beat) or reserved (burst) 101 5 bytes (single-beat) or reserved (burst) 110 6 bytes (single-beat) or reserved (burst) 111 7 bytes (single-beat) or reserved (burst)
8	BURST	Burst transaction for detected error. Read only. 0 Single-beat (≤ 64 bits) transaction 1 Burst transaction
9–10	—	Reserved
11–15	TRANSSRC	Transaction source for detected error. Read only. 00000External (logic external to the core) 10000Core (instruction) 10001Core (data)
16–17	—	Reserved
18–19	TRANSTYPE	Transaction type for detected error. Read only. 00 Snoop (tag/status read) 01 Write 10 Read 11 Reserved
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 that 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.

6.1.6.5.12 L2 Error Address Error Capture Register (L2ERRADDR)

The L2 error address error capture register (L2ERRADDR), shown in Figure 6-21, is a supervisor-level SPR that shows the L2 address corresponding to bits 4–35 of the detected error.

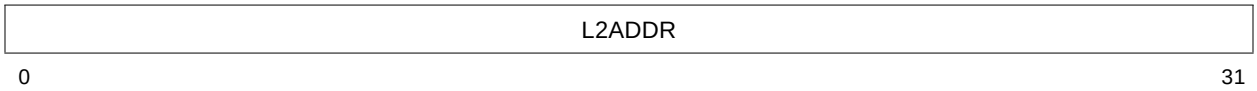


Figure 6-21. L2 Error Address Error Capture Register (L2ERRADDR)

Table 6-22 describes L2ERRADDR.

Table 6-22. L2ERRADDR Field Description

Bits	Name	Description
0–31	L2ADDR	L2 address[4:35] corresponding to detected error (read only)

6.1.6.5.13 L2 Error Address Error Capture Register (L2ERREADDR)

The L2 error address error capture register (L2ERREADDR), shown in Figure 6-22, is a supervisor-level SPR that shows the L2 address corresponding to bits 0–3 of the detected error.

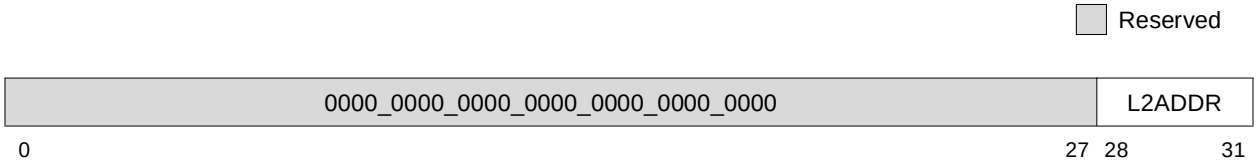


Figure 6-22. L2 Error Address Error Capture Register (L2ERREADDR)

Table 6-23 describes L2ERREADDR.

Table 6-23. L2ERREADDR Field Description

Bits	Name	Description
0–27	—	Reserved
28–31	L2EADDR	L2 address[0:3] corresponding to detected error (read only)

6.1.6.5.14 L2 Error Control Register (L2ERRCTL)

The L2 error control register (L2ERRCTL), shown in Figure 6-23, is a supervisor-level SPR that configures the L2 cache ECC error threshold and provides an L2 error count.

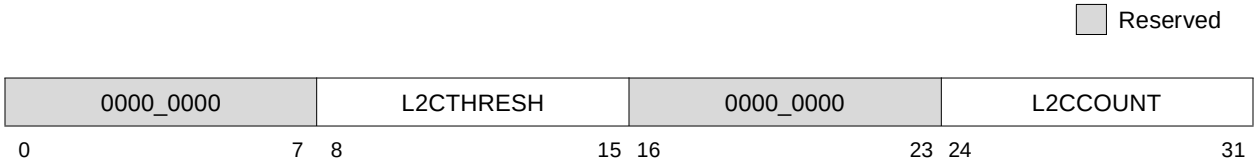


Figure 6-23. L2 Error Control Register (L2ERRCTL)

Table 6-24 describes L2ERRCTL fields.

Table 6-24. 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 that have been detected. If L2CCOUNT equals the ECC single-bit error trigger threshold (L2CTHRESH), an error is reported if single-bit error reporting is enabled (SPECCDIS = 0). Software can write to this field.

6.1.6.5.15 Instruction Cache and Interrupt Control Register (ICTRL)

The instruction cache and interrupt control register (ICTRL), shown in Figure 6-24, is used in configuring interrupts and error reporting for the instruction and data caches. It is accessed as SPR 1011. Control and access to the ICTRL is through the privileged **mtspr**/**mfspr** instructions.

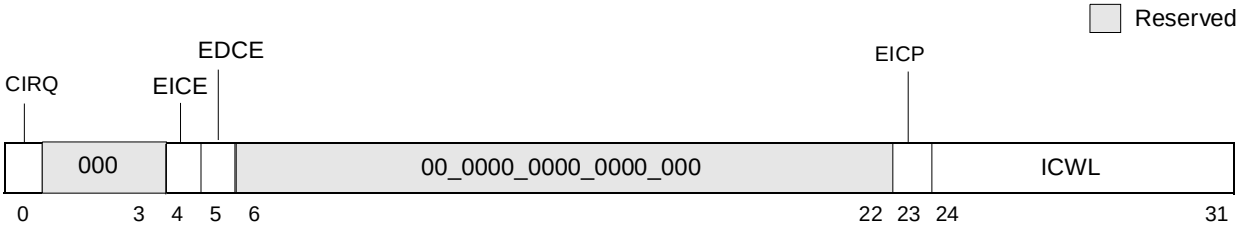


Figure 6-24. Instruction Cache and Interrupt Control Register (ICTRL)

Table 6-25 describes the bit fields for the ICTRL register.

Table 6-25. ICTRL Field Descriptions

Bits	Name	Description
0	CIRQ	CPU interrupt request 0 No processor interrupt request forwarded to interrupt handling. If software clears the CIRQ bit, it does not cancel a previously sent interrupt request. 1 Processor interrupt request sent to the interrupt mechanism. This interrupt request is combined with the external interrupt request (assertion of $\overline{int}$). When interrupts external to the core are enabled with the MSR[EE] bit and either this bit is set or $\overline{int}$ is asserted, the e600 core takes the external interrupt. If there is more than one interrupt request pending (CIRQ and $\overline{int}$ is asserted), only one interrupt is taken. When the external interrupt is taken, the ICTRL[CIRQ] bit is automatically cleared. Note that this mechanism allows a processor to interrupt itself. If software leaves CIRQ set while waiting for the interrupt to be taken, it can poll CIRQ to determine when the interrupt has been taken.
1–3	—	Reserved

Table 6-25. ICTRL Field Descriptions (continued)

Bits	Name	Description
4	EICE[1]	Instruction cache parity error enable 0 When the bit is cleared, any parity error in the L1 instruction cache is masked and does not cause machine checks or checkstop 1 Enables instruction cache parity error reporting. When an instruction cache parity error occurs, a machine check interrupt is taken if MSR[ME] = 1. When this condition occurs, SRR1[1] is set. For details on the machine check interrupt, see the “Interrupts” chapter of the *e600 PowerPC Core Reference Manual*.
5	EDCE[2]	Data cache parity error enable 0 When the bit is cleared, any parity error in the L1 data cache is masked and does not cause machine checks or checkstop 1 Enables data cache parity error reporting. When a data cache parity error occurs, a machine check interrupt is taken if MSR[ME] = 1. When this condition occurs, SRR1[2] is set. For details on the machine check interrupt, see the “Interrupts” chapter of the *e600 PowerPC Core Reference Manual*.
6–8	—	Reserved. Normally cleared. Used in debug. Writing nonzero values may cause boundedly undefined results.
9–22	—	Reserved. Read as zeros and ignores writes.
23	EICP	Enable instruction cache parity checking 0 Instruction cache parity disabled 1 When the EICP bit is set, the parity of any instructions fetched from the L1 instruction cache is checked. Any errors found are reported as instruction cache parity errors in SRR1. If EICE is also set, these instruction cache errors cause a machine check or checkstop. If either EICP or EICE is cleared, instruction cache parity is ignored. Note that when parity checking and error reporting are both enabled, errors are reported even on speculative fetches that are never actually executed. Correct instruction cache parity is always loaded into the L1 instruction cache regardless of whether checking is enabled or not.
24–31	ICWL[1]	Instruction cache way lock 0 Instruction cache way lock disabled. 1 Instruction cache way lock enabled. Each bit in ICWL corresponds to a way of the L1 instruction cache. Bit 24 corresponds to way 0, and bit 31 corresponds to way 7. Setting a bit locks the corresponding way in the instruction cache. Setting all 8 bits of ICWL is equivalent to locking the entire instruction cache. When all 8 ICWL bits are set, the e600 core behaves the same as when HID0[ILOCK] is set. See Section 6.1.6.1, “Hardware Implementation-Dependent Register 0 (HID0),” for details. See the “L1 and L2 Cache Operation” chapter of the *e600 PowerPC Core Reference Manual*. for suggestions on how to keep the PLRU replacement algorithm symmetrical, and for synchronization requirements for modifying ICWL.

1 A context synchronizing instruction must precede and follow an mtspr.

2 A dssall and sync must precede an mtspr and then a sync and context synchronizing instruction must follow. Note that if a user is not using the AltiVec data streaming instructions, then a dssall is not necessary prior to accessing the ICTRL[EDCE] bit.

ICTRL can be accessed with the **mtspr** and **mfspr** instructions using SPR 1011.

6.1.6.5.16 Load/Store Control Register (LDSTCR)

The load/store control register (LDSTCR) provides a way to lock the ways for the L1 data cache. The LDSTCR is shown in Figure 6-29.

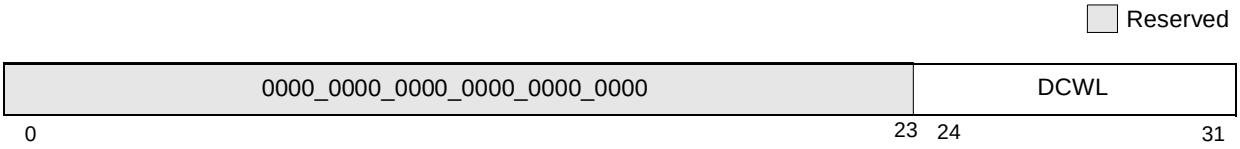


Figure 6-25. Load/Store Control Register (LDSTCR)

Table 6-30 describes the bit fields for the LDSTCR register.

Table 6-26. LDSTCR Field Descriptions

Bits	Name	Description
0–23	—	Reserved. Writing nonzero values may cause boundedly undefined results.
24–31	DCWL	Data cache way lock. Each bit in DCWL corresponds to a way of the L1 data cache. Bit 24 corresponds to way 0, and bit 31 corresponds to way 7. 0 Each cleared bit corresponds to the corresponding way not being locked in the L1 data cache. 1 Each set bit locks the corresponding way in the L1data cache. When DCWL[24–31] are all set, it is equivalent to locking the entire L1 data cache and the e600 core behaves the same as if HID0[DLOCK] is set. the “L1 and L2 Cache Operation” chapter of the *e600 PowerPC Core Reference Manual.* describes how to keep the PLRU replacement algorithm symmetrical and for more information on synchronization requirements with LDSTCR.

The LDSTCR register can be accessed with the **mtspr** and **mfspr** instructions using SPR 1016. For synchronization requirements on the register see Section 6.3.2.4, “Synchronization.”

6.1.6.6 Instruction Address Breakpoint Register (IABR)

The instruction address breakpoint register (IABR), shown in Table 6-26, supports the instruction address breakpoint interrupt. When this interrupt is enabled, instruction fetch addresses are compared with an effective address stored in the IABR. If the word specified in the IABR is fetched, the instruction breakpoint handler is invoked. The instruction that triggers the breakpoint does not execute before the handler is invoked. For more information, see the “Interrupts” chapter of the *e600 PowerPC Core Reference Manual.* The IABR can be accessed with **mtspr** and **mfspr** using SPR 1010. The e600 core requires that an **mtspr**[IABR] be followed by a context synchronizing instruction. The e600 core may not generate a breakpoint response for that context synchronizing instruction if the breakpoint was enabled by **mtspr**[IABR] immediately preceding it. The e600 core cannot block a breakpoint response on the context synchronizing instruction if the breakpoint was disabled by **mtspr**[IABR] immediately preceding it. For more information on synchronization see Section 6.3.2.4.1, “Context Synchronization.”

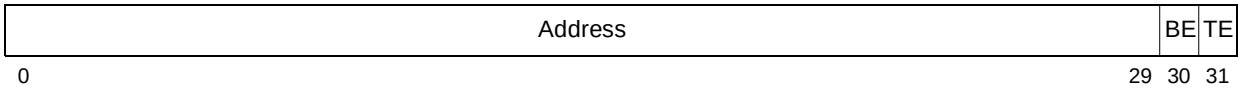


Figure 6-26. Instruction Address Breakpoint Register (IABR)

The IABR bits are described in Table 6-27.

Table 6-27. IABR Field Descriptions

Bits[1]	Name	Description
0–29	Address	Word instruction breakpoint address to be compared with EA[0–29] of the next instruction
30	BE	Breakpoint enabled. Setting this bit enables breakpoint address checking.
31	TE	Translation enable IABR[TE] must equal MSR[IR] in order for a match to be signaled. When IABR[TE] and MSR[IR] = 0 or when IABR[TE] and MSR[IR] = 1, a match is signaled.

1 A context synchronizing instruction must follow an mtspr.

6.1.6.7 Memory Management Registers Used for Software Table Searching

This section describes the registers used by the e600 core when software table searching is enabled (HID0[STEN] = 1) and a TLB miss interrupt occurs. Software table searching is described in detail in the "Memory Management" chapter of the *e600 PowerPC Core Reference Manual.*

6.1.6.7.1 TLB Miss Register (TLBMISS)

The TLBMISS register is automatically loaded by the e600 core when software table searching is enabled (HID0[STEN] = 1) and a TLB miss interrupt occurs. Its contents are used by the TLB miss interrupt handlers (the software table search routines) to start the search process. Note that the e600 core always loads a big-endian address into the TLBMISS register. This register is read-only. The TLBMISS register has the format shown in Figure 6-27 for the e600 core.

Figure 6-27. TLBMISS Register

Table 6-28 describes the bits in the TLBMISS register.

Table 6-28. TLBMISS Register—Field and Bit Descriptions

Bits	Name	Description
0–30	PAGE	Effective page address. Stores EA[0–30] of the access that caused the TLB miss interrupt.
31	LRU	Least recently used way of the addressed TLB set. The LRU bit can be loaded into bit 31 of **r**B, prior to execution of **tlbli** or **tlbld** to select the way to be replaced for a TLB miss. However, this value should be inverted in **r**B prior to execution of **tlbli** or **tlbld** for a TLB miss interrupt caused by the need to update the C-bit.

TLBMISS can be accessed with **mtspr** and **mfspr** using SPR 980.

6.1.6.7.2 Page Table Entry Registers (PTEHI and PTELO)

The PTEHI and PTELO registers are used by the **tlbld** and **tlbli** instructions to create a TLB entry. When software table searching is enabled (HID0[STEN] = 1) and a TLB miss interrupt occurs, the bits of the

page table entry (PTE) for this access are located by software and saved in the PTE registers. Figure 6-28 shows the format for two supervisor registers, PTEHI and PTELO, respectively.

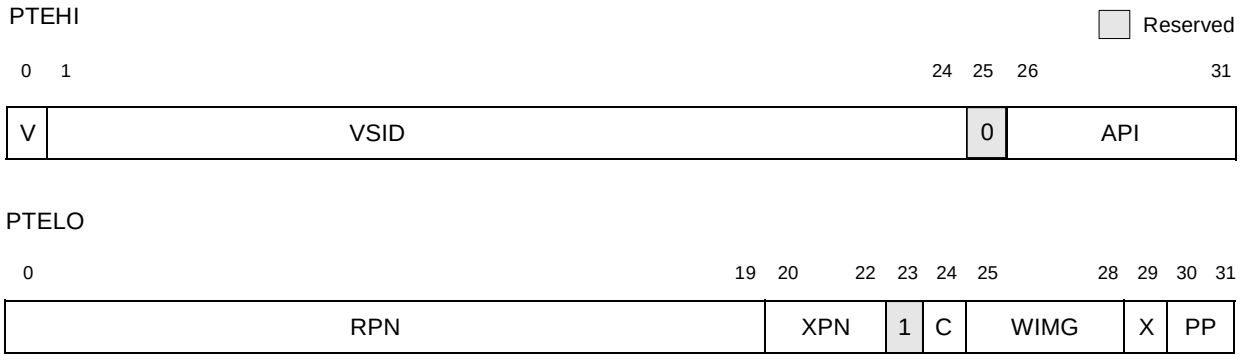


Figure 6-28. PTEHI and PTELO Registers—Extended Addressing

Note that the contents of PTEHI are automatically loaded when any of the three software table search interrupts is taken. PTELO is loaded by the software table search routines (the TLB miss interrupt handlers) based on the valid PTE located in the page tables prior to execution of the **tlbli** or **tlbld** instruction.

Table 6-29 lists the corresponding bit definitions for the PTEHI and PTELO registers.

Table 6-29. PTEHI and PTELO Bit Definitions

Register	Bit	Name	Description
PTEHI	0	V	Entry valid (V = 1) or invalid (V = 0). Always set by the processor on a TLB miss interrupt.
	1–24	VSID	Virtual segment ID. The corresponding SR[VSID] field is copied to this field.
	25	—	Reserved. Corresponds to the hash function identifier in PTE.
	26–31	API	Abbreviated page index. TLB miss interrupts will set this field with bits from TLBMISS[4–9] which are bits from the effective address for the access that caused the software table search operation. The **tlbld** and **tlbli** instructions ignore the API bits in PTEHI register and get the API from the instruction's operand**,** **r**B. However, for future compatibility, the API in **r**B should match the PTEHI[API].
PTELO	0–19	RPN	Physical page number
	20–22	XPN	Extended page number The XPN field provides the physical address bits, PA[0–2].
	23	—	Reserved. Corresponds to the reference bit in a PTE. The referenced bit is not stored in the page tables, so this bit is ignored in the PTELO register. To simplify software, this bit is hard-wired to 1. All the other bits in PTELO correspond to the bits in the low word of the PTE.
	24	C	Changed bit
	25–28	WIMG	Memory/cache control bits
	29	X	Extended page number The X field provides the physical address bit 3, PA[3].
	30–31	PP	Page protection bits

When extended addressing is not enabled, (HID0[XAEN] = 0), the software must clear the PTELO[XPN] and PTELO[X] bits; otherwise whatever values are in the fields become the four most-significant bits of the physical address. **Note:** The PTEHI register is accessed with **mtspr** and **mfspr** as SPR 981 and PTELO is accessed as SPR 982.

6.1.6.8 Thermal Management Register

The e600 core provides an instruction cache throttling mechanism to reduce the instruction execution rate without the complexity and overhead of dynamic clock control. When used with dynamic power management, instruction cache throttling provides the system designer with a flexible way to control device temperature while allowing the processor to continue operating. Note that performance does degrade when instruction cache throttling is enabled to reduce junction temperature; entering short bursts of nap or sleep is a superior method for reducing thermal output and power consumption.

6.1.6.8.1 Instruction Cache Throttling Control Register (ICTC)

Reducing the rate of instruction fetching can control junction temperature without the complexity and overhead of dynamic clock control. System software can control instruction forwarding by writing a nonzero value to the ICTC register, a supervisor-level register shown in Figure 6-29. The overall junction temperature reduction comes from the dynamic power management of each functional unit when the e600 core is idle in between instruction fetches.

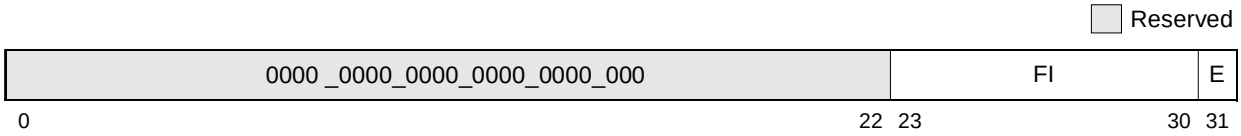


Figure 6-29. Instruction Cache Throttling Control Register (ICTC)

Table 6-30 describes the bit fields for the ICTC register.

Table 6-30. ICTC Field Descriptions

Bits	Name	Description
0–22	—	Reserved. The bits should be cleared.
23–30	INTERVAL	Instruction forwarding interval expressed in core clocks. When throttling is enabled, the interval field specifies the minimum number of cycles between instructions being dispatched. (The core dispatches one instruction every INTERVAL cycle.) The minimum interval for throttling control is 2 cycles. 0x00, 0x01, 0x02 One instruction dispatches every 2 core clocks. 0x03 One instruction dispatches every 3 core clocks ... 0xFF One instruction dispatches every 255 core clocks.
31	E	Enable instruction throttling 0 Instructions dispatch normally. 1 Only one instruction dispatches every INTERVAL cycles.

Instruction cache throttling is enabled by setting ICTC[E] and writing the instruction forwarding interval into ICTC[INTERVAL]. A context synchronizing instruction should be executed after a move to the ICTC

register to ensure that it has taken effect. Enabling, disabling, or changing the instruction forwarding interval affects instruction forwarding immediately.

The ICTC register can be accessed with the **mtspr** and **mfspr** instructions using SPR 1019.

6.1.6.9 Performance Monitor Registers

This section describes the registers used by the performance monitor, which is described in the "Performance Monitor" chapter of the *e600 PowerPC Core Reference Manual.*

6.1.6.9.1 Monitor Mode Control Register 0 (MMCR0)

The monitor mode control register 0 (MMCR0), shown in Figure 6-30, is a 32-bit SPR provided to specify events to be counted and recorded. If the state of MSR[PR] and MSR[PMM] matches a state specified in MMCR0, then counting is enabled. See the "Performance Monitor" chapter of the *e600 PowerPC Core Reference Manual* for further details. The MMCR0 can be accessed only in supervisor mode. User-level software can read the contents of MMCR0 by issuing an **mfspr** instruction to UMMCR0, described in Section 6.1.6.9.2, "User Monitor Mode Control Register 0 (UMMCR0)."

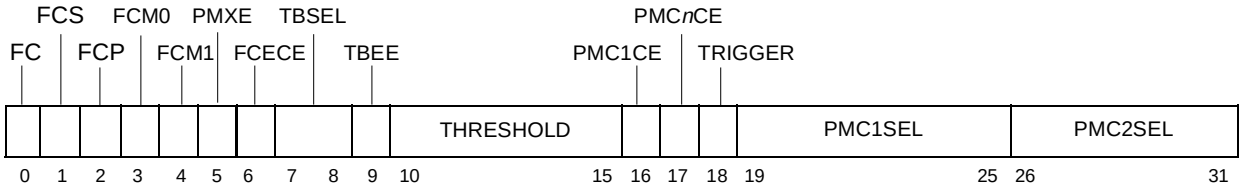


Figure 6-30. Monitor Mode Control Register 0 (MMCR0)

This register is automatically cleared at power-up. Reading this register does not change its contents. Table 6-31 describes MMCR0 fields.

Table 6-31. MMCR0 Field Descriptions

Bits	Name	Description
0	FC	Freeze counters 0 The PMCs are incremented (if permitted by other MMCR bits). 1 The PMCs are not incremented (performance monitor counting is disabled). The processor sets this bit when an enabled condition or event occurs and MMCR0[FCECE] = 1. Note that SIAR is not updated if performance monitor counting is disabled.
1	FCS	Freeze counters in supervisor mode 0 The PMCs are incremented (if permitted by other MMCR bits). 1 The PMCs are not incremented if MSR[PR] = 0.
2	FCP	Freeze counters in user mode 0 The PMCs are incremented (if permitted by other MMCR bits). 1 The PMCs are not incremented if MSR[PR] = 1.
3	FCM1	Freeze counters while mark = 1 0 The PMCs are incremented (if permitted by other MMCR bits). 1 The PMCs are not incremented if MSR[PMM] = 1.

Table 6-31. MMCR0 Field Descriptions (continued)

Bits	Name	Description
4	FCM0	Freeze counters while mark = 0 0 The PMCs are incremented (if permitted by other MMCR bits). 1 The PMCs are not incremented if MSR[PMM] = 0.
5	PMXE	Performance monitor interrupt enable 0 Performance monitor interrupts are disabled. 1 Performance monitor interrupts are enabled until a performance monitor interrupt occurs, at which time MMCR0[PMXE] is cleared. Software can clear PMXE to prevent performance monitor interrupts. Software can also set PMXE and then poll it to determine whether an enabled condition or event occurred.
6	FCECE	Freeze counters on enabled condition or event 0 The PMCs are incremented (if permitted by other MMCR bits). 1 The PMCs are incremented (if permitted by other MMCR bits) until an enabled condition or event occurs when MMCR0[TRIGGER] = 0, at which time MMCR0[FC] is set. If the enabled condition or event occurs when MMCR0[TRIGGER] = 1, FCECE is treated as if it were 0. The use of the trigger and freeze counter conditions depends on the enabled conditions and events described in the “Performance Monitor” chapter of the *e600 PowerPC Core Reference Manual*.
7–8	TBSEL	Time base selector. Selects the time base bit that can cause a time base transition event (the event occurs when the selected bit changes from 0 to 1). 00 TBL[31] 01 TBL[23] 10 TBL[19] 11 TBL[15] Time base transition events can be used to periodically collect information about processor activity. In multiprocessor systems in which the TB registers are synchronized among processors, time base transition events can be used to correlate the performance monitor data obtained by the several processors. For this use, software must specify the same TBSEL value for all the processors in the system. Because the time-base frequency is implementation-dependent, software should invoke a system service program to obtain the frequency before choosing a value for TBSEL.
9	TBEE	Time base event enable 0 Time-base transition events are disabled. 1 Time-base transition events are enabled. A time-base transition is signaled to the performance monitor if the TB bit specified in MMCR0[TBSEL] changes from 0 to 1. Time-base transition events can be used to freeze the counters (MMCR0[FCECE]), trigger the counters (MMCR0[TRIGGER]), or signal an interrupt (MMCR0[PMXE]). Changing the bits specified in MMCR0[TBSEL] while MMCR0[TBEE] is enabled may cause a false 0 to 1 transition that signals the specified action (freeze, trigger, or interrupt) to occur immediately.
10–15	THRESHOLD	Threshold. Contains a threshold value between 0 to 63. Two types of thresholds can be counted. The first type counts any event that lasts longer than the threshold value and uses MMCR2[THRESHMULT] to scale the threshold value by 2 or 32. The second type counts only the events that exceed the threshold value. This type does not use MMCR2[THRESHMULT] to scale the threshold value. By varying the threshold value, software can obtain a profile of the characteristics of the events subject to the threshold. For example, if PMC1 counts cache misses for which the duration exceeds the threshold value, software can obtain the distribution of cache miss durations for a given program by monitoring the program repeatedly using a different threshold value each time.

Table 6-31. MMCR0 Field Descriptions (continued)

Bits	Name	Description
16	PMC1CE	PMC1 condition enable. Controls whether counter negative conditions due to a negative value in PMC1 are enabled. 0 Counter negative conditions for PMC1 are disabled. 1 Counter negative conditions for PMC1 are enabled. Such events can be used to freeze counters (MMCR0[FCECE]), trigger the counters (MMCR0[TRIGGER]), or signal an interrupt (MMCR0[PMXE]).
17	PMC*n*CE	PMC*n* condition enable. Controls whether counter negative conditions due to a negative value in any PMC*n* (that is, in any PMC except PMC1) are enabled. 0 Counter negative conditions for all PMC*n*s are disabled. 1 Counter negative conditions for all PMC*n*s are enabled. These events can be used to freeze the counters (MMCR0[FCECE]), trigger the counters (MMCR0[TRIGGER]), or signal an interrupt (MMCR0[PMXE]).
18	TRIGGER	Trigger 0 The PMCs are incremented (if permitted by other MMCR bits). 1 PMC1 is incremented (if permitted by other MMCR bits). The PMC*n*s are not incremented until PMC1 is negative or an enabled timebase or event occurs, at which time the PMC*n*s resume incrementing (if permitted by other MMCR bits) and MMCR0[TRIGGER] is cleared. The description of FCECE explains the interaction between TRIGGER and FCECE. Uses of TRIGGER include the following: • Resume counting in the PMC*n*s when PMC1 becomes negative without causing a performance monitor interrupt. Then freeze all PMCs (and optionally cause a performance monitor interrupt) when a PMC*n* becomes negative. The PMC*n*s then reflect the events that occurred after PMC1 became negative and before PMC*n* becomes negative. This use requires the following MMCR0 bit settings. – TRIGGER = 1 – PMC1CE = 0 – PMC*n*CE = 1 – TBEE = 0 – FCECE = 1 – PMXE = 1 (if a performance monitor interrupt is desired) • Resume counting in the PMC*n*s when PMC1 becomes negative, and cause a performance monitor interrupt without freezing any PMCs. The PMC*n*s then reflect the events that occurred between the time PMC1 became negative and the time the interrupt handler reads them. This use requires the following MMCR0 bit settings. – TRIGGER = 1 – PMC1CE = 1 – TBEE = 0 – FCECE = 0 – PMXE = 1 The use of the trigger and freeze counter conditions depends on the enabled conditions and events described in the “Performance Monitor” chapter of the *e600 PowerPC Core Reference Manual*.
19–25	PMC1SEL	PMC1 selector. Contains a code (one of at most 128 values) that identifies the event to be counted in PMC1. See the “Performance Monitor” chapter of the *e600 PowerPC Core Reference Manual*.
26–31	PMC2SEL	PMC2 selector. Contains a code (one of at most 64 values) that identifies the event to be counted in PMC2. See the “Performance Monitor” chapter of the *e600 PowerPC Core Reference Manual*.

MMCR0 can be accessed with **mtspr** and **mfspr** using SPR 952.

6.1.6.9.2 User Monitor Mode Control Register 0 (UMMCR0)

The contents of MMCR0 are reflected to UMMCR0, which can be read by user-level software. MMCR0 can be accessed with **mfspr** using SPR 936.

6.1.6.9.3 Monitor Mode Control Register 1 (MMCR1)

The monitor mode control register 1 (MMCR1) functions as an event selector for performance monitor counter registers 3, 4, 5, and 6 (PMC3, PMC4, PMC5, PMC6). The MMCR1 register is shown in Figure 6-31.

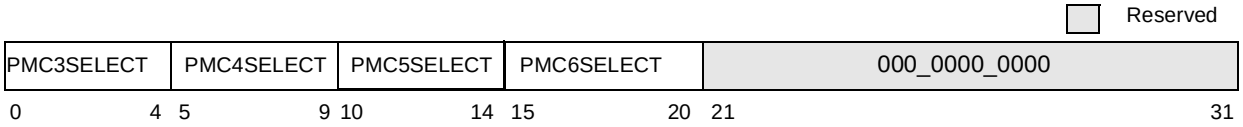


Figure 6-31. Monitor Mode Control Register 1 (MMCR1)

Bit settings for MMCR1 are shown in Table 6-32. The corresponding events are described in Section 6.1.6.9.8, "Performance Monitor Counter Registers (PMC1–PMC6)."

Table 6-32. MMCR1 Field Descriptions

Bits	Name	Description
0–4	PMC3SELECT	PMC3 selector. Contains a code (1 of at most 32 values) that identifies the event to be counted in PMC3. See the "Performance Monitor" chapter of the *e600 PowerPC Core Reference Manual*.
5–9	PMC4SELECT	PMC4 selector. Contains a code (1 of at most 32 values) that identifies the event to be counted in PMC4. See the "Performance Monitor" chapter of the *e600 PowerPC Core Reference Manual*.
10–14	PMC5SELECT	PMC5 selector. Contains a code (1 of at most 32 values) that identifies the event to be counted in PMC5. See the "Performance Monitor" chapter of the *e600 PowerPC Core Reference Manual*.
15–20	PMC6SELECT	PMC6 selector. Contains a code (1 of at most 64 values) that identifies the event to be counted in PMC6. See the "Performance Monitor" chapter of the *e600 PowerPC Core Reference Manual*.
21–31	—	Reserved

MMCR1 can be accessed with **mtspr** and **mfspr** using SPR 956. User-level software can read the contents of MMCR1 by issuing an **mfspr** instruction to UMMCR1, described in Section 6.1.6.9.4, "User Monitor Mode Control Register 1 (UMMCR1)."

6.1.6.9.4 User Monitor Mode Control Register 1 (UMMCR1)

The contents of MMCR1 are reflected to UMMCR1, which can be read by user-level software. MMCR1 can be accessed with **mfspr** using SPR 940.

6.1.6.9.5 Monitor Mode Control Register 2 (MMCR2)

The monitor mode control register 2 (MMCR2) functions as an event selector for performance monitor counter registers 3 and 4 (PMC3 and PMC4). The MMCR2 register is shown in Figure 6-32.

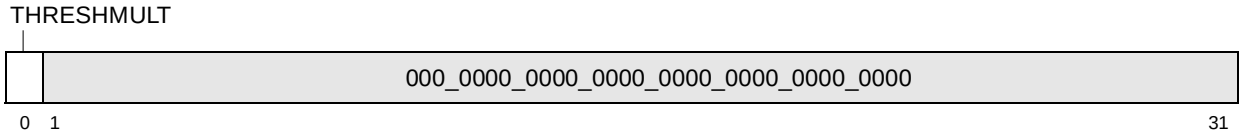


Figure 6-32. Monitor Mode Control Register 2 (MMCR2)

Table 6-33 describes MMCR2 fields.

Table 6-33. MMCR2 Field Descriptions

Bits	Name	Description
0	THRESHMULT	Threshold multiplier. Used to extend the range of the THRESHOLD field, MMCR0[10–15]. 0 Threshold field is multiplied by 2 1 Threshold field is multiplied by 32
1–31	—	Reserved

MMCR2 can be accessed with **mtspr** and **mfspr** using SPR 944. User-level software can read the contents of MMCR2 by issuing an **mfspr** instruction to UMMCR2, described in Section 6.1.6.9.6, "User Monitor Mode Control Register 2 (UMMCR2)."

6.1.6.9.6 User Monitor Mode Control Register 2 (UMMCR2)

The contents of MMCR2 are reflected to UMMCR2, which can be read by user-level software. UMMCR2 can be accessed with the **mfspr** instruction using SPR 928.

6.1.6.9.7 Breakpoint Address Mask Register (BAMR)

The breakpoint address mask register (BAMR), shown in Figure 6-33, is used in conjunction with the events that monitor IABR hits.

Figure 6-33. Breakpoint Address Mask Register (BAMR)

Table 6-34 describes BAMR fields.

Table 6-34. BAMR Field Descriptions

Bit	Name	Description
0–29	MASK[1]	Used with PMC1 event (PMC1 event 42) that monitor IABR hits. The addresses to be compared for an IABR match are affected by the value in BAMR: • IABR hit (PMC1, event 42) occurs if IABR_CMP (that is, IABR AND BAMR) = instruction_address_compare (that is, EA AND BAMR) IABR_CMP[0–29] = IABR[0–29] AND BAMR[0–29] instruction_addr_cmp[0–29] = instruction_addr[0–29] AND BAMR[0–29] Be aware that breakpoint event 42 of PMC1 can be used to trigger performance monitor interrupts when the performance monitor detects an enabled overflow. This feature supports debug purposes and occurs only when IABR[30] is set. To avoid taking one of the above interrupts, make sure that IABR[30] is cleared.
30–31	—	Reserved

1 A context synchronizing instruction must follow the mtspr.

BAMR can be accessed with **mtspr** and **mfspr** using SPR 951. For synchronization requirements on the register see Section 6.3.2.4, “Synchronization.”

6.1.6.9.8 Performance Monitor Counter Registers (PMC1–PMC6)

PMC1–PMC6, shown in Figure 6-34, are 32-bit counters that can be programmed to generate a performance monitor interrupt when they overflow.

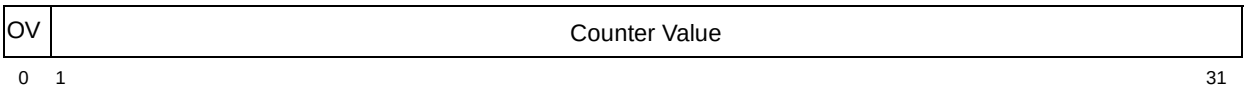


Figure 6-34. Performance Monitor Counter Registers (PMC1–PMC6)

The bits contained in the PMC registers are described in Table 6-35.

Table 6-35. PMC*n* Field Descriptions

Bits	Name	Description
0	OV	Overflow When this bit is set, it indicates that this counter has overflowed and reached its maximum value so that PMCn[OV] = 1.
1–31	Counter value	Counter value Indicates the number of occurrences of the specified event.

Counters overflow when the high-order (sign) bit becomes set; that is, they reach the value 2,147,483,648 (0x8000_0000). However, an interrupt is not generated unless both MMCR0[PMXE] and either MMCR0[PMC1CE] or MMCR0[PMC*c*CE] are also set as appropriate.

Note that the interrupt can be masked by clearing MSR[EE]; the performance monitor condition may occur with MSR[EE] cleared, but the interrupt is not taken until MSR[EE] is set. Setting MMCR0[FCECE] forces counters to stop counting when a counter interrupt or any enabled condition or event occurs. Setting

MMCR0[TRIGGER] forces counters PMC*n* (*n* > 1), to begin counting when PMC1 goes negative or an enabled condition or event occurs.

Software is expected to use the **mtspr** instruction to explicitly set PMC to non-overflowed values. Setting an overflowed value may cause an erroneous interrupt. For example, if both MMCR0[PMXE] and either MMCR0[PMC1CE] or MMCR0[PMC*n*CE] are set and the **mtspr** instruction loads an overflow value, an interrupt may be taken without an event counting having taken place.

The PMC registers can be accessed with the **mtspr** and **mfspr** instructions using the following SPR numbers:

- PMC1 is SPR 953
- PMC2 is SPR 954
- PMC3 is SPR 957
- PMC4 is SPR 958
- PMC5 is SPR 945
- PMC6 is SPR 946

6.1.6.9.9 User Performance Monitor Counter Registers (UPMC1–UPMC6)

The contents of the PMC1–PMC6 are reflected to UPMC1–UPMC6, which can be read by user-level software. The UPMC registers can be read with **mfspr** using the following SPR numbers:

- UPMC1 is SPR 937
- UPMC2 is SPR 938
- UPMC3 is SPR 941
- UPMC4 is SPR 942
- UPMC5 is SPR 929
- UPMC6 is SPR 930

6.1.6.9.10 Sampled Instruction Address Register (SIAR)

The sampled instruction address register (SIAR) is a supervisor-level register that contains the effective address of the last instruction to complete before the performance monitor interrupt is signaled. The SIAR is shown in Figure 6-35.

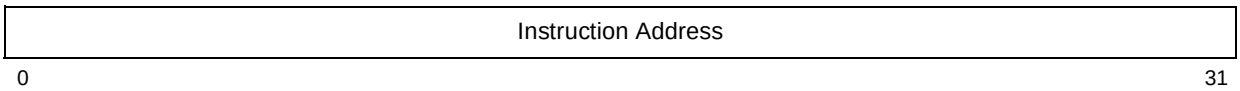


Figure 6-35. Sampled Instruction Address Registers (SIAR)

Note that SIAR is not updated in any of the following conditions:

- Performance monitor counting has been disabled by setting MMCR0[FC].
- Performance monitor interrupt has been disabled by clearing MMCR0[PMXE]

SIAR can be accessed with the **mtspr** and **mfspr** instructions using SPR 955.

6.1.6.9.11 User-Sampled Instruction Address Register (USIAR)

The contents of SIAR are reflected to USIAR, which can be read by user-level software. USIAR can be accessed with the **mfspr** instructions using SPR 939.

6.1.6.9.12 Sampled Data Address Register (SDAR) and User-Sampled Data Address Register (USDAR)

The e600 core does not implement the sampled data address register (SDAR) or the user-level, read-only USDA registers. Note that in previous processors the SDAR and USDAR registers could be written to by boot code without causing an interrupt, this is not the case in the e600 core. A **mtspr** or **mfspr** SDAR or USDAR instruction causes a program interrupt.

6.1.7 Reset Settings

Table 6-36 shows the state of the registers and other resources after a hard reset and before the first instruction is fetched from address 0xFFF0_0100 (the system reset interrupt vector). When a register is not initialized at hard reset, the setting is undefined.

Table 6-36. Settings Caused by Hard Reset (Used at Power-On)

Resource	Setting
BAMR	0x0000_0000
BATs	Undefined
Caches (L1/L2)	Disabled. The caches are not invalidated and must be invalidated in software before they are enabled.
CR	0x0000_0000
CTR	0x0000_0000
DABR	Breakpoint is disabled. Address is undefined.
DAR	0x0000_0000
DEC	0xFFFF_FFFF
DSISR	0x0000_0000
EAR	0x0000_0000
FPRs	Undefined
FPSCR	0x0000_0000
GPRs	Undefined
HID0	0x8000_0000
HID1	0x000*n_n*080 (Note that bits 14–19 are set to match the settings of *pll_cfg*[0:5] at reset. See the “Reset, Clocking, and Initialization” and “Global Utilities” chapters for more information.)
IABR	0x0000_0000 (breakpoint is disabled)
ICTC	0x0000_0000
ICTRL	0x0000_0000
L2CAPTDATAHI	0x0000_0000

Table 6-36. Settings Caused by Hard Reset (Used at Power-On) (continued)

Resource	Setting
L2CAPTDATALO	0x0000_0000
L2CAPTECC	0x0000_0000
L2ERRADDR	0x0000_0000
L2ERRATTR	0x0000_0000
L2ERRCTL	0x0000_0000
L2ERRDET	0x0000_0000
L2ERRDIS	0x0000_0000
L2ERREADDR	0x0000_0000
L2ERRINJCTL	0x0000_0000
L2ERRINJHI	0x0000_0000
L2ERRINJLO	0x0000_0000
L2ERRINTEN	0x0000_0000
LDSTCR	0x0000_0000
LR	0x0000_0000
MMCR*n*	0x0000_0000
MSSCR0	For the MPC8641D, 0x0000_8000 for core 0, 0x0000_8020 for core 1.
MSR	0x0000_0040 (only IP set)
S	0x0000_0000
PMC*n*	Undefined
PTEHI	0x0000_0000
PTELO	0x0000_0000
PVR	For the MPC8641D, 0x8004_0010.
Reservation address	Undefined
Reservation flag	Cleared
SDR1	0x0000_0000
SIAR	0x0000_0000
SPRG0–SPGR7	0x0000_0000
SRs	Undefined
SRR0	0x0000_0000
SRR1	0x0000_0000
TBU and TBL	0x0000_0000
TLBs	Undefined
TLBMISS	0x0000_0000

Table 6-36. Settings Caused by Hard Reset (Used at Power-On) (continued)

Resource	Setting
UMMCR*n*	0x0000_0000
UPMC*n*	0x0000_0000
USIAR	0x0000_0000
VRs	Undefined
VRSAVE	0x0000_0000
VSCR	0x0001_0000
XER	0x0000_0000

6.2 Operand Conventions

This section describes the operand conventions as they are represented in two levels of the PowerPC architecture—UISA and VEA. Detailed descriptions are provided of conventions used for storing values in registers and memory, accessing PowerPC registers, and representation of data in these registers.

6.2.1 Floating-Point Execution Models—UISA

The IEEE Std. 754 standard defines conventions for 64- and 32-bit arithmetic. The standard requires that single-precision arithmetic be provided for single-precision operands. The standard permits double-precision arithmetic instructions to have either (or both) single-precision or double-precision operands but states that single-precision arithmetic instructions should not accept double-precision operands.

The PowerPC UISA follows these guidelines:

- Double-precision arithmetic instructions can have single-precision operands but always produce double-precision results.
- Single-precision arithmetic instructions require all operands to be single-precision and always produce single-precision results.

For arithmetic instructions, conversion from double- to single-precision must be done explicitly by software, while conversion from single- to double-precision is done implicitly by the processor.

All implementations of the PowerPC architecture provide the equivalent of the following execution models to ensure that identical results are obtained. The definition of the arithmetic instructions for infinities, denormalized numbers, and NaNs follow conventions described in the following sections.

Although the double-precision format specifies an 11-bit exponent, exponent arithmetic uses two additional bit positions to avoid potential transient overflow conditions. An extra bit is required when denormalized double-precision numbers are prenormalized. A second bit is required to permit computation of the adjusted exponent value in the following examples when the corresponding interrupt enable bit has a value of 1:

- Underflow during multiplication using a denormalized operand
- Overflow during division using a denormalized divisor

6.2.2 Data Organization in Memory and Data Transfers

Bytes in memory are numbered consecutively starting with 0. Each number is the address of the corresponding byte.

Memory operands can be bytes, half-words, words, double-words, quad-words, or, for the load/store multiple and load/store string instructions, a sequence of bytes or words. The address of a memory operand is the address of its first byte (that is, of its lowest-numbered byte). Operand length is implicit for each instruction.

6.2.3 Alignment and Misaligned Accesses

The operand of a single-register memory access instruction has an alignment boundary equal to its length. An operand's address is misaligned if it is not a multiple of its width.

The concept of alignment is also applied more generally to data in memory. For example, a 12-byte data item is said to be word-aligned if its address is a multiple of four.

Some instructions require their memory operands to have certain alignment. In addition, alignment can affect performance. For single-register memory access instructions, the best performance is obtained when memory operands are aligned.

Instructions are 32 bits (one word) long and must be word-aligned.

The e600 core does not provide hardware support for floating-point memory that is not word-aligned. If a floating-point operand is not word-aligned, the e600 core invokes an alignment interrupt, and it is left up to software to break up the offending memory access operation appropriately. In addition, some non-double-word–aligned memory accesses suffer performance degradation as compared to an aligned access of the same type.

In general, floating-point word accesses should always be word-aligned and floating-point double-word accesses should always be double-word–aligned. Frequent use of misaligned accesses is discouraged because they can degrade overall performance.

6.2.4 Floating-Point Operands

The e600 core provides hardware support for all single- and double-precision floating-point operations for most value representations and all rounding modes. This architecture provides for hardware to implement a floating-point system as defined in ANSI/IEEE Standard 754-1985, *IEEE Standard for Binary Floating Point Arithmetic*. Detailed information about the floating-point execution model can be found in Chapter 3, "Operand Conventions," in the *Programming Environments Manual*.

The e600 core supports non-IEEE Std. 754 mode when FPSCR[29] is set. In this mode, denormalized numbers are treated in a non-IEEE Std. 754 conforming manner. This is accomplished by delivering results that are forced to the value zero.

6.3 Instruction Set Summary

This chapter describes instructions and addressing modes defined for the e600 core. These instructions are divided into the following functional categories:

- Integer instructions—These include arithmetic and logical instructions. For more information, see Section 6.3.4.1, “Integer Instructions.”
- Floating-point instructions—These include floating-point arithmetic instructions, as well as instructions that affect the floating-point status and control register (FPSCR). For more information, see Section 6.3.4.2, “Floating-Point Instructions.”

Load and store instructions—These include integer and floating-point load and store instructions. For more information, see Section 6.3.4.3, “Load and Store Instructions.”

- Flow control instructions—These include branching instructions, condition register logical instructions, trap instructions, and other instructions that affect the instruction flow. For more information, see Section 6.3.4.4, “Branch and Flow Control Instructions.”
- Processor control instructions—These instructions are used for synchronizing memory accesses and managing segment registers. For more information, see Section 6.3.4.6, “Processor Control Instructions—UISA,” Section 6.3.5.1, “Processor Control Instructions—VEA,” and Section 6.3.6.2, “Processor Control Instructions—OEA.”
- Memory synchronization instructions—These instructions are used for memory synchronizing. See Section 6.3.4.7, “Memory Synchronization Instructions—UISA,” and Section 6.3.5.2, “Memory Synchronization Instructions—VEA,” for more information.
- Memory control instructions—These instructions provide control of caches and TLBs. For more information, see Section 6.3.5.3, “Memory Control Instructions—VEA,” and Section 6.3.6.3, “Memory Control Instructions—OEA.”
- External control instructions—These include instructions for use with special input/output devices. For more information, see Section 6.3.5.4, “Optional External Control Instructions.”
- AltiVec instructions–AltiVec technology does not have optional instructions defined, so all instructions listed in the *AltiVec Technology Programming Environments Manual* are implemented for the e600 core. Instructions that are implementation-specific are described in Section 6.6.2, “AltiVec Instructions with Specific Implementations for the e600 Core.”

Note that this grouping of instructions does not necessarily indicate the execution unit that processes a particular instruction or group of instructions. This information, which is useful for scheduling instructions most effectively, is provided in the “Instruction Timing” chapter of the *e600 PowerPC Core Reference Manual*.

Integer instructions operate on word operands. Floating-point instructions operate on single-precision and double-precision floating-point operands. AltiVec instructions operate on byte, half-word, word, and quad-word operands. The PowerPC architecture uses instructions that are 4 bytes long and word-aligned. It provides for byte, half-word, and word operand loads and stores between memory and a set of 32 general-purpose registers (GPRs). It provides for word and double-word operand loads and stores between memory and a set of 32 floating-point registers (FPRs). It also provides for byte, half-word, word, and quad-word operand loads and stores between memory and a set of 32 vector registers (VRs).

Arithmetic and logical instructions do not read or modify memory. To use the contents of a memory location in a computation and then modify the same or another memory location, the memory contents must be loaded into a register, modified, and then written to the target location using load and store instructions.

The description of each instruction includes the mnemonic and a formatted list of operands. To simplify assembly language programming, a set of simplified mnemonics and symbols is provided for some of the frequently-used instructions; see Appendix F, “Simplified Mnemonics,” in the *Programming Environments Manual* for a complete list of simplified mnemonics. Programs written to be portable across the various assemblers for the PowerPC architecture should not assume the existence of mnemonics not described in that document.

6.3.1 Classes of Instructions

The e600 core instructions belong to one of the following three classes:

- Defined
- Illegal
- Reserved

Note that while the definitions of these terms are consistent among the processors that implement the PowerPC architecture, the assignment of these classifications is not. For example, PowerPC instructions defined for 64-bit implementations are treated as illegal by 32-bit implementations such as the e600 core.

The class is determined by examining the primary opcode and the extended opcode, if any. If the opcode, or combination of opcode and extended opcode, is not that of a defined instruction or of a reserved instruction, the instruction is illegal.

Instruction encodings that are now illegal can become assigned to instructions in the architecture or can be reserved by being assigned to processor-specific instructions.

6.3.1.1 Definition of Boundedly Undefined

If instructions are encoded with incorrectly set bits in reserved fields, the results on execution can be said to be boundedly undefined. If a user-level program executes the incorrectly coded instruction, the resulting undefined results are bounded in that a spurious change from user to supervisor state is not allowed, and the level of privilege exercised by the program in relation to memory access and other system resources cannot be exceeded. Boundedly undefined results for a given instruction can vary between implementations and between execution attempts in the same implementation.

6.3.1.2 Defined Instruction Class

Defined instructions are guaranteed to be supported in all implementations of the PowerPC architecture, except as stated in the instruction descriptions in Chapter 8, “Instruction Set,” of the *Programming Environments Manual*. The e600 core provides hardware support for all instructions defined for 32-bit implementations. It does not support the optional **fsqrt**, **fsqrts**, and **tlbia** instructions.

A processor invokes the illegal instruction error handler (part of the program interrupt) when it encounters a PowerPC instruction that has not been implemented. The instruction can be emulated in software, as required.

A defined instruction can have invalid forms. The e600 core provides limited support for instructions represented in an invalid form.

6.3.1.3 Illegal Instruction Class

Illegal instructions can be grouped into the following categories:

- Instructions not defined in the PowerPC architecture.The following primary opcodes are defined as illegal, but can be used in future extensions to the architecture:
  1, 5, 6, 9, 22, 56, 57, 60, 61

  Future versions of the PowerPC architecture can define any of these instructions to perform new functions.
- Instructions defined in the PowerPC architecture but not implemented in a specific implementation. For example, instructions that can be executed on 64-bit processors that implement the PowerPC architecture are considered illegal by 32-bit processors such as the e600 core.

  The following primary opcodes are defined for 64-bit implementations only and are illegal on the e600 core:

  2, 30, 58, 62
- All unused extended opcodes are illegal. The unused extended opcodes can be determined from information in the “e600 Core Instruction Set Listings” appendix of the *e600 PowerPC Core Reference Manual* and Section 6.3.1.4, “Reserved Instruction Class.” Notice that extended opcodes for instructions defined only for 64-bit implementations are illegal in 32-bit implementations, and vice versa. The following primary opcodes have unused extended opcodes:

  17, 19, 31, 59, 63 (Primary opcodes 30 and 62 are illegal for all 32-bit implementations, but as 64-bit opcodes, they have some unused extended opcodes.)
- An instruction consisting of only zeros is guaranteed to be an illegal instruction. This increases the probability that an attempt to execute data or memory that was not initialized invokes the system illegal instruction error handler (a program interrupt). Note that if only the primary opcode consists of all zeros, the instruction is considered a reserved instruction, as described in Section 6.3.1.4, “Reserved Instruction Class.”

The e600 core invokes the system illegal instruction error handler (a program interrupt) when it detects any instruction from this class or any instructions defined only for 64-bit implementations.

See the “Interrupts” chapter of the *e600 PowerPC Core Reference Manual* for additional information about illegal and invalid instruction interrupts. Except for an instruction consisting of binary zeros, illegal instructions are available for additions to the PowerPC architecture.

6.3.1.4 Reserved Instruction Class

Reserved instructions are allocated to specific implementation-dependent purposes not defined by the PowerPC architecture. Attempting to execute a reserved instruction that has not been implemented invokes the illegal instruction error handler (a program interrupt). See “Program Interrupt (0x0_0700),” in Chapter 6, “Interrupts,” in the *Programming Environments Manual* for information about illegal and invalid instruction interrupts.

The PowerPC architecture defines four types of reserved instructions:

- Instructions in the POWER architecture not part of the PowerPC UISA. For details on POWER architecture incompatibilities and how they are handled by processors that implement the PowerPC architecture, see Appendix B, “POWER Architecture Cross Reference,” in the *Programming Environments Manual*.
- Implementation-specific instructions required for the processor to conform to the PowerPC architecture (none of these are implemented in the e600 core)
- All other implementation-specific instructions
- Architecturally allowed extended opcodes

6.3.2 Addressing Modes

This section provides an overview of conventions for addressing memory and for calculating effective addresses as defined by the PowerPC architecture for 32-bit implementations. For more detailed information, see “Conventions,” in Chapter 4, “Addressing Modes and Instruction Set Summary,” of the *Programming Environments Manual*.

6.3.2.1 Memory Addressing

A program references memory using the effective (logical) address computed by the processor when it executes a memory access or branch instruction or when it fetches the next sequential instruction.

Bytes in memory are numbered consecutively starting with zero. Each number is the address of the corresponding byte.

6.3.2.2 Memory Operands

Memory operands can be bytes, half-words, words, double-words, quad-words or, for the load/store multiple and load/store string instructions, a sequence of bytes or words. The address of a memory operand is the address of its first byte (that is, of its lowest-numbered byte). Operand length is implicit for each instruction. The PowerPC architecture supports both big-endian and little-endian byte ordering. The default byte and bit ordering is big endian. See “Byte Ordering,” in Chapter 3, “Operand Conventions,” of the *Programming Environments Manual* for more information about big- and little-endian byte ordering.

The operand of a single-register memory access instruction has a natural alignment boundary equal to the operand length; that is, the natural address of an operand is an integral multiple of its length. A memory operand is said to be aligned if it is aligned at its natural boundary; otherwise it is misaligned. For a detailed discussion about memory operands, see Chapter 3, “Operand Conventions,” of the *Programming Environments Manual*.

6.3.2.3 Effective Address Calculation

An effective address is the 32-bit sum computed by the processor when executing a memory access or branch instruction or when fetching the next sequential instruction. For a memory access instruction, if the sum of the effective address and the operand length exceeds the maximum effective address, the memory operand is considered to wrap around from the maximum effective address through effective address 0, as described in the following paragraphs.

Effective address computations for both data and instruction accesses use 32-bit unsigned binary arithmetic. A carry from bit 0 is ignored.

Load and store operations have the following modes of effective address generation:

- EA = (**r**A|0) + offset (including offset = 0) (register indirect with immediate index)
- EA = (**r**A|0) + **r**B (register indirect with index)

Refer to Section 6.3.4.3.2, “Integer Load and Store Address Generation,” for a detailed description of effective address generation for load and store operations.

Branch instructions have three categories of effective address generation:

- Immediate
- Link register indirect
- Count register indirect

6.3.2.4 Synchronization

The synchronization described in this section refers to the state of the processor that is performing the synchronization.

6.3.2.4.1 Context Synchronization

The System Call (**sc**) and Return from Interrupt (**rfi**) instructions perform context synchronization by allowing previously issued instructions to complete before performing a change in context. Execution of one of these instructions ensures the following:

- No higher priority interrupt exists (**sc**).
- All previous instructions have completed to a point where they can no longer cause an interrupt. If a prior memory access instruction causes direct-store error interrupts, the results are guaranteed to be determined before this instruction is executed.
- Previous instructions complete execution in the context (privilege, protection, and address translation) under which they were issued.
- The instructions following the **sc** or **rfi** instruction execute in the context established by these instructions.

Modifying certain registers requires software synchronization to follow certain register dependencies. Table 6-37 defines specific synchronization procedures that are required when using various SPRs and specific bits within SPRs. Context synchronizing instructions that can be used are: **isync**, **sc**, **rfi**, and any interrupt other than system reset and machine check. If multiple bits are being modified that have different synchronization requirements, the most restrictive requirements can be used. However, a **mtspr**

instruction to modify either HID0[ICE] or HID0[ICFI] should not also modify other HID0 bits that requires synchronization.

Table 6-37. Control Registers Synchronization Requirements

Register	Bits	Synchronization Requirements
BAMR	Any	A context synchronizing instruction must follow the **mtspr**.
DABR	Any	A **dssall** and **sync** must precede the **mtspr** and then a **sync** and a context synchronizing instruction must follow. Note that if a user is not using the AltiVec data streaming instructions, then a **dssall** is not necessary prior to accessing the register.
DBATs	Any	A **dssall** and **sync** must precede the **mtspr** and then a **sync** and a context synchronizing instruction must follow. Note that if a user is not using the AltiVec data streaming instructions, then a **dssall** is not necessary prior to accessing the register.
EAR	Any	A **dssall** and **sync** must precede the **mtspr** and then a **sync** and a context synchronizing instruction must follow. Note that if a user is not using the AltiVec data streaming instructions, then a **dssall** is not necessary prior to accessing register.

Table 6-37. Control Registers Synchronization Requirements (continued)

Register	Bits	Synchronization Requirements
HID0	BHTCLR	A context synchronizing instruction must precede an **mtspr** and a branch instruction should follow. The branch instruction may be either conditional or unconditional. It ensures that all subsequent branch instructions see the newly initialized BHT values. For correct results, the BHT should be disabled (HID0[BHT] = 0) before setting BHTCLR.
	BHT	A context synchronizing instruction must follow the **mtspr**.
	BTIC	
	DPM	
	FOLD	
	LRSTK	
	NAP	
	NHR	
	SLEEP	
	SPD	
	TBEN	
	DCE	A **dssall** and **sync** must precede an **mtspr** and then a **sync** and context synchronizing instruction must follow. Note that if a user is not using the AltiVec data streaming instructions, then a **dssall** is not necessary prior to accessing the HID0[DCE] or HID0[DCFI] bit.
	DCFI	
	DLOCK	
	NOPDST	
	STEN	
	ICE	A context synchronizing instruction must immediately follow an **mtspr**. An **mtspr** instruction for HID0 should not modify either of these bits at the same time it modifies another bit that requires additional synchronization.
	ICFI	
	ILOCK	A context synchronizing instruction must precede and follow an **mtspr**.
	NOPTI	An **mtspr** must follow a **sync** and a context synchronizing instruction.
	SGE	
	XAEN	A **dssall** and **sync** must precede an **mtspr** and then a **sync** and a context-synchronizing instruction must follow. Alteration of HID0[XAEN] must be done with caches and translation disabled. The caches and TLBs must be flushed before they are re-enabled after the XAEN bit is altered. Note that if a user is not using the AltiVec data streaming instructions, then a **dssall** is not necessary prior to accessing the HID0[XAEN] bit.
HID1	Any	A **sync** and context synchronizing instruction must follow an **mtspr**.
IABR	Any	A context synchronizing instruction must follow an **mtspr**.
IBATs	Any	A context synchronizing instruction must follow an **mtspr**.

Table 6-37. Control Registers Synchronization Requirements (continued)

Register	Bits	Synchronization Requirements
ICTRL	EDCE	A **dssall** and **sync** must precede an **mtspr** and then a **sync** and context synchronizing instruction must follow. Note that if a user is not using the AltiVec data streaming instructions, then a **dssall** is not necessary prior to accessing the ICTRL[EDCE] bit.
	ICWL	A context synchronizing instruction must precede and follow an **mtspr**.
	EICE	
L2ERRDIS	Any	A **sync** must precede an **mtspr** and then a **sync** and **isync** must follow.
LDSTCR	Any	A **dssall** and **sync** must precede an **mtspr** and then a **sync** and context synchronizing instruction must follow. Note that if a user is not using the AltiVec data streaming instructions, then a **dssall** is not necessary prior to accessing the register.
MSR	BE	A context synchronizing instruction must follow an **mtmsr** instruction.
	VEC	
	FE0	
	FE1	
	FP	
	SE	
	IR	A context synchronizing instruction must follow an **mtmsr**. When changing the MSR[IR] bit the context synchronizing instruction must reside at both the untranslated and the translated address following the **mtmsr**.
	DR	A **dssall** and **sync** must precede an **mtmsr** and then a **sync** and context synchronizing instruction must follow. Note that if a user is not using the AltiVec data streaming instructions, then a **dssall** is not necessary prior to accessing the MSR[DR] or MSR[PR] bit.
	PR	
	LE	A **dssall** and **sync** must precede an **rfi** to guarantee a solid context boundary. Note that if a user is not using AltiVec data streaming instructions, a **dssall** is not necessary before accessing MSR[LE].
	POW	A **dssall** and **sync** must precede an **mtmsr** instruction and then a context synchronizing instruction must follow.
MSSCR0	Any	A **dssall** and **sync** must precede an **mtspr** instruction and then a **sync** and context synchronizing instruction must follow. Note that if a user is not using the AltiVec data streaming instructions, then a **dssall** is not necessary prior to accessing the register.
SDR1	Any	A **dssall** and **sync** must precede an **mtspr** and then a **sync** and context synchronizing instruction must follow. Note that if a user is not using the AltiVec data streaming instructions, then a **dssall** is not necessary prior to accessing the register.
SR0–SR15	Any	A **dssall** and **sync** must precede an **mtsr** or **mtsrin** instruction and then a **sync** and context synchronizing instruction must follow. Note that if a user is not using the AltiVec data streaming instructions, then a **dssall** is not necessary prior to accessing the register.
Other registers or bits	—	No special synchronization requirements.

6.3.2.4.2 Execution Synchronization

An instruction is execution synchronizing if all previously initiated instructions appear to have completed before the instruction is initiated or, in the case of **sync** and **isync**, before the instruction completes. For example, the Move to Machine State Register (**mtmsr**) instruction is execution synchronizing. It ensures that all preceding instructions have completed execution and cannot cause an interrupt before the instruction executes, but does not ensure subsequent instructions execute in the newly established environment. For example, if the **mtmsr** sets the MSR[PR] bit, unless an **isync** immediately follows the **mtmsr** instruction, a privileged instruction could be executed or privileged access could be performed without causing an interrupt even though MSR[PR] indicates user mode.

6.3.2.4.3 Instruction-Related Interrupts

There are two kinds of interrupts in the e600 core—those caused directly by the execution of an instruction and those caused by an asynchronous event. Either can cause components of the system software to be invoked.

Interrupts can be caused directly by the execution of an instruction as follows:

- An attempt to execute an illegal instruction causes the illegal instruction (program interrupt) handler to be invoked. An attempt by a user-level program to execute the supervisor-level instructions listed below causes the privileged instruction (program interrupt) handler to be invoked. The e600 core provides the following supervisor-level instructions—**dcbi**, **mfmsr**, **mfspr**, **mfsr**, **mfsrin**, **mtmsr**, **mtspr**, **mtsr**, **mtsrin**, **rfi**, **tlbie**, and **tlbsync**. Note that the privilege level of the **mfspr** and **mtspr** instructions depends on the SPR encoding.
- Any **mtspr**, **mfspr**, or **mftb** instruction with an invalid SPR (or TBR) field causes an illegal type program interrupt. Likewise, a program interrupt is taken if user-level software tries to access a supervisor-level SPR. An **mtspr** instruction executing in supervisor mode (MSR[PR] = 0) with the SPR field specifying PVR (read-only register) executes as a no-op.
- An attempt to access memory that is not available (page fault) causes the ISI or DSI interrupt handler to be invoked.
- The execution of an **sc** instruction invokes the system call interrupt handler that permits a program to request the system to perform a service.
- The execution of a trap instruction invokes the program interrupt trap handler.
- The execution of an instruction that causes a floating-point exception while exceptions are enabled in the MSR invokes the program interrupt handler.

A detailed description of exception conditions is provided in the “Interrupts” chapter of the *e600 PowerPC Core Reference Manual.*

6.3.3 Instruction Set Overview

This section provides a brief overview of the PowerPC instructions implemented in the e600 core and highlights any special information with respect to how the e600 core implements a particular instruction. Note that the categories used in this section correspond to those used in Chapter 4, “Addressing Modes and Instruction Set Summary,” in the *Programming Environments Manual*. These categorizations are somewhat arbitrary, are provided for the convenience of the programmer, and do not necessarily reflect the PowerPC architecture specification.

Note that some instructions have the following optional features:

- CR Update—The dot (**.**) suffix on the mnemonic enables the update of the CR.
- Overflow option—The **o** suffix indicates that the overflow bit in the XER is enabled.

6.3.4 PowerPC UISA Instructions

The PowerPC UISA includes the base user-level instruction set (excluding a few user-level cache control, synchronization, and time base instructions), user-level registers, programming model, data types, and addressing modes. This section discusses the instructions defined in the UISA.

6.3.4.1 Integer Instructions

This section describes the integer instructions. These consist of the following:

- Integer arithmetic instructions
- Integer compare instructions
- Integer logical instructions
- Integer rotate and shift instructions

Integer instructions use the content of the GPRs as source operands and place results into GPRs, the XER register, and condition register (CR) fields.

6.3.4.1.1 Integer Arithmetic Instructions

Table 6-38 lists the integer arithmetic instructions for the processors that implement the PowerPC architecture.

Table 6-38. Integer Arithmetic Instructions

Name	Mnemonic	Syntax
Add Immediate	**addi**	**r**D,**r**A,SIMM
Add Immediate Shifted	**addis**	**r**D,**r**A,SIMM
Add	**add** (**add. addo addo.**)	**r**D,**r**A,**r**B
Subtract From	**subf** (**subf. subfo subfo.**)	**r**D,**r**A,**r**B
Add Immediate Carrying	**addic**	**r**D,**r**A,SIMM
Add Immediate Carrying and Record	**addic.**	**r**D,**r**A,SIMM
Subtract from Immediate Carrying	**subfic**	**r**D,**r**A,SIMM
Add Carrying	**addc** (**addc. addco addco.**)	**r**D,**r**A,**r**B
Subtract from Carrying	**subfc** (**subfc. subfco subfco.**)	**r**D,**r**A,**r**B
Add Extended	**adde** (**adde. addeo addeo.**)	**r**D,**r**A,**r**B
Subtract from Extended	**subfe** (**subfe. subfeo subfeo.**)	**r**D,**r**A,**r**B
Add to Minus One Extended	**addme** (**addme. addmeo addmeo.**)	**r**D,**r**A
Subtract from Minus One Extended	**subfme** (**subfme. subfmeo subfmeo.**)	**r**D,**r**A

Table 6-38. Integer Arithmetic Instructions (continued)

Name	Mnemonic	Syntax
Add to Zero Extended	**addze** (**addze. addzeo addzeo.**)	**r**D,**r**A
Subtract from Zero Extended	**subfze** (**subfze. subfzeo subfzeo.**)	**r**D,**r**A
Negate	**neg** (**neg. nego nego.**)	**r**D,**r**A
Multiply Low Immediate	**mulli**	**r**D,**r**A,SIMM
Multiply Low Word	**mullw** (**mullw. mullwo mullwo.**)	**r**D,**r**A,**r**B
Multiply High Word	**mulhw** (**mulhw.**)	**r**D,**r**A,**r**B
Multiply High Word Unsigned	**mulhwu** (**mulhwu.**)	**r**D,**r**A,**r**B
Divide Word	**divw** (**divw. divwo divwo.**)	**r**D,**r**A,**r**B
Divide Word Unsigned	**divwu divwu. divwuo divwuo.**	**r**D,**r**A,**r**B

Although there is no Subtract Immediate instruction, its effect can be achieved by using an **addi** instruction with the immediate operand negated. Simplified mnemonics are provided that include this negation. The **subf** instructions subtract the second operand (**r**A) from the third operand (**r**B). Simplified mnemonics are provided in which the third operand is subtracted from the second operand. See Appendix F, “Simplified Mnemonics,” in the *Programming Environments Manual* for examples.

The UISA states that an implementation that executes instructions that set the overflow enable bit (OE) or the carry bit (CA) can either execute these instructions slowly or prevent execution of the subsequent instruction until the operation completes. The “Instruction Timing” chapter of the *e600 PowerPC Core Reference Manual* describes how the e600 core handles CR dependencies. The summary overflow bit (SO) and overflow bit (OV) in the XER register are set to reflect an overflow condition of a 32-bit result. This can happen only when OE = 1.

6.3.4.1.2 Integer Compare Instructions

The integer compare instructions algebraically or logically compare the contents of register **r**A with either the zero-extended value of the UIMM operand, the sign-extended value of the SIMM operand, or the contents of **r**B. The comparison is signed for the **cmpi** and **cmp** instructions, and unsigned for the **cmpli** and **cmpl** instructions. Table 6-39 summarizes the integer compare instructions.

Table 6-39. Integer Compare Instructions

Name	Mnemonic	Syntax
Compare Immediate	**cmpi**	**crf**D,L,**r**A,SIMM
Compare	**cmp**	**crf**D,L,**r**A,**r**B
Compare Logical Immediate	**cmpli**	**crf**D,L,**r**A,UIMM
Compare Logical	**cmpl**	**crf**D,L,**r**A,**r**B

The **crf**D operand can be omitted if the result of the comparison is to be placed in CR0. Otherwise the target CR field must be specified in **crf**D, using an explicit field number.

For information on simplified mnemonics for the integer compare instructions see Appendix F, “Simplified Mnemonics,” in the *Programming Environments Manual*.

6.3.4.1.3 Integer Logical Instructions

The logical instructions shown in Table 6-40 perform bit-parallel operations on the specified operands. Logical instructions with the CR updating enabled (uses dot suffix) and instructions **andi.** and **andis.** set CR field CR0 to characterize the result of the logical operation. Logical instructions do not affect XER[SO], XER[OV], or XER[CA].

See Appendix F, “Simplified Mnemonics,” in the *Programming Environments Manual* for simplified mnemonic examples for integer logical operations.

Table 6-40. Integer Logical Instructions

Name	Mnemonic	Syntax	Implementation Notes
AND Immediate	**andi.**	**r**A,**r**S,UIMM	—
AND Immediate Shifted	**andis.**	**r**A,**r**S,UIMM	—
OR Immediate	**ori**	**r**A,**r**S,UIMM	The PowerPC architecture defines **ori r**0,**r**0,0 as the preferred form for the no-op instruction. The dispatcher discards this instruction and only dispatches it to the completion queue, but not to any execution unit.
OR Immediate Shifted	**oris**	**r**A,**r**S,UIMM	—
XOR Immediate	**xori**	**r**A,**r**S,UIMM	—
XOR Immediate Shifted	**xoris**	**r**A,**r**S,UIMM	—
AND	**and** (**and**.)	**r**A,**r**S,**r**B	—
OR	**or (or.)**	**r**A,**r**S,**r**B	—
XOR	**xor (xor.)**	**r**A,**r**S,**r**B	—
NAND	**nand (nand.)**	**r**A,**r**S,**r**B	—
NOR	**nor (nor.)**	**r**A,**r**S,**r**B	—
Equivalent	**eqv (eqv.)**	**r**A,**r**S,**r**B	—
AND with Complement	**andc (andc.)**	**r**A,**r**S,**r**B	—
OR with Complement	**orc (orc.)**	**r**A,**r**S,**r**B	—
Extend Sign Byte	**extsb (extsb.)**	**r**A,**r**S	—
Extend Sign Half Word	**extsh (extsh.)**	**r**A,**r**S	—
Count Leading Zeros Word	**cntlzw (cntlzw.)**	**r**A,**r**S	—

6.3.4.1.4 Integer Rotate and Shift Instructions

Rotation operations are performed on data from a GPR, and the result, or a portion of the result, is returned to a GPR. See Appendix F, “Simplified Mnemonics,” in the *Programming Environments Manual* for a complete list of simplified mnemonics that allows simpler coding of often-used functions such as clearing

the leftmost or rightmost bits of a register, left-justifying or right-justifying an arbitrary field, and simple rotates and shifts.

Integer rotate instructions rotate the contents of a register. The result of the rotation is either inserted into the target register under control of a mask (if a mask bit is 1 the associated bit of the rotated data is placed into the target register, and if the mask bit is 0 the associated bit in the target register is unchanged), or ANDed with a mask before being placed into the target register.

The integer rotate instructions are summarized in Table 6-41.

Table 6-41. Integer Rotate Instructions

Name	Mnemonic	Syntax
Rotate Left Word Immediate then AND with Mask	**rlwinm** (**rlwinm.**)	**r**A,**r**S,SH,MB,ME
Rotate Left Word then AND with Mask	**rlwnm** (**rlwnm.**)	**r**A,**r**S,**r**B,MB,ME
Rotate Left Word Immediate then Mask Insert	**rlwimi** (**rlwimi.**)	**r**A,**r**S,SH,MB,ME

The integer shift instructions perform left and right shifts. Immediate-form logical (unsigned) shift operations are obtained by specifying masks and shift values for certain rotate instructions. Simplified mnemonics (shown in Appendix F, “Simplified Mnemonics,” in the *Programming Environments Manual*) are provided to make coding of such shifts simpler and easier to understand.

Multiple-precision shifts can be programmed as shown in Appendix C, “Multiple-Precision Shifts,” in the *Programming Environments Manual*. The integer shift instructions are summarized in Table 6-42.

Table 6-42. Integer Shift Instructions

Name	Mnemonic	Syntax
Shift Left Word	**slw** (**slw.**)	**r**A,**r**S,**r**B
Shift Right Word	**srw** (**srw.**)	**r**A,**r**S,**r**B
Shift Right Algebraic Word Immediate	**srawi** (**srawi.**)	**r**A,**r**S,SH
Shift Right Algebraic Word	**sraw** (**sraw.**)	**r**A,**r**S,**r**B

6.3.4.2 Floating-Point Instructions

This section describes the floating-point instructions, which include the following:

- Floating-point arithmetic instructions
- Floating-point multiply-add instructions
- Floating-point rounding and conversion instructions
- Floating-point compare instructions
- Floating-point status and control register instructions
- Floating-point move instructions

See Section 6.3.4.3, “Load and Store Instructions,” for information about floating-point loads and stores.

The PowerPC architecture supports a floating-point system as defined in IEEE Std. 754, but requires software support to conform with that standard. All floating-point operations conform to IEEE Std.754, except if software sets the non-IEEE mode bit (FPSCR[NI]).

6.3.4.2.1 Floating-Point Arithmetic Instructions

The floating-point arithmetic instructions are summarized in Table 6-43.

Table 6-43. Floating-Point Arithmetic Instructions

Name	Mnemonic	Syntax
Floating Add (Double-Precision)	**fadd (fadd.)**	**fr**D,**fr**A,**fr**B
Floating Add Single	**fadds (fadds.)**	**fr**D,**fr**A,**fr**B
Floating Subtract (Double-Precision)	**fsub (fsub.)**	**fr**D,**fr**A,**fr**B
Floating Subtract Single	**fsubs (fsubs.)**	**fr**D,**fr**A,**fr**B
Floating Multiply (Double-Precision)	**fmul (fmul.)**	**fr**D,**fr**A,**fr**C
Floating Multiply Single	**fmuls (fmuls.)**	**fr**D,**fr**A,**fr**C
Floating Divide (Double-Precision)	**fdiv (fdiv.)**	**fr**D,**fr**A,**fr**B
Floating Divide Single	**fdivs (fdivs.)**	**fr**D,**fr**A,**fr**B
Floating Reciprocal Estimate Single[1]	**fres (fres.)**	**fr**D,**fr**B
Floating Reciprocal Square Root Estimate[1]	**frsqrte (frsqrte.)**	**fr**D,**fr**B
Floating Select[1]	**fsel**	**fr**D,**fr**A,**fr**C,**fr**B

[1] These instructions are optional in the PowerPC architecture.

All single-precision arithmetic instructions are performed using a double-precision format. The floating-point architecture is a single-pass implementation for double-precision products. In most cases, a single-precision instruction using only single-precision operands, in double-precision format, has the same latency as its double-precision equivalent.

6.3.4.2.2 Floating-Point Multiply-Add Instructions

These instructions combine multiply and add operations without an intermediate rounding operation. The floating-point multiply-add instructions are summarized in Table 6-44.

Table 6-44. Floating-Point Multiply-Add Instructions

Name	Mnemonic	Syntax
Floating Multiply-Add (Double-Precision)	**fmadd (fmadd.)**	**fr**D,**fr**A,**fr**C,**fr**B
Floating Multiply-Add Single	**fmadds (fmadds.)**	**fr**D,**fr**A,**fr**C,**fr**B
Floating Multiply-Subtract (Double-Precision)	**fmsub (fmsub.)**	**fr**D,**fr**A,**fr**C,**fr**B
Floating Multiply-Subtract Single	**fmsubs (fmsubs.)**	**fr**D,**fr**A,**fr**C,**fr**B
Floating Negative Multiply-Add (Double-Precision)	**fnmadd (fnmadd.)**	**fr**D,**fr**A,**fr**C,**fr**B
Floating Negative Multiply-Add Single	**fnmadds (fnmadds.)**	**fr**D,**fr**A,**fr**C,**fr**B

Table 6-44. Floating-Point Multiply-Add Instructions (continued)

Name	Mnemonic	Syntax
Floating Negative Multiply-Subtract (Double-Precision)	**fnmsub (fnmsub.)**	**fr**D,**fr**A,**fr**C,**fr**B
Floating Negative Multiply-Subtract Single	**fnmsubs (fnmsubs.)**	**fr**D,**fr**A,**fr**C,**fr**B

6.3.4.2.3 Floating-Point Rounding and Conversion Instructions

The Floating Round to Single-Precision (**frsp**) instruction is used to truncate a 64-bit double-precision number to a 32-bit single-precision floating-point number. The floating-point convert instructions convert a 64-bit double-precision floating-point number to a 32-bit signed integer number.

Examples of uses of these instructions to perform various conversions can be found in Appendix D, “Floating-Point Models,” in the *Programming Environments Manual*.

Table 6-45. Floating-Point Rounding and Conversion Instructions

Name	Mnemonic	Syntax
Floating Round to Single	**frsp (frsp.)**	**fr**D,**fr**B
Floating Convert to Integer Word	**fctiw (fctiw.)**	**fr**D,**fr**B
Floating Convert to Integer Word with Round toward Zero	**fctiwz (fctiwz.)**	**fr**D,**fr**B

6.3.4.2.4 Floating-Point Compare Instructions

Floating-point compare instructions compare the contents of two floating-point registers. The comparison ignores the sign of zero (that is +0 = –0). The floating-point compare instructions are summarized in Table 6-46.

Table 6-46. Floating-Point Compare Instructions

Name	Mnemonic	Syntax
Floating Compare Unordered	**fcmpu**	**crf**D,**fr**A,**fr**B
Floating Compare Ordered	**fcmpo**	**crf**D,**fr**A,**fr**B

6.3.4.2.5 Floating-Point Status and Control Register Instructions

Every FPSCR instruction appears to synchronize the effects of all floating-point instructions executed by a given processor. Executing an FPSCR instruction ensures that all floating-point instructions previously initiated by the given processor appear to have completed before the FPSCR instruction is initiated and that no subsequent floating-point instructions appear to be initiated by the given processor until the FPSCR instruction has completed. The FPSCR instructions are summarized in Table 6-47.

Table 6-47. Floating-Point Status and Control Register Instructions

Name	Mnemonic	Syntax
Move from FPSCR	**mffs (mffs.)**	**fr**D
Move to Condition Register from FPSCR	**mcrfs**	**crf**D,**crf**S

Table 6-47. Floating-Point Status and Control Register Instructions (continued)

Name	Mnemonic	Syntax
Move to FPSCR Field Immediate	**mtfsfi (mtfsfi.)**	**crf**D,IMM
Move to FPSCR Fields	**mtfsf (mtfsf.)**	FM,**fr**B
Move to FPSCR Bit 0	**mtfsb0 (mtfsb0.)**	**crb**D
Move to FPSCR Bit 1	**mtfsb1 (mtfsb1.)**	**crb**D

Implementation Note—The PowerPC architecture states that in some implementations, the Move to FPSCR Fields (**mtfsf**) instruction can perform more slowly when only some of the fields are updated as opposed to all of the fields. In the e600 core, there is no degradation of performance.

6.3.4.2.6 Floating-Point Move Instructions

Floating-point move instructions copy data from one FPR to another. The floating-point move instructions do not modify the FPSCR. The CR update option in these instructions controls the placing of result status into CR1. Table 6-48 summarizes the floating-point move instructions.

Table 6-48. Floating-Point Move Instructions

Name	Mnemonic	Syntax
Floating Move Register	**fmr (fmr.)**	**fr**D,**fr**B
Floating Negate	**fneg (fneg.)**	**fr**D,**fr**B
Floating Absolute Value	**fabs (fabs.)**	**fr**D,**fr**B
Floating Negative Absolute Value	**fnabs (fnabs.)**	**fr**D,**fr**B

6.3.4.3 Load and Store Instructions

Load and store instructions are issued and translated in program order; however, the accesses can occur out of order. Synchronizing instructions are provided to enforce strict ordering. This section describes the load and store instructions, which consist of the following:

- Integer load instructions
- Integer store instructions
- Integer load and store with byte-reverse instructions
- Integer load and store multiple instructions
- Floating-point load instructions
- Floating-point store instructions
- Memory synchronization instructions

Implementation Note—The following describes how the e600 core handles misalignment:

The e600 core provides hardware support for misaligned memory accesses. It performs those accesses within a single cycle if the operand lies within a double-word boundary. Misaligned memory accesses that cross a double-word boundary degrade performance.

Although many misaligned memory accesses are supported in hardware, the frequent use of them is discouraged because they can compromise the overall performance of the processor. Only one outstanding misalignment at a time is supported which means it is non-pipelined.

Accesses that cross a translation boundary can be restarted. That is, a misaligned access that crosses a page boundary is completely restarted if the second portion of the access causes a page fault. This can cause the first access to be repeated.

On some processors, such as the MPC603e, a TLB reload operation causes an instruction restart. On the e600 core, TLB reloads are performed transparently (if hardware table search operations are enabled—HID0[STEN] = 0) and only a page fault causes a restart. If software table searching is enabled (HID0[STEN] = 1) on the e600 core, a TLB miss causes an instruction restart (as it causes a TLB miss interrupt)

6.3.4.3.1 Self-Modifying Code

When a processor modifies a memory location that can be contained in the instruction cache, software must ensure that memory updates are visible to the instruction fetching mechanism. This can be achieved by executing the following instruction sequence (using either **dcbst** or **dcbf**):

```
dcbst (or dcbf)|update memory
sync     |wait for update
icbi     |remove (invalidate) copy in instruction cache
sync     |ensure that ICBI invalidate at the icache has completed
isync    |remove copy in own instruction buffer
```

These operations are required because the data cache is a write-back cache. Because instruction fetching bypasses the data cache, changes to items in the data cache cannot be reflected in memory until the fetch operations complete. The **sync** after the **icbi** is required to ensure that the **icbi** invalidation has completed in the instruction cache.

Special care must be taken to avoid coherency paradoxes in systems that implement unified secondary caches (like the e600 core), and designers should carefully follow the guidelines for maintaining cache coherency that are provided in the VEA, and discussed in Chapter 5, “Cache Model and Memory Coherency,” in the *Programming Environments Manual*.

6.3.4.3.2 Integer Load and Store Address Generation

Integer load and store operations generate effective addresses using register indirect with immediate index mode, register indirect with index mode, or register indirect mode. See Section 6.3.2.3, “Effective Address Calculation,” for information about calculating effective addresses. Note that in some implementations, operations that are not naturally aligned can suffer performance degradation. Refer to the “Interrupts” chapter of the *e600 PowerPC Core Reference Manual* for additional information about load and store address alignment interrupts.

6.3.4.3.3 Register Indirect Integer Load Instructions

For integer load instructions, the byte, half-word, word, or double-word addressed by the EA (effective address) is loaded into **r**D. Many integer load instructions have an update form, in which **r**A is updated with the generated effective address. For these forms, if **r**A ≠ 0 and **r**A ≠ **r**D (otherwise invalid), the EA

is placed into **r**A and the memory element (byte, half-word, word, or double-word) addressed by the EA is loaded into **r**D. Note that the PowerPC architecture defines load with update instructions with operand **r**A = 0 or **r**A = **r**D as invalid forms.

Implementation Notes—The following notes describe the e600 core implementation of integer load instructions:

- The PowerPC architecture cautions programmers that some implementations of the architecture can execute the load half algebraic (**lha**, **lhax**) instructions with greater latency than other types of load instructions. This is not the case for the e600 core; these instructions operate with the same latency as other load instructions.
- The PowerPC architecture cautions programmers that some implementations of the architecture can run the load/store byte-reverse (**lhbrx**, **lbrx**, **sthbrx**, **stwbrx**) instructions with greater latency than other types of load/store instructions. This is not the case for the e600 core. These instructions operate with the same latency as the other load/store instructions.
- The PowerPC architecture describes some preferred instruction forms for load and store multiple instructions and integer move assist instructions that can perform better than other forms in some implementations. None of these preferred forms affect instruction performance on the e600 core. Usage of load/store string instruction is discouraged.
- The PowerPC architecture defines the **lwarx** and **stwcx.** as a way to update memory atomically. In the e600 core, reservations are made on behalf of aligned 32-byte sections of the memory address space. Executing **lwarx** and **stwcx.** to a page marked write-through does cause a DSI interrupt if the page is marked cacheable write-through (WIM = 10x) or caching-inhibited (WIM = x1x), but as with other memory accesses, DSI interrupts can result for other reasons such as protection violations or page faults.

Table 6-49 summarizes the integer load instructions.

Table 6-49. Integer Load Instructions

Name	Mnemonic	Syntax
Load Byte and Zero	**lbz**	**r**D,**d**(r**A**)
Load Byte and Zero Indexed	**lbzx**	**r**D,**r**A,**r**B
Load Byte and Zero with Update	**lbzu**	**r**D,**d**(r**A**)
Load Byte and Zero with Update Indexed	**lbzux**	**r**D,**r**A,**r**B
Load Half Word and Zero	**lhz**	**r**D,**d**(r**A**)
Load Half Word and Zero Indexed	**lhzx**	**r**D,**r**A,**r**B
Load Half Word and Zero with Update	**lhzu**	**r**D,**d**(r**A**)
Load Half Word and Zero with Update Indexed	**lhzux**	**r**D,**r**A,**r**B
Load Half Word Algebraic	**lha**	**r**D,**d**(r**A**)
Load Half Word Algebraic Indexed	**lhax**	**r**D,**r**A,**r**B
Load Half Word Algebraic with Update	**lhau**	**r**D,**d**(r**A**)
Load Half Word Algebraic with Update Indexed	**lhaux**	**r**D,**r**A,**r**B

Table 6-49. Integer Load Instructions (continued)

Name	Mnemonic	Syntax
Load Word and Zero	**lwz**	**r**D,**d**(**r**A)
Load Word and Zero Indexed	**lwzx**	**r**D,**r**A,**r**B
Load Word and Zero with Update	**lwzu**	**r**D,**d**(**r**A)
Load Word and Zero with Update Indexed	**lwzux**	**r**D,**r**A,**r**B

6.3.4.3.4 Integer Store Instructions

For integer store instructions, the contents of **r**S are stored into the byte, half-word, word or double-word in memory addressed by the EA (effective address). Many store instructions have an update form, in which **r**A is updated with the EA. For these forms, the following rules apply:

- If **r**A ≠ 0, the effective address is placed into **r**A.
- If **r**S = **r**A, the contents of register **r**S are copied to the target memory element, then the generated EA is placed into **r**A (**r**S).

The PowerPC architecture defines store with update instructions with **r**A = 0 as an invalid form. In addition, it defines integer store instructions with the CR update option enabled (Rc field, bit 31, in the instruction encoding = 1) to be an invalid form. Table 6-50 summarizes the integer store instructions.

Table 6-50. Integer Store Instructions

Name	Mnemonic	Syntax
Store Byte	**stb**	**r**S,**d**(**r**A)
Store Byte Indexed	**stbx**	**r**S,**r**A,**r**B
Store Byte with Update	**stbu**	**r**S,**d**(**r**A)
Store Byte with Update Indexed	**stbux**	**r**S,**r**A,**r**B
Store Half Word	**sth**	**r**S,**d**(**r**A)
Store Half Word Indexed	**sthx**	**r**S,**r**A,**r**B
Store Half Word with Update	**sthu**	**r**S,**d**(**r**A)
Store Half Word with Update Indexed	**sthux**	**r**S,**r**A,**r**B
Store Word	**stw**	**r**S,**d**(**r**A)
Store Word Indexed	**stwx**	**r**S,**r**A,**r**B
Store Word with Update	**stwu**	**r**S,**d**(**r**A)
Store Word with Update Indexed	**stwux**	**r**S,**r**A,**r**B

6.3.4.3.5 Integer Store Gathering

The e600 core performs store gathering for write-through accesses to nonguarded space or to cache-inhibited stores to nonguarded space if the requirements described in the “L1 and L2 Cache Operation” chapter of the *e600 PowerPC Core Reference Manual* are met. These stores are combined in the load/store unit (LSU) to form a double-word or quad-word and are sent out on the MPX bus as a single

operation. However, stores can be gathered only if the successive stores that meet the criteria are queued and pending. The e600 core also performs store merging as described in the “L1 and L2 Cache Operation” chapter of the *e600 PowerPC Core Reference Manual*.

Store gathering takes place regardless of the address order of the stores. The store gathering and merging feature is enabled by setting HID0[SGE].

If store gathering is enabled and the stores do not fall under the above categories, an **eieio** or **sync** instruction must be used to prevent two stores from being gathered.

6.3.4.3.6 Integer Load and Store with Byte-Reverse Instructions

Table 6-51 describes integer load and store with byte-reverse instructions. When used in a system operating with the default big-endian byte order, these instructions have the effect of loading and storing data in little-endian order. Likewise, when used in a system operating with little-endian byte order, these instructions have the effect of loading and storing data in big-endian order. For more information about big-endian and little-endian byte ordering, see “Byte Ordering,” in Chapter 3, “Operand Conventions,” in the *Programming Environments Manual*.

Table 6-51. Integer Load and Store with Byte-Reverse Instructions

Name	Mnemonic	Syntax
Load Half Word Byte-Reverse Indexed	**lhbrx**	**r**D,**r**A,**r**B
Load Word Byte-Reverse Indexed	**lwbrx**	**r**D,**r**A,**r**B
Store Half Word Byte-Reverse Indexed	**sthbrx**	**r**S,**r**A,**r**B
Store Word Byte-Reverse Indexed	**stwbrx**	**r**S,**r**A,**r**B

6.3.4.3.7 Integer Load and Store Multiple Instructions

The load/store multiple instructions are used to move blocks of data to and from the GPRs. The load multiple and store multiple instructions can have operands that require memory accesses crossing a 4-Kbyte page boundary. As a result, these instructions can be interrupted by a DSI interrupt associated with the address translation of the second page.

The PowerPC architecture defines the Load Multiple Word (**lmw**) instruction with **r**A in the range of registers to be loaded as an invalid form.

Table 6-52. Integer Load and Store Multiple Instructions

Name	Mnemonic	Syntax
Load Multiple Word	**lmw**	**r**D,**d(r**A**)**
Store Multiple Word	**stmw**	**r**S,**d(r**A**)**

6.3.4.3.8 Integer Load and Store String Instructions

The integer load and store string instructions allow movement of data from memory to registers or from registers to memory without concern for alignment. These instructions can be used for a short move between arbitrary memory locations or to initiate a long move between misaligned memory fields.

However, in some implementations, these instructions are likely to have greater latency and take longer to execute, perhaps much longer, than a sequence of individual load or store instructions that produce the same results. Table 6-53 summarizes the integer load and store string instructions.

Table 6-53. Integer Load and Store String Instructions

Name	Mnemonic	Syntax
Load String Word Immediate	**lswi**	**r**D,**r**A,NB
Load String Word Indexed	**lswx**	**r**D,**r**A,**r**B
Store String Word Immediate	**stswi**	**r**S,**r**A,NB
Store String Word Indexed	**stswx**	**r**S,**r**A,**r**B

In the e600 core implementation, operating with little-endian byte order, execution of a load or string instruction will take an alignment interrupt.

Load string and store string instructions can involve operands that are not word-aligned.

For load/store string operations, the e600 core does not combine register values to reduce the number of discrete accesses. However, if store gathering is enabled and the accesses fall under the criteria for store gathering the stores can be combined to enhance performance. At a minimum, additional cache access cycles are required. Usage of load/store string instructions is discouraged.

6.3.4.3.9 Floating-Point Load and Store Address Generation

Floating-point load and store operations generate effective addresses using the register indirect with immediate index addressing mode and register indirect with index addressing mode. Floating-point loads and stores are not supported for direct-store accesses. The use of floating-point loads and stores for direct-store access results in an alignment interrupt.

There are two forms of the floating-point load instruction—single-precision and double-precision operand formats. Because the FPRs support only the floating-point double-precision format, single-precision floating-point load instructions convert single-precision data to double-precision format before loading an operand into an FPR.

Implementation Note—The e600 core treats interrupts as follows:

- The FPU operates in either ignore exceptions mode (MSR[FE0] = MSR[FE1] = 0) or precise mode (any other settings for MSR[FE0,FE1]). For the e600 core, ignore exceptions mode allows floating-point instructions to complete earlier and thus can provide better performance than precise mode.

The floating-point load and store indexed instructions (**lfsx**, **lfsux**, **lfdx**, **lfdux**, **stfsx**, **stfsux**, **stfdx**, **stfdux**) are invalid when the Rc bit is one. The PowerPC architecture defines a load with update instruction with **r**A = 0 as an invalid form. Table 6-54 summarizes the floating-point load instructions.

Table 6-54. Floating-Point Load Instructions

Name	Mnemonic	Syntax
Load Floating-Point Single	**lfs**	**fr**D,**d(r**A**)**
Load Floating-Point Single Indexed	**lfsx**	**fr**D,**r**A,**r**B
Load Floating-Point Single with Update	**lfsu**	**fr**D,**d(r**A**)**
Load Floating-Point Single with Update Indexed	**lfsux**	**fr**D,**r**A,**r**B
Load Floating-Point Double	**lfd**	**fr**D,**d(r**A**)**
Load Floating-Point Double Indexed	**lfdx**	**fr**D,**r**A,**r**B
Load Floating-Point Double with Update	**lfdu**	**fr**D,**d(r**A**)**
Load Floating-Point Double with Update Indexed	**lfdux**	**fr**D,**r**A,**r**B

6.3.4.3.10 Floating-Point Store Instructions

This section describes floating-point store instructions. There are three basic forms of the store instruction—single-precision, double-precision, and integer. The integer form is supported by the optional **stfiwx** instruction. Because the FPRs support only floating-point, double-precision format for floating-point data, single-precision floating-point store instructions convert double-precision data to single-precision format before storing the operands. Table 6-55 summarizes the floating-point store instructions.

Table 6-55. Floating-Point Store Instructions

Name	Mnemonic	Syntax
Store Floating-Point Single	**stfs**	**fr**S,**d(r**A**)**
Store Floating-Point Single Indexed	**stfsx**	**fr**S,**r** B
Store Floating-Point Single with Update	**stfsu**	**fr**S,**d(**rA**)**
Store Floating-Point Single with Update Indexed	**stfsux**	**fr**S,**r** B
Store Floating-Point Double	**stfd**	**fr**S,**d(r**A**)**
Store Floating-Point Double Indexed	**stfdx**	**fr**S,**r**B
Store Floating-Point Double with Update	**stfdu**	**fr**S,**d(**rA**)**
Store Floating-Point Double with Update Indexed	**stfdux**	**fr**S,**r** B
Store Floating-Point as Integer Word Indexed[1]	**stfiwx**	**fr**S,**r**B

[1] The **stfiwx** instruction is optional to the PowerPC architecture

Some floating-point store instructions require conversions in the LSU. Table 6-56 shows conversions the LSU makes when executing a Store Floating-Point Single instruction.

Table 6-56. Store Floating-Point Single Behavior

FPR Precision	Data Type	Action
Single	Normalized	Store
Single	Denormalized	Store
Single	Zero, infinity, QNaN	Store
Single	SNaN	Store

Double	Normalized	If (exp ≤ 896)
		then
		Denormalize and Store
		else
		Store

Double	Denormalized	Store zero
Double	Zero, infinity, QNaN	Store
Double	SNaN	Store

Table 6-57 shows the conversions made when performing a Store Floating-Point Double instruction. Most entries in the table indicate that the floating-point value is simply stored. Only in a few cases are any other actions taken.

Table 6-57. Store Floating-Point Double Behavior

FPR Precision	Data Type	Action
Single	Normalized	Store
Single	Denormalized	Normalize and Store
Single	Zero, infinity, QNaN	Store
Single	SNaN	Store
Double	Normalized	Store
Double	Denormalized	Store
Double	Zero, infinity, QNaN	Store
Double	SNaN	Store

Architecturally, all floating-point numbers are represented in double-precision format within the e600 core. Execution of a store floating-point single (**stfs**, **stfsu**, **stfsx**, **stfsux**) instruction requires conversion from double- to single-precision format. If the exponent is not greater than 896, this conversion requires denormalization. The e600 core supports this denormalization by shifting the mantissa one bit at a time. Anywhere from 1 to 23 clock cycles are required to complete the denormalization, depending upon the value to be stored.

Because of how floating-point numbers are implemented in the e600 core, there is also a case when execution of a store floating-point double (**stfd**, **stfdu**, **stfdx**, **stfdux**) instruction can require internal

shifting of the mantissa. This case occurs when the operand of a store floating-point double instruction is a denormalized single-precision value. The value could be the result of a load floating-point single instruction, a single-precision arithmetic instruction, or a floating round to single-precision instruction. In these cases, shifting the mantissa takes from 1 to 23 clock cycles, depending upon the value to be stored. These cycles are incurred during the store.

6.3.4.4 Branch and Flow Control Instructions

Some branch instructions can redirect instruction execution conditionally based on the value of bits in the CR. When the processor encounters one of these instructions, it scans the execution pipelines to determine whether an instruction in progress can affect the particular CR bit. If no interlock is found, the branch can be resolved immediately by checking the bit in the CR and taking the action defined for the branch instruction.

6.3.4.4.1 Branch Instruction Address Calculation

Branch instructions can alter the sequence of instruction execution. Instruction addresses are always assumed to be word aligned; the processors that ignore the two low-order bits of the generated branch target address.

Branch instructions compute the EA of the next instruction address using the following addressing modes:

- Branch relative
- Branch conditional to relative address
- Branch to absolute address
- Branch conditional to absolute address
- Branch conditional to link register
- Branch conditional to count register

Note that in the e600 core, all branch instructions (**b**, **ba**, **bl**, **bla**, **bc**, **bca**, **bcl**, **bcla**, **bclr**, **bclrl**, **bcctr**, **bcctrl**) are executed in the BPU and condition register logical instructions (**crand**, **cror**, **crxor**, **crnand**, **crnor**, **crandc**, **creqv**, **crorc**, and **mcrf**) are executed by the IU2. Some of these instructions can redirect instruction execution conditionally on the value of CR, CTR, or LR bits. When the CR bits resolve, the branch instruction is either marked as correct or mispredicted. Correcting a mispredicted branch requires that the e600 core flush speculatively executed instructions and restore the machine state to immediately after the branch. This correction can be done when all non-speculative instructions older than the mispredicting branch have completed.

6.3.4.4.2 Branch Instructions

Table 6-58 lists the branch instructions provided by the processors that implement the PowerPC architecture. To simplify assembly language programming, a set of simplified mnemonics and symbols is provided for the most frequently used forms of branch conditional, compare, trap, rotate and shift, and certain other instructions. See Appendix F, “Simplified Mnemonics,” in the *Programming Environments Manual* for a list of simplified mnemonic examples.

Table 6-58. Branch Instructions

Name	Mnemonic	Syntax
Branch	**b (ba bl bla)**	target_addr
Branch Conditional	**bc (bca bcl bcla)**	BO,BI,target_addr
Branch Conditional to Link Register	**bclr (bclrl)**	BO,BI
Branch Conditional to Count Register	**bcctr (bcctrl)**	BO,BI

6.3.4.4.3 Condition Register Logical Instructions

Condition register logical instructions, shown in Table 6-59, and the Move Condition Register Field (**mcrf**) instruction are also defined as flow control instructions.

Table 6-59. Condition Register Logical Instructions

Name	Mnemonic	Syntax
Condition Register AND	**crand**	**crb**D,**crb**A,**crb**B
Condition Register OR	**cror**	**crb**D,**crb**A,**crb**B
Condition Register XOR	**crxor**	**crb**D,**crb**A,**crb**B
Condition Register NAND	**crnand**	**crb**D,**crb**A,**crb**B
Condition Register NOR	**crnor**	**crb**D,**crb**A,**crb**B
Condition Register Equivalent	**creqv**	**crb**D,**crb**A, **crb**B
Condition Register AND with Complement	**crandc**	**crb**D,**crb**A, **crb**B
Condition Register OR with Complement	**crorc**	**crb**D,**crb**A, **crb**B
Move Condition Register Field	**mcrf**	**crf**D,**crf**S

Note that if the LR update option is enabled for any of these instructions, the PowerPC architecture defines these forms of the instructions as invalid.

6.3.4.4.4 Trap Instructions

The trap instructions shown in Table 6-60 are provided to test for a specified set of conditions. If any of the conditions tested by a trap instruction are met, the system trap type program interrupt is taken. For more information, see the “Interrupts” chapter of the *e600 PowerPC Core Reference Manual*. If the tested conditions are not met, instruction execution continues normally.

Table 6-60. Trap Instructions

Name	Mnemonic	Syntax
Trap Word Immediate	**twi**	TO,**r**A,SIMM
Trap Word	**tw**	TO,**r**A,**r**B

See Appendix F, “Simplified Mnemonics,” in the *Programming Environments Manual* for a complete set of simplified mnemonics.

6.3.4.5 System Linkage Instruction—UISA

The System Call (**sc**) instruction permits a program to call on the system to perform a service; see Table 6-61 and also Section 6.3.6.1, “System Linkage Instructions—OEA,” for additional information.

Table 6-61. System Linkage Instruction—UISA

Name	Mnemonic	Syntax
System Call	**sc**	—

Executing this instruction causes the system call interrupt handler to be evoked. For more information, see the “Interrupts” chapter of the *e600 PowerPC Core Reference Manual.*

6.3.4.6 Processor Control Instructions—UISA

Processor control instructions are used to read from and write to the condition register (CR), machine state register (MSR), and special-purpose registers (SPRs). See Section 6.3.5.1, “Processor Control Instructions—VEA,” for the **mftb** instruction and Section 6.3.6.2, “Processor Control Instructions—OEA,” for information about the instructions used for reading from and writing to the MSR and SPRs.

6.3.4.6.1 Move To/From Condition Register Instructions

Table 6-62 summarizes the instructions for reading from or writing to the condition register.

Table 6-62. Move To/From Condition Register Instructions

Name	Mnemonic	Syntax
Move to Condition Register Fields	**mtcrf**	CRM,**r**S
Move to Condition Register from XER	**mcrxr**	**crf**D
Move from Condition Register	**mfcr**	**r**D

Implementation Note—The PowerPC architecture indicates that in some implementations the Move to Condition Register Fields (**mtcrf**) instruction can perform more slowly when only a portion of the fields are updated as opposed to all of the fields. The condition register access latency for the e600 core is the same in both cases, if multiple fields are affected. Note that an **mtcrf** to a single field is handled in the IU1s and latency may be lower if an **mtcrf** multi is split into its component single field pieces by the compiler.

6.3.4.6.2 Move To/From Special-Purpose Register Instructions—UISA

Table 6-63 lists the **mtspr** and **mfspr** instructions.

Table 6-63. Move To/From Special-Purpose Register Instructions (UISA)

Name	Mnemonic	Syntax
Move to Special-Purpose Register	**mtspr**	SPR,**r**S
Move from Special-Purpose Register	**mfspr**	**r**D,SPR

Table 6-64 lists the SPR numbers for user-level PowerPC SPR accesses.

Encodings for the e600-specific user-level SPRs are listed in Table 6-65.

Table 6-64. User-Level PowerPC SPR Encodings

Register Name	SPR[1]			Access	mfspr/mtspr
	Decimal	spr[5–9]	spr[0–4]		
CTR	9	00000	01001	User (UISA)	Both
LR	8	00000	01000	User (UISA)	Both
TBL[2]	268	01000	01100	User (VEA)	**mftb**
TBU [2]	269	01000	01101	User (VEA)	**mftb**
VRSAVE[3]	256	01000	00000	User (AltiVec/UISA)	Both
XER	1	00000	00001	User (UISA)	Both

1 Note that the order of the two 5-bit halves of the SPR number is reversed compared with actual instruction coding. For **mtspr** and **mfspr** instructions, the SPR number coded in assembly language does not appear directly as a 10-bit binary number in the instruction. The number coded is split into two 5-bit halves that are reversed in the instruction, with the high-order 5 bits appearing in bits 16–20 of the instruction and the low-order 5 bits in bits 11–15.

2 The TB registers are referred to as TBRs rather than SPRs and can be written to using the **mtspr** instruction in supervisor mode and the TBR numbers here. The TB registers can be read in user mode using either the **mftb** instruction and specifying TBR 268 for TBL and TBR 269 for TBU.

3 Register defined by the AltiVec technology.

Table 6-65. User-Level SPR Encodings for e600-Defined Registers

Register Name	SPR [1]			Access	mfspr/mtspr
	Decimal	spr[5–9]	spr[0–4]		
UMMCR0	936	11101	01000	User	**mfspr**
UMMCR1	940	11101	01100	User	**mfspr**
UMMCR2	928	11101	00000	User	**mfspr**
UPMC1	937	11101	01001	User	**mfspr**
UPMC2	938	11101	01010	User	**mfspr**
UPMC3	941	11101	01101	User	**mfspr**
UPMC4	942	11101	01110	User	**mfspr**
UPMC5	929	11101	00001	User	**mfspr**
UPMC6	930	11101	00010	User	**mfspr**
USIAR	939	11101	01011	User	**mfspr**

1 Note that the order of the two 5-bit halves of the SPR number is reversed compared with actual instruction coding. For mtspr and mfspr instructions, the SPR number coded in assembly language does not appear directly as a 10-bit binary number in the instruction. The number coded is split into two 5-bit halves that are reversed in the instruction, with the high-order 5 bits appearing in bits 16–20 of the instruction and the low-order 5 bits in bits 11–15.

6.3.4.7 Memory Synchronization Instructions—UISA

Memory synchronization instructions control the order in which memory operations are completed with respect to asynchronous events, and the order in which memory operations are seen by other processors or memory access mechanisms. See the “L1 and L2 Cache Operation” chapter of the *e600 PowerPC Core Reference Manual* for additional information about these instructions and about related aspects of memory synchronization. See Table 6-66 for a summary.

Table 6-66. Memory Synchronization Instructions—UISA

Name	Mnemonic	Syntax	Implementation Notes
Load Word and Reserve Indexed	**lwarx** [1]	**r**D,**r**A,**r**B	Programmers can use **lwarx** with **stwcx.** to emulate common semaphore operations such as test and set, compare and swap, exchange memory, and fetch and add. Both instructions must use the same EA. Reservation granularity is implementation-dependent. The e600 core makes reservations on behalf of aligned 32-byte sections of the memory address space. Executing **lwarx** and **stwcx.** to a page marked write-through (WIMG = 10xx) or caching-inhibited (WIMG = x1xx) or when the data cache is disabled or locked causes a DSI interrupt. If the location is not word-aligned, an alignment interrupt occurs. The **stwcx.** instruction is the only load/store instruction with a valid form if Rc is set. If Rc is zero, executing **stwcx.** sets CR0 to an undefined value.
Store Word Conditional Indexed	**stwcx.**[1]	**r**S,**r**A,**r**B	
Synchronize	**sync**	—	Because it delays execution of subsequent instructions until all previous instructions complete to where they cannot cause an interrupt, **sync** is a barrier against store gathering. Additionally, all load/store cache/MPX bus activities initiated by prior instructions are completed. Touch load operations (**dcbt**, **dcbtst**) must complete address translation, but need not complete on the MPX bus. The **sync** completes after a successful broadcast on the MPX bus. The latency of **sync** depends on the processor state when it is dispatched and on various system-level situations. Note that frequent use of **sync** will degrade performance.

1 Note that the e600 core implements **lwarx** and **stwcx.** as defined for embedded processors by the PowerPC ISA. The execution of an **lwarx** or **stwcx.** instruction to memory marked write-through or cache-inhibited causes a DSI interrupt.

Integrated devices with additional caches should take special care to recognize the hardware signaling caused by a SYNC MPX bus operation and perform the appropriate actions to guarantee that memory references that can be queued internally to the additional cache have been performed globally.

See Section 6.3.5.2, “Memory Synchronization Instructions—VEA,” for details about additional memory synchronization (**eieio**) instructions.

In the PowerPC architecture, the Rc bit must be zero for most load and store instructions. If Rc is set, the instruction form is invalid for **sync** and **lwarx** instructions. If the e600 core encounters one of these invalid instruction forms, it sets CR0 to an undefined value.

6.3.5 PowerPC VEA Instructions

The PowerPC virtual environment architecture (VEA) describes the semantics of the memory model that can be assumed by software processes, and includes descriptions of the cache model, cache control instructions, address aliasing, and other related issues. Implementations that conform to the VEA also adhere to the UISA, but do not necessarily adhere to the OEA.

This section describes additional instructions that are provided by the VEA.

6.3.5.1 Processor Control Instructions—VEA

In addition to the move to condition register instructions (specified by the UISA), the VEA defines the **mftb** instruction (user-level instruction) for reading the contents of the time base register; see the “L1 and L2 Cache Operation” chapter of the *e600 PowerPC Core Reference Manual*. for more information. Table 6-67 shows the **mftb** instruction.

Table 6-67. Move From Time Base Instruction

Name	Mnemonic	Syntax
Move from Time Base	**mftb**	**r**D, TBR

Simplified mnemonics are provided for the **mftb** instruction so it can be coded with the TBR name as part of the mnemonic rather than requiring it to be coded as an operand. See Appendix F, “Simplified Mnemonics,” in the *Programming Environments Manual* for simplified mnemonic examples and for simplified mnemonics for Move from Time Base (**mftb**) and Move from Time Base Upper (**mftbu**), which are variants of the **mftb** instruction rather than of **mfspr**. The **mftb** instruction serves as both a basic and simplified mnemonic. Assemblers recognize an **mftb** mnemonic with two operands as the basic form, and an **mftb** mnemonic with one operand as the simplified form.

Implementation Note—In the e600 core, note the following:

- The e600 core allows user-mode read access to the time base counter through the use of the Move from Time Base (**mftb**) instruction. As a 32-bit implementation of the PowerPC architecture, the e600 core can access TBU and TBL separately only.
- The time base counter is clocked at a frequency that is one-fourth that of the bus clock. Counting is enabled by assertion of the time base enable (*tben*) input signal.

6.3.5.2 Memory Synchronization Instructions—VEA

Memory synchronization instructions control the order in which memory operations are completed with respect to asynchronous events, and the order in which memory operations are seen by other processors or memory access mechanisms. See the “L1 and L2 Cache Operation” chapter of the *e600 PowerPC Core Reference Manual* for more information about these instructions and about related aspects of memory synchronization.

In addition to the **sync** instruction (specified by UISA), the VEA defines the Enforce In-Order Execution of I/O (**eieio**) and Instruction Synchronize (**isync**) instructions. The number of cycles required to complete an **eieio** instruction depends on system parameters and on the core’s state when the instruction is issued. As a result, frequent use of this instruction can degrade performance. Note that the broadcast of these instructions on the MPX bus is controlled by the HID1[SYNCBE] bit.

Table 6-68 describes the memory synchronization instructions defined by the VEA.

Table 6-68. Memory Synchronization Instructions—VEA

Name	Mnemonic	Syntax	Implementation Notes
Enforce In-Order Execution of I/O	**eieio**	—	The **eieio** instruction is dispatched to the LSU and executes after all previous cache-inhibited or write-through accesses are performed; all subsequent instructions that generate such accesses execute after **eieio**. As the **eieio** operation does not affect the caches, it bypasses the L2 cache and is forwarded to the MPX bus. An EIEIO operation is broadcast on the MPX bus to enforce ordering in the memory system external to the core. Because the e600 core does reorder noncacheable accesses, **eieio** may be needed to force ordering. However, if store gathering is enabled and an **eieio** is detected in a store queue, stores are not gathered. Broadcasting **eieio** prevents other peripherals, such as bus bridge chips, from gathering stores.
Instruction Synchronize	**isync**	—	The **isync** instruction is refetch serializing; that is, it causes the e600 core to wait for all prior instructions to complete first then executes, purging all instructions from the core and then refetching the next instruction. The **isync** instruction is not executed until all previous instructions complete to the point where they cannot cause an interrupt. The **isync** instruction does not wait for all pending stores in the store queue to complete. Any instruction after an **isync** sees all effects of prior instructions occurring before the **isync**.

6.3.5.3 Memory Control Instructions—VEA

Memory control instructions can be classified as follows:

- Cache management instructions (user-level and supervisor-level)
- Translation lookaside buffer management instructions (OEA)

This section describes the user-level cache management instructions defined by the VEA. See Section 6.3.6.3, “Memory Control Instructions—OEA,” for information about supervisor-level cache, segment register manipulation, and translation lookaside buffer management instructions. For a complete description of the MPX bus operations caused by cache control instructions, see the “L1 and L2 Cache Operation” chapter of the *e600 PowerPC Core Reference Manual*.

6.3.5.3.1 User-Level Cache Instructions—VEA

The instructions summarized in this section help user-level programs manage caches within the core if they are implemented. See the “L1 and L2 Cache Operation” chapter of the *e600 PowerPC Core Reference Manual*. for more information about cache topics. The following sections describe how these operations are treated with respect to the e600 core’s caches.

As with other memory-related instructions, the effects of cache management instructions on memory are weakly-ordered. If the programmer must ensure that cache or other instructions have been performed with respect to all other processors and system mechanisms, a **sync** instruction must be placed after those instructions.

Note that the e600 core interprets cache control instructions (**icbi**, **dcbi**, **dcbf**, **dcbz**, and **dcbst**) as if they pertain only to the local L1 and L2 caches. A **dcbz** (with M set) is always broadcast on the core interface if it does not hit as modified in any core cache.

All cache control instructions to direct-store space function as no-ops. For information on how cache control instructions affect the L2 cache, see the “L1 and L2 Cache Operation” chapter of the *e600 PowerPC Core Reference Manual*.

The L1 cache can be flushed (if data is not shared and you need only to invalidate the cache) with the HID0[DCFI] and HID0[ICFI] bits. The L2 flush bit is L2CR[l2HWF] and the L2 invalidate bit is L2CR[L2I]. See the “L1 and L2 Cache Operation” chapter of the e600 PowerPC Core Reference Manual for more information.

Table 6-69 summarizes the cache instructions defined by the VEA. Note that these instructions are accessible to user-level programs.

Table 6-69. User-Level Cache Instructions

Name	Mnemonic	Syntax	Implementation Notes
Data Cache Block Touch [1]	**dcbt**	**r**A,**r**B	The VEA defines this instruction to allow for potential system performance enhancements through the use of software-initiated prefetch hints. Implementations are not required to take any action based on execution of this instruction, but they can prefetch the cache block corresponding to the EA into their cache. When **dcbt** executes, the e600 core checks for protection violations (as for a load instruction). This instruction is treated as a no-op for the following cases: • The access causes a protection violation. • The page is mapped cache-inhibited or direct-store (T = 1). • The cache is locked or disabled • HID0[NOPTI] = 1 Otherwise, if no data is in the cache location, the e600 core requests a cache line fill. Data brought into the cache is validated as if it were a load instruction. The memory reference of a **dcbt** sets the referenced bit.
Data Cache Block Touch for Store [1]	**dcbts**t	**r**A,**r**B	This instruction **dcbtst** can be no-oped by setting HID0[NOPTI]. The **dcbtst** instruction behaves similarly to a **dcbt** instruction, except that the line fill request on the MPX bus is signaled as read or read-claim, and the data is marked as exclusive in the L1 data cache if there is no shared response on the MPX bus. More specifically, the following cases occur depending on where the line currently exists or does not exist in the e600 core. • **dcbtst** hits in the L1 data cache. In this case, the **dcbtst** does nothing and the state of the line in the cache is not changed. Thus, if the line was in the shared state, a subsequent store hits on this shared line and incurs the associated latency penalties. • **dcbtst** misses in the L1 data cache and hits in the L2 cache. In this case, the **dcbtst** will reload the L1 data cache with the state found in the L2 cache. Again, if the line was in the shared state in the L2, a subsequent store will hit on this shared line and incur the associated latency penalties. • **dcbtst** misses in L1 data cache and L2 cache. In this case, the e600 core will request the line from memory with read or read-claim and reload the L1 data cache in the exclusive state. A subsequent store will hit on exclusive and can perform the store to the L1 data cache immediately. In addition, a **dcbtst** instruction will be no-oped if the target address of the **dcbtst** is mapped as write-through.

Table 6-69. User-Level Cache Instructions (continued)

Name	Mnemonic	Syntax	Implementation Notes
Data Cache Block Set to Zero	**dcbz**	**r**A,**r**B	The EA is computed, translated, and checked for protection violations. For cache hits, 32 bytes of zeros are written to the cache block and the tag is marked modified. For cache misses with the replacement block marked not modified, the zero reload is performed and the cache block is marked modified. However, if the replacement block is marked modified, the contents are written back to memory first. The instruction takes an alignment interrupt if the cache block is locked or disabled or if the cache is marked WT or CI. If WIMG = xx1x (coherency enforced), the address is broadcast to the MPX bus before the zero reload fill. The interrupt priorities (from highest to lowest) are as follows: 1 Cache disabled—Alignment interrupt 2 Cache is locked—Alignment interrupt 3 Page marked write-through or cache-inhibited—alignment interrupt 4 BAT protection violation—DSI interrupt 5 TLB protection violation—DSI interrupt **dcbz** is broadcast if WIMG = xx1x (coherency enforced).
Data Cache Block Allocate	**dcba**	**r**A,**r**B	The EA is computed, translated, and checked for protection violations. For cache hits, 32 bytes of zeros are written to the cache block and the tag is marked modified. For cache misses with the replacement block marked non-dirty, the zero reload is performed and the cache block is marked modified. However, if the replacement block is marked modified, the contents are written back to memory first. The instruction performs a no-op if the cache is locked or disabled or if the cache is marked WT or CI. If WIMG =xx1x (coherency enforced), the address is broadcast to the MPX bus before the zero reload fill. A no-op occurs for the following: • Cache is disabled • Cache is locked • Page marked write-through or cache-inhibited • BAT protection violation • TLB protection violation **dcba** is broadcast if WIMG = xx1x (coherency enforced).
Data Cache Block Store	**dcbst**	**r**A,**r**B	The EA is computed, translated, and checked for protection violations. • For cache hits with the tag marked not modified, no further action is taken. • For cache hits with the tag marked modified, the cache block is written back to memory and marked exclusive. If WIMG = xx1x (coherency enforced), **dcbst** is broadcast. The instruction acts like a load with respect to address translation and memory protection. It executes regardless of whether the cache is disabled or locked. The interrupt priorities (from highest to lowest) for **dcbst** are as follows: 1 BAT protection violation—DSI interrupt 2 TLB protection violation—DSI interrupt
Data Cache Block Flush	**dcbf**	**r**A,**r**B	The EA is computed, translated, and checked for protection violations: • For cache hits with the tag marked modified, the cache block is written back to memory and the cache entry is invalidated. • For cache hits with the tag marked not modified, the entry is invalidated. • For cache misses, no further action is taken. A **dcbf** is broadcast if WIMG = xx1x (coherency enforced).The instruction acts like a load with respect to address translation and memory protection. It executes regardless of whether the cache is disabled or locked. The interrupt priorities (from highest to lowest) for **dcbf** are as follows: 1 BAT protection violation—DSI interrupt 2 TLB protection violation—DSI interrupt

Table 6-69. User-Level Cache Instructions (continued)

Name	Mnemonic	Syntax	Implementation Notes
Instruction Cache Block Invalidate	**icbi**	**r**A,**r**B	This instruction is broadcast on the MPX bus if WIMG = xx1x. **icbi** should always be followed by a **sync** and an **isync** to make sure that the effects of the **icbi** are seen by the instruction fetches following the **icbi** itself.

[1] A program that uses **dcbt** and **dcbtst** instructions improperly performs less efficiently. To improve performance, HID0[NOPTI] can be set, which causes dcbt and dcbtst to be no-oped at the cache. They do not cause MPX bus activity and cause only a 1-clock execution latency. The default state of this bit is zero which enables the use of these instructions.

6.3.5.4 Optional External Control Instructions

The PowerPC architecture defines an optional external control feature that, if implemented, is supported by the two external control instructions, **eciwx** and **ecowx**. These instructions are not implemented on the e600 core. Note that the EAR, which is accessible only in supervisor mode, must not be enabled. Otherwise, attempted execution of **eciwx**/**ecowx** causes boundedly undefined results.

6.3.6 PowerPC OEA Instructions

The PowerPC operating environment architecture (OEA) includes the structure of the memory management model, supervisor-level registers, and the interrupt model. Implementations that conform to the OEA also adhere to the UISA and the VEA. This section describes the instructions provided by the OEA.

6.3.6.1 System Linkage Instructions—OEA

This section describes the system linkage instructions (see Table 6-70). The user-level **sc** instruction lets a user program call on the system to perform a service and causes the processor to take a system call interrupt. The supervisor-level **rfi** instruction is used for returning from an interrupt handler.

Table 6-70. System Linkage Instructions—OEA

Name	Mnemonic	Syntax	Implementation Notes
System Call	**sc**	—	The **sc** instruction is context-synchronizing.
Return from Interrupt	**rfi**	—	**rfi** is context-synchronizing. For the e600 core, this means that **rfi** works its way to the final stage of the execution pipeline, updates architected registers, and redirects the instruction flow.

6.3.6.2 Processor Control Instructions—OEA

The instructions listed in Table 6-71 provide access to the segment registers for 32-bit implementations. These instructions operate completely independently of the MSR[IR] and MSR[DR] bit settings. Refer to “Synchronization Requirements for Special Registers and for Lookaside Buffers,” in Chapter 2, “Register Set,” of the *Programming Environments Manual* for serialization requirements and other recommended precautions to observe when manipulating the segment registers.

Table 6-71. Segment Register Manipulation Instructions (OEA)

Name	Mnemonic	Syntax	Implementation Notes
Move to Segment Register	**mtsr**	SR,**r**S	—
Move to Segment Register Indirect	**mtsrin**	**r**S,**r**B	—
Move from Segment Register	**mfsr**	**r**D,SR	—
Move from Segment Register Indirect	**mfsrin**	**r**D,**r**B	—

The processor control instructions used to access the MSR and the SPRs are discussed in this section. Table 6-72 lists instructions for accessing the MSR.

Table 6-72. Move To/From Machine State Register Instructions

Name	Mnemonic	Syntax
Move to Machine State Register	**mtmsr**	**r**S
Move from Machine State Register	**mfmsr**	**r**D

The OEA defines encodings of **mtspr** and **mfspr** to provide access to supervisor-level registers. The instructions are listed in Table 6-73.

Table 6-73. Move To/From Special-Purpose Register Instructions (OEA)

Name	Mnemonic	Syntax
Move to Special-Purpose Register	**mtspr**	SPR,**r**S
Move from Special-Purpose Register	**mfspr**	**r**D,SPR

Encodings for the architecture-defined SPRs are listed in Table 6-64. Encodings for e600-specific, supervisor-level SPRs are listed in Table 6-65. Simplified mnemonics are provided for **mtspr** and **mfspr** in Appendix F, “Simplified Mnemonics,” in the *Programming Environments Manual*.

Table 6-74 lists the SPR numbers for supervisor-level PowerPC SPR accesses.

Table 6-74. Supervisor-Level PowerPC SPR Encodings

Register Name	SPR[1]			Access	mfspr/mtspr
	Decimal	spr[5–9]	spr[0–4]		
DABR[2]	1013	11111	10101	Supervisor (OEA)	Both
DAR	19	00000	10011	Supervisor (OEA)	Both
DBAT0L	537	10000	11001	Supervisor (OEA)	Both
DBAT0U	536	10000	11000	Supervisor (OEA)	Both
DBAT1L	539	10000	11011	Supervisor (OEA)	Both
DBAT1U	538	10000	11010	Supervisor (OEA)	Both
DBAT2L	541	10000	11101	Supervisor (OEA)	Both
DBAT2U	540	10000	11100	Supervisor (OEA)	Both

Table 6-74. Supervisor-Level PowerPC SPR Encodings (continued)

Register Name	SPR[1]			Access	mfspr/mtspr
	Decimal	spr[5–9]	spr[0–4]		
DBAT3L	543	10000	11111	Supervisor (OEA)	Both
DBAT3U	542	10000	11110	Supervisor (OEA)	Both
DEC	22	00000	10110	Supervisor (OEA)	Both
DSISR	18	00000	10010	Supervisor (OEA)	Both
EAR [2]	282	01000	11010	Supervisor (OEA)	Both
IBAT0L	529	10000	10001	Supervisor (OEA)	Both
IBAT0U	528	10000	10000	Supervisor (OEA)	Both
IBAT1L	531	10000	10011	Supervisor (OEA)	Both
IBAT1U	530	10000	10010	Supervisor (OEA)	Both
IBAT2L	533	10000	10101	Supervisor (OEA)	Both
IBAT2U	532	10000	10100	Supervisor (OEA)	Both
IBAT3L	535	10000	10111	Supervisor (OEA)	Both
IBAT3U	534	10000	10110	Supervisor (OEA)	Both
MMCR0 [2]	952	11101	11000	Supervisor (OEA)	Both
MMCR1 [2]	956	11101	11100	Supervisor (OEA)	Both
PIR [2]	1023	11111	11111	Supervisor (OEA)	Both
PMC1 [2]	953	11101	11001	Supervisor (OEA)	Both
PMC2 [2]	954	11101	11010	Supervisor (OEA)	Both
PMC3 [2]	957	11101	11101	Supervisor (OEA)	Both
PMC4 [2]	958	11101	11110	Supervisor (OEA)	Both
PMC5 [2]	945	11101	10001	Supervisor (OEA)	Both
PMC6 [2]	946	11101	10010	Supervisor (OEA)	Both
PVR	287	01000	11111	Supervisor (OEA)	**mfspr**
SDR1	25	00000	11001	Supervisor (OEA)	Both
SIAR [2]	955	11101	11011	Supervisor (OEA)	Both
SPRG0	272	01000	10000	Supervisor (OEA)	Both
SPRG1	273	01000	10001	Supervisor (OEA)	Both
SPRG2	274	01000	10010	Supervisor (OEA)	Both
SPRG3	275	01000	10011	Supervisor (OEA)	Both
SRR0	26	00000	11010	Supervisor (OEA)	Both
SRR1	27	00000	11011	Supervisor (OEA)	Both

Table 6-74. Supervisor-Level PowerPC SPR Encodings (continued)

Register Name	SPR[1]			Access	mfspr/mtspr
	Decimal	spr[5–9]	spr[0–4]		
TBL[3]	284	01000	11100	Supervisor (OEA)	**mtspr**
TBU[3]	285	01000	11101	Supervisor (OEA)	**mtspr**

1 Note that the order of the two 5-bit halves of the SPR number is reversed compared with actual instruction coding. For **mtspr** and **mfspr** instructions, the SPR number coded in assembly language does not appear directly as a 10-bit binary number in the instruction. The number coded is split into two 5-bit halves that are reversed in the instruction, with the high-order 5 bits appearing in bits 16–20 of the instruction and the low-order 5 bits in bits 11–15.

2 Optional register defined by the PowerPC architecture.

3 The TB registers are referred to as TBRs rather than SPRs and can be written to using the **mtspr** instruction in supervisor mode and the TBR numbers here. The TB registers can be read in user mode using the **mftb** instruction and specifying TBR 268 for TBL and TBR 269 for TBU.

Encodings for the supervisor-level e600-specific SPRs are listed in Table 6-65.

Table 6-75. Supervisor-Level SPR Encodings for e600-Defined Registers

Register Name	SPR[1]			Access	mfspr/mtspr
	Decimal	spr[5–9]	spr[0–4]		
BAMR	951	11101	10111	Supervisor (OEA)	Both
DBAT4L[2]	569	10001	11001	Supervisor (OEA)	Both
DBAT4U[2]	568	10001	11000	Supervisor (OEA)	Both
DBAT5L[2]	571	10001	11011	Supervisor (OEA)	Both
DBAT5U[2]	570	10001	11010	Supervisor (OEA)	Both
DBAT6L[2]	573	10001	11101	Supervisor (OEA)	Both
DBAT6U[2]	572	10001	11100	Supervisor (OEA)	Both
DBAT7L[2]	575	10001	11111	Supervisor (OEA)	Both
DBAT7U[2]	574	10001	11110	Supervisor (OEA)	Both
HID0	1008	11111	10000	Supervisor (OEA)	Both
HID1	1009	11111	10001	Supervisor (OEA)	Both
IABR	1010	11111	10010	Supervisor (OEA)	Both
IBAT4L[2]	561	10001	10001	Supervisor (OEA)	Both
IBAT4U[2]	560	10001	10000	Supervisor (OEA)	Both
IBAT5L[2]	563	10001	10011	Supervisor (OEA)	Both
IBAT5U[2]	562	10001	10010	Supervisor (OEA)	Both
IBAT6L[2]	565	10001	10101	Supervisor (OEA)	Both
IBAT6U[2]	564	10001	10100	Supervisor (OEA)	Both

Table 6-75. Supervisor-Level SPR Encodings for e600-Defined Registers (continued)

Register Name	SPR [1]			Access	mfspr/mtspr
	Decimal	spr[5–9]	spr[0–4]		
IBAT7L [2]	567	10001	10111	Supervisor (OEA)	Both
IBAT7U [2]	566	10001	10110	Supervisor (OEA)	Both
ICTC	1019	11111	11011	Supervisor (OEA)	Both
ICTRL	1011	11111	10011	Supervisor (OEA)	Both
L2CR	1017	11111	11001	Supervisor (OEA)	Both
L2ERRADDR [2]	995	11111	00011	Supervisor (OEA)	**mfspr**
L2ERRATTR [2]	994	11111	00010	Supervisor (OEA)	Both
L2ERREADDR [2]	996	11111	00100	Supervisor (OEA)	**mfspr**
L2CAPTDATAHI [2]	988	11110	11100	Supervisor (OEA)	**mfspr**
L2CAPTDATALO [2]	989	11110	11101	Supervisor (OEA)	**mfspr**
L2CAPTECC [2]	990	11110	11110	Supervisor (OEA)	**mfspr**
L2ERRCTL [2]	997	11111	00101	Supervisor (OEA)	Both
L2ERRINJCTL [2]	987	11110	11011	Supervisor (OEA)	Both
L2ERRINJHI [2]	985	11110	11001	Supervisor (OEA)	Both
L2ERRINJLO [2]	986	11110	11010	Supervisor (OEA)	Both
L2ERRDET [2]	991	11110	11111	Supervisor (OEA)	Special [3]
L2ERRDIS [2]	992	11111	00000	Supervisor (OEA)	Both
L2ERRINTEN [2]	993	11111	00001	Supervisor (OEA)	Both
LDSTCR	1016	11111	1000	Supervisor (OEA)	Both
MMCR2	944	11101	10000	Supervisor (OEA)	Both
MSSCR0	1014	11111	10110	Supervisor (OEA)	Both
MSSSR0	1015	11111	10111	Supervisor (OEA)	Both
PTEHI	981	11110	10101	Supervisor (OEA)	Both
PTELO	982	11110	10110	Supervisor (OEA)	Both
SPRG4 [2]	276	01000	10100	Supervisor (OEA)	Both
SPRG5 [2]	277	01000	10101	Supervisor (OEA)	Both
SPRG6 [2]	278	01000	100110	Supervisor (OEA)	Both
SPRG7 [2]	279	01000	10111	Supervisor (OEA)	Both
SVR	286	01000	11110	Supervisor (OEA)	**mfspr**
TLBMISS	980	11110	10100	Supervisor (OEA)	Both

[1] Note that the order of the two 5-bit halves of the SPR number is reversed compared with actual instruction coding. For **mtspr** and **mfspr** instructions, the SPR number coded in assembly language does not appear directly as a 10-bit binary number in the instruction. The number coded is split into two 5-bit halves that are reversed in the instruction, with the high-order 5 bits appearing in bits 16–20 of the instruction and the low-order 5 bits in bits 11–15.

[2] e600-specific register that may not be supported on other processors that implement the PowerPC architecture.

[3] Most bits are bit reset/write 1 clear. A write of 0 to a bit does not change it. A write of 1 to a bit clears it. Reads act normally.

6.3.6.3 Memory Control Instructions—OEA

Memory control instructions include the following:

- Cache management instructions (supervisor-level and user-level)
- Translation lookaside buffer management instructions

This section describes supervisor-level memory control instructions. Section 6.3.5.3, “Memory Control Instructions—VEA,” describes user-level memory control instructions.

6.3.6.3.1 Supervisor-Level Cache Management Instruction—OEA

Table 6-76 lists the only supervisor-level cache management instruction.

Table 6-76. Supervisor-Level Cache Management Instruction

Name	Mnemonic	Syntax	Implementation Notes
Data Cache Block Invalidate	**dcbi**	**r**A,**r**B	The **dcbi** instruction is executed identically to the **dcbf** instruction except that it is privileged (supervisor-only). See Section 6.3.5.3.1, “User-Level Cache Instructions—VEA.”

See Section 6.3.5.3.1, “User-Level Cache Instructions—VEA,” for cache instructions that provide user-level programs the ability to manage the caches in the core. If the effective address references a direct-store segment, the instruction is treated as a no-op.

6.3.6.3.2 Translation Lookaside Buffer Management Instructions—OEA

The address translation mechanism is defined in terms of the segment descriptors and page table entries (PTEs) that processors use to locate the logical-to-physical address mapping for a particular access. These segment descriptors and PTEs reside in segment registers within the core and page tables in memory, respectively.

Implementation Note—The e600 core provides two implementation-specific instructions (**tlbld** and **tlbli**) that are used by software table search operations following TLB misses to load TLB entries within the core when HID0[STEN] = 1.

For more information on **tlbld** and **tlbli** refer to Section 6.3.8, “Implementation-Specific Instructions.”

See the “Memory Management” chapter of the *e600 PowerPC Core Reference Manual* for more information about TLB operations. Table 6-77 summarizes the operation of the TLB instructions in the e600 core. Note that the broadcast of **tlbie** and **tlbsync** instructions is enabled by the setting of HID1[ABE].

Table 6-77. Translation Lookaside Buffer Management Instruction

Name	Mnemonic	Syntax	Implementation Notes
TLB Invalidate Entry	**tlbie**	**r**B	Invalidates both ways in both instruction and data TLB entries at the index provided by EA[14–19]. It executes regardless of the MSR[DR] and MSR[IR] settings. To invalidate all entries in both TLBs, the programmer should issue 64 **tlbie** instructions that each successively increment this field.
Load Data TLB Entry	**tlbld**	**r**B	Load Data TLB Entry Loads fields from the PTEHI and PTELO and the EA in **r**B to the way defined in **r**B[31].
Load Instruction TLB Entry	**tlbli**	**r**B	Load Instruction TLB Entry Loads fields from the PTEHI and PTELO and the EA in **r**B to the way defined in **r**B[31].
TLB Synchronize	**tlbsync**	—	TLBSYNC is broadcast.

Implementation Note—The **tlbia** instruction is optional for an implementation if its effects can be achieved through some other mechanism. Therefore, it is not implemented on the e600 core. As described above, **tlbie** can be used to invalidate a particular index of the TLB based on EA[14–19]—a sequence of 64 **tlbie** instructions followed by a **tlbsync** instruction invalidates all the TLB structures (for EA[14–19] = 0, 1, 2, . . . , 63). Attempting to execute **tlbia** causes an illegal instruction program interrupt.

The presence and exact semantics of the TLB management instructions are implementation-dependent. To minimize compatibility problems, system software should incorporate uses of these instructions into subroutines.

6.3.7 Recommended Simplified Mnemonics

The description of each instruction includes the mnemonic and a formatted list of operands. PowerPC-architecture-compliant assemblers support the mnemonics and operand lists. To simplify assembly language programming, a set of simplified mnemonics and symbols is provided for some of the most frequently-used instructions; refer to Appendix F, “Simplified Mnemonics,” in the *Programming Environments Manual* for a complete list. Programs written to be portable across the various assemblers for the PowerPC architecture should not assume the existence of mnemonics not described in this document.

6.3.8 Implementation-Specific Instructions

This section provides the details for the two e600 core implementation-specific instructions—**tlbld** and **tlbli**.

tlbld

tlbld

Load Data TLB Entry

Integer Unit

tlbld **r**B


```
EA ← (rB)
TLB entry created from PTEHI and PTELO
DTLB entry selected by EA[14–19] and rB[31] ← created TLB entry
```

The EA is the contents of **r**B. The **tlbld** instruction loads the contents of the PTEHI special purpose register and PTELO special purpose register into the selected data TLB entry. The set of the data TLB to be loaded is determined by EA[14–19]. The way to be loaded is determined by rB[31]. EA[10–13] are stored in the tag portion of the TLB and are used to match a new EA when a new EA is being translated.

The **tlbld** instruction should only be executed when address translation is disabled (MSR[IR] = 0 and MSR[DR] = 0).

Note that it is possible to execute the **tlbld** instruction when address translation is enabled; however, extreme caution should be used in doing so. If data address translation is enabled (MSR[DR] = 1), **tlbld** must be preceded by a **sync** instruction and succeeded by a context synchronizing instruction.

Note that if extended addressing is not enabled (HID0[XAEN] = 0), then PTELO[20–22] and PTELO[29] should be cleared (zero) by software when executing a **tlbld** instruction.

This is a supervisor-level instruction; it is also a e600-specific instruction, and not part of the PowerPC instruction set.

Other registers altered:

- None

tlbli tlbli

Load Instruction TLB Entry Integer Unit

tlbli **r**B


```
EA ← (rB)
TLB entry created from PTEHI and PTELO
ITLB entry selected by EA[14–19] and rB[31] ← created TLB entry
```

The EA is the contents of **r**B. The **tlbli** instruction loads an instruction TLB entry. The **tlbli** instruction loads the contents of the PTEHI special purpose register and PTELO special purpose register into a selected instruction TLB entry. The set of the instruction TLB to be loaded is determined by EA[14–19]. The way to be loaded is determined by rB[31]. EA[10–13] are stored in the tag portion of the TLB and are used to match a new EA when a new EA is being translated.

The **tlbli** instruction should only be executed when address translation is disabled (MSR[IR] = 0 and MSR[DR] = 0).

Note that it is possible to execute the **tlbli** instruction when address translation is enabled; however, extreme caution should be used in doing so. If instruction address translation is enabled (MSR[IR] = 1), **tlbli** must be followed by a context synchronizing instruction such as **isync** or **rfi**.

Note that if extended addressing is not enabled (HID0[XAEN]=0) then PTELO[20–22] and PTELO[29] should be cleared (set to zero) by software when executing a **tlbli** instruction.

Note also that care should be taken to avoid modification of the instruction TLB entries that translate current instruction prefetch addresses.

This is a supervisor-level instruction; it is also a e600-specific instruction, and not part of the PowerPC instruction set.

Other registers altered:

- None

6.4 AltiVec Instructions

The following sections provide a general summary of the instructions and addressing modes defined by the AltiVec Instruction Set Architecture (ISA). For specific details on the AltiVec instructions see the *AltiVec Technology Programming Environments Manual* and the "AltiVec Technology Implementation" chapter of the *e600 PowerPC Core Reference Manual*." AltiVec instructions belong primarily to the UISA, unless otherwise noted. AltiVec instructions are divided into the following categories:

- Vector integer arithmetic instructions—These include arithmetic, logical, compare, rotate and shift instructions, described in Section 6.3.4.1, "Integer Instructions."
- Vector floating-point arithmetic instructions—These floating-point arithmetic instructions and floating-point modes are described in Section 6.3.4.2, "Floating-Point Instructions."
- Vector load and store instructions—These load and store instructions for vector registers are described in Section 6.5.3, "Vector Load and Store Instructions."
- Vector permutation and formatting instructions—These include pack, unpack, merge, splat, permute, select, and shift instructions, and are described in Section 6.5.5, "Vector Permutation and Formatting Instructions."
- Processor control instructions—These instructions are used to read and write from the AltiVec status and control register, and are described in Section 6.3.4.6, "Processor Control Instructions—UISA."
- Memory control instructions—These instructions are used for managing caches (user level and supervisor level), and are described in Section 6.6.1, "AltiVec Vector Memory Control Instructions—VEA."

This grouping of instructions does not necessarily indicate the execution unit that processes a particular instruction or group of instructions within a processor implementation.

Integer instructions operate on byte, half-word, and word operands. Floating-point instructions operate on single-precision operands. The AltiVec ISA uses instructions that are four bytes long and word-aligned. It provides for byte, half-word, word, and quad-word operand fetches and stores between memory and the vector registers (VRs).

Arithmetic and logical instructions do not read or modify memory. To use the contents of a memory location in a computation and then modify the same or another memory location, the memory contents must be loaded into a register, modified, and then written to the target location using load and store instructions.

The AltiVec ISA supports both big-endian and little-endian byte ordering. The default byte and bit ordering is big-endian; see "Byte Ordering," in Chapter 3, "Operand Conventions," of the *AltiVec Technology Programming Environments Manual* for more information.

6.5 AltiVec UISA Instructions

This section describes the instructions defined in the AltiVec user instruction set architecture (UISA).

6.5.1 Vector Integer Instructions

The following are categories for vector integer instructions:

- Vector integer arithmetic instructions
- Vector integer compare instructions
- Vector integer logical instructions
- Vector integer rotate and shift instructions

Integer instructions use the content of VRs as source operands and also place results into VRs. Setting the Rc bit of a vector compare instruction causes the CR6 field of the PowerPC condition register (CR) to be updated; refer to Section 6.5.1.2, "Vector Integer Compare Instructions," for more details.

The AltiVec integer instructions treat source operands as signed integers unless the instruction is explicitly identified as performing an unsigned operation. For example, both the Vector Add Unsigned Word Modulo (**vadduwm**) and Vector Multiply Odd Unsigned Byte (**vmuloub**) instructions interpret the operands as unsigned integers.

6.5.1.1 Vector Integer Arithmetic Instructions

Table 6-78 lists the integer arithmetic instructions for the processors that implement the PowerPC architecture.

Table 6-78. Vector Integer Arithmetic Instructions

Name	Mnemonic	Syntax
Vector Add Unsigned Integer [b,h,w] Modulo1	**vaddubm** **vadduhm** **vadduwm**	**v**D,**v**A,**v**B
Vector Add Unsigned Integer [b,h,w] Saturate	**vaddubs** **vadduhs** **vadduws**	**v**D,**v**A,**v**B
Vector Add Signed Integer [b.h.w] Saturate	**vaddsbs** **vaddshs** **vaddsws**	**v**D,**v**A,**v**B
Vector Add and Write Carry-out Unsigned Word	**vaddcuw**	**v**D,**v**A,**v**B
Vector Subtract Unsigned Integer Modulo	**vsububm** **vsubuhm** **vsubuwm**	**v**D,**v**A,**v**B
Vector Subtract Unsigned Integer Saturate	**vsububs** **vsubuhs** **vsubuws**	**v**D,**v**A,**v**B

Table 6-78. Vector Integer Arithmetic Instructions (continued)

Name	Mnemonic	Syntax
Vector Subtract Signed Integer Saturate	**vsubsbs** **vsubshs** **vsubsws**	**v**D,**v**A,**v**B
Vector Subtract and Write Carry-out Unsigned Word	**vsubcuw**	**v**D,**v**A,**v**B
Vector Multiply Odd Unsigned Integer [b,h] Modulo	**vmuloub** **vmulouh**	**v**D,**v**A,**v**B
Vector Multiply Odd Signed Integer [b,h] Modulo	**vmulosb** **vmulosh**	**v**D,**v**A,**v**B
Vector Multiply Even Unsigned Integer [b,h] Modulo	**vmuleub** **vmuleuh**	**v**D,**v**A,**v**B
Vector Multiply Even Signed Integer [b,h] Modulo	**vmulesb** **vmulesh**	**v**D,**v**A,**v**B
Vector Multiply-High and Add Signed Half-Word Saturate	**vmhaddshs**	**v**D,**v**A,**v**B, **v**C
Vector Multiply-High Round and Add Signed Half-Word Saturate	**vmhraddshs**	**v**D,**v**A,**v**B,**v**C
Vector Multiply-Low and Add Unsigned Half-Word Modulo	**vmladduhm**	**v**D,**v**A,**v**B,**v**C
Vector Multiply-Sum Unsigned Integer [b,h] Modulo	**vmsumubm** **vmsumuhm**	**v**D,**v**A,**v**B,**v**C
Vector Multiply-Sum Signed Half-Word Saturate	**vmsumshs**	**v**D,**v**A,**v**B,**v**C
Vector Multiply-Sum Unsigned Half-Word Saturate	**vmsumuhs**	**v**D,**v**A,**v**B,**v**C
Vector Multiply-Sum Mixed Byte Modulo	**vmsummbm**	**v**D,**v**A,**v**B,**v**C
Vector Multiply-Sum Signed Half-Word Modulo	**vmsumshm**	**v**D,**v**A,**v**B,**v**C
Vector Sum Across Signed Word Saturate	**vsumsws**	**v**D,**v**A,**v**B
Vector Sum Across Partial (1/2) Signed Word Saturate	**vsum2sws**	**v**D,**v**A,**v**B
Vector Sum Across Partial (1/4) Unsigned Byte Saturate	**vsum4ubs**	**v**D,**v**A,**v**B
Vector Sum Across Partial (1/4) Signed Integer Saturate	**vsum4sbs** **vsum4shs**	**v**D,**v**A,**v**B
Vector Average Unsigned Integer	**vavgub** **vavguh** **vavguw**	**v**D,**v**A,**v**B
Vector Average Signed Integer	**vavgsb** **vavgsh** **vavgsw**	**v**D,**v**A,**v**B
Vector Maximum Unsigned Integer	**vmaxub** **vmaxuh** **vmaxuw**	**v**D,**v**A,**v**B
Vector Maximum Signed Integer	**vmaxsb** **vmaxsh** **vmaxsw**	**v**D,**v**A,**v**B

Table 6-78. Vector Integer Arithmetic Instructions (continued)

Name	Mnemonic	Syntax
Vector Minimum Unsigned Integer	**vminub** **vminuh** **vminuw**	**v**D,**v**A,**v**B
Vector Minimum Signed Integer	**vminsb** **vminsh** **vminsw**	**v**D,**v**A,**v**B

6.5.1.2 Vector Integer Compare Instructions

The vector integer compare instructions algebraically or logically compare the contents of the elements in vector register **v**A with the contents of the elements in **v**B. Each compare result vector is comprised of TRUE (0xFF, 0xFFFF, 0xFFFF_FFFF) or FALSE (0x00, 0x0000, 0x0000_0000) elements of the size specified by the compare source operand element (byte, half word, or word). The result vector can be directed to any VR and can be manipulated with any of the instructions as normal data (for example, combining condition results).

Vector compares provide equal-to and greater-than predicates. Others are synthesized from these by logically combining or inverting result vectors.

The integer compare instructions (shown in Table 6-80) can optionally set the CR6 field of the PowerPC condition register. If Rc = 1 in the vector integer compare instruction, then CR6 is set to reflect the result of the comparison, as follows in Table 6-79.

Table 6-79. CR6 Field Bit Settings for Vector Integer Compare Instructions

CR Bit	CR6 Bit	Vector Compare
24	0	1 Relation is true for all element pairs (that is, **v**D is set to all ones)
25	1	0
26	2	1 Relation is false for all element pairs (that is, register **v**D is cleared)
27	3	0

Table 6-80 summarizes the vector integer compare instructions.

Table 6-80. Vector Integer Compare Instructions

Name	Mnemonic	Syntax
Vector Compare Greater than Unsigned Integer	**vcmpgtub**[.] **vcmpgtuh**[.] **vcmpgtuw**[.]	**v**D,**v**A,**v**B
Vector Compare Greater than Signed Integer	**vcmpgtsb**[.] **vcmpgtsh**[.] **vcmpgtsw**[.]	**v**D,**v**A,**v**B
Vector Compare Equal to Unsigned Integer	**vcmpequb**[.] **vcmpequh**[.] **vcmpequw**[.]	**v**D,**v**A,**v**B

6.5.1.3 Vector Integer Logical Instructions

The vector integer logical instructions shown in Table 6-81 perform bit-parallel operations on the operands.

Table 6-81. Vector Integer Logical Instructions

Name	Mnemonic	Syntax
Vector Logical AND	**vand**	**v**D,**v**A,**v**B
Vector Logical OR	**vor**	**v**D,**v**A,**v**B
Vector Logical XOR	**vxor**	**v**D,**v**A,**v**B
Vector Logical AND with Complement	**vandc**	**v**D,**v**A,**v**B
Vector Logical NOR	**vnor**	**v**D,**v**A,**v**B

6.5.1.4 Vector Integer Rotate and Shift Instructions

The vector integer rotate instructions are summarized in Table 6-82.

Table 6-82. Vector Integer Rotate Instructions

Name	Mnemonic	Syntax
Vector Rotate Left Integer	**vrlb** **vrlh** **vrlw**	**v**D,**v**A,**v**B

The vector integer shift instructions are summarized in Table 6-83.

Table 6-83. Vector Integer Shift Instructions

Name	Mnemonic	Syntax
Vector Shift Left Integer	**vslb** **vslh** **vslw**	**v**D,**v**A,**v**B
Vector Shift Right Integer	**vsrb** **vsrh** **vsrw**	**v**D,**v**A,**v**B
Vector Shift Right Algebraic Integer	**vsrab** **vsrah** **vsraw**	**v**D,**v**A,**v**B

6.5.2 Vector Floating-Point Instructions

This section describes the vector floating-point instructions that include the following:

- Vector floating-point arithmetic instructions
- Vector floating-point rounding and conversion instructions
- Vector floating-point compare instructions
- Vector floating-point estimate instructions

The AltiVec floating-point data format complies with the ANSI/IEEE-754 standard as defined for single precision. A quantity in this format represents a signed normalized number, a signed denormalized number, a signed zero, a signed infinity, a quiet not a number (QNaN), or a signaling NaN (SNaN). Operations conform to the description in the section “AltiVec Floating-Point Instructions-UISA,” in Chapter 3, “Operand Conventions,” of the *AltiVec Technology Programming Environments Manual*.

The AltiVec ISA does not report IEEE exceptions but rather produces default results as specified by the Java/IEEE/C9X Standard; for further details on exceptions see “Floating-Point Exceptions,” in Chapter 3, “Operand Conventions,” of the *AltiVec Technology Programming Environments Manual*.

6.5.2.1 Vector Floating-Point Arithmetic Instructions

The floating-point arithmetic instructions are summarized in Table 6-84.

Table 6-84. Vector Floating-Point Arithmetic Instructions

Name	Mnemonic	Syntax
Vector Add Floating-Point	**vaddfp**	**v**D,**v**A,**v**B
Vector Subtract Floating-Point	**vsubfp**	**v**D,**v**A,**v**B
Vector Maximum Floating-Point	**vmaxfp**	**v**D,**v**A,**v**B
Vector Minimum Floating-Point	**vminfp**	**v**D,**v**A,**v**B

6.5.2.2 Vector Floating-Point Multiply-Add Instructions

Vector multiply-add instructions are critically important to performance because a multiply followed by a data dependent addition is the most common idiom in DSP algorithms. In most implementations, floating-point multiply-add instructions perform with the same latency as either a multiply or add alone, thus doubling performance in comparing to the otherwise serial multiply and adds.

AltiVec floating-point multiply-add instructions fuse (a multiply-add fuse implies that the full product participates in the add operation without rounding, only the final result rounds). This not only simplifies the implementation and reduces latency (by eliminating the intermediate rounding) but also increases the accuracy compared to separate multiply and adds.

The floating-point multiply-add instructions are summarized in Table 6-85.

Table 6-85. Vector Floating-Point Multiply-Add Instructions

Name	Mnemonic	Syntax
Vector Multiply-Add Floating-Point	**vmaddfp**	**v**D,**v**A,**v**C,**v**B
Vector Negative Multiply-Subtract Floating-Point	**vnmsubfp**	**v**D,**v**A,**v**C,**v**B

6.5.2.3 Vector Floating-Point Rounding and Conversion Instructions

All AltiVec floating-point arithmetic instructions use the IEEE Std. 754 default rounding mode round-to-nearest. The AltiVec ISA does not provide the IEEE Std. 754 directed rounding modes.

The AltiVec ISA provides separate instructions for converting floating-point numbers to integral floating-point values for all IEEE Std. 754 rounding modes as follows:

- Round-to-nearest (**vrfin**) (round)
- Round-toward-zero (**vrfiz**) (truncate)
- Round-toward-minus-infinity (**vrfim**) (floor)
- Round-toward-positive-infinity (**vrfip**) (ceiling)

Floating-point conversions to integers (**vctuxs**, **vctsxs**) use round-toward-zero (truncate) rounding. The floating-point rounding instructions are shown in Table 6-86.

Table 6-86. Vector Floating-Point Rounding and Conversion Instructions

Name	Mnemonic	Syntax
Vector Round to Floating-Point Integer Nearest	**vrfin**	**v**D,**v**B
Vector Round to Floating-Point Integer toward Zero	**vrfiz**	**v**D,**v**B
Vector Round to Floating-Point Integer toward Positive Infinity	**vrfip**	**v**D,**v**B
Vector Round to Floating-Point Integer toward Minus Infinity	**vrfim**	**v**D,**v**B
Vector Convert from Unsigned Fixed-Point Word	**vcfux**	**v**D,**v**B,UIMM
Vector Convert from Signed Fixed-Point Word	**vcfsx**	**v**D,**v**B,UIMM
Vector Convert to Unsigned Fixed-Point Word Saturate	**vctuxs**	**v**D,**v**B,UIMM
Vector Convert to Signed Fixed-Point Word Saturate	**vctsxs**	**v**D,**v**B,UIMM

6.5.2.4 Vector Floating-Point Compare Instructions

The floating-point compare instructions are summarized in Table 6-87.

Table 6-87. Vector Floating-Point Compare Instructions

Name	Mnemonic	Syntax
Vector Compare Greater Than Floating-Point [Record]	**vcmpgtfp**[.]	**v**D,**v**A,**v**B
Vector Compare Equal to Floating-Point [Record]	**vcmpeqfp**[.]	**v**D,**v**A,**v**B
Vector Compare Greater Than or Equal to Floating-Point [Record]	**vcmpgeqfp**[.]	**v**D,**v**A,**v**B
Vector Compare Bounds Floating-Point [Record]	**vcmpbfp**[.]	**v**D,**v**A,**v**B

6.5.2.5 Vector Floating-Point Estimate Instructions

The floating-point estimate instructions are summarized in Table 6-88.

Table 6-88. Vector Floating-Point Estimate Instructions

Name	Mnemonic	Syntax
Vector Reciprocal Estimate Floating-Point	**vrefp**	**v**D,**v**B
Vector Reciprocal Square Root Estimate Floating-Point	**vrsqrtefp**	**v**D,**v**B

Table 6-88. Vector Floating-Point Estimate Instructions (continued)

Name	Mnemonic	Syntax
Vector Log2 Estimate Floating-Point	**vlogefp**	**v**D,**v**B
Vector 2 Raised to the Exponent Estimate Floating-Point	**vexptefp**	**v**D,**v**B

6.5.3 Vector Load and Store Instructions

Only very basic load and store operations are provided in the AltiVec ISA. This keeps the circuitry in the memory path fast so the latency of memory operations is minimized. Instead, a powerful set of field manipulation instructions are provided to manipulate data into the desired alignment and arrangement after the data has been brought into the VRs.

Load vector indexed (**lvx**, **lvxl**) and store vector indexed (**stvx**, **stvxl**) instructions transfer an aligned quad-word vector between memory and VRs. Load vector element indexed (**lvebx**, **lvehx**, **lvewx**) and store vector element indexed instructions (**stvebx**, **stvehx**, **stvewx**) transfer byte, half-word, and word scalar elements between memory and VRs.

6.5.3.1 Vector Load Instructions

For vector load instructions, the byte, half-word, word, or quad-word addressed by the EA (effective address) is loaded into **v**D.

The default byte and bit ordering is big-endian as in the PowerPC architecture; see “Byte Ordering,” in Chapter 3, “Operand Conventions,” of the *AltiVec Technology Programming Environments Manual* for information about little-endian byte ordering.

Table 6-89 summarizes the vector load instructions.

Table 6-89. Vector Integer Load Instructions

Name	Mnemonic	Syntax
Load Vector Element Integer Indexed	**lvebx** **lvehx** **lvewx**	**v**D,**r**A,**r**B
Load Vector Element Indexed	**lvx**	**v**D,**r**A,**r**B
Load Vector Element Indexed LRU [1]	**lvxl**	**v**D,**r**A,**r**B

[1] On the e600 core, **lvxl** and **stvxl** are interpreted to be transient. See The “AltiVec Technology Implementation” chapter of the *e600 PowerPC Core Reference Manual*.”

6.5.3.2 Vector Load Instructions Supporting Alignment

The **lvsl** and **lvsr** instructions can be used to create the permute control vector to be used by a subsequent **vperm** instruction. Let X and Y be the contents of **v**A and **v**B specified by **vperm**. The control vector created by **lvsl** causes the **vperm** to select the high-order 16 bytes of the result of shifting the 32-byte value X || Y left by sh bytes (sh = the value in EA[60–63]). The control vector created by **lvsr** causes the **vperm** to select the low-order 16 bytes of the result of shifting X || Y right by sh bytes.

Table 6-90 summarizes the vector alignment instructions.

Table 6-90. Vector Load Instructions Supporting Alignment

Name	Mnemonic	Syntax
Load Vector for Shift Left	**lvsl**	**v**D,**r**A,**r**B
Load Vector for Shift Right	**lvsr**	**v**D,**r**A,**r**B

6.5.3.3 Vector Store Instructions

For vector store instructions, the contents of the VR used as a source (**v**S) are stored into the byte, half word, word or quad word in memory addressed by the effective address (EA). Table 6-91 provides a summary of the vector store instructions.

Table 6-91. Vector Integer Store Instructions

Name	Mnemonic	Syntax
Store Vector Element Integer Indexed	**svetbx** **svethx** **svetwx**	**v**S,**r**A,**r**B
Store Vector Element Indexed	**stvx**	**v**S,**r**A,**r**B
Store Vector Element Indexed LRU[1]	**stvxl**	**v**S,**r**A,**r**B

[1] On the e600 core, **lvxl** and **stvxl** are interpreted to be transient. See the "AltiVec Technology Implementation" chapter of the *e600 PowerPC Core Reference Manual*."

6.5.4 Control Flow

AltiVec instructions can be freely intermixed with existing PowerPC instructions to form a complete program. AltiVec instructions provide a vector compare and select mechanism to implement conditional execution as the preferred mechanism to control data flow in AltiVec programs. In addition, AltiVec vector compare instructions can update the condition register thus providing the communication from AltiVec execution units to PowerPC branch instructions necessary to modify program flow based on vector data.

6.5.5 Vector Permutation and Formatting Instructions

Vector pack, unpack, merge, splat, permute, select, and shift instructions can be used to accelerate various vector math operations and vector formatting. Details of these instructions follow.

6.5.5.1 Vector Pack Instructions

Half-word vector pack instructions (**vpkuhum**, **vpkuhus**, **vpkshus**, **vpkshss**) truncate the sixteen half words from two concatenated source operands producing a single result of sixteen bytes (quad word) using either modulo (2^8), 8-bit signed-saturation, or 8-bit unsigned-saturation to perform the truncation. Similarly, word vector pack instructions (**vpkuwum**, **vpkuwus**, **vpkswus**, **vpksws**) truncate the eight words from two concatenated source operands, producing a single result of eight half words using modulo (2^{16}), 16-bit signed-saturation, or 16-bit unsigned-saturation to perform the truncation.

Table 6-92 describes the vector pack instructions.

Table 6-92. Vector Pack Instructions

Name	Mnemonic	Syntax
Vector Pack Unsigned Integer [h,w] Unsigned Modulo	**vpkuhum** **vpkuwum**	**v**D, **v**A, **v**B
Vector Pack Unsigned Integer [h,w] Unsigned Saturate	**vpkuhus** **vpkuwus**	**v**D, **v**A, **v**B
Vector Pack Signed Integer [h,w] Unsigned Saturate	**vpkshus** **vpkswus**	**v**D, **v**A, **v**B
Vector Pack Signed Integer [h,w] signed Saturate	**vpkshss** **vpkswss**	**v**D, **v**A, **v**B
Vector Pack Pixel	**vpkpx**	**v**D, **v**A, **v**B

6.5.5.2 Vector Unpack Instructions

Byte vector unpack instructions unpack the 8 low bytes (or 8 high bytes) of one source operand into 8 half words using sign extension to fill the most-significant bytes (MSBs). Half word vector unpack instructions unpack the 4 low half words (or 4 high half words) of one source operand into 4 words using sign extension to fill the MSBs.

Two special purpose forms of vector unpack are provided—the Vector Unpack Low Pixel (**vupklpx**) and the Vector Unpack High Pixel (**vupkhpx**) instructions for 1/5/5/5 αRGB pixels. The 1/5/5/5 pixel vector unpack, unpacks the four low 1/5/5/5 pixels (or four 1/5/5/5 high pixels) into four 32-bit (8/8/8/8) pixels. The 1-bit α element in each pixel is sign extended to 8 bits, and the 5-bit R, G, and B elements are each zero extended to 8 bits.

Table 6-93 describes the unpack instructions.

Table 6-93. Vector Unpack Instructions

Name	Mnemonic	Syntax
Vector Unpack High Signed Integer	**vupkhsb** **vupkhsh**	**v**D, **v**B
Vector Unpack High Pixel	**vupkhpx**	**v**D, **v**B
Vector Unpack Low Signed Integer	**vupklsb** **vupklsh**	**v**D, **v**B
Vector Unpack Low Pixel	**vupklpx**	**v**D, **v**B

6.5.5.3 Vector Merge Instructions

Byte vector merge instructions interleave the 8 low bytes or 8 high bytes from two source operands producing a result of 16 bytes. Similarly, half-word vector merge instructions interleave the 4 low half words (or 4 high half words) of two source operands producing a result of 8 half words, and word vector merge instructions interleave the 2 low words or 2 high words from two source operands producing a result

of 4 words. The vector merge instruction has many uses. For example, it can be used to efficiently transpose SIMD vectors. Table 6-94 describes the merge instructions.

Table 6-94. Vector Merge Instructions

Name	Mnemonic	Syntax
Vector Merge High Integer	**vmrghb** **vmrghh** **vmrghw**	**v**D, **v**A, **v**B
Vector Merge Low Integer	**vmrglb** **vmrglh** **vmrglw**	**v**D, **v**A, **v**B

6.5.5.4 Vector Splat Instructions

When a program needs to perform arithmetic vector operations, the vector splat instructions can be used in preparation for performing arithmetic for which one source vector is to consist of elements that all have the same value. Vector splat instructions can be used to move data where it is required. For example to multiply all elements of a vector register (VR) by a constant, the vector splat instructions can be used to splat the scalar into the VR. Likewise, when storing a scalar into an arbitrary memory location, it must be splatted into a VR, and that VR must be specified as the source of the store. This guarantees that the data appears in all possible positions of that scalar size for the store.

Table 6-95. Vector Splat Instructions

Name	Mnemonic	Syntax
Vector Splat Integer	**vspltb** **vsplth** **vspltw**	**v**D, **v**B, UIMM
Vector Splat Immediate Signed Integer	**vspltisb** **vspltish** **vspltisw**	**v**D, SIMM

6.5.5.5 Vector Permute Instructions

Permute instructions allow any byte in any two source VRs to be directed to any byte in the destination vector. The fields in a third source operand specify from which field in the source operands the corresponding destination field is taken. The Vector Permute (**vperm**) instruction is a very powerful one that provides many useful functions. For example, it provides a way to perform table-lookups and data alignment operations. An example of how to use the **vperm** instruction in aligning data is described in “Quad-Word Data Alignment” in Chapter 3, “Operand Conventions,” of the *AltiVec Technology Programming Environments Manual*. Table 6-92 describes the vector permute instruction.

Table 6-96. Vector Permute Instruction

Name	Mnemonic	Syntax
Vector Permute	**vperm**	**v**D, **v**A,**v**B,**v**C

6.5.5.6 Vector Select Instruction

Data flow in the vector unit can be controlled without branching by using a vector compare and the Vector Select (**vsel**) instructions. In this use, the compare result vector is used directly as a mask operand to vector select instructions.The **vsel** instruction selects one field from one or the other of two source operands under control of its mask operand. Use of the TRUE/FALSE compare result vector with select in this manner produces a two instruction equivalent of conditional execution on a per-field basis. Table 6-97 describes the **vsel** instruction.

Table 6-97. Vector Select Instruction

Name	Mnemonic	Syntax
Vector Select	**vsel**	**v**D,**v**A,**v**B,**v**C

6.5.5.7 Vector Shift Instructions

The vector shift instructions shift the contents of one or of two VRs left or right by a specified number of bytes (**vslo**, **vsro**, **vsldoi**) or bits (**vsl**, **vsr**). Depending on the instruction, this shift count is specified either by low-order bits of a VR or by an immediate field in the instruction. In the former case the low-order 7 bits of the shift count register give the shift count in bits ($0 \le count \le 127$). Of these 7 bits, the high-order 4 bits give the number of complete bytes by which to shift and are used by **vslo** and **vsro**; the low-order 3 bits give the number of remaining bits by which to shift and are used by **vsl** and **vsr**.

Table 6-98 describes the vector shift instructions.

Table 6-98. Vector Shift Instructions

Name	Mnemonic	Syntax
Vector Shift Left	**vsl**	**v**D,**v**A,**v**B
Vector Shift Right	**vsr**	**v**D,**v**A,**v**B
Vector Shift Left Double by Octet Immediate	**vsldoi**	**v**D,**v**A,**v**B,SH
Vector Shift Left by Octet	**vslo**	**v**D,**v**A,**v**B
Vector Shift Right by Octet	**vsro**	**v**D,**v**A,**v**B

6.5.5.8 Vector Status and Control Register Instructions

Table 6-99 summarizes the instructions for reading from or writing to the AltiVec status and control register (VSCR), described in the “AltiVec Technology Implementation” chapter of the *e600 PowerPC Core Reference Manual*.”

Table 6-99. Move To/From VSCR Register Instructions

Name	Mnemonic	Syntax
Move to AltiVec Status and Control Register	**mtvscr**	**v**B
Move from AltiVec Status and Control Register	**mfvscr**	**v**B

6.6 AltiVec VEA Instructions

The PowerPC virtual environment architecture (VEA) describes the semantics of the memory model that can be assumed by software processes, and includes descriptions of the cache model, cache-control instructions, address aliasing, and other related issues. Implementations that conform to the VEA also adhere to the UISA, but may not necessarily adhere to the OEA. For further details, see Chapter 4, “Addressing Mode and Instruction Set Summary,” in the *Programming Environments Manual.*

This section describes the additional instructions that are provided by the AltiVec ISA for the VEA.

6.6.1 AltiVec Vector Memory Control Instructions—VEA

Memory control instructions include the following types:

- Cache management instructions (user-level and supervisor-level)
- Translation lookaside buffer (TLB) management instructions

This section briefly summarizes the user-level cache management instructions defined by the AltiVec VEA. See the “L1 and L2 Cache Operation” chapter of the *e600 PowerPC Core Reference Manual*. for more information about supervisor-level cache, segment register manipulation, and TLB management instructions.

The AltiVec architecture specifies the data stream touch instructions **dst(t)**, **dstst(t)**, and it specifies two data stream stop (**dss(all)**) instructions. The e600 core implements all of them. The term **dst***x* used below refers to all of the stream touch instructions.

The instructions summarized in this section provide user-level programs the ability to manage caches within the core. See the “L1 and L2 Cache Operation” chapter of the *e600 PowerPC Core Reference Manual* for more information about cache topics.

Bandwidth between the processor and memory is managed explicitly by the programmer through the use of cache management instructions. These instructions provide a way for software to communicate to the cache hardware how it should prefetch and prioritize the writeback of data. The principal instruction for this purpose is a software-directed cache prefetch instruction called data stream touch (**dst**). Other related instructions are provided for complete control of the software-directed cache prefetch mechanism.

Table 6-100 summarizes the directed prefetch cache instructions defined by the AltiVec VEA. Note that these instructions are accessible to user-level programs.

Table 6-100. AltiVec User-Level Cache Instructions

Name	Mnemonic	Syntax	Implementation Notes
Data Stream Touch (non-transient)	**dst**	**r**A,**r**B,STRM	—
Data Stream Touch Transient	**dstt**	**r**A,**r**B,STRM	Used for last access
Data Stream Touch for Store	**dstst**	**r**A,**r**B,STRM	Not recommended for use in the e600 core
Data Stream Touch for Store Transient	**dststt**	**r**A,**r**B,STRM	Not recommended for use in the e600 core
Data Stream Stop (one stream)	**dss**	STRM	—
Data Stream Stop All	**dssall**	STRM	—

For detailed information for how to use these instructions, see the “AltiVec Technology Implementation” chapter of the *e600 PowerPC Core Reference Manual.*

6.6.2 AltiVec Instructions with Specific Implementations for the e600 Core

The AltiVec architecture specifies Load Vector Indexed LRU (**lvxl**) and Store Vector Indexed LRU (**stvxl**) instructions. The architecture suggests that these instructions differ from regular AltiVec load and store instructions in that they leave cache entries in a least recently used (LRU) state instead of a most recently used (MRU) state. This supports efficient processing of data which is known to have little reuse and poor caching characteristics. The e600 core implements these instructions as suggested. They follow all the cache allocation and replacement policies described in the “L1 and L2 Cache Operation” chapter of the *e600 PowerPC Core Reference Manual*, but they leave their addressed cache entries in the LRU state. In addition, all LRU instructions are also interpreted to be transient and are also treated as described in the “AltiVec Technology Implementation” chapter of the *e600 PowerPC Core Reference Manual.*

Part III
Memory, Peripherals, and I/O Interfaces

Part III defines the memory, peripherals, and I/O interfaces of the MPC8641 and it describes how these blocks interact with one another and with other blocks on the device. The following chapters are included:

- Chapter 7, "MPX Coherency Module (MCM) Overview," defines the e600 coherency module and how it facilitates communication between the e600 core complex, the L2 cache and the other blocks that comprise the coherent memory domain of the MPC8641.

  The MCM provides a mechanism for I/O-initiated transactions to snoop the core complex bus (CCB) of the e600 core in order to maintain coherency across cacheable local memory. It also provides a flexible, easily expandable switch-type structure for e600- and I/O-initiated transactions to be routed (dispatched) to target modules on the MPC8641.
- Chapter 8, "DDR Memory Controllers," describes the DDR SDRAM memory controller of the MPC8641. 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. Special features like ECC error injection support rapid system debug.
- Chapter 9, "Programmable Interrupt Controller (PIC)," describes the embedded programmable interrupt controller (PIC) of the MPC8641. 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 10, "I^2C Interfaces," describes the inter-IC (IIC or I^2C) bus controller of the MPC8641. 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 MPC8641 powers up in boot sequencer mode which allows the I^2C controller to initialize configuration registers.
- Chapter 11, "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 12, "Local Bus Controller," describes the local bus controller of the MPC8641. The main component of the local bus controller (LBC) is its memory controller which provides a seamless interface to many types of memory devices and peripherals. The memory controller is responsible for controlling eight memory banks shared by a high performance SDRAM machine, a general-purpose chip-select machine (GPCM), and up to three user-programmable machines (UPMs). As such, it supports a minimal glue logic interface to synchronous DRAM (SDRAM),

SRAM, EPROM, flash EPROM, burstable RAM, regular DRAM devices, extended data output DRAM devices, and other peripherals.

- Chapter 13, “Enhanced Three-Speed Ethernet Controllers,” describes the enhanced three-speed Ethernet controllers (TSECs) of the MPC8641 interface to 10 Mbps, 100 Mbps, and 1 Gbps Ethernet/IEEE 802.3 networks and devices featuring generic 8-/16-bit FIFO ports.
- Chapter 14, “DMA Controller,” describes the four-channel general-purpose DMA controller of the MPC8641. The DMA controller transfers blocks of data, independent of the e600 core or external hosts. Data movement occurs among RapidIO and the local address space. The DMA controller has four high-speed channels. Both the e600 core and external masters can initiate a DMA transfer. All channels are capable of complex data movement and advanced transaction chaining.
- Chapter 15, “Serial RapidIO Interface,” describes the Serial RapidIO controller, which conforms to Revision 1.2 of the *Parallel RapidIO Interconnect Specification*. This chapter describes the implementation of the serial RapidIO controller.
- Chapter 16, “PCI Express Interface Controller,” complies with the *PCI Express™ Base Specification, Revision 1.0a* (available from http://www.pcisig.org). It is beyond the scope of this manual to document the intricacies of the PCI Express protocol. This chapter describes the PCI Express controller of this device and provides a basic description of the PCI Express protocol.

Chapter 7
MPX Coherency Module (MCM) Overview

7.1 Introduction

The MPX coherency module provides a mechanism for I/O-initiated transactions to be snooped in order to maintain data coherency for cacheable address spaces. It has an MPX bus arbiter which supports two MPX bus ports: one for each core. The MCM also provides a flexible switch structure for routing CPU and I/O-initiated transactions to target modules on the device.

Figure 7-1 shows a high level block diagram of the MPX coherency module (MCM).

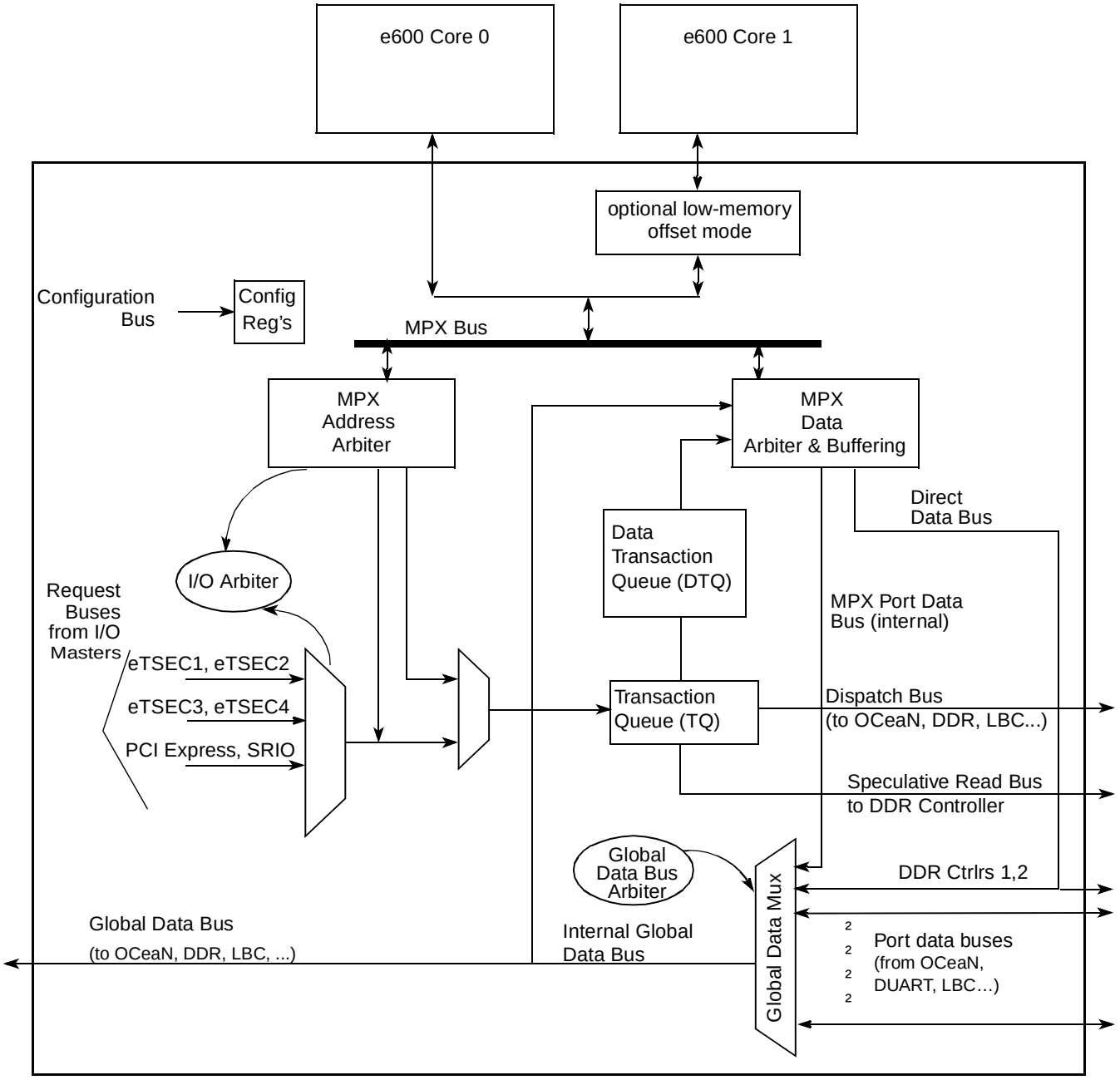


Figure 7-1. MPX Coherency Module (MCM) Overview

7.1.1 Overview

The MCM routes transactions initiated by either of the e600 cores to the appropriate target interface on the device. In a manner analogous to a bridging router in a local area network, the MCM also forwards I/O-initiated transactions that are tagged with the global attribute (snoopable) onto the MPX bus interface. This allows the core caches to snoop these transactions as if they were locally initiated and to take actions to maintain coherency across cacheable memory.

7.2 Features

The MCM includes these distinctive features:

- Supports the MPX bus protocol of the cores with built-in arbiter
  - Built-in arbiter for the two MPX ports: programmable priorities
  - Supports both address tenure streaming and data tenure streaming
  - Support for fully serialized mode on MPX bus
  - Split transaction support
  - Supports 36-bit addresses throughout the device, 64-bit MPX data bus, single beat, 2 beat bursts and 4 beat bursts of data transfer
  - Supports an 8-entry data transaction queue (DTQ) per MPX port
  - Supports out-of-order data transactions on the MPX bus
- Provides full cache coherency between the cores' caches and global I/O transactions
  - Supports data intervention on snoop hits
  - Supports window-of-opportunity and 'cache-to-cache' read intervention (snarfing)
- Supports 1 or 2 wait states from transfer start (TS) to address acknowledge (AACK) on the MPX bus. For a 3:1 clock ratio or greater core-to-platform frequency ratio there is a single wait state. For ratios less than 3:1 there are 2 wait states.
- Has 8 cache lines of data buffering for core-initiated read transactions and 8 cache lines of data buffering for MPX-initiated write transactions.
- Ensures ordering of I/O-initiated transactions on a per-master-ID basis
- Supports speculative read bus for low latency dispatch of reads from local prefetchable memory space (DDR controllers).
- Provides low latency path for returning read data from local memory target to MPX data tenure.
- Error management. Errors can be programmed to generate interrupts to the e600 core, as described in the following sections:
  - Section 7.4.1.4, "Error Detect Register (EDR)"
  - Section 7.4.1.5, "Error Enable Register (EER)"
  - Section 7.4.1.6, "Error Attributes Capture Register (EATR)"
  - Section 7.4.1.7, "Error Low Address Register (ELADR)"
  - Section 7.4.1.8, "Error High Address Register (EHADR)"

7.3 Modes of Operation

There are various programmable options that are controlled by the MPX address bus configuration register (ABCR), the data bus configuration register (DBCR) and the port configuration register (PCR).

The address bus arbiter (ABA) uses a fair arbitration scheme to arbitrate for ownership of the address bus and the data bus. It can use either a least-recently-used (LRU) or round robin policy to arbitrate the address bus. The ABA also supports address tenure streaming and address bus parking. The data bus arbiter uses a round-robin policy to arbitrate the data bus.

7.4 Memory Map/Register Definition

Table 7-1 shows the memory mapped registers of the MCM and their offsets. It lists the offset, name, and a cross-reference to the complete description of each register. Note that the full register address is comprised of CCSRBAR together with the block base address and offset listed in Table 7-1. Undefined 4-byte address spaces within offset 0x000–0xFFF are reserved.

Table 7-1. MCM Memory Map

MCM—Block Base Address 0x0_1000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0x000	MCM MPX address bus configuration register (ABCR)	R/W	0x0000_0003	7.4.1.1/7-5
0x008	MCM MPX data bus configuration register (DBCR)	R/W	0x0000_0003	7.4.1.2/7-6
0x010	MCM MPX port configuration register (PCR)	R/W	0x0*00_0000	7.4.1.3/7-6
0xBF8	MCM IP block revision register 1	R	0x0001_0000	—
0xBFC	MCM IP block revision register 2	R	0x0000_0000	—
0xE00	MCM error detect register (EDR)	w1c	0x0000_0000	7.4.1.4/7-7
0xE08	MCM error enable register (EER)	R/W	0x0000_0000	7.4.1.5/7-9
0xE0C	MCM error attributes capture register (EATR)	R	0x0000_0000	7.4.1.6/7-9
0xE10	MCM error low address capture register (ELADR)	R	0x0000_0000	7.4.1.7/7-11
0xE14	MCM error high address capture register (EHADR)	R	0x0000_0000	7.4.1.8/7-11

7.4.1 Register Descriptions

This section consists of detailed descriptions of the registers summarized in Table 7-1. Note that these registers are shown in big-endian format.

7.4.1.1 Address Bus Configuration Register (ABCR)

The MCM address bus configuration register (ABCR), shown in Figure 7-2, controls arbitration and streaming policies for the MPX bus.

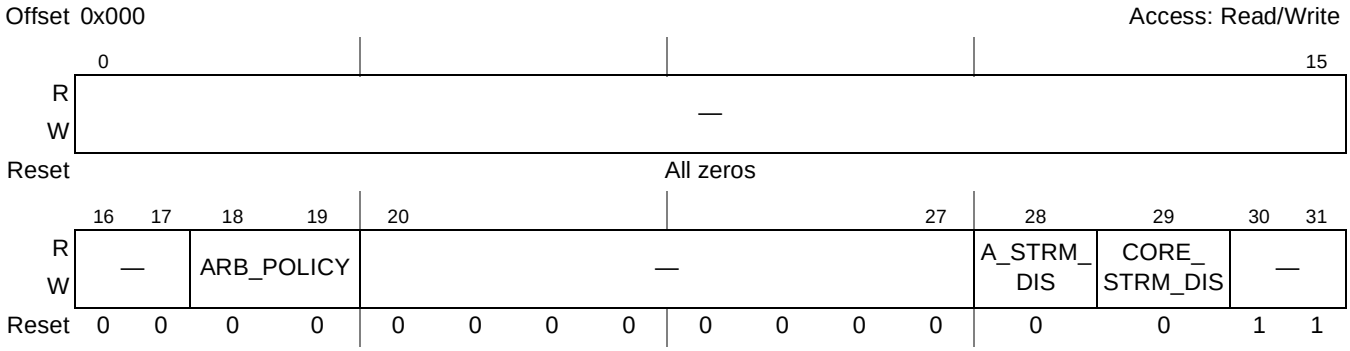


Figure 7-2. Address Bus Configuration Register (ABCR)

Table 7-2 describes the ABCR fields.

Table 7-2. ABCR Field Descriptions

Bits	Name	Description
0–17	—	Reserved
18–19	ARB_POLICY	Defines the address bus arbitration policy that is used on the MPX bus. This field may be used to tune for maximum performance for a particular application. 00 Longest-waiting (LW) 01 Round robin 10 Reserved 11 Reserved
20–27	—	Reserved
28	A_STRM_DIS	Controls whether the MCM, as a master of transactions, streams the address tenures. Note that this bit operates independently from the CORE_STRM_DIS bit. 0 MCM streams address tenures 1 MCM does not stream address tenures.
29	CORE_STRM_DIS	Controls whether the e600 cores can stream commands onto the MPX bus. 0 Address tenures initiated by the e600 cores are streamed. 1 Streaming of address tenures initiated by the e600 cores not allowed.
30–31	—	Reserved

7.4.1.2 Data Bus Configuration Register (DBCR)

The data bus configuration register (DBCR), shown in Figure 7-3, controls the MPX data bus.

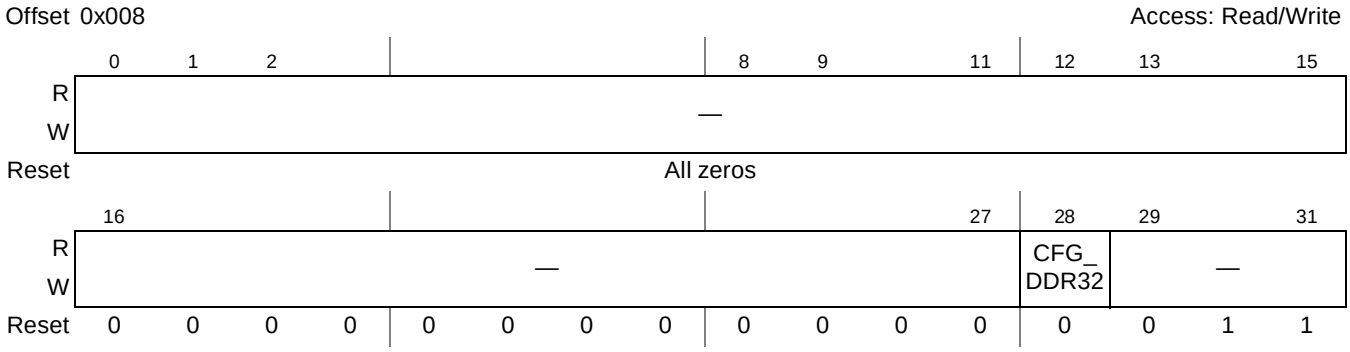


Figure 7-3. Data Bus Configuration Register (DBCR)

Table 7-3 describes the DBCR fields.

Table 7-3. DBCR Field Descriptions

Bits	Name	Description
0–27	—	Reserved
28	CFG_DDR32	MCM configuration for 32-bit DDR controller operation. If the DDR controller is operated in 32-bit mode (DDR_SDRAM_CFG[32_BE] = 1), then this bit must be set to properly configure the MCM prior to enabling the DDR controller. 0 MCM is configured to operate with the DDR controller in 64-bit mode (DDR_SDRAM_CFG[32_BE] = 0). 1 MCM is configured to operate with the DDR controller in 32-bit mode (DDR_SDRAM_CFG[32_BE] = 1).
29–31	—	Reserved

7.4.1.3 Port Configuration Register (PCR)

The port configuration register (PCR), shown in Figure 7-4, enables the cores' access to the MPX bus, and controls the priorities of core accesses with respect to the MCM.

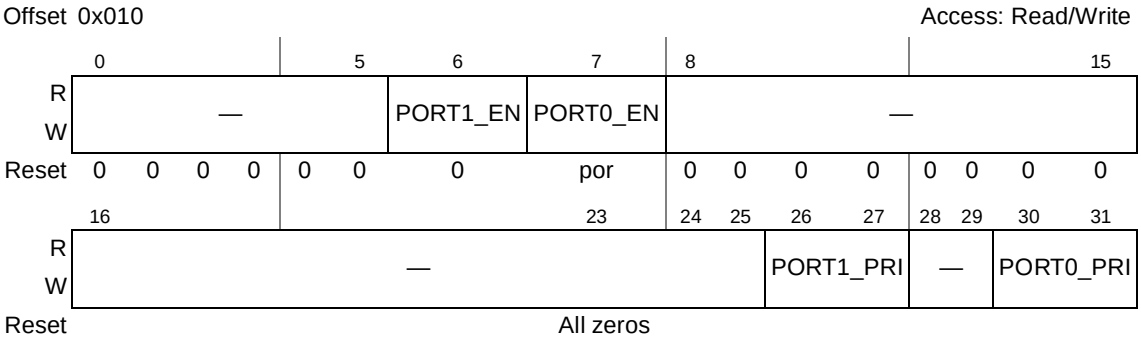


Figure 7-4. Port Configuration Register (PCR)

Table 7-4 describes the PCR fields.

Table 7-4. PCR Field Descriptions

Bits	Name	Description
0–5	—	Reserved
6	PORT1_EN	CPU port 1 enable. Identifies whether MPX port 1 (core 1) can receive bus grants. 0 Port 1 is disabled and does not receive bus grants. Note that this bit is cleared on reset. 1 Port 1 is enabled and receives bus grants in response to bus requests. After this bit is set, it should not be cleared by software. It is not intended for port 1 to be dynamically enabled and disabled. This bit is only intended to end boot holdoff mode for core 1.
7	PORT0_EN	CPU port 0 enable. Identifies whether MPX port 0 (core 0) can receive bus grants. The core 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, core 0 is enabled to boot at the end of the POR sequence. Otherwise, the core cannot fetch its boot vector until an external host sets this PORT0_EN bit. 0 Boot holdoff mode. Port 0 (core 0) arbitration is disabled on the MPX and no bus grants are issued. 1 Port 0 (core 0 is enabled and receives bus grants in response to bus requests. After this bit is set, it should not be cleared by software. It is not intended for port 0 to be dynamically enabled and disabled. This bit is only intended to end boot holdoff mode for core 0. See Section 4.4.3.10, “CPU Boot Configuration,” for more information.
8–2527	—	Reserved
26–27	PORT1_PRI	Identifies the priority level of port 1 (core 1). This priority level is used in streaming mode (when ABCR[A_STRM_DIS] is cleared) to determine whether a port’s MPX bus request can cause the MCM address bus arbiter to terminate another port’s streaming of address tenures if the streaming count has not yet expired. 00 Lowest priority level 01 Second lowest priority level 10 Highest priority level 11 Reserved
30–31	PORT0_PRI	Specifies the priority level of port 0 (core 0). This priority level is used in streaming mode (when ABCR[A_STRM_DIS] is cleared) to determine whether an I/O master’s MPX bus request can cause the MPX arbiter to terminate the core’s streaming of address tenures. 00 Lowest priority level 01 Second lowest priority level 10 Highest priority level 11 Reserved

7.4.1.4 Error Detect Register (EDR)

The error detect register (EDR), shown in Figure 7-5, reports the detection of enabled errors.

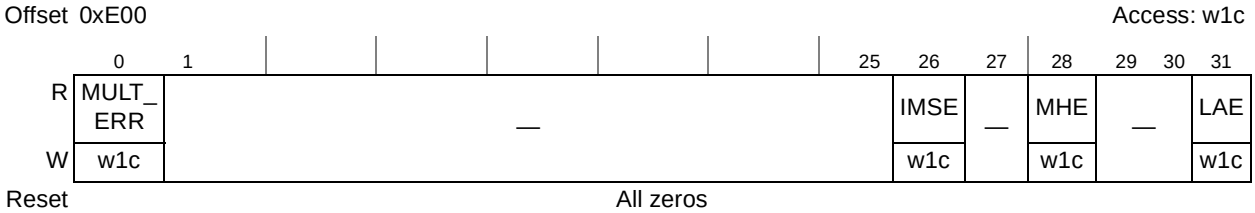


Figure 7-5. Error Detect Register (EDR)

Table 7-5 describes the EDR fields.

Table 7-5. EDR Field Descriptions

Bits	Name	Description
0	MULT_ERR	Multiple errors. Indicates the occurrence of multiple errors of the same type. Write 1 to clear. 0 Multiple errors of the same type were not detected. 1 Multiple errors of the same type were detected. Occurs when an error event is detected and its corresponding EDR bit is still set. When this occurs, the MCM signals an interrupt to the on-chip interrupt controller.
1–25	—	Reserved
26	IMSE	Interleaving index and mode bit error. Set when the interleaving index and mode bits from both DDR memory controllers do not match. This bit is only cleared when both memory controllers have the same values. This error cannot be masked with the EER register. 0 Interleaving index and mode bit error has not occurred. 1 Interleaving index and mode bit error has occurred. When this occurs, the MCM signals an interrupt to the on-chip interrupt controller.
27	—	Reserved
28	MHE	Multiple hit error. The MCM always detects multiple hit errors. This error occurs when an I/O-initiated read maps to a local MCM target, and the two cores both signal a hit. This is either a case where the two cores are trying to intervene on a snoop, or at least one of the cores claims backing store for that address location. In addition, if a core issues a snoop transaction (and the same master signals hit for the transaction that it is mastering), and that transaction also maps to a local MCM target; it is also considered a multiple hit error event. When this error occurs and none of the MCM error capture registers are set, the MCM captures the address and attributes of the transaction into the EATR, ELADR, and EHADR registers. Furthermore, if EER[MHEE] is set while EDR[MHE] is set, the MCM signals an interrupt to the on-chip interrupt controller. This bit is cleared (if it is set) by writing a logic 1 to it. 0 Multiple hit error was not detected on the MPX bus. 1 Multiple hit error was detected on the MPX bus.
29–30	—	Reserved
31	LAE	Local access error. Write 1 to clear. The following cases can generate LAEs in the MCM: • Transaction does not map to any target. In this case the MCM injects read responses (with the corrupt attribute set) and write data is dropped. Note that address-only flush, clean, and intervention transactions will not generate local access errors; however, a tlbie broadcast can generate an LAE if the effective address is not mapped by an LAW. • 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 the OCN target in this case) is not blocked; the LAE error is reported, but the transaction is still sent to its OCN target. When this type of error occurs, and none of the MCM error capture registers are set, the MCM captures the address and attributes of the transaction into the EATR, ELADR, and EHADR registers. Furthermore, if EER[LAEE] is set while EDR[LAE] is set, the MCM signals an interrupt to the on-chip interrupt controller. This bit is cleared (if it is set) by writing a logic 1 to it. 0 Local access error has not occurred. 1 Local access error occurred.

7.4.1.5 Error Enable Register (EER)

The error enable register (EER), shown in Figure 7-6, enables the reporting of MCM-detected errors as interrupts (through the on-chip interrupt controller).

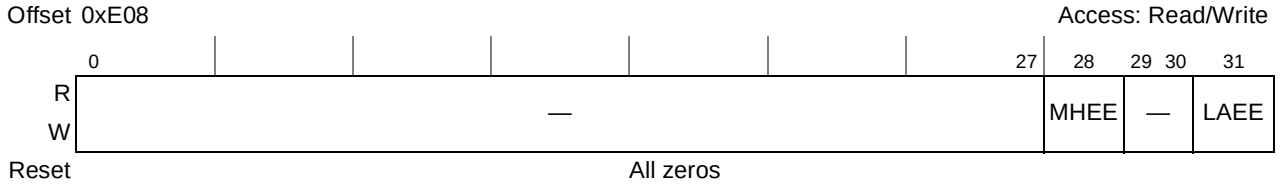


Figure 7-6. Error Enable Register (EER)

Table 7-6 describes the EER fields.

Table 7-6. EER Field Descriptions

Bits	Name	Description
0–27	—	Reserved
28	MHEE	Multiple hit error enable. Enables the reporting of an interrupt in the case of a multiple hit error. If this bit is set and a multiple hit error occurs (setting EDR[MHE]), the MCM signals an interrupt to the on-chip interrupt controller. For more information on multiple hit errors, see the description for EDR[MHE]. 0 Disables reporting of multiple hit errors as interrupts. 1 Enables reporting of multiple hit errors as interrupts.
29–30	—	Reserved
31	LAEE	Local access error enable. Enables the reporting of an interrupt in the case of a local access error. If this bit is set and a local access error occurs (setting EDR[LAE]), the MCM signals an interrupt to the on-chip interrupt controller. For more information on local access errors, see the description for EDR[LAE]. 0 Disables reporting of local access errors as interrupts. 1 Enables reporting of local access errors as interrupts.

7.4.1.6 Error Attributes Capture Register (EATR)

The error attributes capture register (EATR), shown in Figure 7-7, captures relevant information about the transaction that causes an error detected by the MCM.

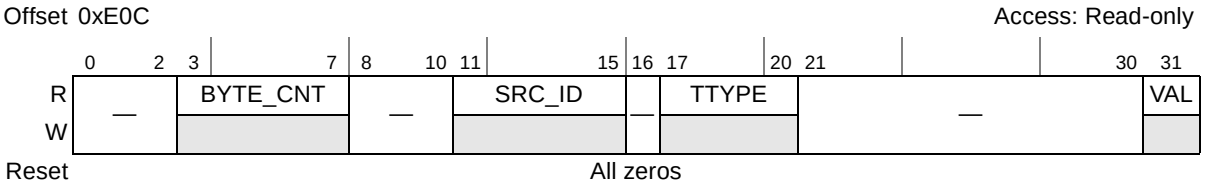


Figure 7-7. Error Attributes Capture Register (EATR)

Table 7-7 describes the EATR fields.

Table 7-7. EATR Field Descriptions

Bits	Name	Description
0–2	—	Reserved
3–7	BYTE_CNT	Byte count. Specifies the transaction byte count. All other values (not shown) are reserved. 00000 32 bytes 00001 1 byte 00010 2 bytes 00011 3 bytes 00100 4 bytes 00101 5 bytes 00110 6 bytes 00111 7 bytes 01000 8 bytes 10000 16 bytes
8–10	—	Reserved
11–15	SRC_ID	Source ID. Specifies the source device mastering the transaction. 00000 PCI Express interface 1 00001 PCI Express interface 2/RapidIO 00010–01001 Reserved 01010 Boot sequencer 01011–01111 Reserved 10000 Core 0 (instruction/data) 10001 Reserved 10010 Core 1 (instruction/data) 10011–10100 Reserved 10101 DMA 10110–10111 Reserved 11000 eTSEC1 11001 eTSEC2 11010 eTSEC3 11011 eTSEC4 11100 RapidIO message unit 11101 RapidIO doorbell unit 11110 RapidIO port-write unit 11111 Reserved
16	—	Reserved
17–20	TTYPE	Transaction type. Defined as follows: 0000 Write 0001 Reserved 0010 Write with allocate 0011 Write with allocate with lock 0100 Address only transaction (other than clean or flush) 0101–0111 Reserved 1000 Read 1001 Read with unlock 1010 Clean or flush 1011 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 EATR does not contain valid information. 1 EATR contains valid information.

7.4.1.7 Error Low Address Register (ELADR)

The error low address register (ELADR), shown in Figure 7-8, captures the lowest 32 bits of the 36-bit physical address of the transaction. The value in this register should be qualified with the EATR[VAL] bit.

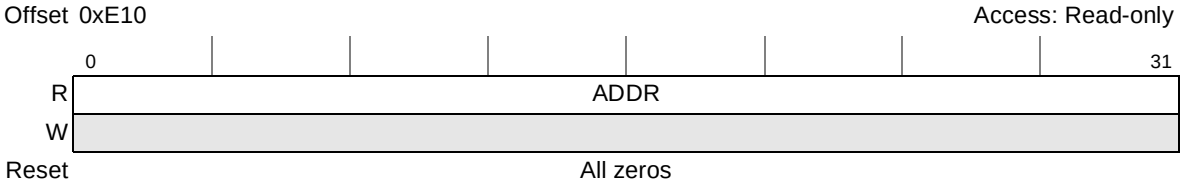


Figure 7-8. Error Low Address Register (ELADR)

Table 7-8 describes the ELADR fields.

Table 7-8. ELADR Field Descriptions

Bits	Name	Description
0–31	ADDR	Address. Specifies the lower-order 32 bits of the 36-bit address of the transaction. Qualified by EATR[VAL].

7.4.1.8 Error High Address Register (EHADR)

The error high address register (EHADR), shown in Figure 7-9, captures the highest 4 bits of the 36-bit physical address of the transaction. The value in this register should be qualified with EATR[VAL].

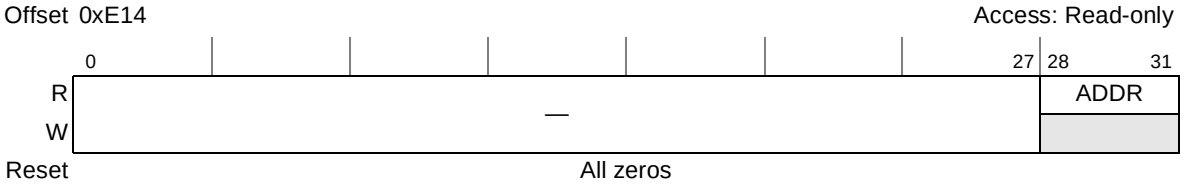


Figure 7-9. Error High Address Register (EHADR)

Table 7-9 describes EHADR fields.

Table 7-9. EHADR Field Descriptions

Bits	Name	Description
0–27	—	Reserved
28–31	ADDR	Address. Specifies the high-order 4 bits of the 36-bit address of the transaction. Qualified by EATR[VAL].

7.5 Functional Description

The following is a general description of MCM operation.

7.5.1 I/O Arbiter

Figure 7-1 shows the I/O arbiter block that manages I/O-initiated address tenure requests arriving on the request buses. The request buses compete for access to the MCM, which can only process one request at a time. The MCM uses two factors to select the winning request bus: the primary factor is requested

bandwidth and the secondary factor is longest waiting/least recently granted status. By default, all requesters start requesting low levels of bandwidth. A starvation avoidance algorithm ensures that low bandwidth requesters make forward progress in the presence of high bandwidth requesters.The transaction from the winning request bus competes with core requests for the MPX bus and entry into the transaction queue.

7.5.2 MPX Address Arbiter

Figure 7-1 shows the MPX address arbiter block which coordinates the entry of new transactions into the MCM's transaction queue. It handles arbitration for requests to use the MPX address bus from the cores and the winning request bus (for I/O-initiated snoopable transactions) and consequently controls when these new transactions can enter the transaction queue.

Because the MPX bus operates most efficiently when it streams transactions from one initiator, the MPX address arbiter alternates grants between streams of transactions from the cores and from the winner of the I/O arbiter. The arbiter uses the priority of the requests to limit streaming. If the priority of a new request is higher than that of a stream in progress, then the higher priority transaction interrupts the stream in progress. The priority of core transactions is set by the PORT*n*_PRI field in PCR.

7.5.3 Transaction Queue

The MCM'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 with the address range defined for each local access window, and the transaction is then routed to the appropriate target interface. Even though the MCM allows for pipelining of transactions (and treats address tenures independently from data tenures), the address tenures of all transactions issued from I/O masters (masters other than the cores) are dispatched to their target interfaces in the same order they are submitted. For those I/O masters requiring higher levels of performance, ordering is only enforced for a given master as is needed, so as not to degrade performance unnecessarily. For those transactions accessing address space marked as snoopable, or space that may be cached by one of the cores, the MCM enforces coherency, mirroring those transactions on the MPX bus for snooping purposes, and taking castouts or interventions from the cores as necessary.

7.5.4 Memory Controller Interleaving

This device supports two DDR memory controllers, and transactions, based on their address, can be directed to either DDR controller 1 or DDR controller 2 through the local access windows, as shown in the Memory Map chapter (add a real xref to the LAW section with the target IDs in the mem_map chapter). A third option for DDR memory is a local access window mapping defined for interleaving transactions between the two memory controllers.

In order to enable memory interleaving for the two DDR controllers, the following fields must be configured:

- A local access window must be set up to direct the desired local accesses to interleaved DDR memory.
- Two fields must be set in the DDR controller.

— CS0_CONFIG[INTLV_EN] must be set to 0b01.
— CS0_CONFIG[INTLV_CTL] must be programmed for one of cache line interleaving, page interleaving, bank interleaving, or super-bank interleaving.

See Section 8.4.1.2, “Chip Select Configuration (CSn_CONFIG),” for information on this register.

The address bit on which the interleaving occurs is based on the value of CS0_CONFIG[INTLV_CTL] and it is not used in the address decode at the memory controller interface.

7.5.5 Global Data Multiplexor

Figure 7-1 shows how the global data multiplexor takes data bus connections and multiplexes them onto one 128-bit wide global data bus. The global data multiplexor allows initiators of write transactions to route data to their targets and read targets to return data to the initiators.

7.5.5.1 Direct Data Bus

In order to further optimize the performance of CPU-to-DDR memory transactions, the MCM has a dedicated direct data bus from the DDR controllers to the MPX bus. Thus, core accesses to one of the DDR controllers can complete in fewer cycles, shortcutting around the global data multiplexing logic.

7.5.6 MPX Interface

Figure 7-1 shows the MPX interface for both MPX address and data tenures. This interface formats MPX address tenures for the MCM transaction queue. It also contains the queueing and buffering needed to manage outstanding MPX data tenures. The buffers receive core-initiated write and I/O-initiated read data from the core write (128-bit wide) and read (128-bit wide) data buses and route it through the global data multiplexor to the global data bus. The buffers also receive core-initiated read and I/O-initiated write data from the global data bus and forward them onto the MPX data bus (64 bits).

Note that core 1 has some unique low-memory address offset capabilities selectable at power-on reset to support separate operating systems running on each core simultaneously. This low-memory offset mode allows the cores to have some area of memory that is private physical address space for each core. This is described in detail in Section 2.1.1, “Low Memory Offset Mode,” and the Reset Chapter.

7.6 Initialization/Application Information

If core 0 is to be used to initialize the device, the corresponding CPU boot configuration power-on reset pin (cfg_cpu_boot) should be pulled high to initially set the PCR[PORT0_EN] bit. See Chapter 4, “Reset, Clocking, and Initialization,” for more information on power-up reset initialization.

If any device other than core 0, such as the boot sequencer or a PCI Express device, is used to initialize the device, the CPU boot configuration power-on reset pin should be pulled low to initially clear PCR[PORT0_EN]. This prevents core 0 from accessing any configuration registers or local memory space during initialization (core 1 is automatically disabled after reset). However, in any such system, one step near the end of the initialization routine must set PCR[PORT0_EN] and possibly PCR[PORT1_EN] to enable the cores.

PCR[PORT0_PRI] and PCR[PORT1_PRI] specify the priority level associated with all core-initiated transactions. These values allow users running time-critical applications to adjust the average response latency of transactions initiated by a core compared to those initiated by I/O masters. These priority levels affect whether the cores' requests can interrupt the streaming of address tenures initiated by the MCM on behalf of I/O masters. Only transactions with a priority greater than the current MPX transaction can interrupt streaming. The higher the core's priority, the lower the average latency needed for it to obtain bus grants from the MCM, because it can interrupt lower priority streaming. The default values of zero give all core-initiated transactions the lowest priority, which prevents the cores from interrupting I/O master transaction streams.

Chapter 8
DDR Memory Controllers

8.1 Introduction

The two fully programmable DDR SDRAM controllers support most JEDEC standard x8, x16, or x32 DDR2 and DDR memories available. In addition, unbuffered and registered DIMMs are supported. However, mixing different memory types or unbuffered and registered DIMMs in the same system is not supported. Built-in error checking and correction (ECC) ensures very low bit-error rates for reliable high-frequency operation. Dynamic power management and auto-precharge modes simplify memory system design. A large set of special features, including ECC error injection, support rapid system debug.

NOTE

In this chapter, the word ‘bank’ refers to a physical bank specified by a chip select; ‘logical bank’ refers to one of the four or eight sub-banks in each SDRAM chip. A sub-bank is specified by the 2 or 3 bits on the bank address (MBA) pins during a memory access.

Figure 8-1 is a high-level block diagram of the DDR memory controller with its associated interfaces. Section 8.5, “Functional Description,” contains detailed figures of the controller.

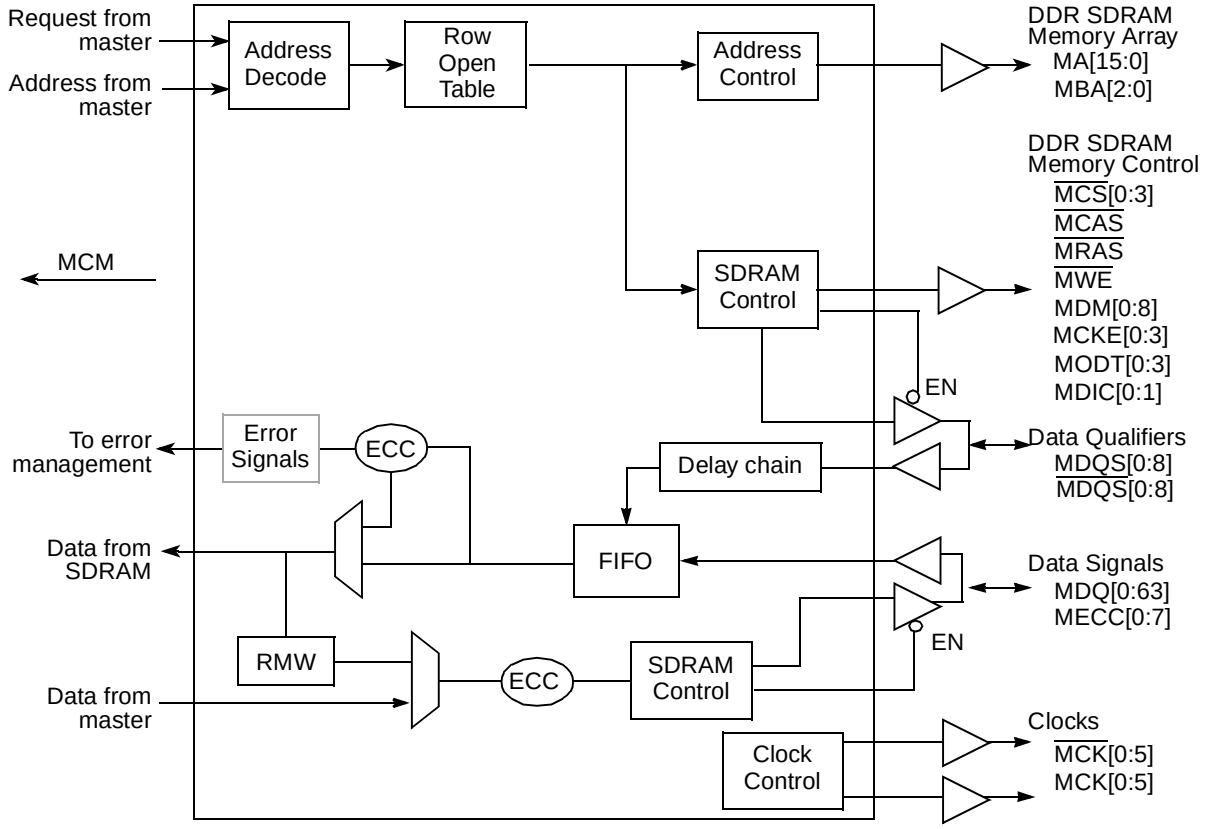


Figure 8-1. DDR Memory Controller Simplified Block Diagram

8.2 Features

The DDR memory controller includes these distinctive features:

- Support for DDR2 and DDR SDRAM
- Dual independent controllers
- 64-/72-bit SDRAM data bus. 32-/40-bit SDRAM for DDR and DDR2
- Programmable settings for meeting all SDRAM timing parameters
- The following SDRAM configurations are supported:
  - As many as four physical banks (chip selects), each bank independently addressable
  - 64-Mbit to 4-Gbit devices depending on internal device configuration with x8/x16/x32 data ports (no direct x4 support)
  - Unbuffered and registered DIMMs
- Chip select interleaving support
- Controller interleaving support
- Support for data mask signals and read-modify-write for sub-double-word writes. Note that a read-modify-write sequence is only necessary when ECC is enabled.
- Support for double-bit error detection and single-bit error correction ECC (8-bit check word across 64-bit data)

- Open page management (dedicated entry for each logical bank)
- Automatic DRAM initialization sequence or software-controlled initialization sequence
- Automatic DRAM data initialization
- Support for up to eight posted refreshes
- Memory controller clock frequency of two times the SDRAM clock with support for sleep power management
- Support for error injection

8.2.1 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].

8.3 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.

8.3.1 Signals Overview

Memory controller signals are grouped as follows:

- Memory interface signals
- Clock signals
- Debug signals

Table 8-1 shows how DDR memory controller external signals are grouped. The device hardware specification has a pinout diagram showing pin numbers. It also lists all electrical and mechanical specifications.

Table 8-1. DDR Memory Interface Signal Summary

Name	Function/Description	Reset	Pins	I/O
DDRC1 Memory Controller				
D1_MDQ[0:63]	DDRC1 Data bus	All zeros	64	I/O
D1_MDQS[0:8]	DDRC1 Data strobes	All zeros	9	I/O
$\overline{\text{D1\_MDQS}}$[0:8]	DDRC1 Complement data strobes	All ones	9	I/O

Table 8-1. DDR Memory Interface Signal Summary (continued)

Name	Function/Description	Reset	Pins	I/O
D1_MECC[0:7]	DDRC1 Error checking and correcting	All zeros	8	I/O
$\overline{\text{D1\_MCAS}}$	DDRC1 Column address strobe	One	1	O
D1_MA[15:0]	DDRC1 Address bus	All zeros	16	O
D1_MBA[2:0]	DDRC1 Logical bank address	All zeros	3	O
$\overline{\text{D1\_MCS}}$[0:3]	DDRC1 Chip selects	All ones	4	O
$\overline{\text{D1\_MWE}}$	DDRC1 Write enable	One	1	O
$\overline{\text{D1\_MRAS}}$	DDRC1 Row address strobe	One	1	O
D1_MDM[0:8]	DDRC1 Data mask	All zeros	9	O
D1_MCK[0:5]	DDRC1 DRAM clock outputs	All zeros	6	O
D1_$\overline{\text{MCK}}$[0:5]	DDRC1 DRAM clock outputs (complement)	All zeros	6	O
D1_MCKE[0:3]	DDRC1 DRAM clock enable	All zeros	4	O
D1_MODT[0:3]	DDRC1 DRAM on-die termination	All zeros	4	O
D1_MDVAL	DDRC1 Memory debug data valid	Zero	1	O
D1_MSRCID[0:4]	DDRC1 Memory debug source ID	All zeros	5	O
D1_MDIC[0:1]	DDRC1 Driver impedance calibration	b10	2	I/O
DDRC2 Memory Controller				
D2_MDQ[0:63]	DDRC2 Data bus	All zeros	64	I/O
D2_MDQS[0:8]	DDRC2 Data strobes	All zeros	9	I/O
$\overline{\text{D2\_MDQS}}$[0:8]	DDRC2 Complement data strobes	All ones	9	I/O
D2_MECC[0:7]	DDRC2 Error checking and correcting	All zeros	8	I/O
$\overline{\text{D2\_MCAS}}$	DDRC2 Column address strobe	One	1	O
D2_MA[15:0]	DDRC2 Address bus	All zeros	16	O
D2_MBA[2:0]	DDRC2 Logical bank address	All zeros	3	O
$\overline{\text{D2\_MCS}}$[0:3]	DDRC2 Chip select	All ones	4	O
$\overline{\text{D2\_MWE}}$	DDRC2 Write enable	One	1	O
$\overline{\text{D2\_MRAS}}$	DDRC2 Row address strobe	One	1	O
D2_MDM[0:8]	DDRC2 Data mask	All zeros	9	O
D2_MCK[0:5]	DDRC2 DRAM clock outputs	All zeros	6	O
$\overline{\text{D2\_MCK}}$[0:5]	DDRC2 DRAM clock outputs (complement)	All ones	6	O
D2_MCKE[0:3]	DDRC2 DRAM clock enable	All zeros	4	O

Table 8-1. DDR Memory Interface Signal Summary (continued)

Name	Function/Description	Reset	Pins	I/O
D2_MODT[0:3]	DDRC2 DRAM on-die termination	All zeros	4	O
D2_MDVAL	DDRC2 Memory debug data valid	Zero	1	O
D2_MSRCID[0:4]	DDRC2 Memory debug source ID	All zeros	5	O
D2_MDIC[0:1]	DDRC2 Driver impedance calibration	b10	2	I/O

Note that some devices implementing two DDR controllers may share one set of MDVAL and MSRCID[0:4] signals between them. Please refer to the signals and debug chapters for clarification on implementation.

Table 8-2 shows the memory address signal mappings.

Table 8-2. Memory Address Signal Mappings

Signal Name (Outputs)		JEDEC DDR DIMM Signals (Inputs)
msb	MA15	A15
	MA14	A14
	MA13	A13
	MA12	A12
	MA11	A11
	MA10	A10 (AP for DDR)[1]
	MA9	A9
	MA8	A8 (alternate AP for DDR)[2]
	MA7	A7
	MA6	A6
	MA5	A5
	MA4	A4
	MA3	A3
	MA2	A2
	MA1	A1
lsb	MA0	A0
msb	MBA2	MBA2
	MBA1	MBA1
lsb	MBA0	MBA0

1 Auto-precharge for DDR signaled on A10 when DDR_SDRAM_CFG[PCHB8] = 0

[2] Auto-precharge for DDR signaled on A8 when DDR_SDRAM_CFG[PCHB8] = 1

8.3.2 Detailed Signal Descriptions

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

Note that this device has two DDR memory controllers. The same set of signals exist on both DDR controllers. The signals are differentiated by the D*n*_ in front of the signal names. When a DDR signal is discussed specifically elsewhere in this book it is referred to without the D*n*_ prefix. For example, the D1_MDQ[0] and D2_MDQ[0] signals are both referred to as just MDQ[0].

8.3.2.1 Memory Interface Signals

Table 8-3 describes the DDR controller memory interface signals.

Table 8-3. Memory Interface Signals—Detailed Signal Descriptions

Signal	I/O	Description	
D*n*_MDQ[0:63]	I/O	Data bus. Both input and output signals on the DDR memory controller.	
	O	As outputs for the bidirectional data bus, these signals operate as described below.	
		State Meaning	Asserted/Negated—Represent the value of data being driven by the DDR memory controller.
		Timing	Assertion/Negation—Driven coincident with corresponding data strobes (MDQS) signal. High impedance—No READ or WRITE command is in progress; data is not being driven by the memory controller or the DRAM.
	I	As inputs for the bidirectional data bus, these signals operate as described below.	
		State Meaning	Asserted/Negated—Represents the state of data being driven by the external DDR SDRAMs.
		Timing	Assertion/Negation—The DDR SDRAM drives data during a READ transaction. High impedance—No READ or WRITE command in progress; data is not being driven by the memory controller or the DRAM.

Table 8-3. Memory Interface Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
D*n*_MDQS[0:8]/ D*n*_$\overline{\text{MDQS}}$[0:8]	I/O	Data strobes. Inputs with read data, outputs with write data. The data strobes may be single ended or differential.	
	O	As outputs, the data strobes are driven by the DDR memory controller during a write transaction. The memory controller always drives these signals low unless a read has been issued and incoming data strobes are expected. This keeps the data strobes from floating high when there are no transactions on the DRAM interface.	
		State Meaning	Asserted/Negated—Driven high when positive capture data is transmitted and driven low when negative capture data is transmitted. Centered in the data “eye” for writes; coincident with the data eye for reads. Treated as a clock. Data is valid when signals toggle. See Table 8-42 for byte lane assignments.
		Timing	Assertion/Negation—If a WRITE command is registered at clock edge *n*, data strobes at the DRAM assert centered in the data eye on clock edge *n* + 1. See the JEDEC DDR SDRAM specification for more information.
	I	As inputs, the data strobes are driven by the external DDR SDRAMs during a read transaction. The data strobes are used by the memory controller to synchronize data latching.	
		State Meaning	Asserted/Negated—Driven high when positive capture data is received and driven low when negative capture data is received. Centered in the data eye for writes; coincident with the data eye for reads. Treated as a clock. Data is valid when signals toggle. See Table 8-42 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.
D*n*_MECC[0:7]	I/O	Error checking and correcting codes. Input and output signals for the DDR controller’s bidirectional ECC bus. MECC[0:5] function in both normal and debug modes.	
	O	As normal mode outputs the ECC signals represent the state of ECC driven by the DDR controller on writes. As debug mode outputs MECC[0:5] provide source ID and data-valid information. See Section 8.5.11, “Error Checking and Correcting (ECC),” and Section 19.4.2.2, “Debug Information on ECC Pins,” for more details.	
		State Meaning	Asserted/Negated—Represents the state of ECC being driven by the DDR controller on writes.
		Timing	Assertion/Negation—Same timing as MDQ High impedance—Same timing as MDQ
	I	As inputs, the ECC signals represent the state of ECC driven by the SDRAM devices on reads.	
		State Meaning	Asserted/Negated—Represents the state of ECC being driven by the DDR SDRAMs on reads.
		Timing	Assertion/Negation—Same timing as MDQ High impedance—Same timing as MDQ

Table 8-3. Memory Interface Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
D*n*_MA[15:0]	O	Address bus. Memory controller outputs for the address to the DRAM. MA[15:0] carry 16 of the address bits for the DDR memory interface corresponding to the row and column address bits. MA0 is the lsb of the address output from the memory controller.	
		State Meaning	Asserted/Negated—Represents the address driven by the DDR memory controller. Contains different portions of the address depending on the memory size and the DRAM command being issued by the memory controller. See Table 8-46Table 8-47 for a complete description of the mapping of these signals.
		Timing	Assertion/Negation—The address is always driven when the memory controller is enabled. It is valid when a transaction is driven to DRAM (when $\overline{\text{MCS}}n$ is active). High impedance—When the memory controller is disabled
D*n*_MBA[2:0]	O	Logical bank address. Outputs that drive the logical (or internal) bank address pins of the SDRAM. Each SDRAM supports four or eight addressable logical sub-banks. Bit zero of the memory controller's output bank address must be connected to bit zero of the SDRAM's input bank address. MBA0, the least-significant bit of the three bank address signals, is asserted during the mode register set command to specify the extended mode register.	
		State Meaning	Asserted/Negated—Selects the DDR SDRAM logical (or internal) bank to be activated during the row address phase and selects the SDRAM internal bank for the read or write operation during the column address phase of the memory access. Table 8-46Table 8-47 describes the mapping of these signals in all cases.
		Timing	Assertion/Negation—Same timing as MA*n* High impedance—Same timing as MA*n*
D*n*_$\overline{\text{MCAS}}$	O	Column address strobe. Active-low SDRAM address multiplexing signal. $\overline{\text{MCAS}}$ is asserted for read or write transactions and for mode register set, refresh, and precharge commands.	
		State Meaning	Asserted—Indicates that a valid SDRAM column address is on the address bus for read and write transactions. See Table 8-51 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 8.4.1.4, "DDR SDRAM Timing Configuration 0 (TIMING_CFG_0)," Section 8.4.1.5, "DDR SDRAM Timing Configuration 1 (TIMING_CFG_1)," Section 8.4.1.6, "DDR SDRAM Timing Configuration 2 (TIMING_CFG_2)," and Section 8.4.1.3, "DDR SDRAM Timing Configuration 3 (TIMING_CFG_3)." High impedance—$\overline{\text{MCAS}}$ is always driven unless the memory controller is disabled.
$\overline{\text{D}n\text{\_MRAS}}$	O	Row address strobe. Active-low SDRAM address multiplexing signal. Asserted for activate commands. In addition; used for mode register set commands and refresh commands.	
		State Meaning	Asserted—Indicates that a valid SDRAM row address is on the address bus for read and write transactions. See Table 8-51 for more information on the states required on $\overline{\text{MRAS}}$ for various other SDRAM commands. Negated—The row address is not guaranteed to be valid.
		Timing	Assertion/Negation—Assertion and negation timing is directed by the values described in Section 8.4.1.4, "DDR SDRAM Timing Configuration 0 (TIMING_CFG_0)," Section 8.4.1.5, "DDR SDRAM Timing Configuration 1 (TIMING_CFG_1)," Section 8.4.1.6, "DDR SDRAM Timing Configuration 2 (TIMING_CFG_2)," and Section 8.4.1.3, "DDR SDRAM Timing Configuration 3 (TIMING_CFG_3)." High impedance—$\overline{\text{MRAS}}$ is always driven unless the memory controller is disabled.

Table 8-3. Memory Interface Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
D*n*_$\overline{\text{MCS}}$[0:3]	O	Chip selects. Four chip selects supported by the memory controller.	
		State Meaning	Asserted—Selects a physical SDRAM bank to perform a memory operation as described in Section 8.4.1.1, "Chip Select Memory Bounds (CSn_BNDS)," and Section 8.4.1.2, "Chip Select Configuration (CSn_CONFIG)." The DDR controller asserts one of the $\overline{\text{MCS}}$[0:3] signals to begin a memory cycle. Negated—Indicates no SDRAM action during the current cycle.
		Timing	Assertion/Negation—Asserted to signal any new transaction to the SDRAM. The transaction must adhere to the timing constraints set in TIMING_CFG_0–TIMING_CFG_3. High impedance—Always driven unless the memory controller is disabled.
D*n*_$\overline{\text{MWE}}$	O	Write enable. Asserted when a write transaction is issued to the SDRAM. This is also used for mode registers set commands and precharge commands.	
		State Meaning	Asserted—Indicates a memory write operation. See Table 8-51 for more information on the states required on $\overline{\text{MWE}}$ for various other SDRAM commands. Negated—Indicates a memory read operation.
		Timing	Assertion/Negation—Similar timing as $\overline{\text{MRAS}}$ and $\overline{\text{MCAS}}$. Used for write commands. High impedance—$\overline{\text{MWE}}$ is always driven unless the memory controller is disabled.
D*n*_MDM[0:8]	O	DDR SDRAM data output mask. Masks unwanted bytes of data transferred during a write. They are needed to support sub-burst-size transactions (such as single-byte writes) on SDRAM where all I/O occurs in multi-byte bursts. MDM0 corresponds to the most significant byte (MSB) and MDM7 corresponds to the LSB, while MDM8 corresponds to the ECC byte. Table 8-42 shows byte lane encodings.	
		State Meaning	Asserted—Prevents writing to DDR SDRAM. Asserted when data is written to DRAM if the corresponding byte(s) should be masked for the write. Note that the MDM*n* signals are active-high for the DDR controller. MDM*n* is part of the DDR command encoding. Negated—Allows the corresponding byte to be read from or written to the SDRAM.
		Timing	Assertion/Negation—Same timing as MDQx as outputs. High impedance—Always driven unless the memory controller is disabled.
D*n*_MODT[0:3]	O	On-Die termination. Memory controller outputs for the ODT to the DRAM. MODT[0:3] represents the on-die termination for the associated data, data masks, ECC, and data strobes.	
		State Meaning	Asserted/Negated—Represents the ODT driven by the DDR memory controller.
		Timing	Assertion/Negation—Driven in accordance with JEDEC DRAM specifications for on-die termination timings. It is configured through the CS*n*_CONFIG[ODT_RD_CFG] and CS*n*_CONFIG[ODT_WR_CFG] fields. High impedance—Always driven.
D*n*_MDIC[0:1]	I/O	Driver impedance calibration. Note that the MDIC signals require the use of 18.2-Ω precision 1% resistors; MDIC0 must be pulled to GND, while MDIC1 must be pulled to GV_{DD}. SeeSection 8.4.1.20, "DDR Control Driver Register 2 (DDRCDR_2)," for more information on these signals.	
		State Meaning	These pins are used for automatic calibration of the DDR IOs.
		Timing	These are driven for four DRAM cycles at a time while the DDR controller is executing the automatic driver compensation.

8.3.2.2 Clock Interface Signals

Table 8-4 contains the detailed descriptions of the clock signals of the DDR controller.

Table 8-4. Clock Signals—Detailed Signal Descriptions

Signal	I/O	Description	
MCK[0:5], $\overline{\text{MCK}}$[0:5]	O	DRAM clock outputs and their complements. See Section 8.5.4.1, "Clock Distribution."	
		State Meaning	Asserted/Negated—The JEDEC DDR SDRAM specifications require true and complement clocks. A clock edge is seen by the SDRAM when the true and complement cross.
		Timing	Assertion/Negation—Timing is controlled by the DDR_CLK_CNTL register at offset 0x130.
MCKE[0:3]	O	Clock enable. Output signals used as the clock enables to the SDRAM. MCKE[0:3] can be negated to stop clocking the DDR SDRAM. The MCKE signals should be connected to the same rank of memory as the corresponding $\overline{\text{MCS}}$ and MODT signals. For example, MCKE[0] should be connected to the same rank of memory as $\overline{\text{MCS}}$[0] and MODT[0].	
		State Meaning	Asserted—Clocking to the SDRAM is enabled. Negated—Clocking to the SDRAM is disabled and the SDRAM should ignore signal transitions on MCK or $\overline{\text{MCK}}$. MCK/$\overline{\text{MCK}}$ are don't cares while MCKE[0:3] are negated.
		Timing	Assertion/Negation—Asserted when DDR_SDRAM_CFG[MEM_EN] is set. Can be negated when entering dynamic power management or self refresh. Are asserted again when exiting dynamic power management or self refresh. High impedance—Always driven.

8.3.2.3 Debug Signals

The debug signals MSRCID[0:4] and MDVAL have no function in normal DDR controller operation. A detailed description of these signals can be found in Section 19.4.2, "DDR SDRAM Interface Debug."

8.4 Memory Map/Register Definition

Table 8-5 shows the register memory map for the DDR memory controllers. Note that while the table lists only the DDRC1 registers, the offsets are defined for both DDRC1 and DDRC2. Memory-mapped registers for DDRC1 begin at offset 0x0_2000 and the memory-mapped registers for DDRC2 begin at offset 0x0_6000.

Table 8-5. DDR Memory Controller Memory Map

Offset	Register	Access	Reset	Section/Page
DDRC1 Registers				
0x000	CS0_BNDS—Chip select 0 memory bounds	R/W	0x0000_0000	8.4.1.1/8-12
0x008	CS1_BNDS—Chip select 1 memory bounds	R/W	0x0000_0000	8.4.1.1/8-12
0x010	CS2_BNDS—Chip select 2 memory bounds	R/W	0x0000_0000	8.4.1.1/8-12
0x018	CS3_BNDS—Chip select 3 memory bounds	R/W	0x0000_0000	8.4.1.1/8-12
0x080	CS0_CONFIG—Chip select 0 configuration	R/W	0x0000_0000	8.4.1.2/8-13
0x084	CS1_CONFIG—Chip select 1 configuration	R/W	0x0000_0000	8.4.1.2/8-13
0x088	CS2_CONFIG—Chip select 2 configuration	R/W	0x0000_0000	8.4.1.2/8-13
0x08C	CS3_CONFIG—Chip select 3 configuration	R/W	0x0000_0000	8.4.1.2/8-13

Table 8-5. DDR Memory Controller Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x100	TIMING_CFG_3—DDR SDRAM timing configuration 3	R/W	0x0000_0000	8.4.1.3/8-15
0x104	TIMING_CFG_0—DDR SDRAM timing configuration 0	R/W	0x0011_0105	8.4.1.4/8-16
0x108	TIMING_CFG_1—DDR SDRAM timing configuration 1	R/W	0x0000_0000	8.4.1.5/8-18
0x10C	TIMING_CFG_2—DDR SDRAM timing configuration 2	R/W	0x0000_0000	8.4.1.6/8-20
0x110	DDR_SDRAM_CFG—DDR SDRAM control configuration	R/W	0x0200_0000	8.4.1.7/8-22
0x114	DDR_SDRAM_CFG_2—DDR SDRAM control configuration 2	R/W	0x0000_0000	8.4.1.8/8-24
0x118	DDR_SDRAM_MODE—DDR SDRAM mode configuration	R/W	0x0000_0000	8.4.1.9/8-26
0x11C	DDR_SDRAM_MODE_2—DDR SDRAM mode configuration 2	R/W	0x0000_0000	8.4.1.10/8-27
0x120	DDR_SDRAM_MD_CNTL—DDR SDRAM mode control	R/W	0x0000_0000	8.4.1.11/8-27
0x124	DDR_SDRAM_INTERVAL—DDR SDRAM interval configuration	R/W	0x0000_0000	8.4.1.12/8-30
0x128	DDR_DATA_INIT—DDR SDRAM data initialization	R/W	0x0000_0000	8.4.1.13/8-30
0x130	DDR_SDRAM_CLK_CNTL—DDR SDRAM clock control	R/W	0x0200_0000	8.4.1.14/8-31
0x140–0x144	Reserved	—	—	—
0x148	DDR_INIT_ADDR—DDR training initialization address	R/W	0x0000_0000	8.4.1.15/8-31
0x14C	DDR_INIT_EXT_ADDRESS—DDR training initialization extended address	R/W	0x0000_0000	8.4.1.16/8-32
0x150–0xB1F	Reserved	—	—	—
0xB20	DDRDSR_1—DDR Debug Status Register 1	R	0x0000_0000	8.4.1.17/8-33
0xB24	DDRDSR_2—DDR Debug Status Register 2	R	0x0000_0000	8.4.1.18/8-33
0xB28	DDRCDR_1—DDR Control Driver Register 1	R/W	0x0000_0000	8.4.1.19/8-34
0xB2C	DDRCDR_2—DDR Control Driver Register 2	R/W	0x0000_0000	8.4.1.20/8-35
0xB30–0xBF7	Reserved	—	—	—
0xBF8	DDR_IP_REV1—DDR IP block revision 1	R	0x*nnnn_nnnn*[1]	8.4.1.21/8-36
0xBFC	DDR_IP_REV2—DDR IP block revision 2	R	0x00*nn*_00*nn*[1]	8.4.1.22/8-37
0xE00	DATA_ERR_INJECT_HI—Memory data path error injection mask high	R/W	0x0000_0000	8.4.1.23/8-37
0xE04	DATA_ERR_INJECT_LO—Memory data path error injection mask low	R/W	0x0000_0000	8.4.1.24/8-38
0xE08	ECC_ERR_INJECT—Memory data path error injection mask ECC	R/W	0x0000_0000	8.4.1.25/8-38
0xE20	CAPTURE_DATA_HI—Memory data path read capture high	R/W	0x0000_0000	8.4.1.26/8-39
0xE24	CAPTURE_DATA_LO—Memory data path read capture low	R/W	0x0000_0000	8.4.1.27/8-39
0xE28	CAPTURE_ECC—Memory data path read capture ECC	R/W	0x0000_0000	8.4.1.28/8-40
0xE40	ERR_DETECT—Memory error detect	w1c	0x0000_0000	8.4.1.29/8-40
0xE44	ERR_DISABLE—Memory error disable	R/W	0x0000_0000	8.4.1.30/8-41
0xE48	ERR_INT_EN—Memory error interrupt enable	R/W	0x0000_0000	8.4.1.31/8-42
0xE4C	CAPTURE_ATTRIBUTES—Memory error attributes capture	R/W	0x0000_0000	8.4.1.32/8-43
0xE50	CAPTURE_ADDRESS—Memory error address capture	R/W	0x0000_0000	8.4.1.33/8-44
0xE54	CAPTURE_EXT_ADDRESS—Memory error extended address capture	R/W	0x0000_0000	8.4.1.34/8-44

Table 8-5. DDR Memory Controller Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0xE58	ERR_SBE—Single-Bit ECC memory error management	R/W	0x0000_0000	8.4.1.35/8-45
DDRC2 Registers				
Block Base Address: 0x0_6000				
DDRC2 registers **Note:** All registers defined for DDRC1 are also defined for DDRC2; the base address of DDRC2 registers is 0x0_6*nnn*.				

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

8.4.1 Register Descriptions

This section describes the DDR memory controller registers. Shading indicates reserved fields that should not be written.

8.4.1.1 Chip Select Memory Bounds (CS*n*_BNDS)

The chip select bounds registers (CS*n*_BNDS) define the starting and ending address of the memory space that corresponds to the individual chip selects. Note that the size specified in CS*n*_BNDS should equal the size of physical DRAM. Also, note that EA*n* must be greater than or equal to SA*n*.

If chip select interleaving is enabled, all fields in the lower interleaved chip select are used, and the other chip selects' bounds registers are unused. For example, if chip selects 0 and 1 are interleaved, all fields in CS0_BNDS are used, and all fields in CS1_BNDS are unused.

CS*n*_BNDS are shown in Figure 8-2.

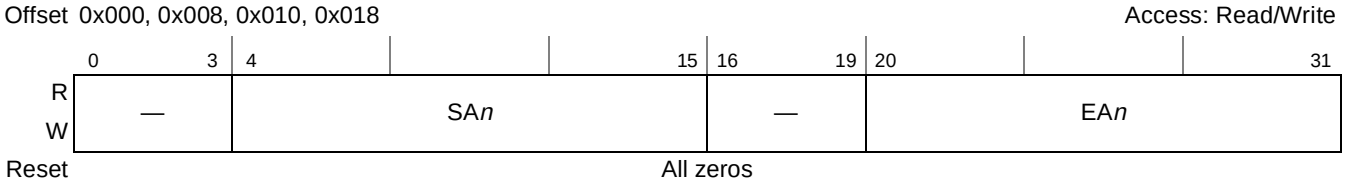


Figure 8-2. Chip Select Bounds Registers (CS*n*_BNDS)

Table 8-6 describes the CS*n*_BNDS register fields.

Table 8-6. CS*n*_BNDS Field Descriptions

Bits	Name	Description
0–3	—	Reserved
4–15	SA*n*	Starting address for chip select (bank) *n*. This value is compared against the 12 msbs of the 36-bit address.
16–19	—	Reserved
20–31	EA*n*	Ending address for chip select (bank) *n*. This value is compared against the 12 msbs of the 36-bit address.

8.4.1.2 Chip Select Configuration (CS*n*_CONFIG)

The chip select configuration (CS*n*_CONFIG) registers shown in Figure 8-3 enable the DDR chip selects and set the number of row and column bits used for each chip select. These registers should be loaded with the correct number of row and column bits for each SDRAM. Because CS*n*_CONFIG[ROW_BITS_CS_*n*, COL_BITS_CS_*n*] establish address multiplexing, the user should take great care to set these values correctly.

If chip select interleaving is enabled, then all fields in the lower interleaved chip select are used, and the other registers' fields are unused, with the exception of the ODT_RD_CFG and ODT_WR_CFG fields. For example, if chip selects 0 and 1 are interleaved, all fields in CS0_CONFIG are used, but only the ODT_RD_CFG and ODT_WR_CFG fields in CS1_CONFIG are used.

Offset 0x080, 0x084, 0x088, 0x08C — Access: Read/Write

	0	1	2	3	4 – 7	8	9 – 11	12	13 – 15
R W	CS_*n*_EN	—	INTLV_EN	—	INTLV_CTL	AP_*n*_EN	ODT_RD_CFG	—	ODT_WR_CFG
Reset	All zeros								

	16 – 17	18 – 20	21 – 23	24 – 28	29 – 31
R W	BA_BITS_CS_*n*	—	ROW_BITS_CS_*n*	—	COL_BITS_CS_*n*
Reset	All zeros				

Figure 8-3. Chip Select Configuration Register (CS*n*_CONFIG)

Table 8-7 describes the CS*n*_CONFIG register fields.

Table 8-7. CS*n*_CONFIG Field Descriptions

Bits	Name	Description
0	CS_*n*_EN	Chip select *n* enable 0 Chip select *n* is not active 1 Chip select *n* is active and assumes the state set in CS*n*_BNDS.
1	—	Reserved
2	INTLV_EN	Memory controller interleave enable Note: This field is only available in CS0_CONFIG. If memory controller interleaving is enabled, then the data bus widths must be programmed identically for the 2 memory controllers. 0 No interleaving 1 Interleaving between 2 memory controllers If this bit is set, INTLV_CTL must be set to select the bit(s) used to control interleaving.For more information on memory controller interleaving, see Section 7.5.4, "Memory Controller Interleaving."
3	—	Reserved

Table 8-7. CS*n*_CONFIG Field Descriptions (continued)

Bits	Name	Description
4–7	INTLV_CTL	Interleaving control—Controls the bit(s) used to select the memory controller when memory controller interleaving is enabled (INTLV_EN is set) Note: This field is only available in CS0_CONFIG. 0000 Cache line interleaving. In this mode, bit 30 of the 36-bit physical address is used to select the memory controller. 0001 Page interleaving. In this mode, the bit used to select the controller is the bit to the left of the highest-order column bits. 0010 Bank interleaving. In this mode, the bit used to select the controller is the bit to the left of the bank select bits, which are to the left of the highest-order column bits. 0011 Super-bank interleaving. Super-bank interleaving should only be used if chip select interleaving is enabled. In this mode, the decoded address bit to the left of the chip select interleaved bit(s) is used for the memory controller interleaving. 0100–1111Reserved For more information on memory controller interleaving, see Section 7.5.4, “Memory Controller Interleaving.”
8	AP_*n*_EN	Chip select *n* auto-precharge enable 0 Chip select *n* is only auto-precharged if global auto-precharge mode is enabled (DDR_SDRAM_INTERVAL[BSTOPRE] = 0). 1 Chip select *n* always issues an auto-precharge for read and write transactions.
9–11	ODT_RD_CFG	ODT for reads configuration. Note that CAS latency plus additive latency must be at least 3 cycles for ODT_RD_CFG to be enabled. ODT should only be used with DDR2 memories. 000 Never assert ODT for reads 001 Assert ODT only during reads to CS*n* 010 Assert ODT only during reads to other chip selects 011 Assert ODT only during reads to other DIMM modules. It is assumed that CS0 and CS1 are on the same DIMM module, whereas CS2 and CS3 are on a separate DIMM module. 100 Assert ODT for all reads 101–111Reserved
12	—	Reserved
13–15	ODT_WR_CFG	ODT for writes configuration. Note that write latency plus additive latency must be at least 3 cycles for ODT _WR_CFG to be enabled. ODT should only be used with DDR2 memories. 000 Never assert ODT for writes 001 Assert ODT only during writes to CS*n* 010 Assert ODT only during writes to other chip selects 011 Assert ODT only during writes to other DIMM modules. It is assumed that CS0 and CS1 are on the same DIMM module, whereas CS2 and CS3 are on a separate DIMM module. 100 Assert ODT for all writes 101–111Reserved
16–17	BA_BITS_CS_*n*	Number of bank bits for SDRAM on chip select *n*. These bits correspond to the sub-bank bits driven on MBA*n* in Table 8-47Table 8-46and Table 8-47. 00 2 logical bank bits 01 3 logical bank bits 10–11Reserved
18–20	—	Reserved

Table 8-7. CS*n*_CONFIG Field Descriptions (continued)

Bits	Name	Description
21–23	ROW_BITS_CS_*n*	Number of row bits for SDRAM on chip select *n*. See Table 8-47Table 8-46and Table 8-47 for details. 000 12 row bits 001 13 row bits 010 14 row bits 011 15 row bits 100 16 row bits 101–111 Reserved
24–28	—	Reserved
29–31	COL_BITS_CS_*n*	Number of column bits for SDRAM on chip select *n*. For DDR, the decoding is as follows: 000 8 column bits 001 9 column bits 010 10 column bits 011 11 column bits 100–111 Reserved

8.4.1.3 DDR SDRAM Timing Configuration 3 (TIMING_CFG_3)

DDR SDRAM timing configuration register 3, shown in Figure 8-4, sets the extended refresh recovery time, which is combined with TIMING_CFG_1[REFREC] to determine the full refresh recovery time.

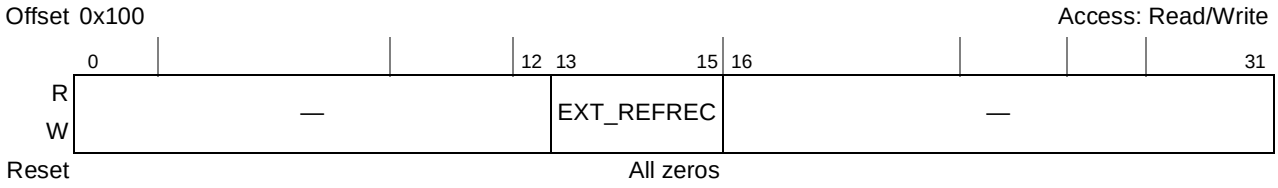


Figure 8-4. DDR SDRAM Timing Configuration 3 (TIMING_CFG_3)

Table 8-8 describes TIMING_CFG_3 fields.

Table 8-8. TIMING_CFG_3 Field Descriptions

Bits	Name	Description
0–12	—	Reserved, should be cleared.
13–15	EXT_REFREC	Extended refresh recovery time (t_{RFC}). Controls the number of clock cycles from a refresh command until an activate command is allowed. This field is concatenated with TIMING_CFG_1[REFREC] to obtain a 7bit value for the total refresh recovery. Note that hardware adds an additional 8 clock cycles to the final, 7bit value of the refresh recovery. t_{RFC} = {EXT_REFREC || REFREC} + 8, such that t_{RFC} is calculated as follows: 000 0 clocks 001 16 clocks 010 32 clocks 011 48 clocks 100 64 clocks 101 80 clocks 110 96 clocks 111 112 clocks
16–31	—	Reserved, should be cleared.

8.4.1.4 DDR SDRAM Timing Configuration 0 (TIMING_CFG_0)

DDR SDRAM timing configuration register 0, shown in Figure 8-5, sets the number of clock cycles between various SDRAM control commands.

Offset 0x104 Access: Read/Write

	0 1	2 3	4 5	6 7	8	9 11	12	13 15	16 19	20 23	24 27	28 31
R W	RWT	WRT	RRT	WWT	—	ACT_PD_EXIT	—	PRE_PD_EXIT	—	ODT_PD_EXIT	—	MRS_CYC
Reset	0 0	0 0	0 0	0 0	0	0 0 1	0	0 0 1	0 0 0 0	0 0 0 1	0 0 0 0	0 1 0 1

Figure 8-5. DDR SDRAM Timing Configuration 0 (TIMING_CFG_0)

Table 8-9 describes TIMING_CFG_0 fields.

Table 8-9. TIMING_CFG_0 Field Descriptions

Bits	Name	Description
0–1	RWT	Read-to-write turnaround (t_{RTW}). Specifies how many extra cycles are added between a read to write turnaround. If 0 clocks is chosen, then the DDR controller uses a fixed number based on the CAS latency and write latency. Choosing a value other than 0 adds extra cycles past this default calculation. As a default the DDR controller determines the read-to-write turnaround as CL – WL + BL/2 + 2. In this equation, CL is the CAS latency rounded up to the next integer, WL is the programmed write latency, and BL is the burst length. 00 0 clocks 10 2 clocks 01 1 clock 11 3 clocks
2–3	WRT	Write-to-read turnaround. Specifies how many extra cycles are added between a write to read turnaround. If 0 clocks is chosen, then the DDR controller uses a fixed number based on the, read latency, and write latency. Choosing a value other than 0 adds extra cycles past this default calculation. As a default, the DDR controller determines the write-to-read turnaround as WL – CL + BL/2 + 1. In this equation, CL is the CAS latency rounded down to the next integer, WL is the programmed write latency, and BL is the burst length. 00 0 clocks 10 2 clocks 01 1 clock 11 3 clocks
4–5	RRT	Read-to-read turnaround. Specifies how many extra cycles are added between reads to different chip selects. As a default, 3 cycles are required between read commands to different chip selects. Extra cycles may be added with this field. Note: If 8-beat bursts are enabled, then 5 cycles are the default. Note that DDR2 does not support 8-beat bursts. 00 0 clocks 10 2 clocks 01 1 clock 11 3 clocks
6–7	WWT	Write-to-write turnaround. Specifies how many extra cycles are added between writes to different chip selects. As a default, 2 cycles are required between write commands to different chip selects. Extra cycles may be added with this field. Note: If 8-beat bursts are enabled, then 4 cycles are the default. Note that DDR2 does not support 8-beat bursts. 00 0 clocks 10 2 clocks 01 1 clock 11 3 clocks
8	—	Reserved, should be cleared.

Table 8-9. TIMING_CFG_0 Field Descriptions (continued)

Bits	Name	Description
9–11	ACT_PD_EXIT	Active powerdown exit timing (t_{XARD} and t_{XARDS}). Specifies how many clock cycles to wait after exiting active powerdown before issuing any command. 000 Reserved 001 1 clock 010 2 clocks 011 3 clocks 100 4 clocks 101 5 clocks 110 6 clocks 111 7 clocks
12	—	Reserved, should be cleared.
13–15	PRE_PD_EXIT	Precharge powerdown exit timing (t_{XP}). Specifies how many clock cycles to wait after exiting precharge powerdown before issuing any command. 000 Reserved 001 1 clock 010 2 clocks 011 3 clocks 100 4 clocks 101 5 clocks 110 6 clocks 111 7 clocks
16–19	—	Reserved, should be cleared.
20–23	ODT_PD_EXIT	ODT powerdown exit timing (t_{AXPD}). Specifies how many clocks must pass after exiting powerdown before ODT may be asserted. 0000 0 clock 0001 1 clock 0010 2 clocks 0011 3 clocks 0100 4 clocks 0101 5 clocks 0110 6 clocks 0111 7 clocks 1000 8 clocks 1001 9 clocks 1010 10 clocks 1011 11 clocks 1100 12 clocks 1101 13 clocks 1110 14 clocks 1111 15 clocks
24–27	—	Reserved, should be cleared.
28–31	MRS_CYC	Mode register set cycle time (t_{MRD}). Specifies the number of cycles that must pass after a Mode Register Set command until any other command. 0000 Reserved 0001 1 clock 0010 2 clocks 0011 3 clocks 0100 4 clocks 0101 5 clocks 0110 6 clocks 0111 7 clocks 1000 8 clocks 1001 9 clocks 1010 10 clocks 1011 11 clocks 1100 12 clocks 1101 13 clocks 1110 14 clocks 1111 15 clocks

8.4.1.5 DDR SDRAM Timing Configuration 1 (TIMING_CFG_1)

DDR SDRAM timing configuration register 1, shown in Figure 8-6, sets the number of clock cycles between various SDRAM control commands.

Offset 0x108 Access: Read/Write

	0	1–3	4–7	8	9–11	12–15	16–19	20	21–23	24	25–27	28	29–31
R W	—	PRETOACT	ACTTOPRE	—	ACTTORW	CASLAT	REFREC	—	WRREC	—	ACTTOACT	—	WRTORD
Reset	All zeros												

Figure 8-6. DDR SDRAM Timing Configuration 1 (TIMING_CFG_1)

Table 8-10 describes TIMING_CFG_1 fields.

Table 8-10. 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 001 1 clock 010 2 clocks 011 3 clocks 100 4 clocks 101 5 clocks 110 6 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 16 clocks 0001 17 clocks 0010 18 clocks 0011 19 clocks 0100 4 clocks 0101 5 clocks 0110 6 clocks 0111 7 clocks ... 1111 15 clocks
8	—	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 001 1 clock 010 2 clocks 011 3 clocks 100 4 clocks 101 5 clocks 110 6 clocks 111 7 clocks

Table 8-10. TIMING_CFG_1 Field Descriptions (continued)

Bits	Name	Description
12–15	CASLAT	$\overline{\text{MCAS}}$ latency from READ command. Number of clock cycles between registration of a READ command by the SDRAM and the availability of the first output data. If a READ command is registered at clock edge n and the latency is m clocks, data is available nominally coincident with clock edge $n + m$. This value must be programmed at initialization as described in Section 8.4.1.8, "DDR SDRAM Control Configuration 2 (DDR_SDRAM_CFG_2).") 0000 Reserved 0001 1 clock 0010 1.5 clocks 0011 2 clocks 0100 2.5 clocks 0101 3 clocks 0110 3.5 clocks 0111 4 clocks 1000 4.5 clocks 1001 5 clocks 1010 5.5 clocks 1011 6 clocks 1100 6.5 clocks 1101 7 clocks 1110 7.5 clocks 1111 8 clocks
16–19	REFREC	Refresh recovery time (t_{RFC}). Controls the number of clock cycles from a refresh command until an activate command is allowed. This field is concatenated with TIMING_CFG_3[EXTREFREC] to obtain a 7-bit value for the total refresh recovery. Note that hardware adds an additional 8 clock cycles to the final, 7-bit value of the refresh recovery, such that t_{RFC} is calculated as follows: t_{RFC} = {EXT_REFREC || REFREC} + 8. 0000 8 clocks 0001 9 clocks 0010 10 clocks 0011 11 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 Reserved 001 1 clock 010 2 clocks 011 3 clocks 100 4 clocks 101 5 clocks 110 6 clocks 111 7 clocks
24	—	Reserved, should be cleared.
25–27	ACTTOACT	Activate-to-activate interval (t_{RRD}). Number of clock cycles from an activate command until another activate command is allowed for a different logical bank in the same physical bank (chip select). 000 Reserved 001 1 clock 010 2 clocks 011 3 clocks 100 4 clocks 101 5 clocks 110 6 clocks 111 7 clocks
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

8.4.1.6 DDR SDRAM Timing Configuration 2 (TIMING_CFG_2)

DDR SDRAM timing configuration 2, shown in Figure 8-7, sets the clock delay to data for writes.

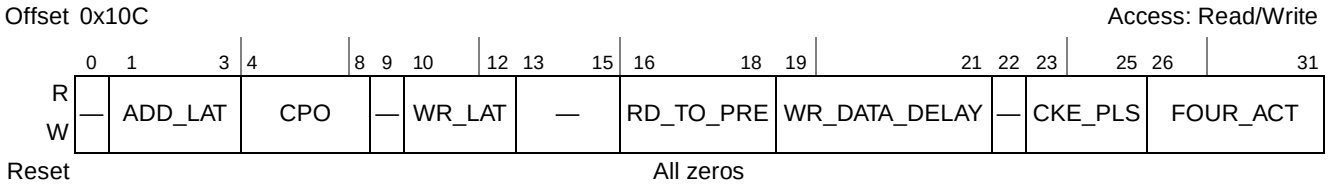


Figure 8-7. DDR SDRAM Timing Configuration 2 Register (TIMING_CFG_2)

Table 8-11 describes the TIMING_CFG_2 fields.

Table 8-11. TIMING_CFG_2 Field Descriptions

Bits	Name	Description
0	—	Reserved
1–3	ADD_LAT	Additive latency. The additive latency must be set to a value less than TIMING_CFG_1[ACTTORW]. (DDR2-specific) 000 0 clocks 001 1 clock 010 2 clocks 011 3 clocks 100 4 clocks 101 5 clocks 110 Reserved 111 Reserved
4–8	CPO[1]	$\overline{\text{MCAS}}$-to-preamble override. Defines the number of DRAM cycles between when a read is issued and when the corresponding DQS preamble is valid for the memory controller. For these decodings, “READ_LAT” is equal to the CAS latency plus the additive latency. 00000READ_LAT + 1 00001Reserved 00010READ_LAT 00011READ_LAT + 1/4 00100READ_LAT + 1/2 00101READ_LAT + 3/4 00110READ_LAT + 1 00111READ_LAT + 5/4 01000READ_LAT + 3/2 01001READ_LAT + 7/4 01010READ_LAT + 2 01011READ_LAT + 9/4 01100READ_LAT + 5/2 01101READ_LAT + 11/4 01110READ_LAT + 3 01111READ_LAT + 13/4 10000READ_LAT + 7/2 10001READ_LAT + 15/4 10010READ_LAT + 4 10011READ_LAT + 17/4 10100READ_LAT + 9/2 10101READ_LAT + 19/4 10110–11111 Reserved
9	—	Reserved
10–12	WR_LAT	Write latency. Note that the total write latency for DDR2 is equal to WR_LAT + ADD_LAT; the write latency for DDR1 is 1. 000 Reserved 001 1 clock 010 2 clocks 011 3 clocks 100 4 clocks 101 5 clocks 110 6 clocks 111 7 clocks

Table 8-11. TIMING_CFG_2 Field Descriptions (continued)

Bits	Name	Description
13–15	—	Reserved
16–18	RD_TO_PRE	Read to precharge (t_{RTP}). For DDR2, with a non-zero ADD_LAT value, takes a minimum of ADD_LAT + t_{RTP} cycles between read and precharge. For DDR1 with burst length of 4, must be set to 010; for DDR1 with burst length of 8, must be set to 100. 000 Reserved 001 1 cycle 010 2 cycles 011 3 cycles 100 4 cycles 101–111 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. See Section 8.5.7, “DDR SDRAM Write Timing Adjustments,” for details. 000 0 clock delay 001 1/4 clock delay 010 1/2 clock delay 011 3/4 clock delay 100 1 clock delay 101 5/4 clock delay 110 3/2 clock delay 111 Reserved
22	—	Reserved
23–25	CKE_PLS	Minimum CKE pulse width (t_{CKE})Can be set to 001 for DDR1. 000 Reserved 001 1 cycle 010 2 cycles 011 3 cycles 100 4 cycles 101–111 Reserved
26–31	FOUR_ACT	Window for four activates (t_{FAW}). This is applied to DDR2 with eight logical banks only. Must be set to 000001 for DDR1. 000000Reserved 0000011 cycle 0000102 cycles 0000113 cycles 0001004 cycles ... 01001119 cycles 01010020 cycles 010101–111111 Reserved

1 For CPO decodings other than 00000 and 11111, ‘READ_LAT’ is rounded up to the next integer value.

8.4.1.7 DDR SDRAM Control Configuration (DDR_SDRAM_CFG)

The DDR SDRAM control configuration register, shown in Figure 8-8, enables the interface logic and specifies certain operating features such as self refreshing, error checking and correcting, registered DIMMs, and dynamic power management.

Offset 0x110 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MEM_EN	SREN	ECC_EN	RD_EN	—	SDRAM_TYPE			—		DYN_PWR	—	32_BE	8_BE	NCAP	—
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	2T_EN	BA_INTLV_CTL							—		x32_EN	PCHB8	HSE	—	MEM_HALT	BI
Reset	All zeros															

Figure 8-8. DDR SDRAM Control Configuration Register (DDR_SDRAM_CFG)

Table 8-12 describes the DDR_SDRAM_CFG fields.

Table 8-12. DDR_SDRAM_CFG Field Descriptions

Bits	Name	Description
0	MEM_EN	DDR SDRAM interface logic enable. 0 SDRAM interface logic is disabled. 1 SDRAM interface logic is enabled. Must not be set until all other memory configuration parameters have been appropriately configured by initialization code.
1	SREN	Self refresh enable (during sleep). 0 SDRAM self refresh is disabled during sleep. Whenever self-refresh is disabled, the system is responsible for preserving the integrity of SDRAM during sleep. 1 SDRAM self refresh is enabled during sleep.
2	ECC_EN	ECC enable. Note that uncorrectable read errors may cause an interrupt. 0 No ECC errors are reported. No ECC interrupts are generated. 1 ECC is enabled.
3	RD_EN	Registered DIMM enable. Specifies the type of DIMM used in the system. 0 Indicates unbuffered DIMMs. 1 Indicates registered DIMMs. Note that RD_EN and 2T_EN must not both be set at the same time.
4	—	Reserved
5–7	SDRAM_TYPE	Type of SDRAM device to be used. This field is used when issuing the automatic hardware initialization sequence to DRAM through Mode Register Set and Extended Mode Register Set commands. Default value is 010 designating DDR1 SDRAM. 000–001Reserved 010 DDR1 SDRAM 011 DDR2 SDRAM 100 Reserved 101 Reserved 110 Reserved 111 Reserved

Table 8-12. DDR_SDRAM_CFG Field Descriptions (continued)

Bits	Name	Description
8–9	—	Reserved
10	DYN_PWR	Dynamic power management mode 0 Dynamic power management mode is disabled. 1 Dynamic power management mode is enabled. If there is no ongoing memory activity, the SDRAM CKE signal is negated.
11	—	Reserved
12	32_BE	32-bit bus enable. 0 64-bit bus is used. 1 32-bit bus is used.If the DDR controller is configured to operate in 32-bit mode (32_BE = 1), then DBCR[CFG_DDR32] in the MCM configuration registers must be set to properly configure the MCM prior to enabling the DDR controller.
13	8_BE	8-beat burst enable. 0 4-beat bursts are used on the DRAM interface. 1 8-beat bursts are used on the DRAM interface. **Note:** DDR1 (SDRAM_TYPE = 010) must use 8-beat bursts when using 32-bit bus mode (32_BE = 1) and 4-beat bursts when using 64-bit bus mode; DDR2 (SDRAM_TYPE = 011) must use 4-beat bursts, even when using 32-bit bus mode
14	NCAP	Non-concurrent auto-precharge. Some older DDR DRAMs do not support concurrent auto precharge. If one of these devices is used, then this bit needs to be set if auto precharge is used. 0 DRAMs in system support concurrent auto-precharge. 1 DRAMs in system do not support concurrent auto-precharge.
15	—	Reserved
16	2T_EN	Enable 2T timing. 0 1T timing is enabled. The DRAM command/address are held for only 1 cycle on the DRAM bus. 1 2T timing is enabled. The DRAM command/address are held for 2 full cycles on the DRAM bus for every DRAM transaction. However, the chip select is only held for the second cycle. Note that RD_EN and 2T_EN must not both be set at the same time.
17–23	BA_INTLV_CTL	Bank (chip select) interleaving control. Set this field only if you wish to use bank interleaving. ('*x*' denotes a don't care bit value. All unlisted field values are reserved.) 0000000No external memory banks are interleaved 1000000External memory banks 0 and 1 are interleaved 0100000External memory banks 2 and 3 are interleaved 1100000External memory banks 0 and 1 are interleaved together **and** banks 2 and 3 are interleaved together *xx*00100External memory banks 0 through 3 are all interleaved together
24–25	—	Reserved
26	x32_EN	x32 enable. 0 Either x8 or x16 discrete DRAM chips are used. In this mode, each data byte has a dedicated corresponding data strobe. 1 x32 discrete DRAM chips are used. In this mode, DQS0 is used to capture DQ[0:31], DQS4 is used to capture DQ[32:63] and DQS8 is used to capture ECC[0:7].
27	PCHB8	Precharge bit 8 enable. 0 MA[10] is used to indicate the auto-precharge and precharge all commands. 1 MA[8] is used to indicate the auto-precharge and precharge all commands. If x32_EN is cleared, then PCHB8 should be cleared as well.

Table 8-12. DDR_SDRAM_CFG Field Descriptions (continued)

Bits	Name	Description
28	HSE	Global half-strength override Sets I/O driver impedance to half strength. This impedance is used by the MDIC, address/command, data, and clock impedance values, but only if automatic hardware calibration is disabled and the corresponding group's software override is disabled in the DDR control driver register(s) described in Section 8.4.1.19, "DDR Control Driver Register 1 (DDRCDR_1)," This bit should be cleared if using automatic hardware calibration. 0 I/O driver impedance is configured to full strength. 1 I/O driver impedance is configured to half strength.
29	—	Reserved
30	MEM_HALT	DDR memory controller halt. When this bit is set, the memory controller does not accept any new data read/write transactions to DDR SDRAM until the bit is cleared again. This can be used when bypassing initialization and forcing MODE REGISTER SET commands through software. 0 DDR controller accepts new transactions. 1 DDR controller finishes any remaining transactions, and then it remains halted until this bit is cleared by software.
31	BI	Bypass initialization 0 DDR controller cycles through initialization routine based on SDRAM_TYPE 1 Initialization routine is bypassed. Software is responsible for initializing memory through DDR_SDRAM_MODE2 register. If software is initializing memory, then the MEM_HALT bit can be set to prevent the DDR controller from issuing transactions during the initialization sequence. Note that the DDR controller does not issue a DLL reset to the DRAMs when bypassing the initialization routine, regardless of the value of DDR_SDRAM_CFG[DLL_RST_DIS]. If a DLL reset is required, then the controller should be forced to enter and exit self refresh after the controller is enabled. See Section 8.4.1.15, "DDR Initialization Address (DDR_INIT_ADDR)," for details on avoiding ECC errors in this mode.

8.4.1.8 DDR SDRAM Control Configuration 2 (DDR_SDRAM_CFG_2)

The DDR SDRAM control configuration register 2, shown in Figure 8-9, provides more control configuration for the DDR controller.

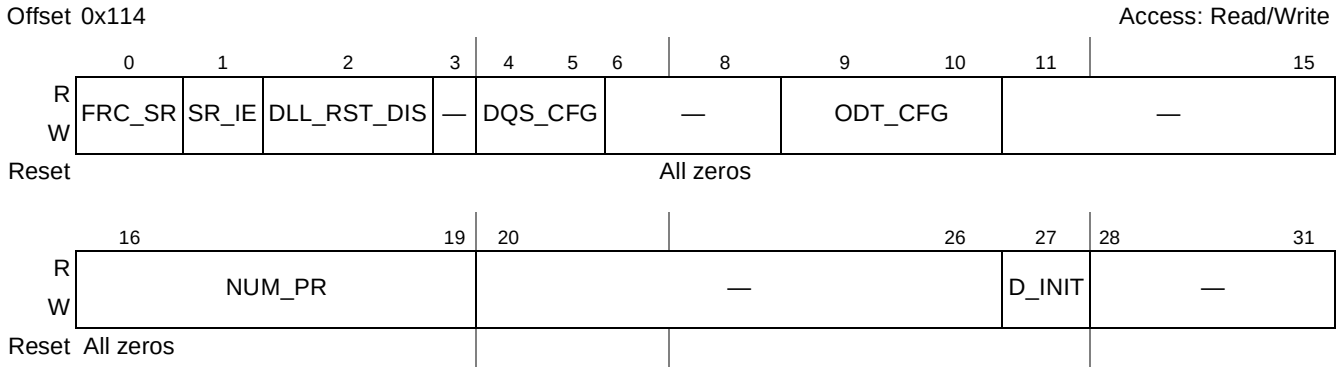


Figure 8-9. DDR SDRAM Control Configuration Register 2 (DDR_SDRAM_CFG_2)

Table 8-13 describes the DDR_SDRAM_CFG_2 fields.

Table 8-13. DDR_SDRAM_CFG_2 Field Descriptions

Bits	Name	Description
0	FRC_SR	Force self refresh 0 DDR controller operates in normal mode. 1 DDR controller enters self-refresh mode.
1	SR_IE	Self-refresh interrupt enable. The DDR controller can be placed into self refresh mode by forcing the PIC to assert IRQ_OUT. This is considered a 'panic interrupt' by the DDR controller, and it enters self refresh as soon as possible. DDR_SDRAM_CFG[SREN] must also be set if the panic interrupt is used. 0 DDR controller does not enter self-refresh mode if panic interrupt is asserted. 1 DDR controller enters self-refresh mode if panic interrupt is asserted.
2	DLL_RST_DIS	DLL reset disable. The DDR controller typically issues a DLL reset to the DRAMs when exiting self refresh. However, this function may be disabled by setting this bit during initialization. 0 DDR controller issues a DLL reset to the DRAMs when exiting self refresh. 1 DDR controller does not issue a DLL reset to the DRAMs when exiting self refresh.
3	—	Reserved
4–5	DQS_CFG	DQS configuration 00 Only true DQS signals are used. 01 Differential DQS signals are used for DDR2 support. 10 Reserved 11 Reserved
6–8	—	Reserved
9–10	ODT_CFG	ODT configuration. This field defines how ODT is driven to the on-chip IOs. See Section 8.4.1.19, "DDR Control Driver Register 1 (DDRCDR_1)," which defines the termination value that is used. (DDR2-specific, must be cleared for DDR1) 00 Never assert ODT to internal IOs 01 Assert ODT to internal IOs only during writes to DRAM 10 Assert ODT to internal IOs only during reads to DRAM 11 Always keep ODT asserted to internal IOs
16–19	NUM_PR	Number of posted refreshes. This determines how many posted refreshes, if any, can be issued at one time. Note that if posted refreshes are used, then this field, along with DDR_SDRAM_INTERVAL[REFINT], must be programmed such that the maximum t_{ras} specification cannot be violated. For example, some DDR1 SDRAMs are not able to use more than 3 posted refreshes because the required refresh interval could then exceed the maximum constraint for t_{ras}. 0000 Reserved 0001 1 refresh is issued at a time 0010 2 refreshes is issued at a time 0011 3 refreshes is issued at a time ... 1000 8 refreshes is issued at a time 1001–1111 Reserved
20–26	—	Reserved, should be cleared.

Table 8-13. DDR_SDRAM_CFG_2 Field Descriptions (continued)

Bits	Name	Description
27	D_INIT	DRAM data initialization This bit is set by software, and it is cleared by hardware. If software sets this bit before the memory controller is enabled, the controller automatically initializes DRAM after it is enabled. This bit is automatically cleared by hardware once the initialization is completed. This data initialization bit should only be set when the controller is idle. 0 There is not data initialization in progress, and no data initialization is scheduled 1 The memory controller initializes memory once it is enabled. This bit remains asserted until the initialization is complete. The value in DDR_DATA_INIT register is used to initialize memory.
28–31	—	Reserved

8.4.1.9 DDR SDRAM Mode Configuration (DDR_SDRAM_MODE)

The DDR SDRAM mode configuration register, shown in Figure 8-10, sets the values loaded into the DDR's mode registers.

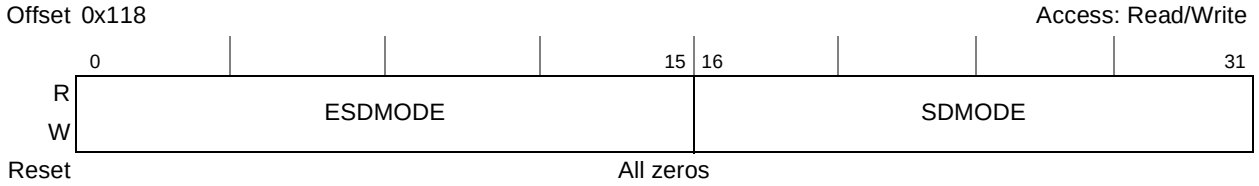


Figure 8-10. DDR SDRAM Mode Configuration Register (DDR_SDRAM_MODE)

Table 8-14 describes the DDR_SDRAM_MODE fields.

Table 8-14. DDR_SDRAM_MODE Field Descriptions

Bits	Name	Description
0–15	ESDMODE	Extended SDRAM mode. Specifies the initial value loaded into the DDR SDRAM extended mode register. The range and meaning of legal values is specified by the DDR SDRAM manufacturer. When this value is driven onto the address bus (during the DDR SDRAM initialization sequence), MA[0] presents the lsb of ESDMODE, which, in the big-endian convention shown in Figure 8-10, corresponds to ESDMODE[15]. The msb of the SDRAM extended mode register value must be stored at ESDMODE[0].
16–31	SDMODE	SDRAM mode. Specifies the initial value loaded into the DDR SDRAM mode register. The range of legal values is specified by the DDR SDRAM manufacturer. When this value is driven onto the address bus (during DDR SDRAM initialization), MA[0] presents the lsb of SDMODE, which, in the big-endian convention shown in Figure 8-10, corresponds to SDMODE[15]. The msb of the SDRAM mode register value must be stored at SDMODE[0]. Because the memory controller forces SDMODE[7] to certain values depending on the state of the initialization sequence, (for resetting the SDRAM's DLL) the corresponding bits of this field are ignored by the memory controller. Note that SDMODE[7] is mapped to MA[8].

8.4.1.10 DDR SDRAM Mode 2 Configuration (DDR_SDRAM_MODE_2)

The DDR SDRAM mode 2 configuration register, shown in Figure 8-11, sets the values loaded into the DDR's extended mode 2 and 3 registers (for DDR2).

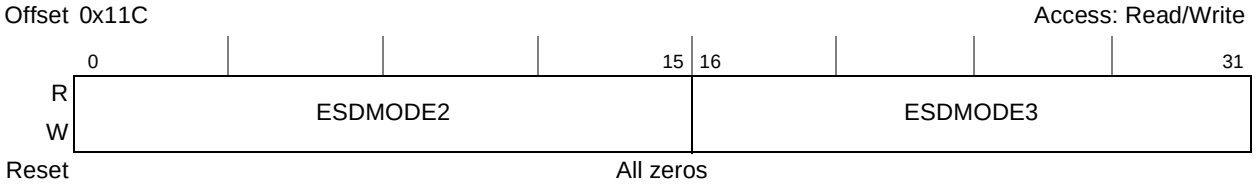


Figure 8-11. DDR SDRAM Mode 2 Configuration Register (DDR_SDRAM_MODE_2)

Table 8-15 describes the DDR_SDRAM_MODE_2 fields.

Table 8-15. DDR_SDRAM_MODE_2 Field Descriptions

Bits	Name	Description
0–15	ESDMODE2	Extended SDRAM mode 2. Specifies the initial value loaded into the DDR SDRAM extended 2 mode register. The range and meaning of legal values is specified by the DDR SDRAM manufacturer. When this value is driven onto the address bus (during the DDR SDRAM initialization sequence), MA[0] presents the lsb bit of ESDMODE2, which, in the big-endian convention shown in Figure 8-11, corresponds to ESDMODE2[15]. The msb of the SDRAM extended mode 2 register value must be stored at ESDMODE2[0].
16–31	ESDMODE3	Extended SDRAM mode 3. Specifies the initial value loaded into the DDR SDRAM extended 3 mode register. The range of legal values of legal values is specified by the DDR SDRAM manufacturer. When this value is driven onto the address bus (during DDR SDRAM initialization), MA[0] presents the lsb of ESDMODE3, which, in the big-endian convention shown in Figure 8-11, corresponds to ESDMODE3[15]. The msb of the SDRAM extended mode 3 register value must be stored at ESDMODE3[0].

8.4.1.11 DDR SDRAM Mode Control Register (DDR_SDRAM_MD_CNTL)

The DDR SDRAM mode control register, shown in Figure 8-12, allows the user to carry out the following tasks:

- Issue a mode register set command to a particular chip select
- Issue an immediate refresh to a particular chip select
- Issue an immediate precharge or precharge all command to a particular chip select
- Force the CKE signals to a specific value

Table 8-16 describes the fields of this register. Table 8-17 shows the user how to set the fields of this register to accomplish the above tasks.

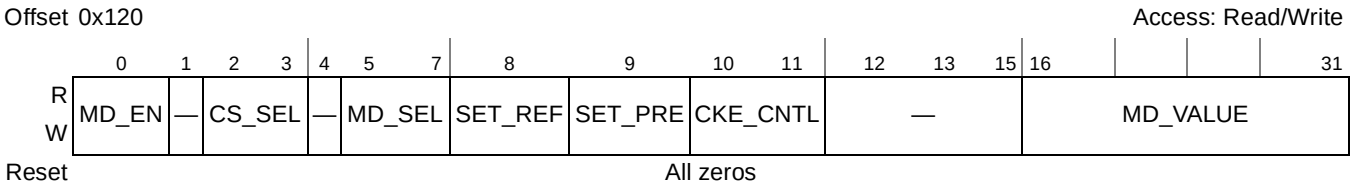


Figure 8-12. DDR SDRAM Mode Control Register (DDR_SDRAM_MD_CNTL)

Table 8-16 describes the DDR_SDRAM_MD_CNTL fields.

NOTE

Note that MD_EN, SET_REF, and SET_PRE are mutually exclusive; only one of these fields can be set at a time.

Table 8-16. DDR_SDRAM_MD_CNTL Field Descriptions

Bits	Name	Description
0	MD_EN	Mode enable. Setting this bit specifies that valid data in MD_VALUE is ready to be written to DRAM as one of the following commands: • MODE REGISTER SET • EXTENDED MODE REGISTER SET • EXTENDED MODE REGISTER SET 2 • EXTENDED MODE REGISTER SET 3 The specific command to be executed is selected by setting MD_SEL. In addition, the chip select must be chosen by setting CS_SEL.MD_EN is set by software and cleared by hardware once the command has been issued. 0 Indicates that no mode register set command needs to be issued. 1 Indicates that valid data contained in the register is ready to be issued as a mode register set command.
1	—	Reserved
2–3	CS_SEL	Select chip select. Specifies the chip select that is driven active due to any command forced by software in DDR_SDRAM_MD_CNTL. 00 Chip select 0 is active 01 Chip select 1 is active 10 Chip select 2 is active 11 Chip select 3 is active
4	—	Reserved
5–7	MD_SEL	Mode register select. MD_SEL specifies one of the following: • During a mode select command, selects the SDRAM mode register to be changed • During a precharge command, selects the SDRAM logical bank to be precharged. A precharge all command ignores this field. • During a refresh command, this field is ignored. Note that MD_SEL contains the value that is presented onto the memory bank address pins (MBA*n*) of the DDR controller. 000 MR 001 EMR 010 EMR2 011 EMR3
8	SET_REF	Set refresh. Forces an immediate refresh to be issued to the chip select specified by DDR_SDRAM_MD_CNTL[CS_SEL]. This bit is set by software and cleared by hardware once the command has been issued. 0 Indicates that no refresh command needs to be issued. 1 Indicates that a refresh command is ready to be issued.
9	SET_PRE	Set precharge. Forces a precharge or precharge all to be issued to the chip select specified by DDR_SDRAM_MD_CNTL[CS_SEL]. This bit is set by software and cleared by hardware once the command has been issued. 0 Indicates that no precharge all command needs to be issued. 1 Indicates that a precharge all command is ready to be issued.

Table 8-16. DDR_SDRAM_MD_CNTL Field Descriptions (continued)

Bits	Name	Description
10–11	CKE_CNTL	Clock enable control. Allows software to globally clear or set all CKE signals issued to DRAM. Once software has forced the value driven on CKE, that value continues to be forced until software clears the CKE_CNTL bits. At that time, the DDR controller continues to drive the CKE signals to the same value forced by software until another event causes the CKE signals to change (such as, self refresh entry/exit, power down entry/exit). 00 CKE signals are not forced by software. 01 CKE signals are forced to a low value by software. 10 CKE signals are forced to a high value by software. 11 Reserved
12–15	—	Reserved
16–31	MD_VALUE	Mode register value. This field, which specifies the value that is presented on the memory address pins of the DDR controller during a mode register set command, is significant only when this register is used to issue a mode register set command or a precharge or precharge all command. For a mode register set command, this field contains the data to be written to the selected mode register. For a precharge command, only bit five is significant: 0 Issue a precharge command; MD_SEL selects the logical bank to be precharged 1 Issue a precharge all command; all logical banks are precharged

Table 8-17 shows how DDR_SDRAM_MD_CNTL fields should be set for each of the tasks described above.

Table 8-17. Settings of DDR_SDRAM_MD_CNTL Fields

Field	Mode Register Set	Refresh	Precharge	Clock Enable Signals Control
MD_EN	1	0	0	—
SET_REF	0	1	0	—
SET_PRE	0	0	1	—
CS_SEL	Chooses chip select (CS)			—
MD_SEL	Select mode register. See Table 8-16.	—	Selects logical bank	—
MD_VALUE	Value written to mode register	—	Only bit five is significant. See Table 8-16.	—
CKE_CNTL	0	0	0	See Table 8-16.

8.4.1.12 DDR SDRAM Interval Configuration (DDR_SDRAM_INTERVAL)

The DDR SDRAM interval configuration register, shown in Figure 8-13, 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.

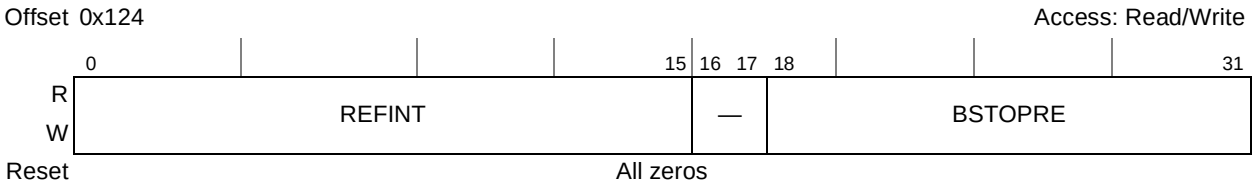


Figure 8-13. DDR SDRAM Interval Configuration Register (DDR_SDRAM_INTERVAL)

Table 8-18 describes the DDR_SDRAM_INTERVAL fields.

Table 8-18. DDR_SDRAM_INTERVAL Field Descriptions

Bits	Name	Description
0–15	REFINT	Refresh interval. Represents the number of memory bus clock cycles between refresh cycles. Depending on DDR_SDRAM_CFG_2[NUM_PR], some number of rows are refreshed in each DDR SDRAM physical bank during each refresh cycle. The value for REFINT depends on the specific SDRAMs used and the interface clock frequency. Refreshes are not issued when the REFINT is set to all 0s.
16–17	—	Reserved
18–31	BSTOPRE	Precharge interval. Sets the duration (in memory bus clocks) that a page is retained after a DDR SDRAM access. If BSTOPRE is zero, the DDR memory controller uses auto-precharge read and write commands rather than operating in page mode. This is called global auto-precharge mode.

8.4.1.13 DDR SDRAM Data Initialization (DDR_DATA_INIT)

The DDR SDRAM data initialization register, shown in Figure 8-14, provides the value that is used to initialize memory if DDR_SDRAM_CFG2[D_INIT] is set.

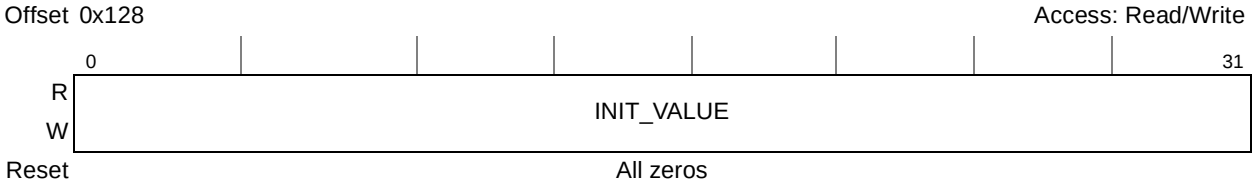


Figure 8-14. DDR SDRAM Data Initialization Configuration Register (DDR_DATA_INIT)

Table 8-19 describes the DDR_DATA_INIT fields.

Table 8-19. DDR_DATA_INIT Field Descriptions

Bits	Name	Description
0–31	INIT_VALUE	Initialization value. Represents the value that DRAM is initialized with if DDR_SDRAM_CFG2[D_INIT] is set.

8.4.1.14 DDR SDRAM Clock Control (DDR_SDRAM_CLK_CNTL)

The DDR SDRAM clock control configuration register, shown in Figure 8-15, provides a 1/8-cycle clock adjustment.

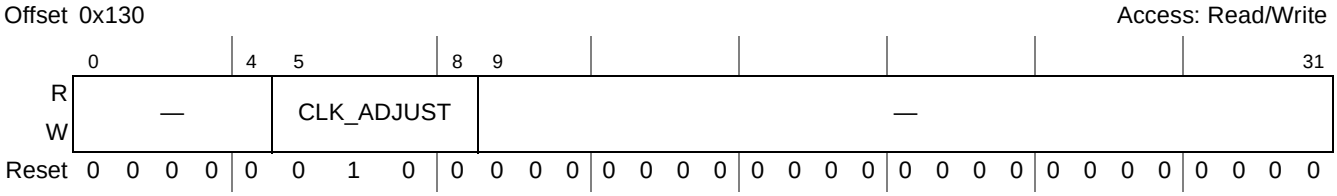


Figure 8-15. DDR SDRAM Clock Control Configuration Register (DDR_SDRAM_CLK_CNTL)

Table 8-20 describes the DDR_SDRAM_CLK_CNTL fields.

Table 8-20. DDR_SDRAM_CLK_CNTL Field Descriptions

Bits	Name	Description
0–4	—	Reserved
5–8	CLK_ADJUST	Clock adjust 0000 Clock is launched aligned with address/command 0001 Clock is launched 1/8 applied cycle after address/command 0010 Clock is launched 1/4 applied cycle after address/command 0011 Clock is launched 3/8 applied cycle after address/command 0100 Clock is launched 1/2 applied cycle after address/command 0101 Clock is launched 5/8 applied cycle after address/command 0110 Clock is launched 3/4 applied cycle after address/command 0111 Clock is launched 7/8 applied cycle after address/command 1000 Clock is launched 1 applied cycle after address/command 1001–1111Reserved
9–31	—	Reserved

8.4.1.15 DDR Initialization Address (DDR_INIT_ADDR)

The DDR SDRAM initialization address register, shown in Figure 8-16, provides the address that is used for the data strobe to data skew adjustment and automatic $\overline{\text{CAS}}$ to preamble calibration after POR.

NOTE

After the skew adjustment, this address contains bad ECC data. This is not important at POR, as all of memory should be subsequently initialized if ECC is enabled (either by software or through the use of DDR_SDRAM_CFG_2[D_INIT]).

If an $\overline{\text{HRESET}}$ has been issued after the DRAM is in self-refresh mode, however, memory is not initialized, so this address should be written to using an 8- or 32-byte transaction to avoid possible ECC errors if this address could later be accessed.

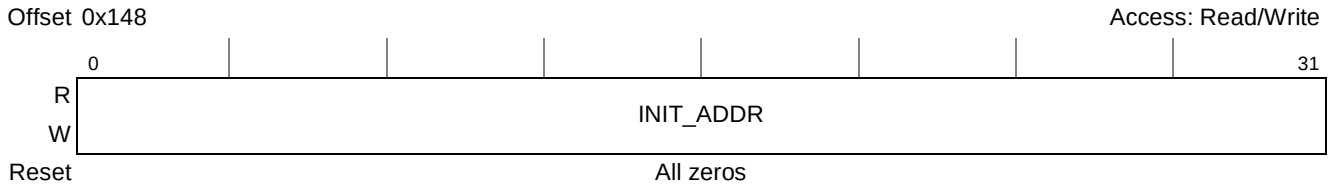


Figure 8-16. DDR Initialization Address Configuration Register (DDR_INIT_ADDR)

Table 8-21 describes the DDR_INIT_ADDR fields.

Table 8-21. DDR_INIT_ADDR Field Descriptions

Bits	Name	Description
0–31	INIT_ADDR	Initialization address. Represents the address that is used for the data strobe to data skew adjustment and automatic CAS to preamble calibration at POR. This address is written to during the initialization sequence.

8.4.1.16 DDR Initialization Enable Extended Address (DDR_INIT_EXT_ADDR)

The DDR SDRAM initialization extended address register, shown in Figure 8-17, provides the extended address that is used for the data strobe to data skew adjustment and automatic $\overline{\text{CAS}}$ to preamble calibration after POR.

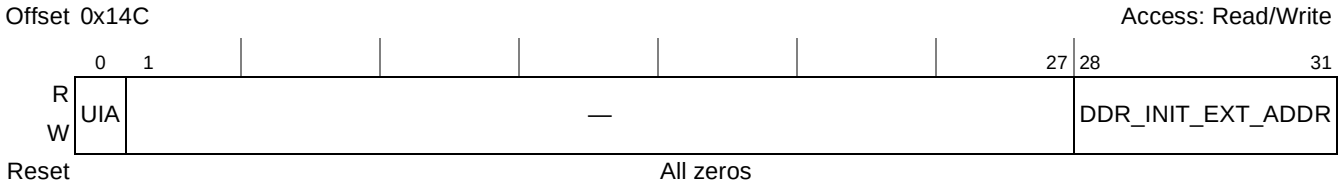


Figure 8-17. DDR Initialization Extended Address Configuration Register (DDR_INIT_EXT_ADDR)

Table 8-22 describes the DDR_INIT_EXT_ADDR fields.

Table 8-22. DDR_INIT_EXT_ADDR Field Descriptions

Bits	Name	Description
0	UIA	Use initialization address. 0 Use the default address for training sequence as calculated by the controller. This is the first valid address in the first enabled chip select. 1 Use the initialization address programmed in DDR_INIT_ADDR and DDR_INIT_EXT_ADDR.
1–27	—	Reserved, should be cleared.
28–31	INIT_EXT_ADDR	Initialization extended address. Represents the extended address that is used for the data strobe to data skew adjustment and automatic $\overline{\text{CAS}}$ to preamble calibration at POR. This extended address is written to during the initialization sequence.

8.4.1.17 DDR Debug Status Register 1 (DDRDSR_1)

The DDRDSR_1 register, shown in Figure 8-18, contains the DDR driver compensation input value and the current settings of the P and N FET impedance for MDIC*n*, command/control, and data.

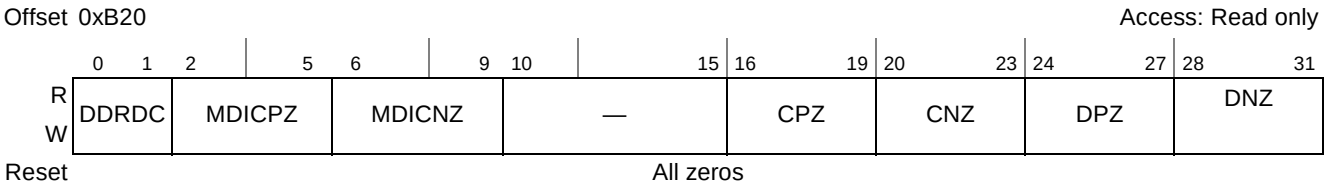


Figure 8-18. DDR Debug Status Register 1 (DDRDSR_1)

Table 8-23 describes the DDRDSR_1 fields.

Table 8-23. DDRDSR_1 Field Descriptions

Bits	Name	Description
0–1	DDRDC	DDR driver compensation input value
2–5	MDICPZ	Current setting of PFET driver MDIC impedance
6–9	MDICNZ	Current setting of NFET driver MDIC impedance
10–15	—	Reserved, should be cleared.
16–19	CPZ	Current setting of PFET driver command impedance
20–23	CNZ	Current setting of NFET driver command impedance
24–27	DPZ	Current setting of PFET driver data impedance
28–31	DNZ	Current setting of NFET driver data impedance

8.4.1.18 DDR Debug Status Register 2 (DDRDSR_2)

The DDRDSR_2 register, shown in Figure 8-19, contains the current settings of the P and N FET impedance for the DDR drivers for clocks.

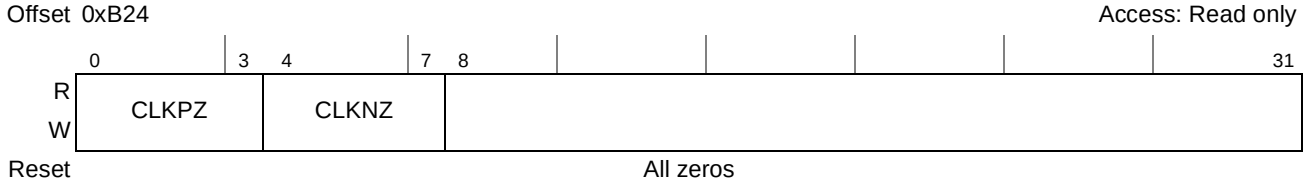


Figure 8-19. DDR Debug Status Register 2 (DDRDSR_2)

Table 8-24 describes the DDRDSR_2 fields.

Table 8-24. DDRDSR_2 Field Descriptions

Bits	Name	Description
0–3	CLKPZ	Current setting of PFET driver clock impedance
4–7	CLKNZ	Current setting of NFET driver clock impedance
8–31	—	Reserved

8.4.1.19 DDR Control Driver Register 1 (DDRCDR_1)

DDRCDR_1, shown in Figure 8-20, sets the driver hardware compensation enable, the DDR MDIC driver P/N impedance, ODT termination value for IOs, driver software override enable for MDIC, driver software override enable for address/command, driver software override enable for data, the DDR address/command driver P/N impedance, and the DDR data driver P/N impedance.

The fields in DDRCDR_1, other than DDRCDR_1[ODT], are used to enable driver calibration with the MDIC[0:1] pins. This can be used to calibrate the DDR drivers to 18 ohms. However, this should only be used for full-strength driver applications. If half strength is desired, then this calibration should remain disabled.

The hardware DDR driver calibration is enabled via DDRCDR_1[DHC_EN]. If this is used, then it should be set before DDR_SDRAM_CFG[MEM_EN] is set.

Software can be used to calibrate the drivers instead of the automatic hardware calibration. If software calibration is used, the following steps should be taken:

1. Set DDRCDR_1[DSO_MDIC_EN] and ensure that DDRCDR_1[DHC_EN] is cleared
2. Set the highest impedance (value 0000) for DDRCDR_1[DSO_MDICPZ]
3. Set DDRCDR_1[DSO_MDIC_PZ_OE] to enable the output enable for MDIC[0]
4. After at least 4 cycles, read DDRDSR_1[0]. If the value is 0, then use the next lowest impedance, and read DDRDSR_1[0] again. Once a value of 1 is detected, then leave DDRCDR_1[DSO_MDICPZ] at the calibrated value
5. Clear DDRCDR_1[DSO_MDIC_PZ_OE]
6. After DDRCDR_1[DSO_MDICPZ} is calibrated, set a value of 0000 for DDRCDR_1[DSO_MDICNZ]
7. Set DDRCDR_1[DSO_MDIC_NZ_OE] to enable the output enable for MDIC[1]
8. After at least 4 cycles, read DDRDSR_1[1]. If the value is 1, then use the next lowest impedance, and read DDRDSR_1[1] again. Once a value of 0 is detected, then leave DDRCDR_1[DSO_MDICNZ] at the calibrated value
9. Clear DDRCDR_1[DSO_MDIC_NZ_OE]

Note that the legal impedance values (from highest impedance to lowest impedance) are:

- 0000
- 1000
- 1100

- 1110
- 1111

Offset 0xB28 Access: Read/Write

	0	1	2	5	6	9	10	11	12	13	14	15
R W	DHC_EN	DSO_MDIC_EN	DSO_MDICPZ		DSO_MDICNZ		DSO_MDIC_PZ_OE	DSO_MDIC_NZ_OE	ODT	—	DSO_C_EN	DSO_D_EN
Reset	All zeros											

	16	19	20	23	24	27	28	31
R W	DSO_CPZ		DSO_CNZ		DSO_DPZ		DSO_DNZ	
Reset	All zeros							

Figure 8-20. DDR Control Driver Register 1 (DDRCDR_1)

Table 8-25 describes the DDRCDR_1 fields.

Table 8-25. DDRCDR_1 Field Descriptions

Bits	Name	Description
0	DHC_EN	DDR driver hardware compensation enable
1	DSO_MDIC_EN	Driver software override enable for MDIC
2–5	DSO_MDICPZ	DDR driver software MDIC p-impedance override
6–9	DSO_MDICNZ	DDR driver software MDIC n-impedance override
10	DSO_MDIC_PZ_OE	Driver software override p-impedance output enable
11	DSO_MDIC_NZ_OE	Driver software override n-impedance output enable
12	ODT	ODT termination value for IOs 0 102.5 ohm 1 175.5 ohm
13	—	Reserved
14	DSO_C_EN	Driver software override enable for address/command
15	DSO_D_EN	Driver software override enable for data
16–19	DSO_CPZ	DDR driver software command p-impedance override
20–23	DSO_CNZ	DDR driver software command n-impedance override
24–27	DSO_DPZ	Driver software data p-impedance override
28–31	DSO_DNZ	Driver software data n-impedance override

8.4.1.20 DDR Control Driver Register 2 (DDRCDR_2)

The DDRCDR_2, shown in Figure 8-21, sets the driver software override enable for clocks, and the DDR clocks driver P/N impedance.

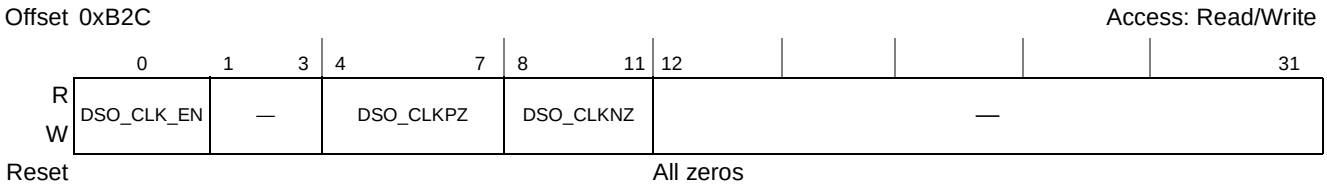


Figure 8-21. DDR Control Driver Register 2 (DDRCDR_2)

Table 8-26 describes the DDRCDR_2 fields.

Table 8-26. DDRCDR_2 Field Descriptions

Bits	Name	Description
0	DSO_CLK_EN	Driver software override enable for clocks
1–3	—	Reserved
4–7	DSO_CLKPZ	Driver software clocks p-impedance override
8–11	DSO_CLKNZ	Driver software clocks n-impedance override
12–31	—	Reserved

8.4.1.21 DDR IP Block Revision 1 (DDR_IP_REV1)

The DDR IP block revision 1 register, shown in Figure 8-22, provides read-only fields with the IP block ID, along with major and minor revision information.

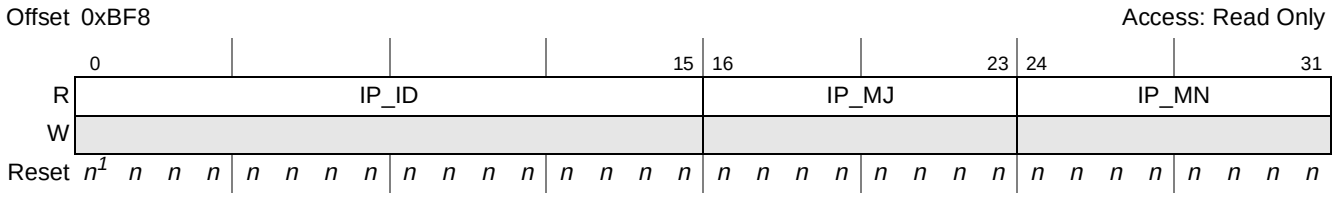


Figure 8-22. DDR IP Block Revision 1 (DDR_IP_REV1)

1 For reset values, see Table 8-27.

Table 8-27 describes the DDR_IP_REV1 fields.

Table 8-27. DDR_IP_REV1 Field Descriptions

Bits	Name	Description
0–15	IP_ID	IP block ID. For the DDR controller, this value is 0x0002.
16–23	IP_MJ	Major revision. This is currently set to 0x03.
24–31	IP_MN	Minor revision. This is currently set to 0x01.

8.4.1.22 DDR IP Block Revision 2 (DDR_IP_REV2)

The DDR IP block revision 2 register, shown in Figure 8-23, provides read-only fields with the IP block integration and configuration options.

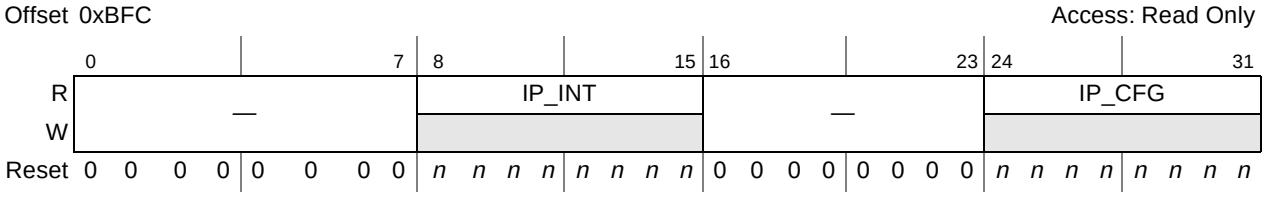


Figure 8-23. DDR IP Block Revision 2 (DDR_IP_REV2)

Table 8-28 describes the DDR_IP_REV2 fields.

Table 8-28. DDR_IP_REV2 Field Descriptions

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

8.4.1.23 Memory Data Path Error Injection Mask High (DATA_ERR_INJECT_HI)

The memory data path error injection mask high register is shown in Figure 8-24.

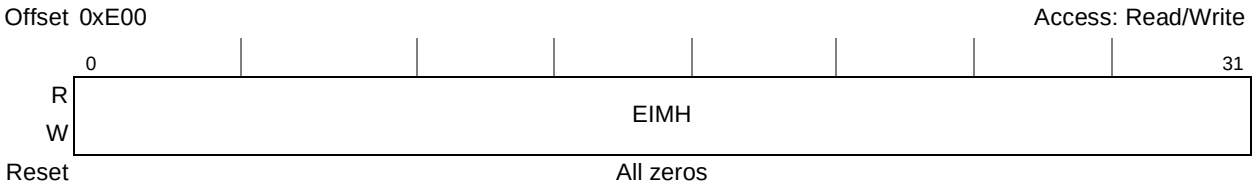


Figure 8-24. Memory Data Path Error Injection Mask High Register (DATA_ERR_INJECT_HI)

Table 8-29 describes the DATA_ERR_INJECT_HI fields.

Table 8-29. 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.

8.4.1.24 Memory Data Path Error Injection Mask Low (DATA_ERR_INJECT_LO)

The memory data path error injection mask low register is shown in Figure 8-25.

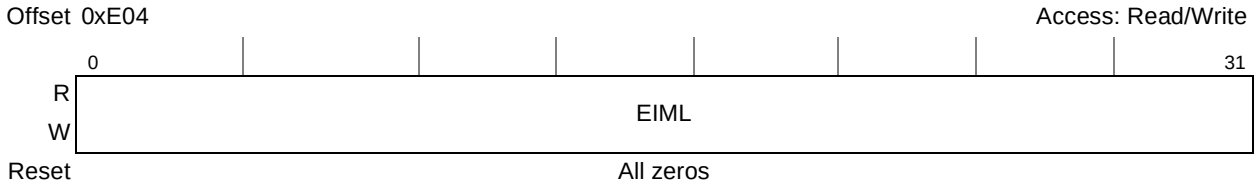


Figure 8-25. Memory Data Path Error Injection Mask Low Register (DATA_ERR_INJECT_LO)

Table 8-30 describes the DATA_ERR_INJECT_LO fields.

Table 8-30. 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.

8.4.1.25 Memory Data Path Error Injection Mask ECC (ERR_INJECT)

The memory data path error injection mask ECC register, shown in Figure 8-26, sets the ECC mask, enables errors to be written to ECC memory, and allows the ECC byte to mirror the most significant data byte.

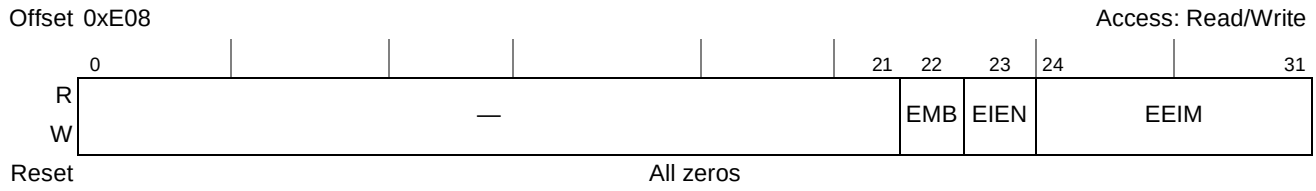


Figure 8-26. Memory Data Path Error Injection Mask ECC Register (ERR_INJECT)

Table 8-31 describes the ERR_INJECT fields.

Table 8-31. 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. Note that error injection should not be enabled until the memory controller has been enabled through DDR_SDRAM_CFG[MEM_EN].
24–31	EEIM	ECC error injection mask. Setting a mask bit causes the corresponding ECC bit to be inverted on memory bus writes.

8.4.1.26 Memory Data Path Read Capture High (CAPTURE_DATA_HI)

The memory data path read capture high register, shown in Figure 8-27, stores the high word of the read data path during error capture.

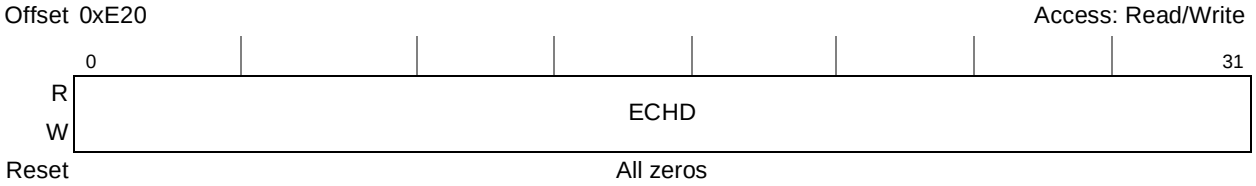


Figure 8-27. Memory Data Path Read Capture High Register (CAPTURE_DATA_HI)

Table 8-32 describes the CAPTURE_DATA_HI fields.

Table 8-32. 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.

8.4.1.27 Memory Data Path Read Capture Low (CAPTURE_DATA_LO)

The memory data path read capture low register, shown in Figure 8-28, stores the low word of the read data path during error capture.

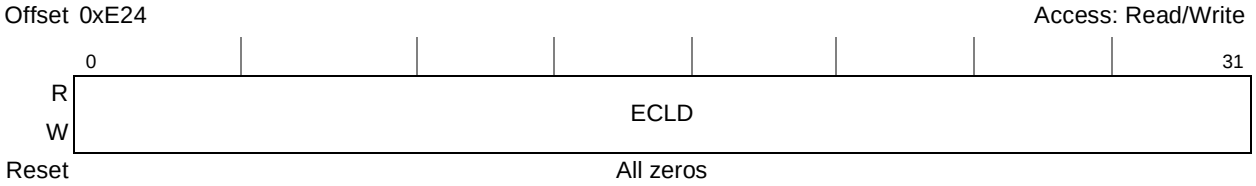


Figure 8-28. Memory Data Path Read Capture Low Register (CAPTURE_DATA_LO)

Table 8-33 describes the CAPTURE_DATA_LO fields.

Table 8-33. 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.

8.4.1.28 Memory Data Path Read Capture ECC (CAPTURE_ECC)

The memory data path read capture ECC register, shown in Figure 8-29, stores the ECC syndrome bits that were on the data bus when an error was detected.

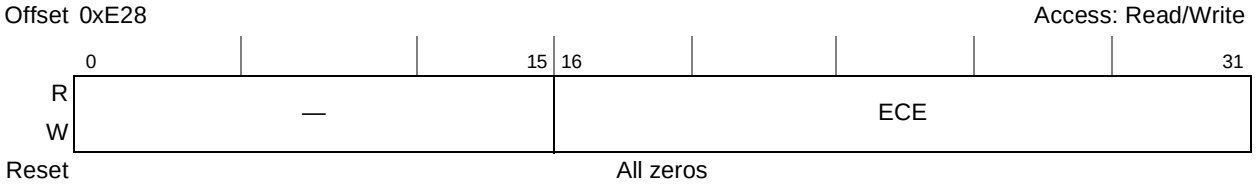


Figure 8-29. Memory Data Path Read Capture ECC Register (CAPTURE_ECC)

Table 8-34 describes the CAPTURE_ECC fields.

Table 8-34. CAPTURE_ECC Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	ECE	Error capture ECC. Captures the ECC bits on the data path whenever errors are detected. 16:23—8-bit ECC code for 1st 32 bits 24:31—8-bit ECC code for 2nd 32 bits **Note:** In 64-bit mode, only 24:31 should be used, although 16:23 shows the 8-bit ECC code replicated.

8.4.1.29 Memory Error Detect (ERR_DETECT)

The memory error detect register stores the detection bits for multiple memory errors, single- and multiple-bit ECC errors, and memory select errors. It is a read/write register. A bit can be cleared by writing a one to the bit. System software can determine the type of memory error by examining the contents of this register. If an error is disabled with ERR_DISABLE, the corresponding error is never detected or captured in ERR_DETECT.

ERR_DETECT is shown in Figure 8-30.

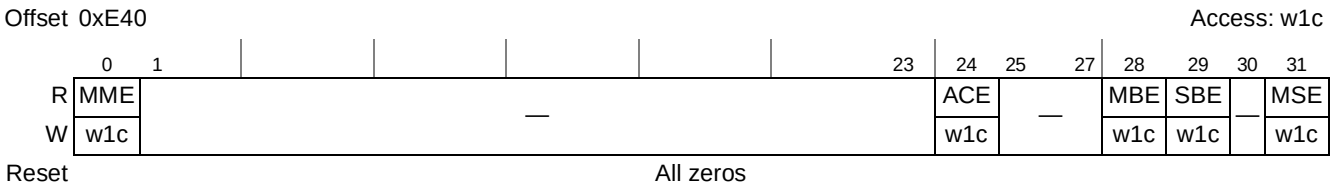


Figure 8-30. Memory Error Detect Register (ERR_DETECT)

Table 8-35 describes the ERR_DETECT fields.

Table 8-35. 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.
1–23	—	Reserved
24	ACE	Automatic calibration error. This bit is cleared by software writing a 1. 0 An automatic calibration error has not been detected. 1 An automatic calibration error has been detected.
25–27	—	Reserved
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.

8.4.1.30 Memory Error Disable (ERR_DISABLE)

The memory error disable register, shown in Figure 8-31, allows selective disabling of the DDR controller's error detection circuitry. Disabled errors are not detected or reported.

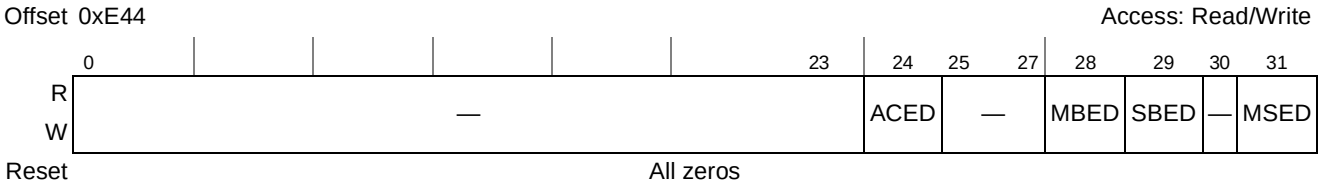


Figure 8-31. Memory Error Disable Register (ERR_DISABLE)

Table 8-36 describes the ERR_DISABLE fields.

Table 8-36. ERR_DISABLE Field Descriptions

Bits	Name	Description
0–23	—	Reserved
24	ACED	Automatic calibration error disable 0 Automatic calibration errors are enabled. 1 Automatic calibration errors are disabled.
25–27	—	Reserved

Table 8-36. ERR_DISABLE Field Descriptions (continued)

Bits	Name	Description
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. 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.

8.4.1.31 Memory Error Interrupt Enable (ERR_INT_EN)

The memory error interrupt enable register, shown in Figure 8-32, 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).

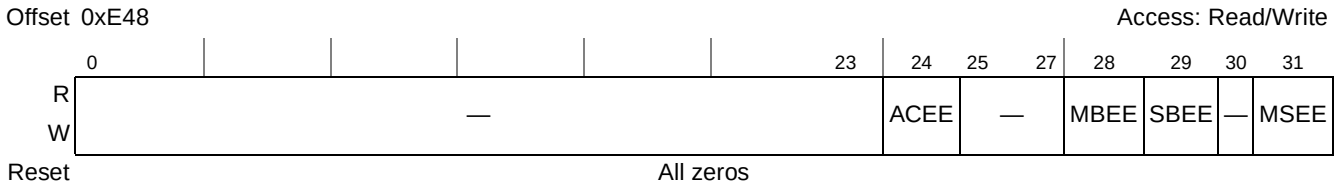


Figure 8-32. Memory Error Interrupt Enable Register (ERR_INT_EN)

Table 8-37 describes the ERR_INT_EN fields.

Table 8-37. ERR_INT_EN Field Descriptions

Bits	Name	Description
0–23	—	Reserved
24	ACEE	Automatic calibration error interrupt enable 0 Automatic calibration errors cannot generate interrupts. 1 Automatic calibration errors generate interrupts.
25–27	—	Reserved
28	MBEE	Multiple-bit ECC error interrupt enable. Note that uncorrectable read errors may cause an interrupt. 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.

8.4.1.32 Memory Error Attributes Capture (CAPTURE_ATTRIBUTES)

The memory error attributes capture register, shown in Figure 8-33, sets attributes for errors including type, size, source, and others.

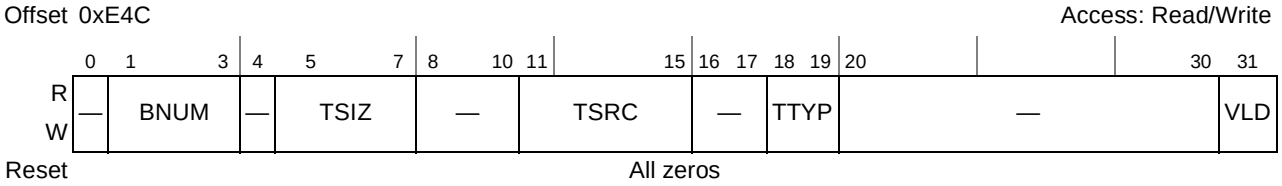


Figure 8-33. Memory Error Attributes Capture Register (CAPTURE_ATTRIBUTES)

Table 8-38 describes the CAPTURE_ATTRIBUTES fields.

Table 8-38. CAPTURE_ATTRIBUTES Field Descriptions

Bits	Name	Description
0	—	Reserved
1–3	BNUM	Data beat number. Captures the doubleword number for the detected error. Relevant only for ECC errors.
4	—	Reserved
5–7	TSIZ	Transaction size for the error. Captures the transaction size in double words. 000 4 double words 001 1 double word 010 2 double words 011 3 double words Others Reserved
8–10	—	Reserved
11–15	TSRC	Transaction source for the error 00000 PCI Express interface 1 00001 PCI Express interface 2/RapidIO 00010–01001 Reserved 01010 Boot sequencer 01011–01111 Reserved 10000 Core 0 (instruction/data) 10001 Reserved 10010 Core 1 (instruction/data) 10011–10100 Reserved 10101 DMA 10110–10111 Reserved 11000 eTSEC 1 11001 eTSEC 2 11010 eTSEC 3 11011 eTSEC 4 11100 RapidIO message unit 11101 RapidIO doorbell unit 11110 RapidIO port-write unit 11111 Reserved
		100xy are used by the QUICC Engine block as follows: x = TSRC[3] Least significant bit of SNUM y = TSCRC[4] CETM bit from RBMR/TBMR registers. RBMR/TBMR registers are described in protocol chapters. 101xx Reserved 11xxx Reserved **Note:** TSRC reflects the source of transaction and is used for debug purposes.
16–17	—	Reserved

Table 8-38. CAPTURE_ATTRIBUTES Field Descriptions (continued)

Bits	Name	Description
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.

8.4.1.33 Memory Error Address Capture (CAPTURE_ADDRESS)

The memory error address capture register, shown in Figure 8-34, holds the 32 lsbs of a transaction when a DDR ECC error is detected.

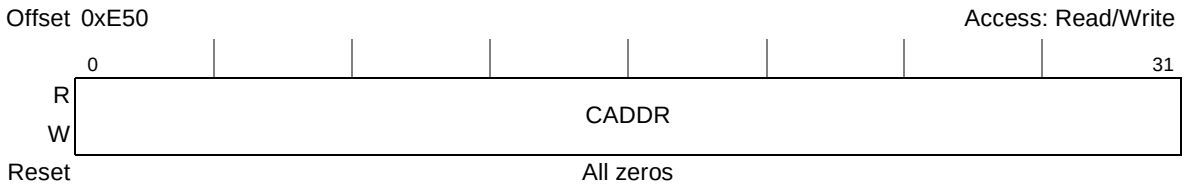


Figure 8-34. Memory Error Address Capture Register (CAPTURE_ADDRESS)

Table 8-39 describes the CAPTURE_ADDRESS fields.

Table 8-39. 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.

8.4.1.34 Memory Error Extended Address Capture (CAPTURE_EXT_ADDRESS)

The memory error extended address capture register, shown in Figure 8-35, holds the four most significant transaction bits when an error is detected.

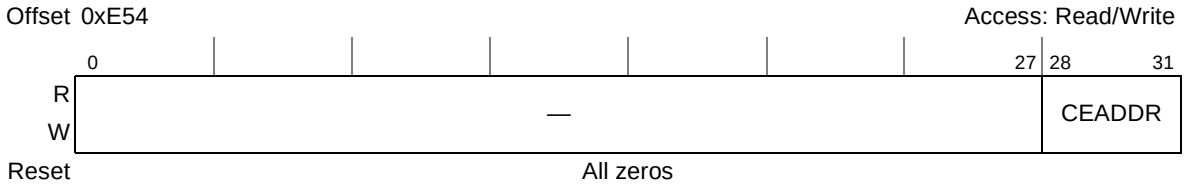


Figure 8-35. Memory Error Extended Address Capture Register (CAPTURE_EXT_ADDRESS)

Table 8-40 describes the CAPTURE_EXT_ADDRESS fields.

Table 8-40. CAPTURE_EXT_ADDRESS Field Descriptions

Bits	Name	Description
0–27	—	Reserved
28–31	CEADDR	Captured extended address. Captures the 4 msbs of the transaction address when an error is detected

8.4.1.35 Single-Bit ECC Memory Error Management (ERR_SBE)

The single-bit ECC memory error management register, shown in Figure 8-36, 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.

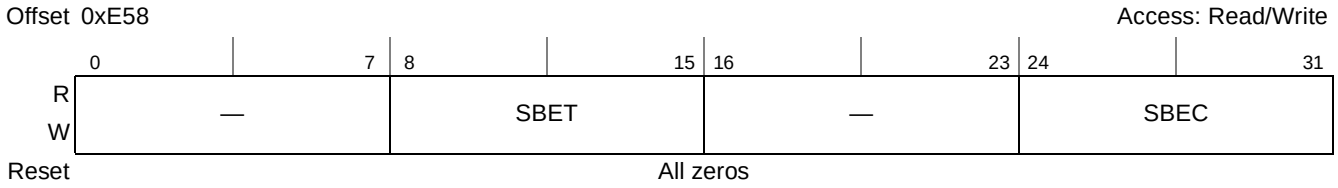


Figure 8-36. Single-Bit ECC Memory Error Management Register (ERR_SBE)

Table 8-41 describes the ERR_SBE fields.

Table 8-41. 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.

8.5 Functional Description

The DDR SDRAM controller controls processor and I/O interactions with system memory. It provides support for JEDEC-compliant DDR2 and DDR SDRAMs. The memory system allows a wide range of memory devices to be mapped to any arbitrary chip select, and support is provided for registered DIMMs and unbuffered DIMMs. However, registered DIMMs cannot be mixed with unbuffered DIMMs.

Figure 8-37 is a high-level block diagram of the DDR memory controller. Requests are received from the internal mastering device and the address is decoded to generate the physical bank, logical bank, row, and column addresses. The transaction is compared with values in the row open table to determine if the

address maps to an open page. If the transaction does not map to an open page, an active command is issued.

The memory interface supports as many as four physical banks of 64-/72-bit wide or 32-/40bit wide memory. Bank sizes up to 4 Gbytes are supported, providing up to a maximum of 16 Gbytes of DDR main memory.

Programmable parameters allow for a variety of memory organizations and timings. Optional error checking and correcting (ECC) protection is provided for the DDR SDRAM data bus. Using ECC, the DDR memory controller detects and corrects all single-bit errors within the 64- or 32-bit data bus, detects all double-bit errors within the 64- or 32-bit data bus, and detects all errors within a nibble. The controller allows as many as 32 pages to be open simultaneously. The amount of time (in clock cycles) the pages remain open is programmable with DDR_SDRAM_INTERVAL[BSTOPRE].

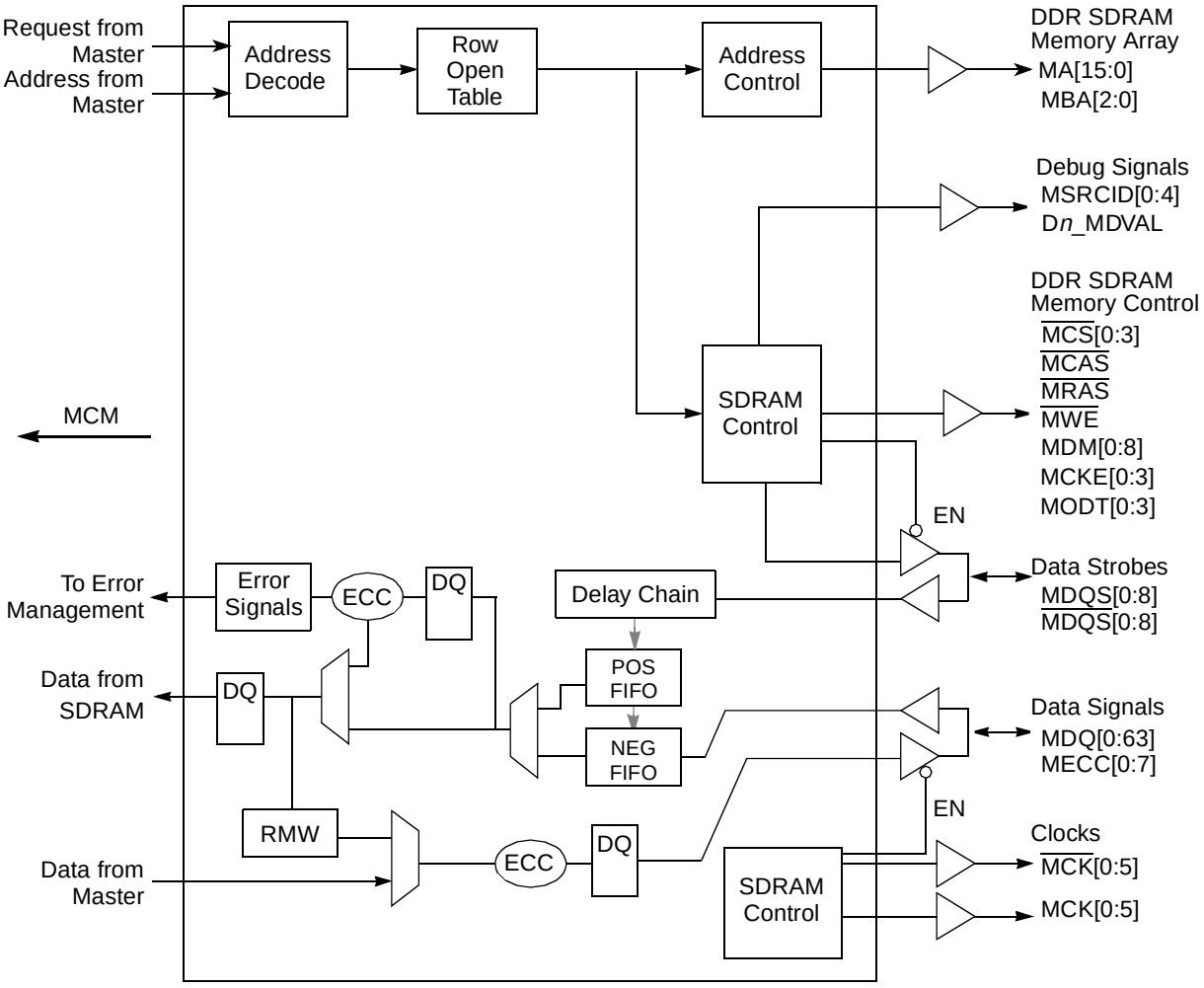


Figure 8-37. DDR Memory Controller Block Diagram

Read and write accesses to memory are burst oriented; accesses start at a selected location and continue for a programmed number of higher locations (4 or 8) in a programmed sequence. Accesses to closed pages start with the registration of an ACTIVE command followed by a READ or WRITE. (Accessing open pages does not require an ACTIVE command.) The address bits registered coincident with the activate

command specifies the logical bank and row to be accessed. The address coincident with the READ or WRITE command specify the logical bank and starting column for the burst access.

The data interface is source synchronous, meaning whatever sources the data also provides a clocking signal to synchronize data reception. These bidirectional data strobes (MDQS[0:8]) are inputs to the controller during reads and outputs during writes. The DDR SDRAM specification requires the data strobe signals to be centered within the data tenure during writes and to be offset by the controller to the center of the data tenure during reads. This delay is implemented in the controller for both reads and writes.

When ECC is enabled, 1 clock cycle is added to the read path to check ECC and correct single-bit errors. ECC generation does not add a cycle to the write path.

The address and command interface is also source synchronous, although 1/8 cycle adjustments are provided for adjusting the clock alignment.

Figure 8-38 shows an example DDR SDRAM configuration with four logical banks.

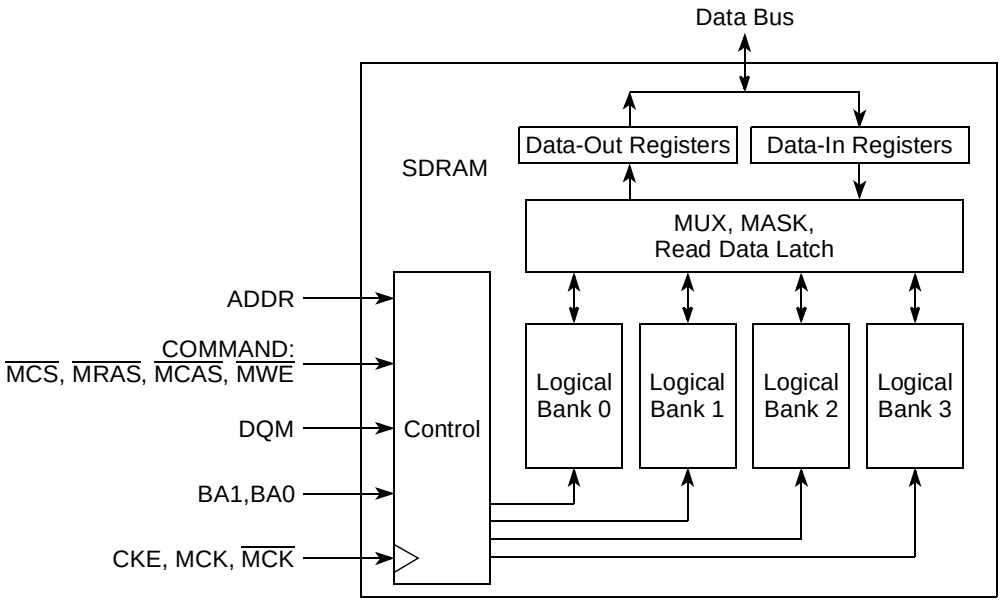


Figure 8-38. Typical Dual Data Rate SDRAM Internal Organization

Figure 8-39 shows some typical signal connections.

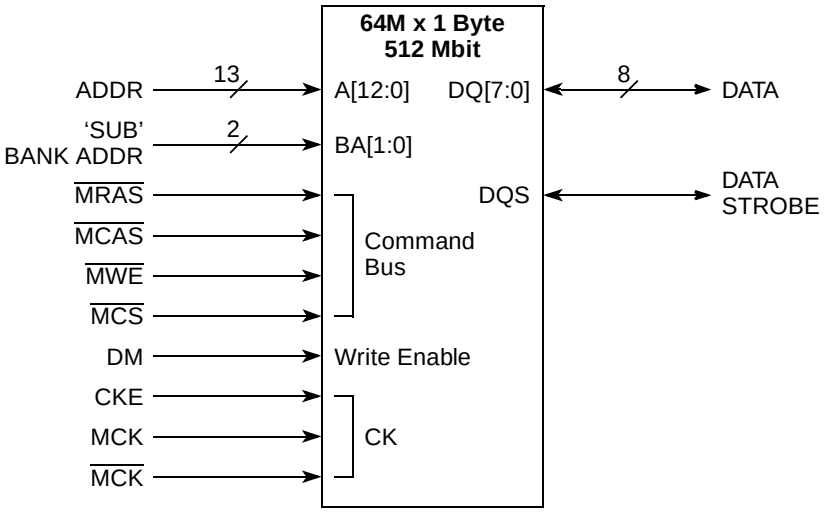
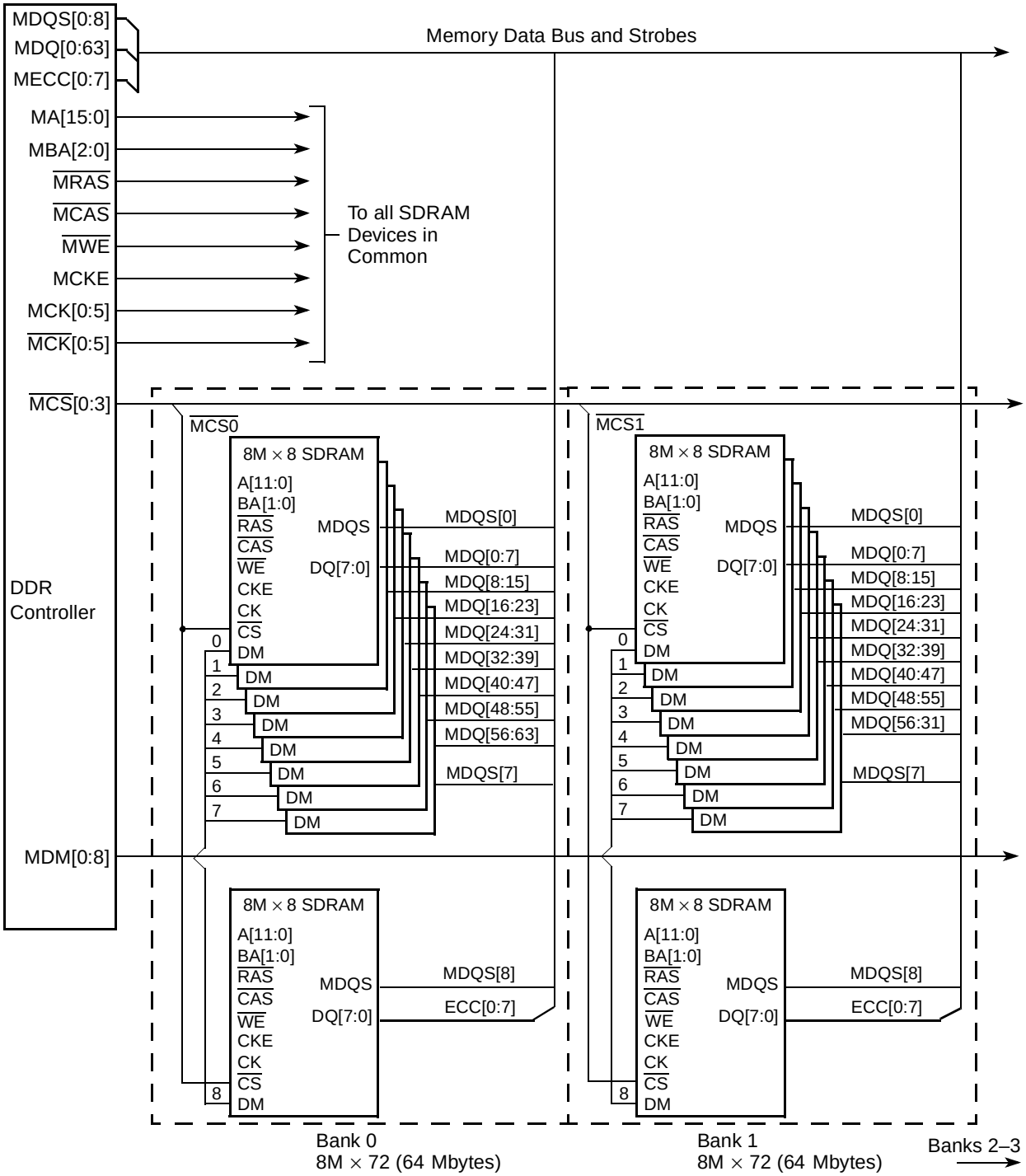


Figure 8-39. Typical DDR SDRAM Interface Signals

Figure 8-40 shows an example DDR SDRAM configuration with four physical banks each comprised of nine 8M × 8 DDR modules for a total of 256 Mbytes of system memory. One of the nine modules is used for the memory's ECC checking function. Certain address and control lines may require buffering. Analysis of the device's AC timing specifications, desired memory operating frequency, capacitive loads, and board routing loads can assist the system designer in deciding signal buffering requirements. The DDR memory controller drives 16 address pins, but in this example the DDR SDRAM devices use only 12 bits.



1. All signals are connected in common (in parallel) except for $\overline{\text{MCS}}$[0:3], MCK[0:5], MDM[0:8], and the data bus signals.
2. Each of the $\overline{\text{MCS}}$[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 8-40. Example 256-Mbyte DDR SDRAM Configuration With ECC

Section 8.5.12, "Error Management," explains how the DDR memory controller handles errors.

8.5.1 DDR SDRAM Interface Operation

The DDR memory controller supports many different DDR SDRAM configurations. SDRAMs with different sizes can be used in the same system. Sixteen multiplexed address signals and three logical bank select signals support device densities from 64 Mbits to 4 Gbits. Four chip select ($\overline{CS}$) signals support up to two DIMMs of memory. The DDR SDRAM physical banks can be built from standard memory modules or directly-attached memory devices. The data path to individual physical banks is 64 or 32 bits wide, 72 or 40 bits with ECC. The DDR memory controller supports physical bank sizes from 16 Mbytes to 4 Gbytes. The physical banks can be constructed using x8, x16, or x32 memory devices. The memory technologies supported are 64 Mbits, 128 Mbits, 256 Mbits, 512 Mbits, 1 Gbit, 2 Gbits, and 4 Gbits. Nine data qualifier (DQM) signals provide byte selection for memory accesses.

NOTE

An 8-bit DDR SDRAM device has a DQM signal and eight data signals (DQ[0:7]). A 16-bit DDR SDRAM device has two DQM signals associated with specific halves of the 16 data signals (DQ[0:7] and DQ[8:15]).

When ECC is enabled, all memory accesses are performed on double-word boundaries (that is, all DQM signals are set simultaneously). However, when ECC is disabled, the memory system uses the DQM signals for byte lane selection.

Table 8-42 shows the DDR memory controller's relationships between data byte lane0–7, MDM[0:7], MDQS[0:7], and MDQ[0:63] when DDR SDRAM memories are used with x8 or x16 devices.

Table 8-42. 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]

8.5.1.1 Supported DDR SDRAM Organizations

Although the DDR memory controller multiplexes row and column address bits onto 16 memory address signals and 3 logical bank select signals, a physical bank may be implemented with memory devices requiring fewer than 31 address bits. The physical bank may be configured to provide from 12 to 16 row address bits, plus 2 or 3 logical bank-select bits and from 8–11 column address bits.

Table 8-44 describe DDR SDRAM device configurations supported by the DDR memory controller.

NOTE

DDR SDRAM is limited to 30 total address bits.

Table 8-43. Supported DDR1 SDRAM Device Configurations

SDRAM Device	Device Configuration	Row x Column x Sub-Bank Bits	64-Bit Bank Size	Four Banks of Memory
64 Mbits	8 Mbits x 8	12 x 9 x 2	64 Mbytes	256 Mbytes
64 Mbits[1]	4 Mbits x 16	12 x 8 x 2	32 Mbytes	128 Mbytes
128 Mbits	16 Mbits x 8	12 x 10 x 2	128 Mbytes	512 Mbytes
128 Mbits	8 Mbits x 16	12 x 9 x 2	64 Mbytes	256 Mbytes
256 Mbits	32 Mbits x 8	13 x 10 x 2	256 Mbytes	1 Gbyte
256 Mbits	16 Mbits x 16	13 x 9 x 2	128 Mbytes	512 Mbytes
512 Mbits	64 Mbits x 8	13 x 11 x 2	512 Mbytes	2 Gbytes
512 Mbits	32 Mbits x 16	13 x 10 x 2	256 Mbytes	1 Gbyte
1 Gbit	128 Mbits x 8	14 x 11 x 2	1 Gbyte	4 Gbytes
1 Gbit	64 Mbits x 16	14 x 10 x 2	512 Mbytes	2 Gbytes
2 Gbits	256 Mbits x 8	15 x 11 x 2	2 Gbytes	8 Gbytes
2 Gbits	128 Mbits x 16	15 x 10 x 2	1 Gbyte	4 Gbytes
4 Gbits	512 Mbits x 8	16 x 11 x 2	4 Gbytes	16 Gbytes
4 Gbits	256 Mbits x 16	16 x 10 x 2	2 Gbytes	8 Gbytes

1 This configuration is not supported in 16-bit bus mode.

Table 8-44. Supported DDR2 SDRAM Device Configurations

SDRAM Device	Device Configuration	Row x Column x Sub-Bank Bits	64-Bit Bank Size	Four Banks of Memory
256 Mbits	32 Mbits x 8	13 x 10 x 2	256 Mbytes	1 Gbyte
256 Mbits	16 Mbits x 16	13 x 9 x 2	128 Mbytes	512 Mbytes
512 Mbits	64 Mbits x 8	14 x 10 x 2	512 Mbytes	2 Gbytes
512 Mbits	32 Mbits x 16	13 x 10 x 2	256 Mbytes	1 Gbyte
1 Gbit	128 Mbits x 8	14 x 10 x 3	1 Gbyte	4 Gbytes
1 Gbit	64 Mbits x 16	13 x 10 x 3	512 Mbytes	2 Gbytes
2 Gbits	256 Mbits x 8	15 x 10 x 3	2 Gbytes	8 Gbytes
2 Gbits	128Mbits x 16	14 x 10 x 3	1 Gbyte	4 Gbytes
4 Gbits	512 Mbits x 8	16 x 10 x 3	4 Gbytes	16 Gbytes
4 Gbits	256 Mbits x 16	15 x 10 x 3	2 Gbytes	8 Gbytes

If a transaction request is issued to the DDR memory controller and the address does not lie within any of the programmed address ranges for an enabled chip select, a memory select error is flagged. Errors are described in detail in Section 8.5.12, “Error Management.”

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.

8.5.2 DDR SDRAM Address Multiplexing

Table 8-45, Table 8-46 Table 8-47, and Table 8-48 show the address bit encodings for each DDR SDRAM configuration. The address presented at the memory controller signals MA[15:0] use MA[15] as the msb and MA[0] as the lsb. Also, MA[10] is used as the auto-precharge bit in DDR1/DDR2 modes for reads and writes, so the column address can never use MA[10].

Table 8-45. DDR1 Address Multiplexing for 64-Bit Data Bus with Interleaving Disabled

Row x Col		msb	Address from Core Master																												lsb
		4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33–35
16 x 11 x 2	$\overline{\text{MRAS}}$	15	14	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	
16 x 10 x 2	$\overline{\text{MRAS}}$		15	14	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	
15 x 11 x 2	$\overline{\text{MRAS}}$		14	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	
15 x 10 x 2	$\overline{\text{MRAS}}$			14	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	
14 x 11 x 2	$\overline{\text{MRAS}}$			13	12	11	10	9	8	7	6	5	4	3	2	1	0														
	MBA																	1	0												
	$\overline{\text{MCAS}}$																			11	9	8	7	6	5	4	3	2	1	0	
14 x 10 x 2	$\overline{\text{MRAS}}$				13	12	11	10	9	8	7	6	5	4	3	2	1	0													
	MBA																		1	0											
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0	
13 x 11 x 2	$\overline{\text{MRAS}}$				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 x 10 x 2	$\overline{\text{MRAS}}$					12	11	10	9	8	7	6	5	4	3	2	1	0													
	MBA																		1	0											
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0	

Table 8-45. DDR1 Address Multiplexing for 64-Bit Data Bus with Interleaving Disabled (continued)

Row x Col		msb	Address from Core Master																												lsb
		4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33–35
13 x 9 x 2	$\overline{\text{MRAS}}$						12	11	10	9	8	7	6	5	4	3	2	1	0												
	MBA																			1	0										
	$\overline{\text{MCAS}}$																					8	7	6	5	4	3	2	1	0	
12 x 10 x 2	$\overline{\text{MRAS}}$						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 x 9 x 2	$\overline{\text{MRAS}}$							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 x 8 x 2	$\overline{\text{MRAS}}$								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	

Table 8-46. DDR1 Address Multiplexing for 32-Bit Data Bus with Interleaving Disabled

Row x Col		msb	Address from Core Master																													lsb
		4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34–35
16 x 11 x 2	$\overline{\text{MRAS}}$		15	14	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	
16 x 10 x 2	$\overline{\text{MRAS}}$			15	14	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	
15 x 11 x 2	$\overline{\text{MRAS}}$			14	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	
15 x 10 x 2	$\overline{\text{MRAS}}$				14	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	
14 x 11 x 2	$\overline{\text{MRAS}}$				13	12	11	10	9	8	7	6	5	4	3	2	1	0														
	MBA																		1	0												
	$\overline{\text{MCAS}}$																				11	9	8	7	6	5	4	3	2	1	0	
14 x 10 x 2	$\overline{\text{MRAS}}$					13	12	11	10	9	8	7	6	5	4	3	2	1	0													
	MBA																			1	0											
	$\overline{\text{MCAS}}$																					9	8	7	6	5	4	3	2	1	0	

Table 8-46. DDR1 Address Multiplexing for 32-Bit Data Bus with Interleaving Disabled (continued)

Row x Col		msb	Address from Core Master																													lsb
		4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34–35
13 x 11 x 2	$\overline{\text{MRAS}}$					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 x 10 x 2	$\overline{\text{MRAS}}$						12	11	10	9	8	7	6	5	4	3	2	1	0													
	MBA																			1	0											
	$\overline{\text{MCAS}}$																					9	8	7	6	5	4	3	2	1	0	
13 x 9 x 2	$\overline{\text{MRAS}}$							12	11	10	9	8	7	6	5	4	3	2	1	0												
	MBA																				1	0										
	$\overline{\text{MCAS}}$																						8	7	6	5	4	3	2	1	0	
12 x 10 x 2	$\overline{\text{MRAS}}$							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 x 9 x 2	$\overline{\text{MRAS}}$								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 x 8 x 2	$\overline{\text{MRAS}}$									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	

Table 8-47. DDR2 Address Multiplexing for 64-Bit Data Bus with Interleaving Disabled

Row x Col		msb	Address from Core Master																													lsb
		4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33–35
16 x 10 x 3	$\overline{\text{MRAS}}$	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0															
	MBA																	2	1	0												
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0		
15 x 10 x 3	$\overline{\text{MRAS}}$		14	13	12	11	10	9	8	7	6	5	4	3	2	1	0															
	MBA																	2	1	0												
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0		
14 x 10 x 3	$\overline{\text{MRAS}}$			13	12	11	10	9	8	7	6	5	4	3	2	1	0															
	MBA																	2	1	0												
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0		
14 x 10 x 2	$\overline{\text{MRAS}}$				13	12	11	10	9	8	7	6	5	4	3	2	1	0														
	MBA																		1	0												
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0		

Table 8-47. DDR2 Address Multiplexing for 64-Bit Data Bus with Interleaving Disabled (continued)

Row x Col		msb	Address from Core Master																												lsb
		4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33–35
13 x 10 x 3	$\overline{\text{MRAS}}$				12	11	10	9	8	7	6	5	4	3	2	1	0														
	MBA																	2	1	0											
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0	
13 x 10 x 2	$\overline{\text{MRAS}}$					12	11	10	9	8	7	6	5	4	3	2	1	0													
	MBA																		1	0											
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0	
13 x 9 x 2	$\overline{\text{MRAS}}$						12	11	10	9	8	7	6	5	4	3	2	1	0												
	MBA																			1	0										
	$\overline{\text{MCAS}}$																					8	7	6	5	4	3	2	1	0	

Table 8-48. DDR2 Address Multiplexing for 32-Bit Data Bus with Interleaving Disabled

Row x Col		msb	Address from Core Master																													lsb
		4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34–35
16 x 10 x 3	$\overline{\text{MRAS}}$		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0														
	MBA																		2	1	0											
	$\overline{\text{MCAS}}$																					9	8	7	6	5	4	3	2	1	0	
15 x 10 x 3	$\overline{\text{MRAS}}$			14	13	12	11	10	9	8	7	6	5	4	3	2	1	0														
	MBA																		2	1	0											
	$\overline{\text{MCAS}}$																					9	8	7	6	5	4	3	2	1	0	
14 x 10 x 3	$\overline{\text{MRAS}}$				13	12	11	10	9	8	7	6	5	4	3	2	1	0														
	MBA																		2	1	0											
	$\overline{\text{MCAS}}$																					9	8	7	6	5	4	3	2	1	0	
14 x 10 x 2	$\overline{\text{MRAS}}$					13	12	11	10	9	8	7	6	5	4	3	2	1	0													
	MBA																			1	0										
	$\overline{\text{MCAS}}$																					9	8	7	6	5	4	3	2	1	0	
13 x 10 x 3	$\overline{\text{MRAS}}$					12	11	10	9	8	7	6	5	4	3	2	1	0														
	MBA																		2	1	0											
	$\overline{\text{MCAS}}$																					9	8	7	6	5	4	3	2	1	0	

Table 8-48. DDR2 Address Multiplexing for 32-Bit Data Bus with Interleaving Disabled (continued)

Row x Col		msb	Address from Core Master																												lsb
		4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34–35
13 x 10 x 2	$\overline{\text{MRAS}}$						12	11	10	9	8	7	6	5	4	3	2	1	0													
	MBA																			1	0										
	$\overline{\text{MCAS}}$																					9	8	7	6	5	4	3	2	1	0	
13 x 9 x 2	$\overline{\text{MRAS}}$							12	11	10	9	8	7	6	5	4	3	2	1	0												
	MBA																				1	0										
	$\overline{\text{MCAS}}$																						8	7	6	5	4	3	2	1	0	

Chip select interleaving is supported for the memory controller, and is programmed in DDR_SDRAM_CFG[BA_INTLV_CTL]. Interleaving is supported between chip selects 0 and 1 or chip selects 2 and 3. In addition, interleaving between all four chip selects can be enabled. When interleaving is enabled, the chip selects being interleaved must use the same size of memory. If two chip selects are interleaved, then 1 extra bit in the address decode is used for the interleaving to determine which chip select to access. If four chip selects are interleaved, then two extra bits are required in the address decode.

Table 8-49 illustrates examples of address decode when interleaving between two chip selects, and Table 8-50 shows examples of address decode when interleaving between four chip selects.

Table 8-49. Example of Address Multiplexing for 64-Bit Data Bus Interleaving between Two Banks

Row x Col		msb	Address from Core Master																													lsb
		4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33–35
14 x 10 x 3	$\overline{\text{MRAS}}$		13	12	11	10	9	8	7	6	5	4	3	2	1	0															
	MBA																CS SEL	2	1	0											
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0	
14 x 10 x 2	$\overline{\text{MRAS}}$			13	12	11	10	9	8	7	6	5	4	3	2	1	0														
	MBA																	CS SEL	1	0											
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0	
13 x 10 x 3	$\overline{\text{MRAS}}$			12	11	10	9	8	7	6	5	4	3	2	1	0															
	MBA																CS SEL	2	1	0											
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0	
13 x 10 x 2	$\overline{\text{MRAS}}$				12	11	10	9	8	7	6	5	4	3	2	1	0														
	MBA																	CS SEL	1	0											
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0	

Table 8-50. Example of Address Multiplexing for 64-Bit Data Bus Interleaving between Four Banks

Row x Col		msb	Address from Core Master																												lsb
		4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33–35
14 x 10 x 3	$\overline{\text{MRAS}}$	13	12	11	10	9	8	7	6	5	4	3	2	1	0	CS SEL															
	MBA																	2	1	0											
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0	
14 x 10 x 2	$\overline{\text{MRAS}}$		13	12	11	10	9	8	7	6	5	4	3	2	1	0	CS SEL														
	MBA																		1	0											
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0	
13 x 10 x 3	$\overline{\text{MRAS}}$		12	11	10	9	8	7	6	5	4	3	2	1	0	CS SEL															
	MBA																	2	1	0											
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0	
13 x 10 x 2	$\overline{\text{MRAS}}$			12	11	10	9	8	7	6	5	4	3	2	1	0	CS SEL														
	MBA																		1	0											
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0	

8.5.3 JEDEC Standard DDR SDRAM Interface Commands

The following section describes the commands and timings the controller uses when operating in DDR2 or DDR modes.

All read or write accesses to DDR SDRAM are performed by the DDR memory controller using JEDEC standard DDR SDRAM interface commands. The SDRAM device samples command and address inputs on rising edges of the memory clock; data is sampled using both the rising and falling edges of DQS. Data read from the DDR SDRAM is also sampled on both edges of DQS.

The following DDR SDRAM interface commands (summarized in Table 8-51) are provided by the DDR controller. All actions for these commands are described from the perspective of the SDRAM device.

- Row activate—Latches row address and initiates memory read of that row. Row data is latched in SDRAM sense amplifiers and must be restored by a precharge command before another row activate occurs.
- Precharge—Restores data from the sense amplifiers to the appropriate row. Also initializes the sense amplifiers in preparation for reading another row in the memory array, (performing another activate command). Precharge must occur after read or write, if the row address changes on the next open page mode access.
- Read—Latches column address and transfers data from the selected sense amplifier to the output buffer as determined by the column address. During each succeeding clock edge, additional data is driven without additional read commands. The amount of data transferred is determined by the burst size which defaults to 4.

- Write—Latches column address and transfers data from the data pins to the selected sense amplifier as determined by the column address. During each succeeding clock edge, additional data is transferred to the sense amplifiers from the data pins without additional write commands. The amount of data transferred is determined by the data masks and the burst size, which is set to four by the DDR memory controller.
- Refresh (similar to $\overline{\text{MCAS}}$ before $\overline{\text{MRAS}}$)—Causes a row to be read in all logical banks (JEDEC SDRAM) as determined by the refresh row address counter. This refresh row address counter is internal to the SDRAM. After being read, the row is automatically rewritten in the memory array. All logical banks must be in a precharged state before executing a refresh. The memory controller also supports posted refreshes, where several refreshes may be executed at once, and the refresh interval may be extended.
- Mode register set (for configuration)—Allows setting of DDR SDRAM options. These options are: $\overline{\text{MCAS}}$ latency, additive latency (for DDR2), write recovery (for DDR2), burst type, and burst length. $\overline{\text{MCAS}}$ latency may be chosen as provided by the preferred SDRAM (some SDRAMs provide $\overline{\text{MCAS}}$ latency {1,2,3}, some provide $\overline{\text{MCAS}}$ latency {1,2,3,4,5}, 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. A burst length of 8 is supported for DDR1memory only. For DDR2 in 32-bit bus mode, all 32-byte burst accesses from the platform are split into two 16-byte (that is, 4 beat) accesses to the SDRAMs in the memory controller. The mode register set command is performed by the DDR memory controller during system initialization. Parameters such as mode register data, $\overline{\text{MCAS}}$ latency, burst length, and burst type, are set by software in DDR_SDRAM_MODE[SDMODE] and transferred to the SDRAM array by the DDR memory controller after DDR_SDRAM_CFG[MEM_EN] is set. If DDR_SDRAM_CFG[BI] is set to bypass the automatic initialization, then the MODE registers can be configured through software through use of the DDR_SDRAM_MD_CNTL register.
- Self refresh (for long periods of standby)—Used when the device is in standby for very long periods of time. Automatically generates internal refresh cycles to keep the data in all memory banks refreshed. Before execution of this command, the DDR controller places all logical banks in a precharged state.

Table 8-51. DDR SDRAM Command Table

Operation	CKE Prev.	CKE Current	$\overline{\text{MCS}}$	$\overline{\text{MRAS}}$	$\overline{\text{MCAS}}$	$\overline{\text{MWE}}$	MBA	MA10	MA
Activate	H	H	L	L	H	H	Logical bank select	Row	Row
Precharge select logical bank	H	H	L	L	H	L	Logical bank select	L	X
Precharge all logical banks	H	H	L	L	H	L	X	H	X
Read	H	H	L	H	L	H	Logical bank select	L	Column
Read with auto-precharge	H	H	L	H	L	H	Logical bank select	H	Column
Write	H	H	L	H	L	L	Logical bank select	L	Column

Table 8-51. DDR SDRAM Command Table (continued)

Operation	CKE Prev.	CKE Current	$\overline{\text{MCS}}$	$\overline{\text{MRAS}}$	$\overline{\text{MCAS}}$	$\overline{\text{MWE}}$	MBA	MA10	MA
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

8.5.4 DDR SDRAM Interface Timing

The DDR memory controller supports four-beat bursts to SDRAM. For single-beat reads, the DDR memory controller performs a four- (or eight-) beat burst read, but ignores the last three (or seven) beats. Single-beat writes are performed by masking the last three (or seven) beats of the four- (or eight-) beat burst using the data mask MDM[0:8]. If ECC is disabled, writes smaller than double words are performed by appropriately activating the data mask. If ECC is enabled, the controller performs a read-modify write.

NOTE

If a second read or write is pending, reads shorter than four beats are not terminated early even if some data is irrelevant.

To accommodate available memory technologies across a wide spectrum of operating frequencies, the DDR memory controller allows the setting of the intervals defined in Table 8-52 with granularity of one memory clock cycle, except for CASLAT, which can be programmed with 1/2 clock granularity.

Table 8-52. DDR SDRAM Interface Timing Intervals

Timing Intervals	Definition
ACTTOACT	The number of clock cycles from a bank-activate command until another bank-activate command within a physical bank. This interval is listed in the AC specifications of the SDRAM as t_{RRD}.
ACTTOPRE	The number of clock cycles from an activate command until a precharge command is allowed. This interval is listed in the AC specifications of the SDRAM as t_{RAS}.
ACTTORW	The number of clock cycles from an activate command until a read or write command is allowed. This interval is listed in the AC specifications of the SDRAM as t_{RCD}.
BSTOPRE	The number of clock cycles to maintain a page open after an access. The page open duration counter is reloaded with BSTOPRE each time the page is accessed (including page hits). When the counter expires, the open page is closed with a SDRAM precharge bank command as soon as possible.
CASLAT	Used in conjunction with additive latency to obtain the READ latency. The number of clock cycles between the registration of a READ command by the SDRAM and the availability of the first piece of output data. If a READ command is registered at clock edge n, and the read latency is m clocks, the data is available nominally coincident with clock edge $n + m$.
PRETOACT	The number of clock cycles from a precharge command until an activate or a refresh command is allowed. This interval is listed in the AC specifications of the SDRAM as t_{RP}

Table 8-52. DDR SDRAM Interface Timing Intervals (continued)

Timing Intervals	Definition
REFINT	Refresh interval. Represents the number of memory bus clock cycles between refresh cycles. Depending on DDR_SDRAM_CFG_2[NUM_PR], some number of rows are refreshed in each SDRAM bank during each refresh cycle. The value of REFINT depends on the specific SDRAMs used and the frequency of the interface as t_{RP}.
REFREC	The number of clock cycles from the refresh command until an activate command is allowed. This can be calculated by referring to the AC specification of the SDRAM device. The AC specification indicates a maximum refresh to activate interval in nanoseconds.
WR_DATA_DELAY	Provides different options for the timing between a write command and the write data strobe. This allows write data to be sent later than the nominal time to meet the SDRAM timing requirement between the registration of a write command and the reception of a data strobe associated with the write command. The specification dictates that the data strobe may not be received earlier than 75% of a cycle, or later than 125% of a cycle, from the registration of a write command. This parameter is not defined in the SDRAM specification. It is implementation-specific, defined for the DDR memory controller in TIMING_CFG_2.
WRREC	The number of clock cycles from the last beat of a write until a precharge command is allowed. This interval, write recovery time, is listed in the AC specifications of the SDRAM as t_{WR}.
WRTORD	Last write pair to read command. Controls the number of clock cycles from the last write data pair to the subsequent read command to the same bank as t_{WTR}.

The value of the above parameters (in whole clock cycles) must be set by boot code at system start-up (in the TIMING_CFG_0, TIMING_CFG_1, TIMING_CFG_2, and TIMING_CFG_3 registers as described in Section 8.4.1.4, “DDR SDRAM Timing Configuration 0 (TIMING_CFG_0),” Section 8.4.1.5, “DDR SDRAM Timing Configuration 1 (TIMING_CFG_1),” Section 8.4.1.6, “DDR SDRAM Timing Configuration 2 (TIMING_CFG_2),” and Section 8.4.1.3, “DDR SDRAM Timing Configuration 3 (TIMING_CFG_3),”) and be kept in the DDR memory controller configuration register space.

The following figures show SDRAM timing for various types of accesses. System software is responsible (at reset) for optimally configuring SDRAM timing parameters. The programmable timing parameters apply to both read and write timing configuration. The configuration process must be completed and the DDR SDRAM initialized before any accesses to SDRAM are attempted.

Figure 8-41 through Figure 8-44 show DDR SDRAM timing for various types of accesses; see Figure 8-41 for a single-beat read operation, Figure 8-42 for a single-beat write operation, and Figure 8-44 for a burst-write operation. Note that all signal transitions occur on the rising edge of the memory bus clock and that single-beat read operations are identical to burst-reads. These figures assume the CLK_ADJUST is set to 1/2 DRAM cycle, an additive latency of 0 DRAM cycles is used, and the write latency is 1 DRAM cycle (for DDR1).

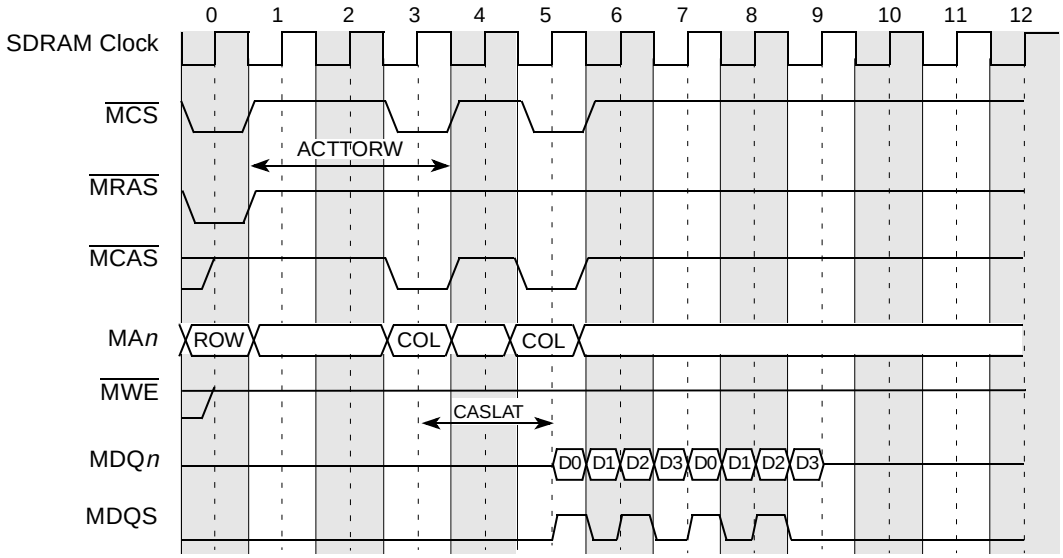


Figure 8-41. DDR SDRAM Burst Read Timing—ACTTORW = 3, $\overline{\text{MCAS}}$ Latency = 2

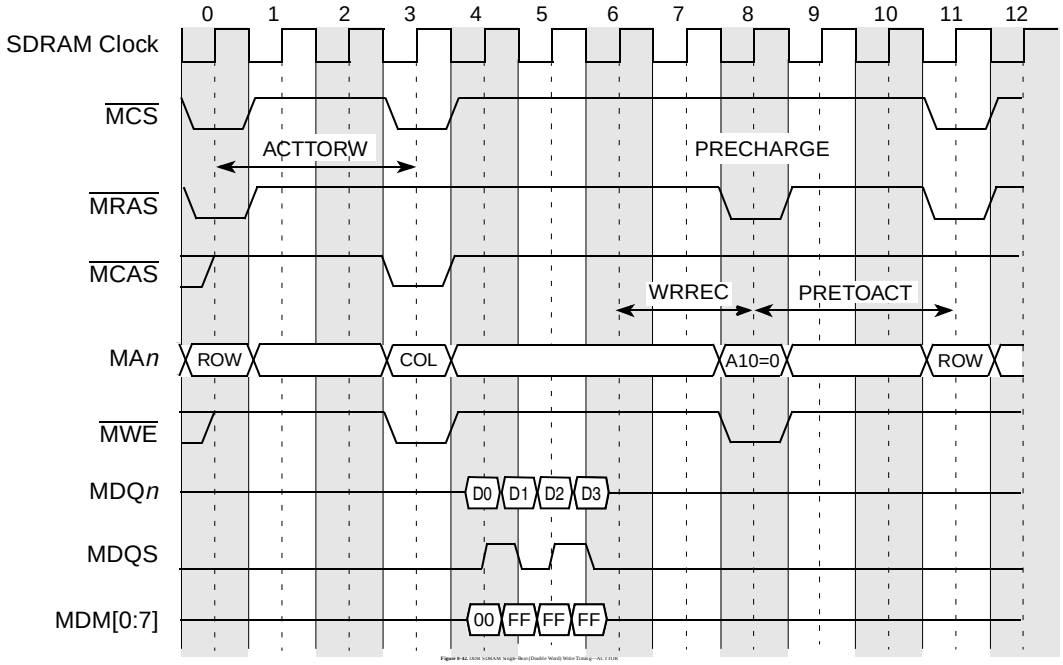


Figure 8-43. DDR SDRAM Single-Beat (Double Word) Write Timing—ACTTORW = 3

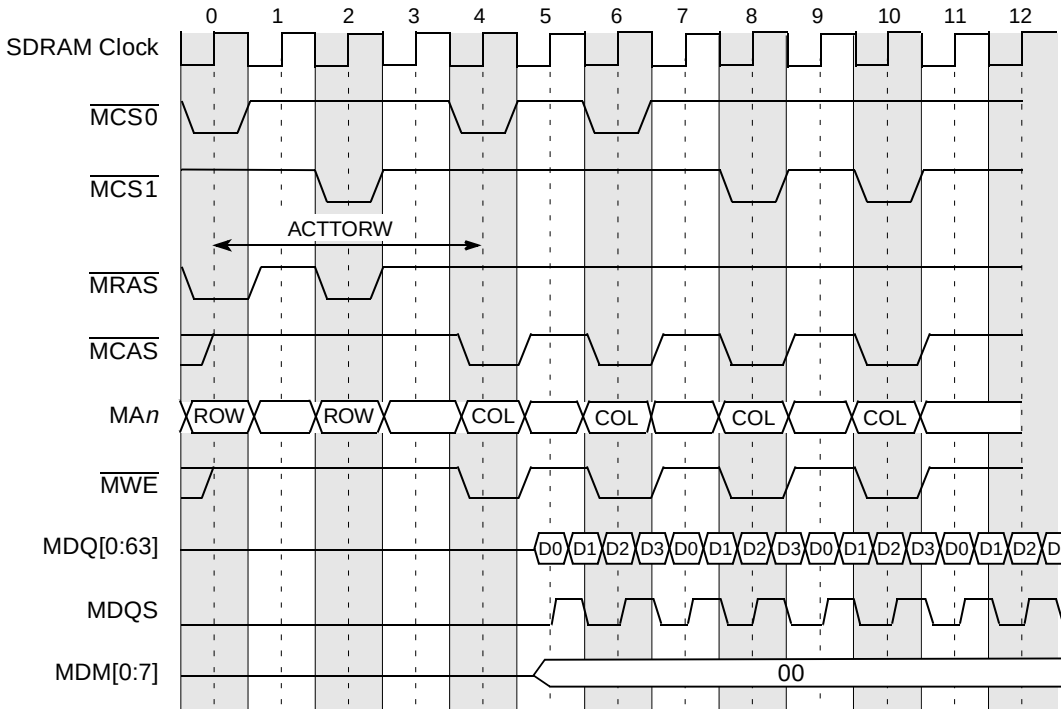


Figure 8-44. DDR SDRAM 4-Beat Burst Write Timing—ACTTORW = 4

8.5.4.1 Clock Distribution

- If running with many devices, zero-delay PLL clock buffers, JEDEC-JESD82 standard, should be used. These buffers were designed for DDR applications.
- A 72 bit x 64 Mbytes DDR bank has 9-byte-wide DDR chips, resulting in 18 DDR chips in a two-bank system. In this case, each MCK/$\overline{\text{MCK}}$ signal pair should drive exactly three devices.
- PCB traces for DDR clock signals should be short, all on the same layer, and of equal length and loading.
- DDR SDRAM manufacturers provide detailed information on PCB layout and termination issues.

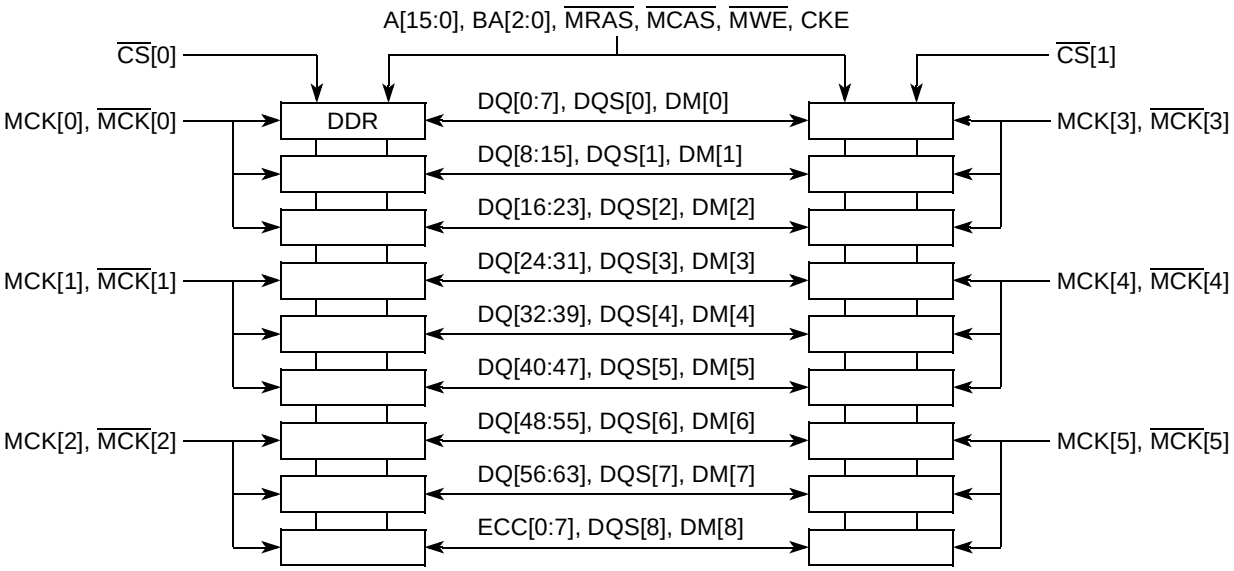


Figure 8-45. DDR SDRAM Clock Distribution Example for x8 DDR SDRAMs

8.5.5 DDR SDRAM Mode-Set Command Timing

The DDR memory controller transfers the mode register set commands to the SDRAM array, and it uses the setting of TIMING_CFG_0[MRS_CYC] for the Mode Register Set cycle time.

Figure 8-46 shows the timing of the mode-set command. The first transfer corresponds to the ESDMODE code; the second corresponds to SDMODE. The Mode Register Set cycle time is set to 2 DRAM cycles.

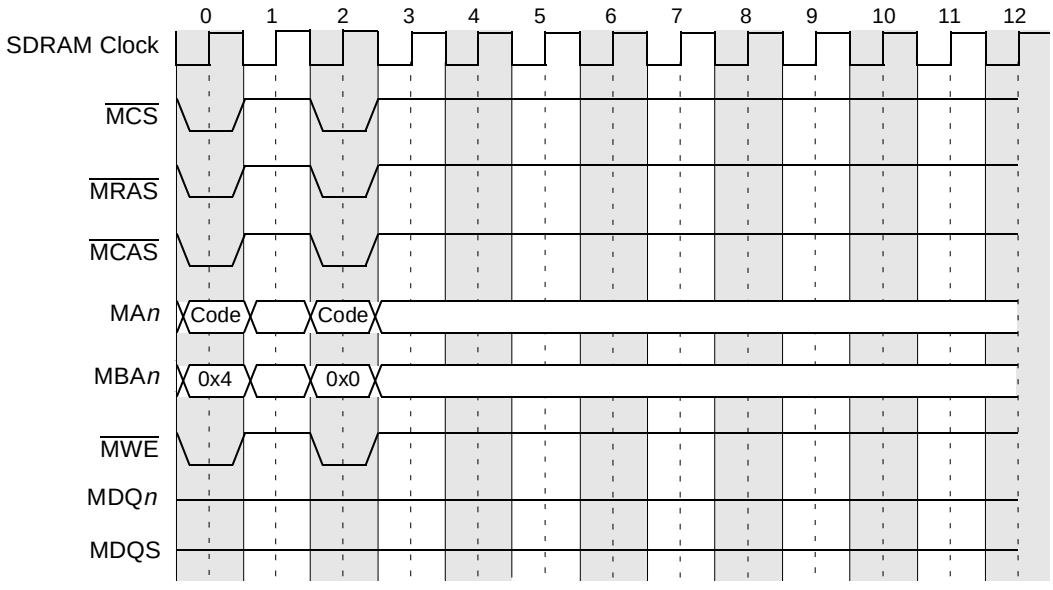


Figure 8-46. DDR SDRAM Mode-Set Command Timing

8.5.6 DDR SDRAM Registered DIMM Mode

To reduce loading, registered DIMMs latch the DDR SDRAM control signals internally before using them to access the array. Setting DDR_SDRAM_CFG[RD_EN] compensates for this delay on the DIMMs' control bus by delaying the data and data mask writes (on SDRAM buses) by an extra SDRAM clock cycle.

NOTE

Application system board must assert the reset signal on DDR memory devices until software is able to program the DDR memory controller configuration registers, and must deassert the reset signal on DDR memory devices before DDR_SDRAM_CFG[MEM_EN] is set. This ensures that the DDR memory devices are held in reset until a stable clock is provided and, further, that a stable clock is provided before memory devices are released from reset.

Figure 8-47 shows the registered DDR SDRAM DIMM single-beat write timing.

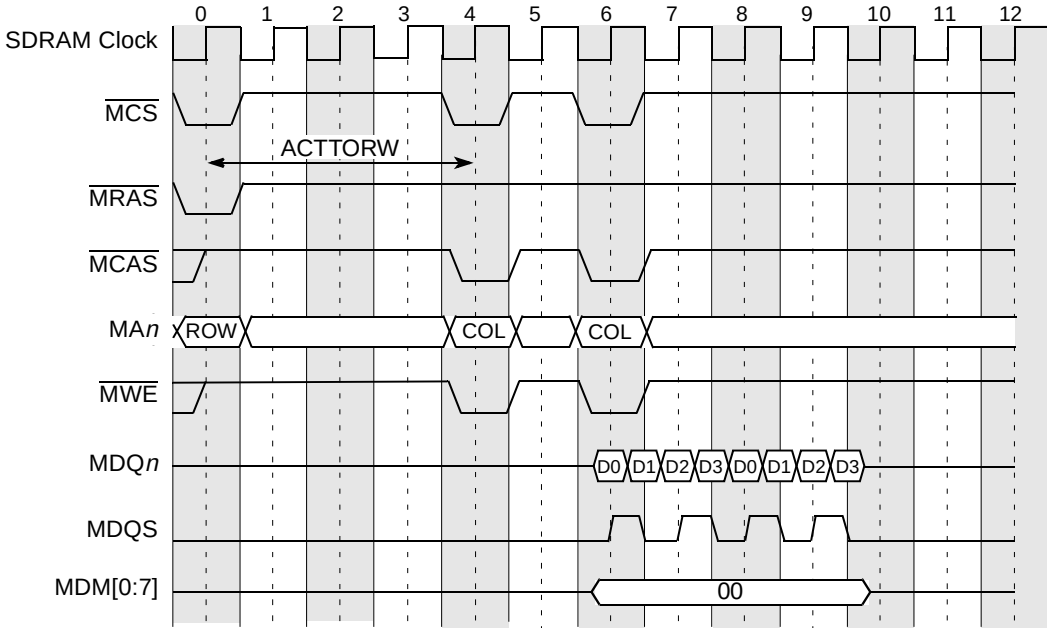


Figure 8-47. Registered DDR SDRAM DIMM Burst Write Timing

8.5.7 DDR SDRAM Write Timing Adjustments

The DDR memory controller facilitates system design flexibility by providing a write timing adjustment parameter, write data delay, (TIMING_CFG_2[WR_DATA_DELAY]) for data and DQS. The DDR SDRAM specification requires DQS be received no sooner than 75% of an SDRAM clock period—and no later than 125% of a clock period—from the capturing clock edge of the command/address at the SDRAM. The WR_DATA_DELAY parameter may be used to meet this timing requirement for a variety of system configurations, ranging from a system with one DIMM to a fully populated system with two DIMM. TIMING_CFG_2[WR_DATA_DELAY] specifies how much to delay the launching of DQS and

data from the first clock edge occurring one SDRAM clock cycle after the command is launched. The delay increment step sizes are in 1/4 SDRAM clock periods starting with the default value of 0.

Figure 8-48 shows the use of the WR_DATA_DELAY parameter.

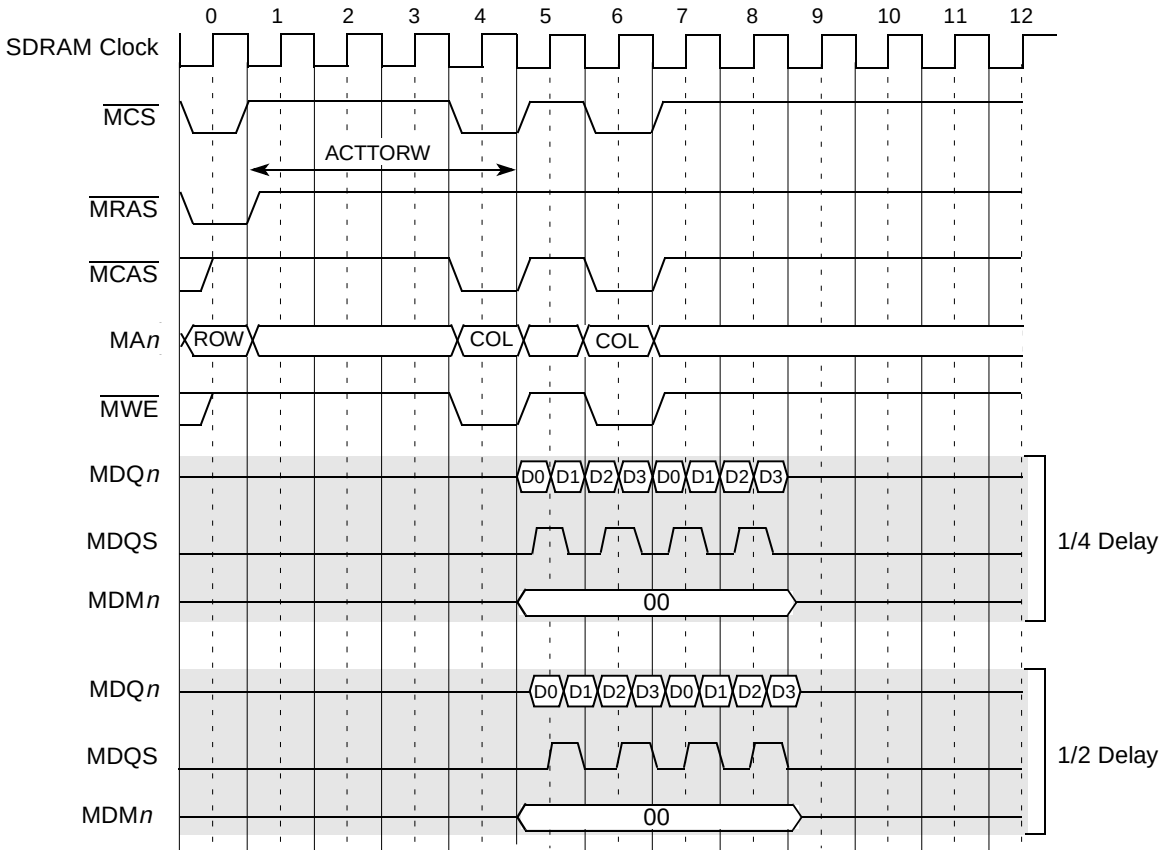


Figure 8-48. Write Timing Adjustments Example for Write Latency = 1

8.5.8 DDR SDRAM Refresh

The DDR memory controller supports auto-refresh and self-refresh. Auto refresh is used during normal operation and is controlled by the DDR_SDRAM_INTERVAL[REFINT] value; self-refresh is used only when the DDR memory controller is set to enter a sleep power management state. The REFINT value, which represents the number of memory bus clock cycles between refresh cycles, must allow for possible outstanding transactions to complete before a refresh request is sent to the memory after the REFINT value is reached. If a memory transaction is in progress when the refresh interval is reached, the refresh cycle waits for the transaction to complete. In the worst case, the refresh cycle must wait the number of bus clock cycles required by the longest programmed access. To ensure that the latency caused by a memory transaction does not violate the device refresh period, it is recommended that the programmed value of REFINT be less than that required by the SDRAM.

When a refresh cycle is required, the DDR memory controller does the following:

1. Completes all current memory requests.

2. Closes all open pages with a PRECHARGE-ALL command to each DDR SDRAM bank with an open page (as indicated by the row open table).
3. Issues one or more auto-refresh commands to each DDR SDRAM bank (as identified by its chip select) to refresh one row in each logical bank of the selected physical bank.

The auto-refresh commands are staggered across the four possible banks to reduce the system's instantaneous power requirements. Three sets of auto refresh commands are issued on consecutive cycles when the memory is fully populated with two DIMMs. The initial PRECHARGE-ALL commands are also staggered in three groups for convenience. It is important to note that when entering self-refresh mode, only one refresh command is issued simultaneously to all physical banks. For this entire refresh sequence, no cycle optimization occurs for the usual case where fewer than four banks are installed. After the refresh sequence completes, any pending memory request is initiated after an inactive period specified by TIMING_CFG_1 [REFREC] and TIMING_CFG_3[EXT_REFREC]. In addition, posted refreshes are supported to allow the refresh interval to be set to a larger value.

8.5.8.1 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 8-49 (TIMING_CFG_1 [REFREC] = 10 in this example).

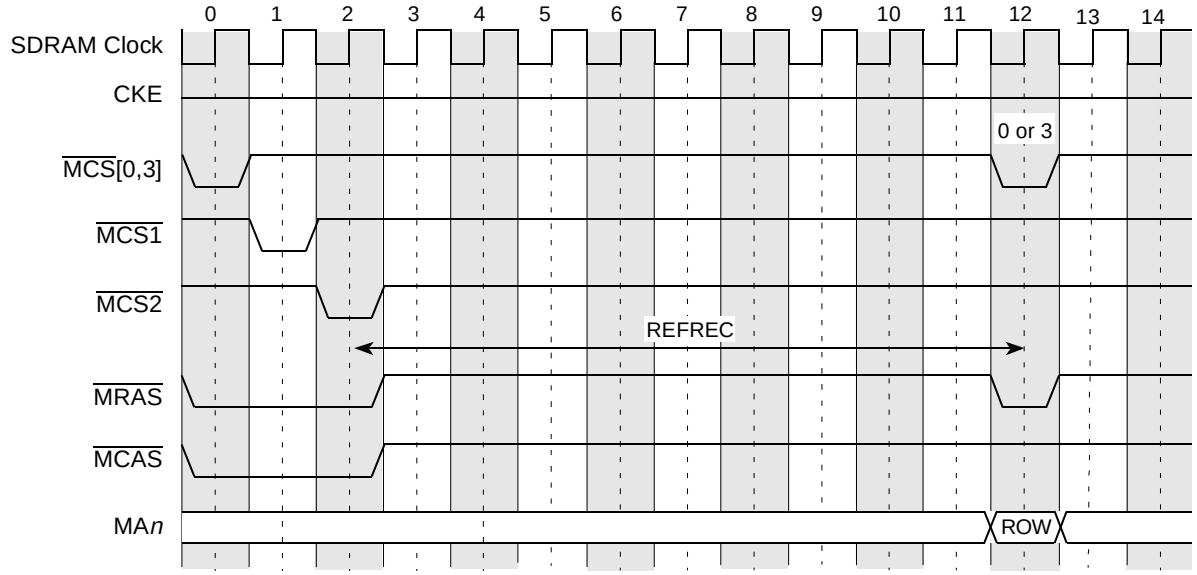


Figure 8-49. DDR SDRAM Bank Staggered Auto Refresh Timing

System software is responsible for optimal configuration of TIMING_CFG_1 [REFREC] and TIMING_CFG_3[EXT_REFREC] at reset. Configuration must be completed before DDR SDRAM accesses are attempted.

8.5.8.2 DDR SDRAM Refresh and Power-Saving Modes

In full-on mode, the DDR memory controller supplies the normal auto refresh to SDRAM. In sleep mode, the DDR memory controller can be configured to take advantage of self-refreshing SDRAMs or to provide no refresh support. Self-refresh support is enabled with the SREN memory control parameter.

Table 8-53 summarizes the refresh types available in each power-saving mode.

Table 8-53. DDR SDRAM Power-Saving Modes Refresh Configuration

Power Saving Mode	Refresh Type	SREN
Sleep	Self	1
	None	—

Note that in the absence of refresh support, system software must preserve DDR SDRAM data (such as by copying the data to disk) before entering the power-saving mode.

The dynamic power-saving mode uses the CKE DDR SDRAM pin to dynamically power down when there is no system memory activity. The CKE pin is negated when both of the following conditions are met:

- No memory refreshes are scheduled
- No memory accesses are scheduled

CKE is reasserted when a new access or refresh is scheduled or the dynamic power mode is disabled. This mode is controlled with DDR_SDRAM_CFG[DYN_PWR_MGMT].

Dynamic power management mode offers tight control of the memory system's power consumption by trading power for performance through the use of CKE. Powering up the DDR SDRAM when a new memory reference is scheduled causes an access latency penalty, depending on whether active or precharge powerdown is used, along with the settings of TIMING_CFG_0[ACT_PD_EXIT] and TIMING_CFG_0[PRE_PD_EXIT]. A penalty of 1 cycle is shown in Figure 8-50.

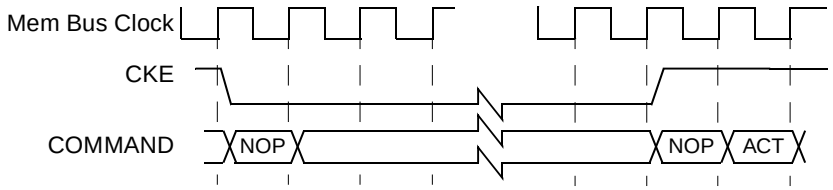


Figure 8-50. DDR SDRAM Power-Down Mode

8.5.8.2.1 Self-Refresh in Sleep Mode

The entry and exit timing for self-refreshing SDRAMs is shown in Figure 8-51 and Figure 8-52.

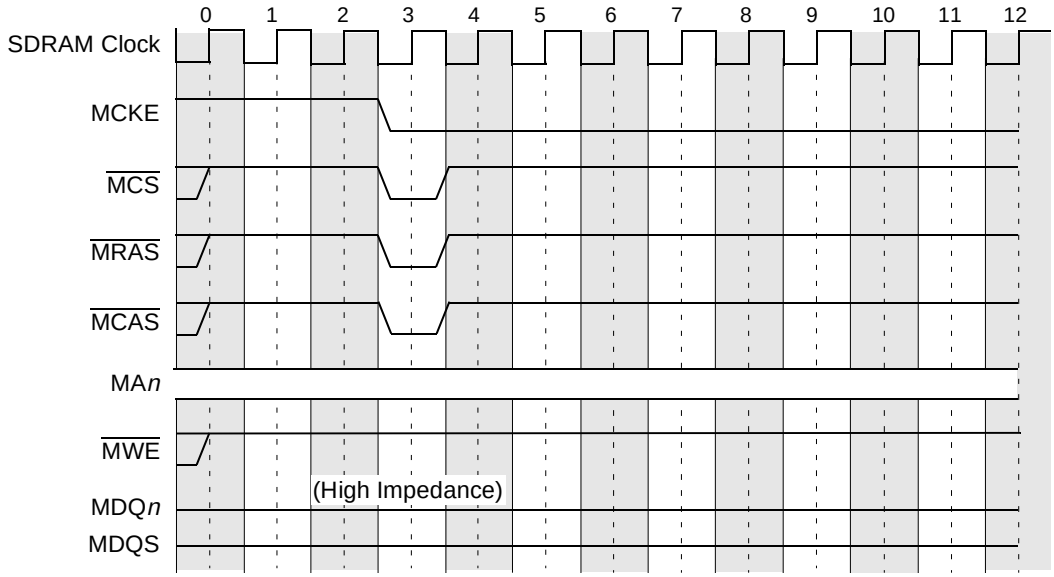


Figure 8-51. DDR SDRAM Self-Refresh Entry Timing

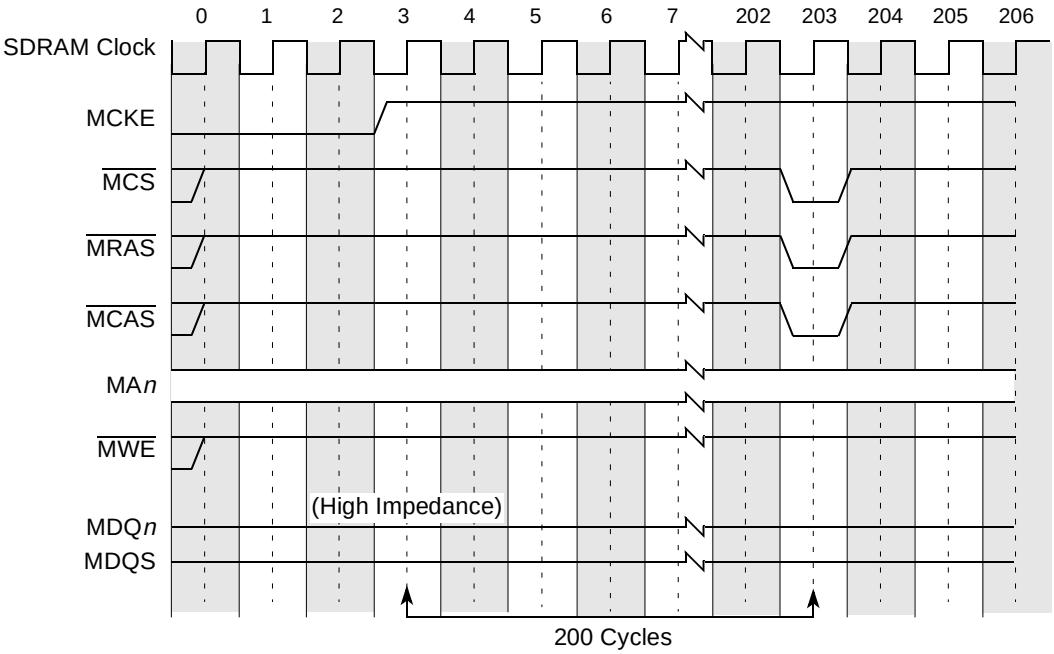


Figure 8-52. DDR SDRAM Self-Refresh Exit Timing

8.5.9 DDR Data Beat Ordering

Transfers to and from memory are always performed in four- or eight-beat bursts (four beats = 32 bytes when a 64-bit bus is used). For transfer sizes other than four or eight beats, the data transfers are still operated as four- or eight-beat bursts. If ECC is enabled and either the access is not doubleword aligned or the size is not a multiple of a doubleword, a full read-modify-write is performed for a write to SDRAM. If ECC is disabled or both the access is doubleword aligned with a size that is a multiple of a doubleword, the data masks (MDM[0:8] (MDM[0:4] for 32-bit bus) can be used to prevent the writing of unwanted data to SDRAM. The DDR memory controller also uses data masks to prevent all unintended full double words from writing to SDRAM. For example, if a write transaction is desired with a size of one double word (8 bytes), then the second, third, and fourth beats of data are not written to DRAM.

Table 8-54 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 8-54. Memory Controller–Data Beat Ordering

Transfer Size	Starting Double-Word Offset	Double-Word Sequence[1] to/from DRAM and Queues
1 double word	0 1 2 3	**0** - 1 - 2 - 3 **1** - 2 - 3 - 0 **2** - 3 - 0 - 1 **3** - 0 - 1 - 2
2 double words	0 1 2	**0 - 1** - 2 - 3 **1 - 2** - 3 - 0 **2 - 3** - 0 - 1
3 double words	0 1	**0 - 1 - 2** - 3 **1 - 2 - 3** - 0

[1] All underlined **Double**-word offsets are valid for the transaction.

8.5.10 Page Mode and Logical Bank Retention

The DDR memory controller supports an open/closed page mode with an allowable open page for each logical bank of DRAM used. In closed page mode for DDR SDRAMs, the DDR memory controller uses the SDRAM auto-precharge feature, which allows the controller to indicate that the page must be automatically closed by the DDR SDRAM after the READ or WRITE access. This is performed 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 using more banks, especially in systems which use many different channels. Page mode is disabled by clearing DDR_SDRAM_INTERVAL[BSTOPRE] or setting CSn_CONFIG[AP_nEN].

8.5.11 Error Checking and Correcting (ECC)

The DDR memory controller supports error checking and correcting (ECC) for the data path between the core master and system memory. The memory detects all double-bit errors, detects all multi-bit errors within a nibble, and corrects all single-bit errors. Other errors may be detected, but are not guaranteed to be corrected or detected. Multiple-bit errors are always reported when error reporting is enabled. When a single-bit error occurs, the single-bit error counter register is incremented, and its value compared to the single-bit error trigger register. An error is reported when these values are equal. The single-bit error registers can be programmed such that minor memory faults are corrected and ignored, but a catastrophic memory failure generates an interrupt.

For writes that are smaller than 64 bits, the DDR memory controller performs a double-word read from system memory of the address for the write (checking for errors), and merges the write data with the data read from memory. Then, a new ECC code is generated for the merged double word. The data and ECC code is then written to memory. If a multi-bit error is detected on the read, the transaction completes the read-modify-write to keep the DDR memory controller from hanging. However, the corrupt data is masked on the write, so the original contents in SDRAM remain unchanged.

The syndrome encodings for the ECC code are shown in Table 8-55 and Table 8-56.

In 32-bit mode, Table 8-55 is split into 2 halves. The first half, consisting of rows 0–31, is used to calculate the ECC bits for the first 32 data bits of any 64-bit granule of data. This always applies to the odd data beats on the DDR data bus. The second half of the table, consisting of rows 32–63, is used to calculate the ECC bits for the second 32 bits of any 64-bit granule of data. This always applies to the even data beats on the DDR data bus.

Table 8-55. DDR SDRAM ECC Syndrome Encoding

Data Bit	Syndrome Bit							
	0	1	2	3	4	5	6	7
0	•	•						•
1	•		•					•
2	•			•				•
3	•				•			•
4	•	•				•		
5	•		•			•		
6	•			•		•		
7	•				•	•		
8	•	•					•	

Data Bit	Syndrome Bit							
	0	1	2	3	4	5	6	7
32			•	•				•
33			•		•			•
34	•		•		•			
35		•	•		•			
36			•	•		•		
37			•		•	•		
38	•		•		•	•		•
39		•	•		•	•		•
40			•	•			•	

Table 8-55. 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
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 8-56. 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								•

8.5.12 Error Management

The DDR memory controller detects four different kinds of errors: training, single-bit, multi-bit, and memory select errors. The following discussion assumes all the relevant error detection, correction, and reporting functions are enabled as described in Section 8.4.1.31, “Memory Error Interrupt Enable (ERR_INT_EN),” Section 8.4.1.30, “Memory Error Disable (ERR_DISABLE),” and Section 8.4.1.29, “Memory Error Detect (ERR_DETECT).”

Single-bit errors are counted and reported based on the ERR_SBE value. When a single-bit error is detected, the DDR memory controller does the following:

- Corrects the data
- Increments the single-bit error counter ERR_SBE[SBEC]
- Generates a critical interrupt if the counter value ERR_SBE[SBEC] equals the programmable threshold ERR_SBE[SBET]
- Completes the transaction normally

If a multi-bit error is detected for a read, the DDR memory controller logs the error and generates the interrupt (if enabled, as described in Section 8.4.1.30, “Memory Error Disable (ERR_DISABLE)”). Another error the DDR memory controller detects is a memory select error, which causes the DDR memory controller to log the error and generate a critical interrupt (if enabled, as described in Section 8.4.1.29, “Memory Error Detect (ERR_DETECT)”). This error is detected if the address from the memory request does not fall into any of the enabled, programmed chip select address ranges. For all memory select errors, the DDR memory controller does not issue any transactions onto the pins after the first read has returned data strobes. If the DDR memory controller is not using sample points, then a dummy transaction is issued to DDR SDRAM with the first enabled chip select. In this case, the source port on the pins is forced to 0x1F to show the transaction is not real. Table 8-57 shows the errors with their descriptions. The final error the memory controller detects is the automatic calibration error. This error is set if the memory controller detects an error during its training sequence.

Table 8-57. 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 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.		

8.6 Initialization/Application Information

System software must configure the DDR memory controller, using a memory polling algorithm at system start-up, to correctly map the size of each bank in memory. Then, the DDR memory controller uses its bank map to assert the appropriate $\overline{\text{MCS}n}$ signal for memory accesses according to the provided bank depths. System software must also configure the DDR memory controller at system start-up to appropriately multiplex the row and column address bits for each bank. Refer to row-address configuration in Section 8.4.1.2, “Chip Select Configuration (CSn_CONFIG).” Address multiplexing occurs according to these configuration bits.

At system reset, initialization software (boot code) must set up the programmable parameters in the memory interface configuration registers. See Section 8.4.1, “Register Descriptions,” for more detailed descriptions of the configuration registers. These parameters are shown in Table 8-58.

Table 8-58. Memory Interface Configuration Register Initialization Parameters

Name	Description	Parameter		Section/Page
CSn_BNDS	Chip select memory bounds	SAn EAn		8.4.1.1/8-12
CSn_CONFIG	Chip select configuration	CS_n_EN AP_n_EN ODT_RD_CFG ODT_WR_CFG	BA_BITS_CS_n ROW_BITS_CS_n COL_BITS_CS_n	8.4.1.2/8-13
TIMING_CFG_3	Extended timing parameters for fields in TIMING_CFG_1	EXT_REFREC		8.4.1.3/8-15
TIMING_CFG_0	Timing configuration	RWT WRT RRT WWT	ACT_PD_EXIT PRE_PD_EXIT ODT_PD_EXIT MRS_CYC	8.4.1.4/8-16
TIMING_CFG_1	Timing configuration	PRETOACT ACTTOPRE ACTTORW CASLAT	REFREC WRREC ACTTOACT WRTORD	8.4.1.5/8-18
TIMING_CFG_2	Timing configuration	ADD_LAT CPO WR_LAT RD_TO_PRE	WR_DATA_DELAY CKE_PLS FOUR_ACT	8.4.1.6/8-20

Table 8-58. Memory Interface Configuration Register Initialization Parameters (continued)

Name	Description	Parameter	Section/Page
DDR_SDRAM_CFG	Control configuration	SREN ECC_EN RD_EN SDRAM_TYPE DYN_PWR 32_BE 8_BE DBW NCAP 2T_EN BA_INTLV_CTL x32_EN HSE BI	8.4.1.7/8-22
DDR_SDRAM_CFG_2	Control configuration	DLL_RST_DIS DQS_CFG ODT_CFG NUM_PR D_INIT	8.4.1.8/8-24
DDR_SDRAM_MODE	Mode configuration	ESDMODE SDMODE	8.4.1.9/8-26
DDR_SDRAM_MODE_2	Mode configuration	ESDMODE2 ESDMODE3	8.4.1.10/8-27
DDR_SDRAM_INTERVAL	Interval configuration	REFINT BSTOPRE	8.4.1.12/8-30
DDR_DATA_INIT	Data initialization configuration register	INIT_VALUE	8.4.1.13/8-30
DDR_SDRAM_CLK_CNTL	Clock adjust	CLK_ADJUST	8.4.1.14/8-31
DDR_INIT_ADDR	Initialization address	INIT_ADDR	8.4.1.15/8-31
DDR_INIT_EXT_ADDR	Extended initialization address	INIT_EXT_ADDR	8.4.1.16/8-32

8.6.1 Programming Differences between Memory Types

Depending on the memory type used, certain fields must be programmed differently. Table 8-59 illustrates the differences in certain fields for different memory types. Note: This table does not list all fields that must be programmed.

Table 8-59. Programming Differences between Memory Types

Parameter	Description	Differences		Section/Page
AP*n*_EN	Chip Select *n* Auto Precharge Enable	DDR1	Can be used to place chip select *n* in auto precharge mode	8.4.1.2/8-13
		DDR2	Can be used to place chip select *n* in auto precharge mode	

Table 8-59. Programming Differences between Memory Types (continued)

Parameter	Description	Differences		Section/Page
ODT_RD_CFG	Chip Select ODT Read Configuration	DDR1	Should always be set to 000	8.4.1.2/8-13
		DDR2	Can be enabled to assert ODT if desired. This could be set differently depending on system topology. However, systems with only 1 chip select typically not uses ODT when issuing reads to the memory.	
ODT_WR_CFG	Chip Select ODT Write Configuration	DDR1	Should always be set to 000	8.4.1.2/8-13
		DDR2	Can be enabled to assert ODT if desired. This could be set differently depending on system topology. However, ODT typically is set to assert for the chip select that is getting written to (value would be set to 001).	
ODT_PD_EXIT	ODT Powerdown Exit	DDR1	Should be set to 0001	8.4.1.4/8-16
		DDR2	Should be set according to the DDR2 specifications for the memory used. The JEDEC parameter this applies to is t_{AXPD}.	
PRETOACT	Precharge to Activate Timing	DDR1	Should be set according to the specifications for the memory used (t_{RP})	8.4.1.5/8-18
		DDR2	Should be set according to the specifications for the memory used (t_{RP})	
ACTTOPRE	Activate to Precharge Timing	DDR1	Should be set, along with the Extended Activate to Precharge Timing, according to the specifications for the memory used (t_{RAS})	8.4.1.5/8-18
		DDR2	Should be set, along with the Extended Activate to Precharge Timing, according to the specifications for the memory used (t_{RAS})	
ACTTORW	Activate to Read/Write Timing	DDR1	Should be set according to the specifications for the memory used (t_{RCD})	8.4.1.5/8-18
		DDR2	Should be set according to the specifications for the memory used (t_{RCD})	
CASLAT	CAS Latency	DDR1	Should be set, along with the Extended CAS Latency, to the desired CAS latency	8.4.1.5/8-18
		DDR2	Should be set, along with the Extended CAS Latency, to the desired CAS latency	
REFREC	Refresh Recovery	DDR1	Should be set, along with the Extended Refresh Recovery, to the specifications for the memory used (t_{RFC})	8.4.1.5/8-18
		DDR2	Should be set, along with the Extended Refresh Recovery, to the specifications for the memory used (T_{RFC})	

Table 8-59. Programming Differences between Memory Types (continued)

Parameter	Description	Differences		Section/Page
WRREC	Write Recovery	DDR1	Should be set according to the specifications for the memory used (t_{WR})	8.4.1.5/8-18
		DDR2	Should be set according to the specifications for the memory used (t_{WR})	
ACTTOACT	Activate *A* to Activate *B*	DDR1	Should be set according to the specifications for the memory used (t_{RRD})	8.4.1.5/8-18
		DDR2	Should be set according to the specifications for the memory used (t_{RRD})	
WRTORD	Write to Read Timing	DDR1	Should be set according to the specifications for the memory used (t_{WTR})	8.4.1.5/8-18
		DDR2	Should be set according to the specifications for the memory used (t_{WTR})	
ADD_LAT	Additive Latency	DDR1	Should be set to 000	8.4.1.6/8-20
		DDR2	Should be set to the desired additive latency. This must be set to a value less than TIMING_CFG_1[ACTTORW]	
WR_LAT	Write Latency	DDR1	Should be set to 001	8.4.1.6/8-20
		DDR2	Should be set to CAS latency – 1 cycle. For example, if the CAS latency if 5 cycles, then this field should be set to 100 (4 cycles).	
RD_TO_PRE	Read to Precharge Timing	DDR1	Should be set to 010 if burst length is 4 and 100 if burst length is 8	8.4.1.6/8-20
		DDR2	Should be set according to the specifications for the memory used (t_{RTP}). Time between read and precharge for non-zero value of additive latency (AL) is a minimum of AL + t_{RTP} cycles.	
CKE_PLS	Minimum CKE Pulse Width	DDR1	Can be set to 001	8.4.1.6/8-20
		DDR2	Should be set according to the specifications for the memory used (t_{CKE})	
FOUR_ACT	Four Activate Window	DDR1	Should be set to 00001	8.4.1.6/8-20
		DDR2	Should be set according to the specifications for the memory used (t_{FAW}). Only applies to eight logical banks.	
RD_EN	Registered DIMM Enable	DDR1	If registered DIMMs are used, then this field should be set to 1	8.4.1.7/8-22
		DDR2	If registered DIMMs are used, then this field should be set to 1	
8_BE	8-beat burst enable	DDR1	If a 32-bit bus is used, and 8-beat bursts are desired, then this field should be set to 1	8.4.1.7/8-22
		DDR2	Should be set to 0	

Table 8-59. Programming Differences between Memory Types (continued)

Parameter	Description	Differences		Section/Page
2T_EN	2T Timing Enable	DDR1	In heavily loaded systems, this can be set to 1 to gain extra timing margin on the interface at the cost of address/command bandwidth.	8.4.1.7/8-22
		DDR2	In heavily loaded systems, this can be set to 1 to gain extra timing margin on the interface at the cost of address/command bandwidth.	
DLL_RST_DIS	DLL Reset Disable	DDR1	Should typically be set to 0, unless it is desired to bypass the DLL reset when exiting self refresh.	8.4.1.8/8-24
		DDR2	Should typically be set to 0, unless it is desired to bypass the DLL reset when exiting self refresh.	
DQS_CFG	DQS Configuration	DDR1	Should be set to 00	8.4.1.8/8-24
		DDR2	Can be set to either 00 or 01, depending on if differential strobes are used	
ODT_CFG	ODT Configuration	DDR1	Should be set to 00	8.4.1.8/8-24
		DDR2	Can be set for termination at the IOs according to system topology. Typically, if ODT is enabled, then the internal IOs should be set up for termination only during reads to DRAM.	
BSTOPR	Burst To Precharge Interval	DDR1	Can be set to any value, depending on the application. Auto precharge can be enabled by setting this field to all 0s.	8.4.1.12/8-30
		DDR2	Can be set to any value, depending on the application. Auto precharge can be enabled by setting this field to all 0s.	

8.6.2 DDR SDRAM Initialization Sequence

After configuration of all parameters is complete, system software must set DDR_SDRAM_CFG[MEM_EN] to enable the memory interface. Note that 200 µs must elapse after DRAM clocks are stable (DDR_SDRAM_CLK_CNTL[CLK_ADJUST] is set and any chip select is enabled) before MEM_EN can be set, so a delay loop in the initialization code may be necessary if software is enabling the memory controller. If DDR_SDRAM_CFG[BI] is not set, the DDR memory controller conducts an automatic initialization sequence to the memory, which follows the memory specifications. If the bypass initialization mode is used, then software can initialize the memory through the DDR_SDRAM_MD_CNTL register.

Chapter 9
Programmable Interrupt Controller (PIC)

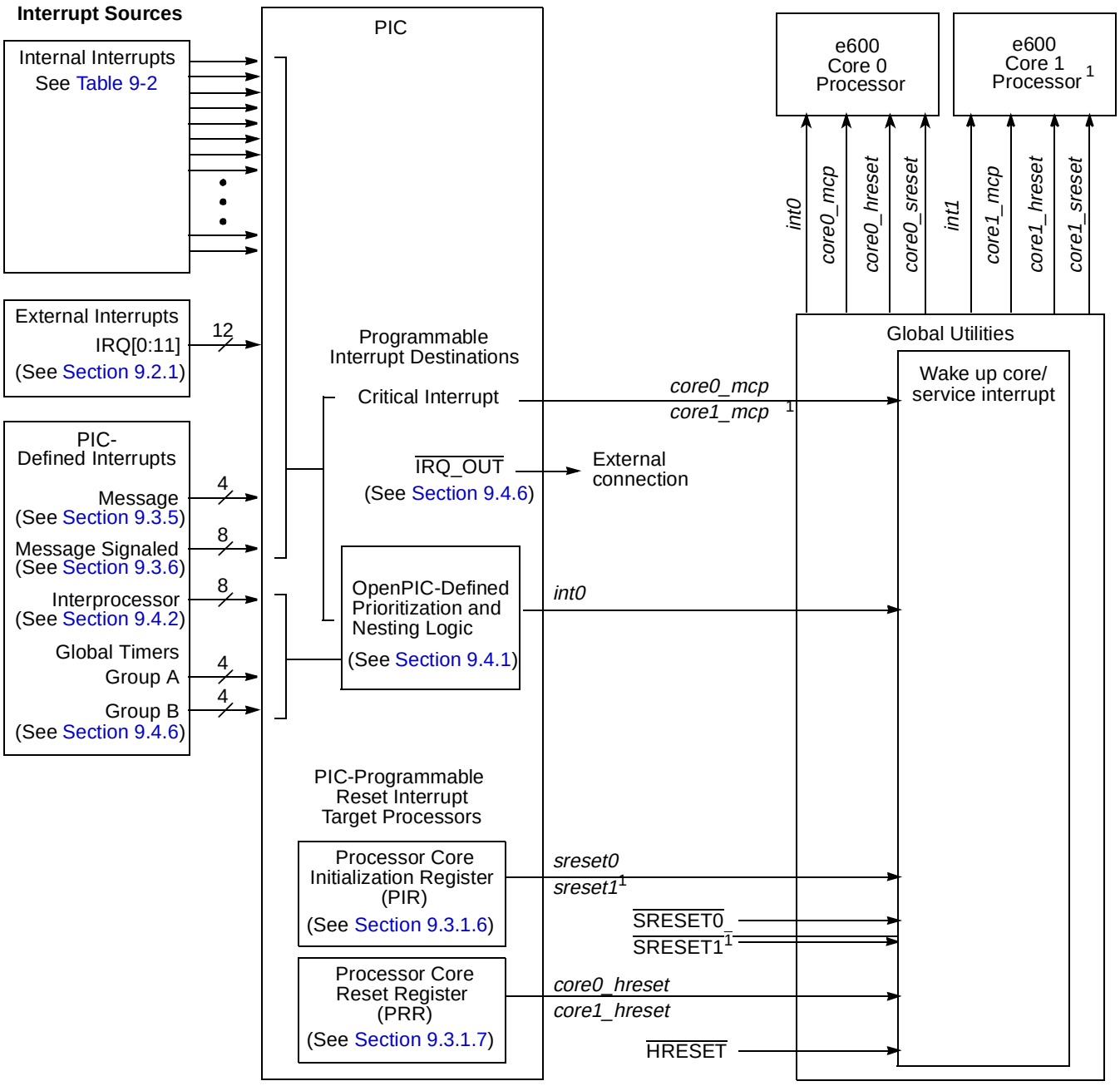
This chapter describes the programmable interrupt controller (PIC) interrupt protocol, various types of interrupt sources controlled by the PIC, and the PIC registers with some programming guidelines.

9.1 Introduction

The PIC conforms to the OpenPIC architecture. The interrupt controller provides multiprocessor interrupt management, and is responsible for receiving hardware-generated interrupts from different sources (both internal and external), prioritizing them, and delivering them to a CPU for servicing. References to the multiple processors in the MPC8641D correspond to the two e600 cores. The MPC8641 has a single core that corresponds to core 0 with respect to the PIC.

9.1.1 Overview

Figure 9-1 is a block diagram showing the relationship of the various functional blocks and how the signals external to the PIC are connected to other blocks on the device, including the cores.



1 Not supported in single-processor implementations.

Figure 9-1. Interrupt Sources Block Diagram Features

The PIC has the following features:

- Support for the following interrupt sources:
- External—Off-chip signals, IRQ[0:11]
- Internal interrupt sources; see Table 9-3. These are on-chip sources from peripheral logic within the integrated device signalling error conditions that need to be addressed by software.
- Interrupts generated from within the PIC itself, which are as follows:
  - Global timers A and B internal to the PIC
  - Interprocessor interrupts (IPI)—Intended for communication between different processor cores on the same device. (Can be used for self-interrupt in single-core devices.)
  - Message registers—From within the PIC. Triggered on register write, cleared on read. Used for interprocessor communication.
  - Shared message signaled registers—From within the PIC. Triggered on register write, cleared on read. Used for cross-program communication.
    - Four 32-bit message interrupt channels.
    - Two groups of four global 32-bit timers clocked with the MPX clock or the RTC input. Timers within each group can be concatenated to time longer durations.
- Three types of programmable interrupt outputs:
  - External interrupt (*int0* and *int1*). Any of the PIC interrupt sources can be programmed to direct interrupt requests to *int0* and *int1*. Handling of such interrupt requests follows the OpenPIC specification, which guarantees that the highest priority interrupt supersedes lower priority interrupts. Section 9.4.1.2, “Interrupts Routed to *int*,” describes how the PIC logic handles these interrupts.
  - Critical interrupt (*cint0* and *cint1*). Connected to the core’s machine check interrupt input.
  - $\overline{\text{IRQ\_OUT}}$.

    Section 9.4.1.1, “Interrupts Routed to *cint* or IRQ_OUT,” describes how the PIC logic supports this interrupt.
- Programming model compliant with the OpenPIC architecture.
  - Message, interprocessor and global timer interrupts. (Note that the interprocessor and global timer interrupts can only be routed to *int*.)
  - The following OpenPIC-defined features support only interrupts routed to the *int* signal:
    - Fully-nested interrupt delivery, guaranteeing that the interrupt source with the highest priority is given precedence over lower priority interrupts, including any that are in service.
    - 16 programmable interrupt priority levels
    - Support for identifying and handling spurious interrupts
- Support for two processors.
  - Interrupts can be routed to processor core 0 or 1
  - Multi-cast delivery mode for interprocessor and global timer interrupts allowing these interrupts to be routed to either core 0 or 1, or both cores.
- Processor core initialization control

- Processor core hardware reset control
- Programmable resetting of the PIC through the global configuration register
- Support for connection of external interrupt controller device such as an 8259 programmable interrupt controller. In 8259 mode, an interrupt causes assertion of a local (that is, internal to the integrated device) interrupt output signal, $\overline{\text{IRQ\_OUT}}$.
- Pass-through mode (PIC disabled) in which the PIC directs interrupts off-chip for external servicing. See Section 9.1.4.2, “Pass-Through Mode (GCR[M] = 0).”

9.1.2 The PIC in Multiple-Processor Implementations

In a multiprocessor implementation, the PIC is replicated for each core, and where necessary, the duplicated resources are indicated with a 0 or a 1. For example, *int0* identifies the *int* signal to processor 0 and *int1* identifies the *int* signal to processor 1. Other resources associated with the cores are identified by the number. For example, setting PIR[P1] triggers a reset of core 1, and setting PIR[P0] triggers a reset of core 0.

However, where the distinction is not necessary, none is made and such resources are referred to generically. For example, the *int* signal refers to the behavior of either *int0* or *int1*.

In a multiprocessor system, it is possible for the PIC to direct interrupts to the other processor. This functionality is supported by certain multiprocessor aspects to the programming model. For example, an interrupt source in processor 0 can be programmed to target *int1* by setting the P1 bit in its destination register, *x*IDR*n*.

Note that although they are part of the programming model, such resources are reserved in a single-processor device.

9.1.3 Interrupts to the Processor Core

The external interrupt signal, *int*, is the main interrupt output from the PIC to the processor core.

The interrupt sources can also specify the critical interrupt output, c*int0* or c*int1*, if the corresponding *x*IDR*n*[CI0] or *x*IDR*n*[CI1] is set.

The PIC also defines the PIR, described in Section 9.3.1.6, “Processor Core Initialization Register (PIR),” which can be used to reset the core. The PRR, described in Section 9.3.1.7, “Processor Reset Register (PRR),” which can be used to trigger a hard reset. Processor core interrupts generated by the PIC are described in Table 9-1.

Table 9-1. Processor Core Interrupts Generated by the PIC—Types and Sources

Source	Internal External Message Message Signaled Global Timer Interprocessor	Internal External Message Message Signaled		PIR	PRR
PIC output	*int0* or *int1*	*cint0* or *cint1*	$\overline{\text{IRQ\_OUT}}$	*sreset0* or *sreset1*	*core0_hreset* or *core1_hreset*
Core input	*int0* or *int1*	*core0_mcp* or *core1_mcp*		*core0_sreset* or *core1_sreset*	*core0_hreset* or *core1_hreset*
Interrupt type	External interrupt	Machine check		Soft reset	Hard reset
Description	PIC signals an external interrupt in corresponding core.	Causes machine check condition in corresponding core.	$\overline{\text{IRQ\_OUT}}$ assertion	One of the following: • External $\overline{\text{SRESET0}}$ or $\overline{\text{SRESET1}}$ assertion (and negation) • *sreset0* or *sreset1* output from PIC due to PIR update.	One of the following: • $\overline{\text{HRESET}}$ assertion (and negation) • *core0_hreset* or *core1_hreset* outputs due to PRR update.

9.1.4 Modes of Operation

Mixed or pass-through mode of operation is chosen by setting or clearing GCR[M], as described in Section 9.3.1.4, “Global Configuration Register (GCR).”

9.1.4.1 Mixed Mode (GCR[M] = 1)

In mixed mode, external and internal interrupts are delivered using the normal priority and delivery mechanisms detailed in Section 9.4.1, “Flow of Interrupt Control.”

9.1.4.2 Pass-Through Mode (GCR[M] = 0)

The PIC provides a mechanism to support alternate external interrupt controllers such as the PC/AT-compatible 8259 interrupt controller architecture. After a hard reset, the PIC defaults to pass-through mode, in which active-high interrupts from external source IRQ0 are passed directly to core 0 as shown in Figure 9-2; all other external interrupt signals are ignored. Thus, the interrupt signal from an external interrupt controller can be connected to IRQ0 and cause direct interrupts to the processor core 0. The PIC does not perform a vector fetch from an 8259 interrupt controller.

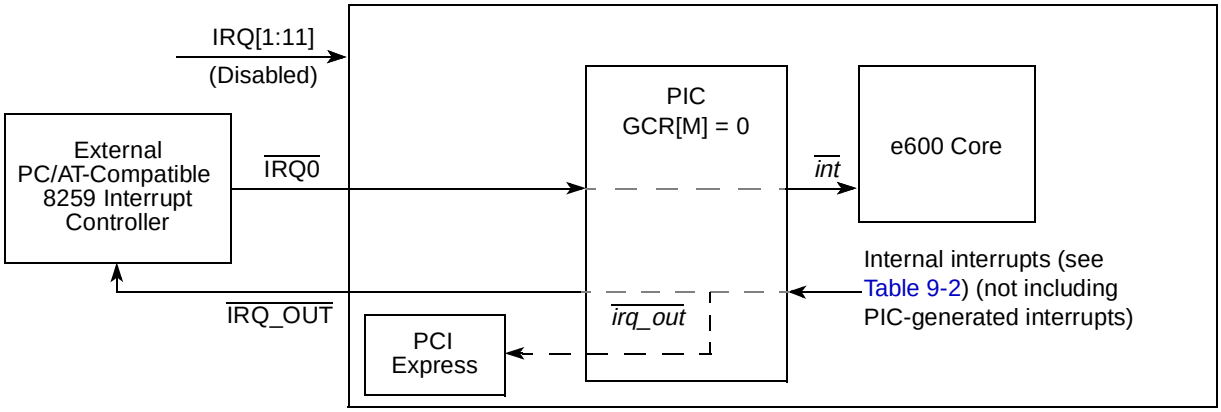


Figure 9-2. Pass-Through Mode Example

When pass-through mode is enabled, the internally-generated interrupts shown in Table 9-3 are not forwarded to core 0. Instead, the PIC passes the raw interrupts from the internal sources to IRQ_OUT. Note that when the PCI Express controller is configured as an endpoint (EP) device, the *irq_out* signal from the PIC may used to automatically generate an outbound PCI Express MSI transaction toward the remote interrupt controller resource on the root complex (RC). See Section 16.4.2.1.2, "Hardware MSI Generation."

Note that in pass-through mode, interrupts generated within the PIC (global timers, interprocessor, and message register interrupts) are disabled. If internal or PIC-generated interrupts must be reported internally to the processor, mixed mode must be used.

9.1.5 Interrupt Sources

The PIC can receive separate interrupts from the following sources:

- External—Off-chip signals, IRQ[0:11]
- Internal—On-chip sources from peripheral logic within the integrated device. See Table 9-2.
- Global timers A and B internal to the PIC
- Interprocessor interrupts (IPI)—Intended for communication between different processor cores on the same device. (Can be used for self-interrupt in single-core implementations.)
- Message registers—From within the PIC. Triggered on register write, cleared on read. Used for interprocessor communication.
- Shared message signaled registers—From within the PIC. Triggered on register write, cleared on read. Used for cross-program communication.

9.1.5.1 Interrupt Routing—Mixed Mode

When an interrupt request is delivered to the PIC, the corresponding interrupt destination register is checked to determine where the request should be routed, as follows:

- If *x*IDR*n*[EP] = 1 (and all other destination bits are zero), the interrupt is routed off-chip to the external IRQ_OUT signal. Or if the PCI Express controller is in EP mode and automatically generates a PCI Express MSI transaction. See Section 16.4.2.1.2, "Hardware MSI Generation."

- If either, but not both, *x*IDR*n*[CI0] or *x*IDR*n*[CI1] is set (and all other destination bits are zero), the interrupt is routed to *cint0* or *cint1*.
- If *x*IDR*n*[P0] is set (and all other destination bits are zero) the interrupt is routed to *int0*. Setting *x*IDR*n*[P1] likewise routs the interrupt to *int1* In this case, the interrupt is latched by the interrupt pending register (IPR) and the interrupt flow is as described in Section 9.4.1, "Flow of Interrupt Control." Note that multicasting interrupts (global timer and interprocessor interrupts) can set both P0 and P1; other interrupt sources cannot.

9.1.5.2 Interrupt Destinations

Following its reset (by default), the PIC directs all timer, shared message signaled, and interrupts from external and internal sources to *int* output (connected to the *int* signal of the processor core).

Interprocessor and global timers interrupts can be programmed to be routed to either core's *int* signal or to both cores (multi-casting).

All other interrupts have more destination options, but only one destination can be chosen for a single non–multi-casting interrupt. Instead of being routed to *int*, these interrupts can be routed to the core through $\overline{\text{IRQ\_OUT}}$ or *cint*. These options are selected by writing to the EP or CI fields in the appropriate destination register. The global utilities block routes the critical interrupts to either *core_mcp0* or *core_mcp1*, depending on which bit is set in the interrupt's destination register.

9.1.5.3 Internal Interrupt Sources

Table 9-2 shows the assignments of the internal interrupt sources and how they are mapped to the registers that control them. Only the internal interrupts used are listed; that is, the numbers are not consecutive.

Table 9-2. Internal Interrupt Assignments

Internal Interrupt Number	Interrupt Source
1	MCM
2	DDR DRAM controllers 1 and 2
3	LBC controller
4	DMA channel 1
5	DMA channel 2
6	DMA channel 3
7	DMA channel 4
8	PCI Express Port1
9	PCI Express Port2
12	UART2
13	eTSEC 1 transmit
14	eTSEC 1 receive
15	eTSEC 3 transmit
16	eTSEC 3 receive
17	eTSEC 3 error

Table 9-2. Internal Interrupt Assignments (continued)

Internal Interrupt Number	Interrupt Source
18	eTSEC 1 error
19	eTSEC 2 transmit
20	eTSEC 2 receive
21	eTSEC 4 transmit
22	eTSEC 4 receive
23	eTSEC 4 error
24	eTSEC 2 error
26	UART1
27	I^2C controllers
28	Performance monitor
32	SRIO error/write-port unit
33	SRIO outbound doorbell
34	SRIO inbound doorbell
37	SRIO outbound message unit1
38	SRIO inbound message unit 1
39	SRIO outbound message unit 2
40	SRIO inbound message unit 2

9.2 External Signal Descriptions

The following sections describe the PIC signals.

9.2.1 Signal Overview

The PIC external interface signals are described in Table 9-3. There are 12 distinct external interrupt request input signals (IRQ[0:11]) and 1 interrupt request output signal $\overline{\text{IRQ\_OUT}}$. As Table 9-3 shows, 8 of the IRQ inputs are also used for delivering INTx signals for the PCI Express root complexes.

9.2.2 Detailed Signal Descriptions

Table 9-3 provides detailed descriptions of the external PIC signals.

Table 9-3. Interrupt Signals—Detailed Signal Descriptions

Signal	I/O	Description	
IRQ[0:11]	I	Interrupt request 0–11. The polarity and sense of each of these signals is programmable. All of these inputs can be driven asynchronously. **Note:** Some interrupt request signals IRQ*n* may share PIC external interrupt registers with PCI Express INTx signaling. See Section 9.4.5, "PCI Express INTx," for more information.	
		State Meaning	Asserted—When an external interrupt signal is asserted (according to the programmed polarity), the PIC checks its priority and the interrupt is conditionally passed to the processor designated in the corresponding destination register. In pass-through mode, only interrupts detected on IRQ0 are passed directly to core 0. Negated—There is no incoming interrupt from that source.
		Timing	Assertion—All of these inputs can be asserted asynchronously. Negation—Interrupts programmed as level-sensitive must remain asserted until serviced. Timing requirements for edge-sensitive interrupts can be found in the *Hardware Specifications*.
$\overline{\text{IRQ\_OUT}}$	O	Interrupt request out. 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 9.1.4, "Modes of Operation."	
		State Meaning	Asserted—At least one interrupt is currently being signalled to the external system. Negated—Indicates no interrupt source currently routed to $\overline{\text{IRQ\_OUT}}$.
		Timing	Because external interrupts are asynchronous with respect to the system clock, both assertion and negation of $\overline{\text{IRQ\_OUT}}$ occurs asynchronously with respect to the interrupt source. All timing given here is approximate. Assertion—Internal interrupt source: 2 MPX 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 MPX clock cycles External interrupt: 4 cycles. Message interrupts: 2 cycles after message register cleared.
$\overline{\text{MCP\_0}}$ and $\overline{\text{MCP\_1}}$	I	Machine check processor *n*. Assertion causes a machine check interrupt to the corresponding core. Note that if the core is not configured to process machine check interrupts (MSR[ME] = 0), assertion of $\overline{\text{MCP}n}$ causes a core checkstop condition. Note that internal sources for the internal *core_mcp n* can also cause a machine check interrupt to the processor core as described in Section 17.4.1.11, "Machine Check Summary Register (MCPSUMR)," Table 9-1 and Table 5-2.	
		State Meaning	Asserted—Integrated logic should direct the corresponding core to take a machine check interrupt or enter the checkstop state as directed by the MSR. Negated—Machine check handling is not being requested by the external system.
		Timing	Assertion—May occur at any time, asynchronous to any clock. Negation—Because $\overline{\text{MCP}n}$ is edge-triggered, it can be negated one clock after its assertion.

9.3 Memory Map/Register Definition

The PIC programmable register map occupies 256 Kbytes of memory-mapped space. Reading undefined portions of the memory map returns all zeros; writing has no effect. All PIC registers are 32 bits wide and, although located on 128-bit address boundaries, should be accessed only as 32-bit quantities.

The PIC address offset map, shown in Table 9-4, is divided into three areas:

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

Table 9-4. PIC Register Address Map

Offset	Register	Access	Reset	Section/Page
PIC Register Address Map—Global Registers—Block Base Address: 0x4_0000				
0x4_0000	BRR1—Block revision register 1	R	0x0040_0300	9.3.1.1/9-18
0x4_0010	BRR2—Block revision register 2	R	0x0000_0001	9.3.1.2/9-19
0x4_0020–0x4_0030	Reserved	—	—	—
0x4_0040	IPIDR0—Interprocessor interrupt 0 (IPI 0) dispatch register	W	0x0000_0000	9.3.8.1/9-49
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	9.3.8.2/9-49
0x4_0090	WHOAMI—Who am I register	R	n/a	9.3.8.3/9-50
0x4_00A0	IACK—Interrupt acknowledge register	R	0x0000_0000	9.3.8.4/9-51
0x4_00B0	EOI—End of interrupt register	W	0x0000_0000	9.3.8.5/9-51
0x4_00C0–0x4_0FF0	Reserved	—	—	—
0x4_1000	FRR—Feature reporting register	R	0x0057_01020 x0067_0002	9.3.1.3/9-19
0x4_1010	Reserved	—	—	—
0x4_1020	GCR—Global configuration register	R/W	0x0000_0000	9.3.1.4/9-20
0x4_1030	Reserved	—	—	—
0x4_1040–0x4_1070	Vendor reserved	—	—	—
0x4_1080	VIR—Vendor identification register	R	0x0000_0000	9.3.1.5/9-21
0x4_1090	PIR—Processor core initialization register	R/W	0x0000_0000	9.3.1.6/9-21
0x4_1098	PRR—Processor reset register	R/W	0x0000_0000	9.3.1.7/9-22
0x4_10A0	IPIVPR0—IPI 0 vector/priority register	R/W	0x8000_0000	9.3.1.8/9-23
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	9.3.1.9/9-23
Global Timer Group A Registers				
0x4_10F0	TFRRA—Timer frequency reporting register (Group A)	R/W	0x0000_0000	9.3.2.1/9-24
0x4_1100	GTCCRA0—Global timer 0 current count register (Group A)	R	0x0000_0000	9.3.2.2/9-25
0x4_1110	GTBCRA0—Global timer 0 base count register (Group A)	R/W	0x8000_0000	9.3.2.3/9-25

Table 9-4. PIC Register Address Map (continued)

Offset	Register	Access	Reset	Section/Page
0x4_1120	GTVPRA0—Global timer 0 vector/priority register (Group A)	R/W	0x8000_0000	9.3.2.4/9-26
0x4_1130	GTDRA0—Global timer 0 destination register (Group A)	R/W	0x0000_0001	9.3.2.5/9-27
0x4_1140	GTCCRA1—Global timer 1 current count register (Group A)	R	0x0000_0000	9.3.2.2/9-25
0x4_1150	GTBCRA1—Global timer 1 base count register (Group A)	R/W	0x8000_0000	9.3.2.3/9-25
0x4_1160	GTVPRA1—Global timer 1 vector/priority register (Group A)	R/W	0x8000_0000	9.3.2.4/9-26
0x4_1170	GTDRA1—Global timer 1 destination register (Group A)	R/W	0x0000_0001	9.3.2.5/9-27
0x4_1180	GTCCRA2—Global timer 2 current count register (Group A)	R	0x0000_0000	9.3.2.2/9-25
0x4_1190	GTBCRA2—Global timer 2 base count register (Group A)	R/W	0x8000_0000	9.3.2.3/9-25
0x4_11A0	GTVPRA2—Global timer 2 vector/priority register (Group A)	R/W	0x8000_0000	9.3.2.4/9-26
0x4_11B0	GTDRA2—Global timer 2 destination register (Group A)	R/W	0x0000_0001	9.3.2.5/9-27
0x4_11C0	GTCCRA3—Global timer 3 current count register (Group A)	R	0x0000_0000	9.3.2.2/9-25
0x4_11D0	GTBCRA3—Global timer 3 base count register (Group A)	R/W	0x8000_0000	9.3.2.3/9-25
0x4_11E0	GTVPRA3—Global timer 3 vector/priority register (Group A)	R/W	0x8000_0000	9.3.2.4/9-26
0x4_11F0	GTDRA3—Global timer 3 destination register (Group A)	R/W	0x0000_0001	9.3.2.5/9-27
0x4_1200–0x4_12F0	Reserved	—	—	—
0x4_1300	TCRA—Timer control register (Group A)	R/W	0x0000_0000	9.3.2.6/9-27
0x4_1308	ERQSR—External interrupt summary register	R	0x0000_0000	9.3.3.1/9-29
0x4_1310	IRQSR0—IRQ_OUT summary register 0	R	0x0000_0000	9.3.3.2/9-30
0x4_1320	IRQSR1—IRQ_OUT summary register 1	R	0x0000_0000	9.3.3.3/9-31
0x4_1324	IRQSR2—IRQ_OUT summary register 2	R	0x0000_0000	9.3.3.4/9-31
0x4_1330	CISR0—Critical interrupt summary register 0	R	0x0000_0000	9.3.3.5/9-32
0x4_1340	CISR1—Critical interrupt summary register 1	R	0x0000_0000	9.3.3.6/9-32
0x4_1344	CISR2—Critical interrupt summary register 2	R	0x0000_0000	9.3.3.7/9-33
0x4_1350	PM0MR0—Performance monitor 0 mask register 0	R/W	0xFFFF_FFFF	9.3.4.1/9-33
0x4_1360	PM0MR1—Performance monitor 0 mask register 1	R/W	0xFFFF_FFFF	9.3.4.2/9-34
0x4_1364	PM0MR2—Performance monitor 0 mask register 2	R/W	0xFFFF_0000	9.3.4.2/9-34
0x4_1370	PM1MR0—Performance monitor 1 mask register 0	R/W	0xFFFF_FFFF	9.3.4.1/9-33
0x4_1380	PM1MR1—Performance monitor 1 mask register 1	R/W	0xFFFF_FFFF	9.3.4.2/9-34
0x4_1384	PM1MR2—Performance monitor 1 mask register 2	R/W	0xFFFF_0000	9.3.4.2/9-34
0x4_1390	PM2MR0—Performance monitor 2 mask register 0	R/W	0xFFFF_FFFF	9.3.4.1/9-33
0x4_13A0	PM2MR1—Performance monitor 2 mask register 1	R/W	0xFFFF_FFFF	9.3.4.2/9-34
0x4_13A4	PM2MR2—Performance monitor 2 mask register 2	R/W	0xFFFF_0000	9.3.4.2/9-34
0x4_13B0	PM3MR0—Performance monitor 3 mask register 0	R/W	0xFFFF_FFFF	9.3.4.1/9-33
0x4_13C0	PM3MR1—Performance monitor 3 mask register 1	R/W	0xFFFF_FFFF	9.3.4.2/9-34
0x4_13C4	PM3MR2—Performance monitor 3 mask register 2	R/W	0xFFFF_0000	9.3.4.2/9-34
0x4_13D0–0x4_13F0	Reserved	—	—	—

Table 9-4. PIC Register Address Map (continued)

Offset	Register	Access	Reset	Section/Page
0x4_1400	MSGR0—Message register 0	R/W	0x0000_0000	9.3.5.1/9-35
0x4_1410	MSGR1—Message register 1	R/W	0x0000_0000	9.3.5.1/9-35
0x4_1420	MSGR2—Message register 2	R/W	0x0000_0000	9.3.5.1/9-35
0x4_1430	MSGR3—Message register 3	R/W	0x0000_0000	9.3.5.1/9-35
0x4_1440–0x4_14F0	Reserved	—	—	—
0x4_1500	MER—Message enable register	R/W	0x0000_0000	9.3.5.2/9-36
0x4_1510	MSR—Message status register	R/W	0x0000_0000	9.3.5.3/9-36
0x4_1520–0x4_15F0	Reserved	—	—	—
0x4_1600	MSIR0—Shared message signaled interrupt register 0	RC	0x0000_0000	9.3.6.1/9-37
0x4_1610	MSIR1—Shared message signaled interrupt register 1	RC	0x0000_0000	9.3.6.1/9-37
0x4_1620	MSIR2—Shared message signaled interrupt register 2	RC	0x0000_0000	9.3.6.1/9-37
0x4_1630	MSIR3—Shared message signaled interrupt register 3	RC	0x0000_0000	9.3.6.1/9-37
0x4_1640	MSIR4—Shared message signaled interrupt register 4	RC	0x0000_0000	9.3.6.1/9-37
0x4_1650	MSIR5—Shared message signaled interrupt register 5	RC	0x0000_0000	9.3.6.1/9-37
0x4_1660	MSIR6—Shared message signaled interrupt register 6	RC	0x0000_0000	9.3.6.1/9-37
0x4_1670	MSIR7—Shared message signaled interrupt register 7	RC	0x0000_0000	9.3.6.1/9-37
0x4_1680–0x4_1700	Reserved	—	—	—
0x4_1720	MSISR—Shared message signaled interrupt status register	R	0x0000_0000	9.3.6.2/9-38
0x4_1740	MSIIR—Shared message signaled interrupt index register	W	0x0000_0000	9.3.6.3/9-38
0x4_1750–0x4_20E0	Reserved	—	—	—
Global Timer Group B Registers				
0x4_20F0	TFRRB—Timer frequency reporting register group B	R/W	0x0000_0000	9.3.2.1/9-24
0x4_2100	GTCCRB0—Global timer current count register group B 0	R	0x0000_0000	9.3.2.2/9-25
0x4_2110	GTBCRB0—Global timer base count register group B 0	R/W	0x8000_0000	9.3.2.3/9-25
0x4_2120	GTVPRB0—Global timer vector/priority register group B 0	R/W	0x8000_0000	9.3.2.4/9-26
0x4_2130	GTDRB0—Global timer destination register group B 0	R/W	0x0000_0001	9.3.2.5/9-27
0x4_2140	GTCCRB1—Global timer current count register group B 1	R	0x0000_0000	9.3.2.2/9-25
0x4_2150	GTBCRB1—Global timer base count register group B 1	R/W	0x8000_0000	9.3.2.3/9-25
0x4_2160	GTVPRB1—Global timer vector/priority register group B 1	R/W	0x8000_0000	9.3.2.4/9-26
0x4_2170	GTDRB1—Global timer destination register group B 1	R/W	0x0000_0001	9.3.2.5/9-27
0x4_2180	GTCCRB2—Global timer current count register group B 2	R	0x0000_0000	9.3.2.2/9-25
0x4_2190	GTBCRB2—Global timer base count register group B 2	R/W	0x8000_0000	9.3.2.3/9-25
0x4_21A0	GTVPRB2—Global timer vector/priority register group B 2	R/W	0x8000_0000	9.3.2.4/9-26
0x4_21B0	GTDRB2—Global timer destination register group B 2	R/W	0x0000_0001	9.3.2.5/9-27

Table 9-4. PIC Register Address Map (continued)

Offset	Register	Access	Reset	Section/Page
0x4_21C0	GTCCRB3—Global timer current count register group B 3	R	0x0000_0000	9.3.2.2/9-25
0x4_21D0	GTBCRB3—Global timer base count register group B 3	R/W	0x8000_0000	9.3.2.3/9-25
0x4_21E0	GTVPRB3—Global timer vector/priority register group B 3	R/W	0x8000_0000	9.3.2.4/9-26
0x4_21F0	GTDRB3—Global timer destination register group B 3	R/W	0x0000_0001	9.3.2.5/9-27
0x4_2200–0x4_22F0	Reserved	—	—	—
0x4_2300	TCRB—Timer control register (Group B)	R/W	0x0000_0000	9.3.2.6/9-27
0x4_2310–0x4_FFF0	Reserved	—	—	—
PIC Register Address Map—Interrupt Source Configuration Registers—Block Base Address: 0x5_0000				
0x5_0000	EIVPR0—External interrupt 0 (IRQ0) vector/priority register or PEX1-INTA vector/priority register	R/W	0x8000_0000	9.3.7.1/9-42
0x5_0010	EIDR0—External interrupt 0 (IRQ0) destination register or PEX1-INTA destination register	R/W	0x0000_0001	9.3.7.2/9-43
0x5_0020	EIVPR1—External interrupt 1 (IRQ1) vector/priority register or PEX1-INTB vector/priority register	R/W	0x8000_0000	9.3.7.1/9-42
0x5_0030	EIDR1—External interrupt 1 (IRQ1) destination register or PEX1-INTB destination register	R/W	0x0000_0001	9.3.7.2/9-43
0x5_0040	EIVPR2—External interrupt 2 (IRQ2) vector/priority register or PEX1-INTC vector/priority register	R/W	0x8000_0000	9.3.7.1/9-42
0x5_0050	EIDR2—External interrupt 2 (IRQ2) destination register or PEX1-INTC destination register	R/W	0x0000_0001	9.3.7.2/9-43
0x5_0060	EIVPR3—External interrupt 3 (IRQ3) vector/priority register or PEX1-INTD vector/priority register	R/W	0x8000_0000	9.3.7.1/9-42
0x5_0070	EIDR3—External interrupt 3 (IRQ3) destination register or PEX1-INTD destination register	R/W	0x0000_0001	9.3.7.2/9-43
0x5_0080	EIVPR4—External interrupt 4 (IRQ4) vector/priority register or PEX2-INTA vector/priority register	R/W	0x8000_0000	9.3.7.1/9-42
0x5_0090	EIDR4—External interrupt 4 (IRQ4) destination register or PEX2-INTA destination register	R/W	0x0000_0001	9.3.7.2/9-43
0x5_00A0	EIVPR5—External interrupt 5 (IRQ5) vector/priority register or PEX2-INTB vector/priority register	R/W	0x8000_0000	9.3.7.1/9-42
0x5_00B0	EIDR5—External interrupt 5 (IRQ5) destination register or PEX2-INTB destination register	R/W	0x0000_0001	9.3.7.2/9-43
0x5_00C0	EIVPR6—External interrupt 6 (IRQ6) vector/priority register or PEX2-INTC vector/priority register	R/W	0x8000_0000	9.3.7.1/9-42
0x5_00D0	EIDR6—External interrupt 6 (IRQ6) destination register or PEX2-INTC destination register	R/W	0x0000_0001	9.3.7.2/9-43
0x5_00E0	EIVPR7—External interrupt 7 (IRQ7) vector/priority register or PEX2-INTD vector/priority register	R/W	0x8000_0000	9.3.7.1/9-42
0x5_00F0	EIDR7—External interrupt 7 (IRQ7) destination register or PEX2-INTD destination register	R/W	0x0000_0001	9.3.7.2/9-43
0x5_0100	EIVPR8—External interrupt 8 (IRQ8) vector/priority register	R/W	0x8000_0000	9.3.7.1/9-42

Table 9-4. PIC Register Address Map (continued)

Offset	Register	Access	Reset	Section/Page
0x5_0110	EIDR8—External interrupt 8 (IRQ8) destination register	R/W	0x0000_0001	9.3.7.2/9-43
0x5_0120	EIVPR9—External interrupt 9 (IRQ9) vector/priority register	R/W	0x8000_0000	9.3.7.1/9-42
0x5_0130	EIDR9—External interrupt 9 (IRQ9) destination register	R/W	0x0000_0001	9.3.7.2/9-43
0x5_0140	EIVPR10—External interrupt 10 (IRQ10) vector/priority register	R/W	0x8000_0000	9.3.7.1/9-42
0x5_0150	EIDR10—External interrupt 10 (IRQ10) destination register	R/W	0x0000_0001	9.3.7.2/9-43
0x5_0160	EIVPR11—External interrupt 11 (IRQ11) vector/priority register	R/W	0x8000_0000	9.3.7.1/9-42
0x5_0170	EIDR11—External interrupt 11 (IRQ11) destination register	R/W	0x0000_0001	9.3.7.2/9-43
0x5_0180–0x5_01F0	Reserved	—	—	—
0x5_0200	IIVPR0—Internal interrupt 0 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x5_0210	IIDR0—Internal interrupt 0 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x5_0220	IIVPR1—Internal interrupt 1 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x5_0230	IIDR1—Internal interrupt 1 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x5_0240	IIVPR2—Internal interrupt 2 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x5_0250	IIDR2—Internal interrupt 2 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x5_0260	IIVPR3—Internal interrupt 3 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x5_0270	IIDR3—Internal interrupt 3 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x5_0280	IIVPR4—Internal interrupt 4 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x5_0290	IIDR4—Internal interrupt 4 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x5_02A0	IIVPR5—Internal interrupt 5 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x5_02B0	IIDR5—Internal interrupt 5 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x5_02C0	IIVPR6—Internal interrupt 6 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x5_02D0	IIDR6—Internal interrupt 6 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x5_02E0	IIVPR7—Internal interrupt 7 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x5_02F0	IIDR7—Internal interrupt 7 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x5_0300	IIVPR8—Internal interrupt 8 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x5_0310	IIDR8—Internal interrupt 8 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x5_0320	IIVPR9—Internal interrupt 9 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x5_0330	IIDR9—Internal interrupt 9 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x5_0340	IIVPR10—Internal interrupt 10 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x5_0350	IIDR10—Internal interrupt 10 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x5_0360	IIVPR11—Internal interrupt 11 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x5_0370	IIDR11—Internal interrupt 11 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x5_0380	IIVPR12—Internal interrupt 12 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x5_0390	IIDR12—Internal interrupt 12 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x5_03A0	IIVPR13—Internal interrupt 13 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x5_03B0	IIDR13—Internal interrupt 13 destination register	R/W	0x0000_0001	9.3.7.4/9-45

Table 9-4. PIC Register Address Map (continued)

Offset	Register	Access	Reset	Section/Page
0x5_03C0	IIVPR14—Internal interrupt 14 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x5_03D0	IIDR14—Internal interrupt 14 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x5_03E0	IIVPR15—Internal interrupt 15 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x5_03F0	IIDR15—Internal interrupt 15 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x5_0400	IIVPR16—Internal interrupt 16 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x5_0410	IIDR16—Internal interrupt 16 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x5_0420	IIVPR17—Internal interrupt 17 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x5_0430	IIDR17—Internal interrupt 17 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x5_0440	IIVPR18—Internal interrupt 18 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x5_0450	IIDR18—Internal interrupt 18 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x5_0460	IIVPR19—Internal interrupt 19 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x5_0470	IIDR19—Internal interrupt 19 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x5_0480	IIVPR20—Internal interrupt 20 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x5_0490	IIDR20—Internal interrupt 20 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x5_04A0	IIVPR21—Internal interrupt 21 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x5_04B0	IIDR21—Internal interrupt 21 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x5_04C0	IIVPR22—Internal interrupt 22 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x5_04D0	IIDR22—Internal interrupt 22 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x5_04E0	IIVPR23—Internal interrupt 23 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x5_04F0	IIDR23—Internal interrupt 23 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x5_0500	IIVPR24—Internal interrupt 24 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x5_0510	IIDR24—Internal interrupt 24 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x5_0520	IIVPR25—Internal interrupt 25 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x5_0530	IIDR25—Internal interrupt 25 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x5_0540	IIVPR26—Internal interrupt 26 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x5_0550	IIDR26—Internal interrupt 26 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x5_0560	IIVPR27—Internal interrupt 27 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x5_0570	IIDR27—Internal interrupt 27 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x5_0580	IIVPR28—Internal interrupt 28 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x5_0590	IIDR28—Internal interrupt 28 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x5_05A0	IIVPR29—Internal interrupt 29 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x5_05B0	IIDR29—Internal interrupt 29 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x5_05C0	IIVPR30—Internal interrupt 30 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x5_05D0	IIDR30—Internal interrupt 30 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x5_05E0	IIVPR31—Internal interrupt 31 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x5_05F0	IIDR31—Internal interrupt 31 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x5_0600	IIVPR32—Internal interrupt 32 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44

Table 9-4. PIC Register Address Map (continued)

Offset	Register	Access	Reset	Section/Page
0x5_0610	IIDR32—Internal interrupt 32 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x5_0620	IIVPR33—Internal interrupt 33 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x5_0630	IIDR33—Internal interrupt 33 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x5_0640	IIVPR34—Internal interrupt 34 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x5_0650	IIDR34—Internal interrupt 34 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x5_0660	IIVPR35—Internal interrupt 35 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x5_0670	IIDR35—Internal interrupt 35 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x5_0680	IIVPR36—Internal interrupt 36 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x5_0690	IIDR36—Internal interrupt 36 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x5_06A0	IIVPR37—Internal interrupt 37 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x5_06B0	IIDR37—Internal interrupt 37 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x5_06C0	IIVPR38—Internal interrupt 38 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x5_06D0	IIDR38—Internal interrupt 38 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x5_06E0	IIVPR39—Internal interrupt 39 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x5_06F0	IIDR39—Internal interrupt 39 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x5_0700	IIVPR40—Internal interrupt 40 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x5_0710	IIDR40—Internal interrupt 40 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x5_0720	IIVPR41—Internal interrupt 41 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x5_0730	IIDR41—Internal interrupt 41 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x5_0740	IIVPR42—Internal interrupt 42 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x5_0750	IIDR42—Internal interrupt 42 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x5_0760	IIVPR43—Internal interrupt 43 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x5_0770	IIDR43—Internal interrupt 43 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x5_0780	IIVPR44—Internal interrupt 44 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x5_0790	IIDR44—Internal interrupt 44 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x5_07A0	IIVPR45—Internal interrupt 45 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x5_07B0	IIDR45—Internal interrupt 45 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x5_07C0	IIVPR46—Internal interrupt 46 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x5_07D0	IIDR46—Internal interrupt 46 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x5_07E0	IIVPR47—Internal interrupt 47 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x5_07F0	IIDR47—Internal interrupt 47 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x5_0800–0x5_15F0	Reserved	—	—	—
0x5_1600	MIVPR0—Messaging interrupt 0 (MSG 0) vector/priority register	R/W	0x8000_0000	9.3.7.5/9-46
0x5_1610	MIDR0—Messaging interrupt 0 (MSG 0) destination register	R/W	0x0000_0001	9.3.7.6/9-46
0x5_1620	MIVPR1—Messaging interrupt 1 (MSG 1) vector/priority register	R/W	0x8000_0000	9.3.7.5/9-46
0x5_1630	MIDR1—Messaging interrupt 1 (MSG 1) destination register	R/W	0x0000_0001	9.3.7.6/9-46

Table 9-4. PIC Register Address Map (continued)

Offset	Register	Access	Reset	Section/Page
0x5_1640	MIVPR2—Messaging interrupt 2 (MSG 2) vector/priority register	R/W	0x8000_0000	9.3.7.5/9-46
0x5_1650	MIDR2—Messaging interrupt 2 (MSG 2) destination register	R/W	0x0000_0001	9.3.7.6/9-46
0x5_1660	MIVPR3—Messaging interrupt 3 (MSG 3) vector/priority register	R/W	0x8000_0000	9.3.7.5/9-46
0x5_1670	MIDR3—Messaging interrupt 3 (MSG 3) destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x5_1680–0x5_1BF0	Reserved	—	—	—
0x5_1C00	MSIVPR0—Shared message signaled interrupt vector/priority register 0	R/W	0x8000_0000	9.3.6.4/9-39
0x5_1C10	MSIDR0—Shared message signaled interrupt destination register 0	R/W	0x0000_0001	9.3.6.5/9-40
0x5_1C20	MSIVPR1—Shared message signaled interrupt vector/priority register 1	R/W	0x8000_0000	9.3.6.4/9-39
0x5_1C30	MSIDR1—Shared message signaled interrupt destination register 1	R/W	0x0000_0001	9.3.6.5/9-40
0x5_1C40	MSIVPR2—Shared message signaled interrupt vector/priority register 2	R/W	0x8000_0000	9.3.6.4/9-39
0x5_1C50	MSIDR2—Shared message signaled interrupt destination register 2	R/W	0x0000_0001	9.3.6.5/9-40
0x5_1C60	MSIVPR3—Shared message signaled interrupt vector/priority register 3	R/W	0x8000_0000	9.3.6.4/9-39
0x5_1C70	MSDIR3—Shared message signaled interrupt destination register 3	R/W	0x0000_0001	9.3.6.5/9-40
0x5_1C80	MSIVPR4—Shared message signaled interrupt vector/priority register 4	R/W	0x8000_0000	9.3.6.4/9-39
0x5_1C90	MSIDR4—Shared message signaled interrupt destination register 4	R/W	0x0000_0001	9.3.6.5/9-40
0x5_1CA0	MSIVPR5—Shared message signaled interrupt vector/priority register 5	R/W	0x8000_0000	9.3.6.4/9-39
0x5_1CB0	MSIDR5—Shared message signaled interrupt destination register 5	R/W	0x0000_0001	9.3.6.5/9-40
0x5_1CC0	MSIVPR6—Shared message signaled interrupt vector/priority register 6	R/W	0x8000_0000	9.3.6.4/9-39
0x5_1CD0	MSIDR6—Shared message signaled interrupt destination register 6	R/W	0x0000_0001	9.3.6.5/9-40
0x5_1CE0	MSIVPR7—Shared message signaled interrupt vector/priority register 7	R/W	0x8000_0000	9.3.6.4/9-39
0x5_1CF0	MSDIR7—Shared message signaled interrupt destination register 7	R/W	0x0000_0001	9.3.6.5/9-40
0x5_1D00–0x5_FFF0	Reserved	—	—	—
PIC Register Address Map—Per-CPU Registers Block Base Address: 0x6_0000				
0x6_0000–0x6_0030	Reserved	—	—	—
0x6_0040	IPIDR0—Processor core 0 interprocessor 0 dispatch register	W	all zeros	9.3.8.1/9-49
0x6_0050	IPIDR1—Processor core 0 interprocessor 1 dispatch register			
0x6_0060	IPIDR2—Processor core 0 interprocessor 2 dispatch register			
0x6_0070	IPIDR3—Processor core 0 interprocessor 3 dispatch register			
0x6_0080	CTPR0—Processor core 0 current task priority register	R/W	0x0000_000F	9.3.8.2/9-49
0x6_0090	WHOAMI0—Processor core 0 who am I register	R	n/a	9.3.8.3/9-50
0x6_00A0	IACK0—Processor core 0 interrupt acknowledge register	R	all zeros	9.3.8.4/9-51
0x6_00B0	EOI0—Processor core 0 end of interrupt register	W	all zeros	9.3.8.5/9-51
0x6_00C0–0x6_0FF0	Reserved	—	—	—
0x6_1000–0x6_1030	Reserved	—	—	—

Table 9-4. PIC Register Address Map (continued)

Offset	Register	Access	Reset	Section/Page
0x6_1040	IPIDR0—Processor core 1 interprocessor 0 dispatch register [1]	W	all zeros	9.3.8.1/9-49
0x6_1050	IPIDR1—Processor core 1 interprocessor 1 dispatch register [1]			
0x6_1060	IPIDR2—Processor core 1 interprocessor 2 dispatch register [1]			
0x6_1070	IPIDR3—Processor core 1 interprocessor 3 dispatch register [1]			
0x6_1080	CTPR1—Processor core 1 current task priority register [1]	R/W	0x0000_000F	9.3.8.2/9-49
0x6_1090	WHOAMI1—Processor core 1 who am I register [1]	R	n/a	9.3.8.3/9-50
0x6_10A0	IACK1—Processor core 1 interrupt acknowledge register [1]	R	all zeros	9.3.8.4/9-51
0x6_10B0	EOI1—Processor core 1 end of interrupt register [1]	W	all zeros	9.3.8.5/9-51
0x6_10C0–0x7_FFFC	Reserved	—	—	—

[1] Not supported for single-processor implementations.

9.3.1 Global Registers

Although most PIC registers have one address, some are replicated for each processor core in a multiprocessor device. For such registers, each core accesses its separate registers using the same address, the address decoding being sensitive to the processor core ID. A copy of the per-CPU registers is available to each processor core at the same physical address, that is, in a private access address space that acts like an alias to a processor's own copy of the per-CPU registers. As shown in Figure 9-45, the ID of the core initiating the read/write transaction determines which processor's per-CPU registers to access. For more information, see Section 9.3.8, "Per-CPU (Private Access) Registers."

NOTE

Register fields designated as write-1-to-clear are cleared only by writing ones to them. Writing zeros to them has no effect.

9.3.1.1 Block Revision Register 1 (BRR1)

BRR1, shown in Figure 9-3, provides information about the PIC IP block.

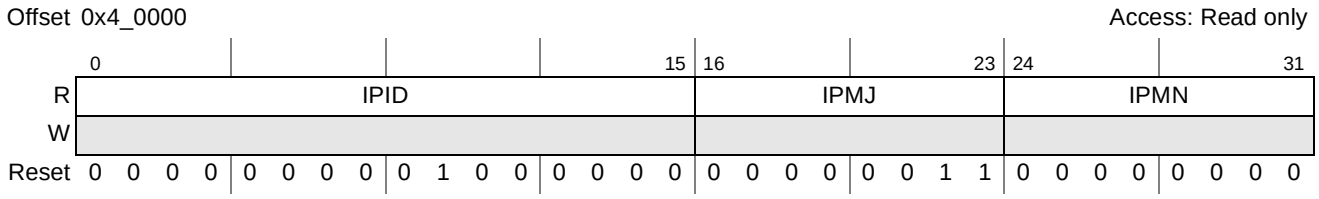


Figure 9-3. Block Revision Register 1 (BRR1)

Table 9-7 describes the BRR1 fields.

Table 9-5. BRR1 Field Descriptions

Bits	Name	Description
0–15	IPID	IP block ID.
16–23	IPMJ	The major revision of the IP block. 0x03 = PIC block used in the MPC8641 and MPC8641D
24–31	IPMN	The minor revision of the IP block.

9.3.1.2 Block Revision Register 2 (BRR2)

BRR2, shown in Figure 9-4, provides information about the IP block integration option and IP block configuration options.

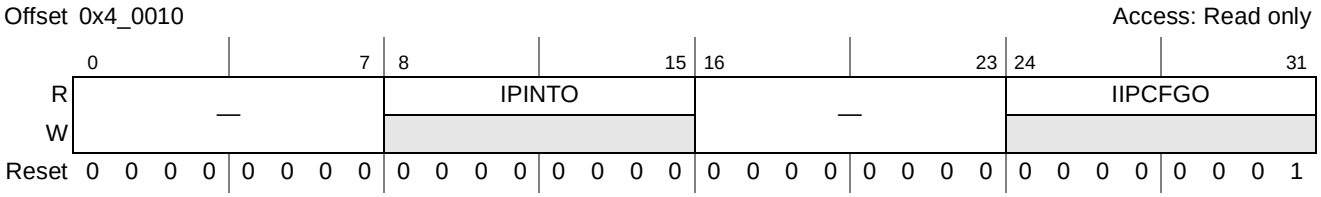


Figure 9-4. Block Revision Register 2 (BRR2)

Table 9-6 describes the BRR2 fields.

Table 9-6. BRR2 Field Descriptions

Bits	Name	Description
0–7	—	Reserved, should be cleared.
8–15	IPINTO	IP block integration options
16–23	—	Reserved, should be cleared.
24–31	IPCFGO	IP block configuration options

9.3.1.3 Feature Reporting Register (FRR)

FRR, shown in Figure 9-5, provides information about interrupt and processor core configurations. It also informs the programming environment of the controller version.

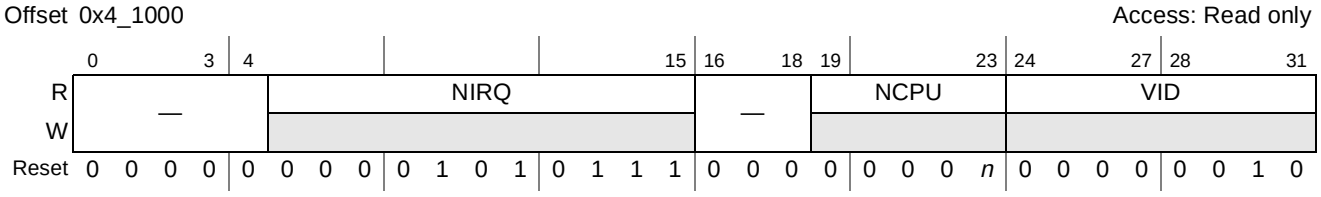


Figure 9-5. Feature Reporting Register (FRR)

Table 9-7 describes the FRR fields.

Table 9-7. FRR Field Descriptions

Bits	Name	Description
0–4	—	Reserved, should be cleared.
5–15	NIRQ	Number of interrupts. Holds the binary value of the number of the highest interrupt source supported minus one. The value is 87 (0x57 or 0b000_0101_0111) because this device supports 88 interrupts: 12 external sources, 48 internal sources (see Table 9-3), 8 timer sources, 8 interprocessor sources, 4 messaging sources, and 8 shared message signaled sources. A zero in this field corresponds to one source.
16–18	—	Reserved, should be cleared
19–23	NCPU	Number of CPUs. The number of the highest physical CPUs (or processor cores) supported minus one. 00000MPC8641. Single core—core0 00001MPC8641D. Two cores—core0 and core1
24–31	VID	Version ID. Reports the OpenPIC specification revision level supported by this interrupt controller implementation. The VID field's value of two (0x02) corresponds to revision 1.2 which is the revision level currently supported.

9.3.1.4 Global Configuration Register (GCR)

GCR, shown in Figure 9-6, controls the PIC's operating mode, and allows software to reset the PIC.

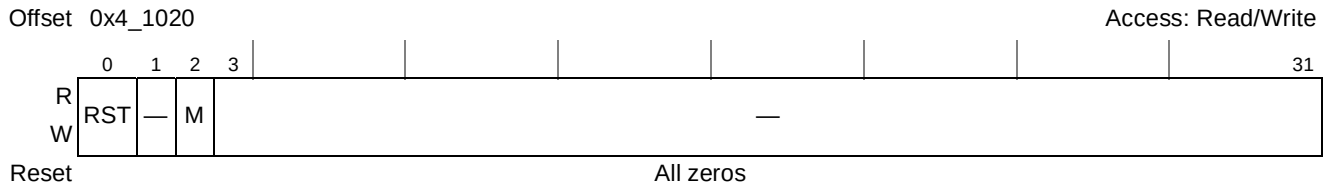


Figure 9-6. Global Configuration Register (GCR)

Table 9-8 describes the GCR fields.

Table 9-8. GCR Field Descriptions

Bits	Name	Description
0	RST	Reset. Setting RST forces the PIC to be reset. Cleared automatically when the reset sequence is complete. See Section 9.4.8, "Resetting the PIC," for more information.
1	—	Reserved, should be cleared.
2	M	Mode. PIC operating mode. Section 9.1.4, "Modes of Operation," provides details about these modes. 0 Pass-through mode. On-chip PIC is disabled and interrupts detected on IRQ0 are passed directly to core 0. 1 Mixed mode. Interrupts are handled by the normal priority and delivery mechanisms of the PIC.
3–31	—	Reserved, should be cleared.

9.3.1.5 Vendor Identification Register (VIR)

VIR, shown in Figure 9-7, is defined by the OpenPIC specifications and is provided for compliance. The zero value for VIR[VENDORID] indicates a generic OpenPIC-compliant device, which makes the other VIR fields meaningless.

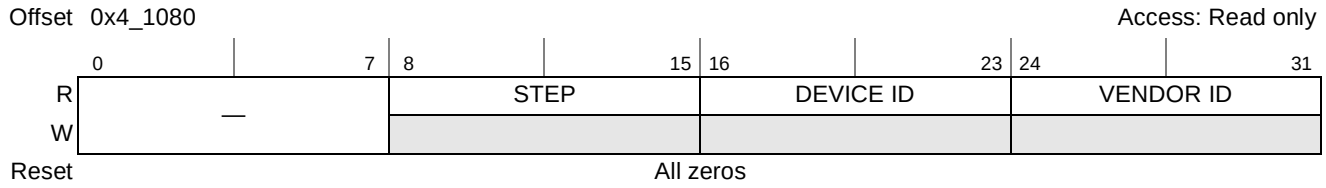


Figure 9-7. Vendor Identification Register (VIR)

Table 9-9 describes the VIR fields.

Table 9-9. VIR Field Descriptions

Bits	Name	Description
0–7	—	Reserved, should be cleared.
8–15	STEP	Stepping. Indicates the silicon revision for this device. Has no meaning if VENDOR ID value is zero.
16–23	DEVICE ID	Device identification. Vendor-specified identifier for this device. Has no meaning if VENDOR ID is zero.
24–31	VENDOR ID	Vendor identification. Specifies the manufacturer of this part. A value of zero implies a generic OpenPIC-compliant device.

9.3.1.6 Processor Core Initialization Register (PIR)

PIR, shown in Figure 9-8, provides a way for software to generate a core reset. Setting P1 or P0 causes the respective *sreset1* or *sreset0* signal to assert. These signals are routed to the appropriate *core_sreset* signals. Note that after requesting a core reset using this register the applicable bit should not be cleared until the requested core reset has occurred. However, if one core is used to reset another one, the core being reset can effectively be held off indefinitely from issuing its initial boot vector fetch to the platform by leaving its appropriate PIR[Px] bit asserted. Clearing it releases the core to fetch.Note that although the OpenPIC architecture was defined to support up to 32 processing cores, only fields corresponding to processors on the device are implemented.

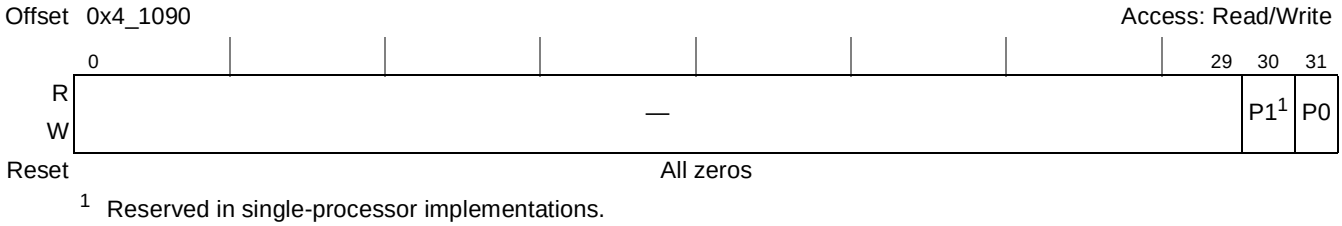


Figure 9-8. Processor Core Initialization Register (PIR)

Table 9-10 describes the PIR fields.

Table 9-10. PIR Field Descriptions

Bits	Name	Description
0–29	—	Reserved, should be cleared.
30	P1	Processor core 1 reset. Setting this bit causes the PIC to assert the *sreset1* signal. Reserved in single-processor implementations.
31	P0	Processor core 0 reset. Setting this bit causes the PIC to assert the *sreset0* signal.

9.3.1.7 Processor Reset Register (PRR)

The PRR provides a way for software on another device to perform the equivalent of a hardware reset (*hreset*) to the processor cores by writing to an PIC register. A *core0_hreset* and *core1_hreset* signal to the cores is asserted when a one is written to the corresponding processor reset field; these signals are held active until a zero is written to those same bits. This produces a pulse to the *core_hreset* input of the corresponding core or cores. Because of hardware-specific requirements on the width of such a pulse, to ensure successful hardware reset, the system must ensure a minimum adequate pulse width on *core_hreset*.

For proper system operation, a core should be reset in this way only if the core is already in nap or sleep state. Because a core in either state cannot perform the necessary write to cause a soft reset, a core cannot put itself into soft reset.

Note that although the OpenPIC architecture is defined to support up to 32 processing cores, only fields corresponding to processors on the device are implemented, as shown in Figure 9-9.

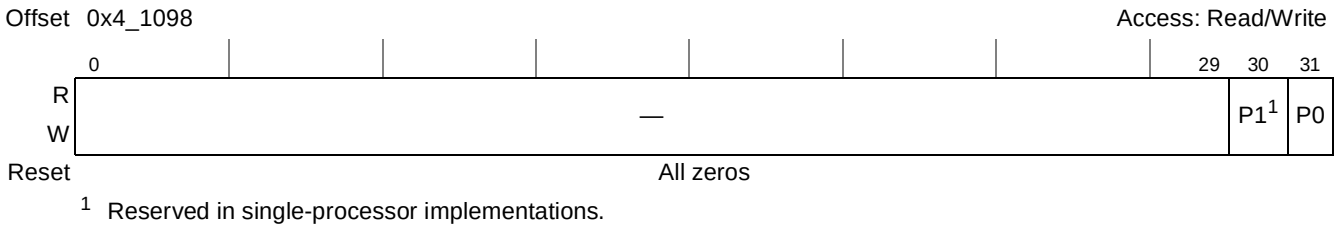


Figure 9-9. Processor Reset Register (PRR)

Table 9-11 describes the PRR fields.

Table 9-11. PRR Field Descriptions

Bits	Name	Description
0–29	—	Reserved, should be cleared.
30	P1	Processor 1 core reset. Setting P1 causes the PIC to assert the *core1_hreset* signal to processor 1. Reserved in single-processor implementations.
31	P0	Processor 0 core reset. Setting P0 causes the PIC to assert the *core0_hreset* signal to processor 0.

9.3.1.8 Interprocessor Interrupt Vector/Priority Registers (IPIVPR0–IPIVPR3)

IPIVPRs, shown in Figure 9-10, contain the interrupt vector and priority fields for the four interprocessor interrupt channels. There is one vector/priority register per channel. The VECTOR and PRIORITY values should not be changed while IPIVPR*n*[A] is set.

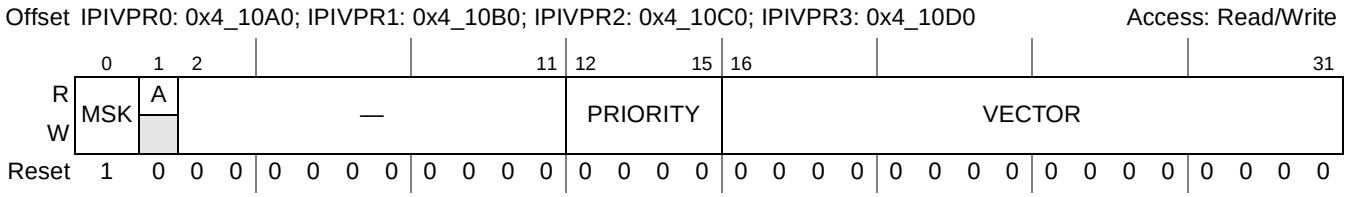


Figure 9-10. Interprocessor Interrupt Vector/Priority Register (IPIVPR*n*)

Table 9-12 describes the IPIVPR*n* fields.

Table 9-12. IPIVPR*n* Field Descriptions

Bits	Name	Description
0	MSK	Mask. Mask interrupts from this source. MSK affects only interrupts routed to *int*. 0 An interrupt request is generated if the corresponding IPR bit is set. 1 Further interrupts from this source are disabled.
1	A	Activity. Indicates an interrupt has been requested or is in service. The VECTOR and PRIORITY values should not be changed while this bit is set. Affects only interrupts routed to *int*. 0 No current interrupt activity associated with this source. 1 The interrupt field for this source is set in the IPR or ISR.
2–11	—	Reserved, should be cleared.
12–15	PRIORITY	Priority. Specifies the interrupt priority. The lowest priority is 0 and the highest priority is 15. A priority level of 0 inhibits signalling of this interrupt to the core. Affects only interrupts routed to *int*.
16–31	VECTOR	Vector (Affects only interrupts routed to *int*). Contains the value returned when IACK is read and this interrupt resides in the corresponding interrupt request register (IRR) for that core, as shown in Figure 9-51.

9.3.1.9 Spurious Vector Register (SVR)

SVR, shown in Figure 9-11, contains the 16-bit vector returned to the processor core when the corresponding IACK register is read for a spurious interrupt.

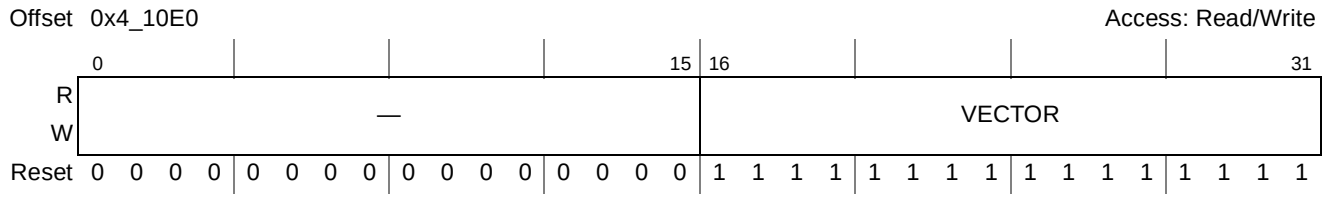


Figure 9-11. Spurious Vector Register (SVR)

Table 9-13 describes the SVR fields.

Table 9-13. SVR Field Descriptions

Bits	Name	Description
0–15	—	Reserved, should be cleared.
16–31	VECTOR	Spurious interrupt vector. Value returned when IACK is read during a spurious vector fetch. Section 9.4.1.2.4, "Spurious Vector Generation," gives information about the conditions that may cause a spurious vector fetch.

9.3.2 Global Timer Registers

The two independent groups of global timer registers, group A and group B, are identical in their functionality, except that they appear at different locations within the PIC register map. Note that each of the four timers within an *x* group have four individual configuration registers (GTCCR*xn*, GTBCR*xn*, GTVPR*xn*, GTDR*xn*), but they are only shown once in this section. These two groups of timers cannot be cascaded together.

9.3.2.1 Timer Frequency Reporting Register (TFRRA–TFRRB)

The TFRRs, shown in Figure 9-12, are written by software to report the clocking frequency of the PIC timers. Note that although TFRRs are read/write, the PIC ignores the register values.

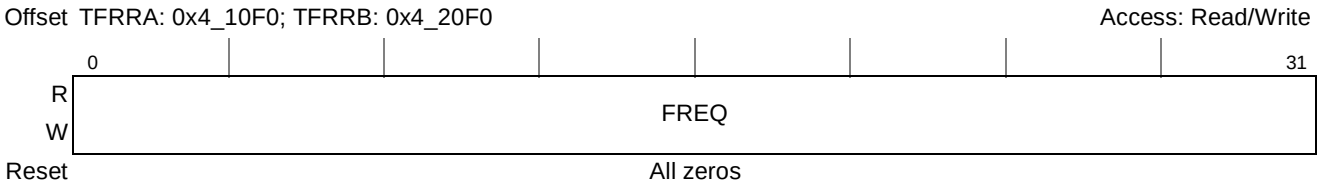


Figure 9-12. Timer Frequency Reporting Registers (TFRR*x*)

Table 9-14 describes the TFRR*x* registers.

Table 9-14. TFRR*x* Field Descriptions

Bits	Name	Description
0–31	FREQ	Timer frequency (in ticks/second (Hz)). Used to communicate the frequency of the global timers' clock source, (either the MPX clock or the frequency of the RTC signal), to user software. TFRR*x* is set only by software for later use by other applications and its value in no way affects the operating frequency of the global timers. The timers operate at a ratio of this clock frequency, as set by TCR*x*[CLKR]. See Section 9.3.2.6, "Timer Control Registers (TCRA–TCRB)."

9.3.2.2 Global Timer Current Count Registers (GTCCRA0–GTCCRA3, GTCCRB0–GTCCRB3)

The GTCCRs, shown in Figure 9-13, contain the current count for each of the four PIC timers in each of the two groups.

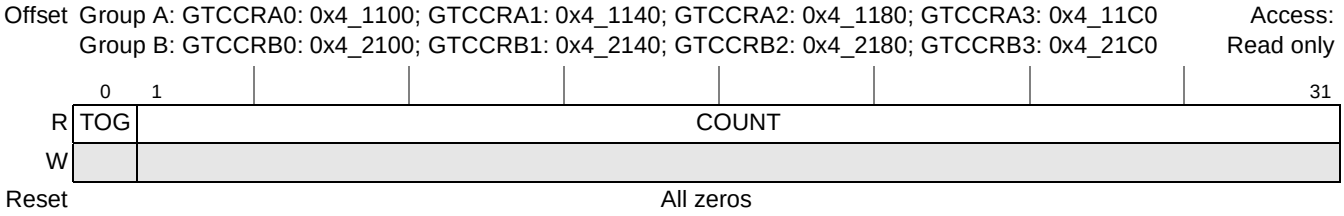


Figure 9-13. Global Timer Current Count Registers (GTCCR*xn*)

Table 9-15 describes the GTCCR*xn* fields.

Table 9-15. GTCCR*xn* Field Descriptions

Bits	Name	Description
0	TOG	Toggle. Toggles when the current count decrements to zero. Cleared when GTBCR*xn*[CI] goes from 1 to 0.
1–31	COUNT	Current count. Decremented while GTBCR*xn*[CI] is zero. When the timer count reaches zero, an interrupt is generated (provided it is not masked), the toggle bit is inverted, and the count is reloaded. For non-cascaded timers, the reload value is the contents of the corresponding GTBCR*xn*. Cascaded timers are reloaded with either all ones, or the GTBCR*xn* contents, depending on the value of TCR*n*[ROVR]. See Section 9.3.2.6, “Timer Control Registers (TCRA–TCRB),” for more details.

9.3.2.3 Global Timer Base Count Registers (GTBCRA0–GTBCRA3, GTBCRB0–GTBCRB3)

The GTBCRs contain the base counts for each of the four PIC timers in each of the two groups, as shown in Figure 9-14. This value is reloaded into the corresponding GTCCR*xn* when the current count reaches zero. Note that when zero is written to the base count field, (and GTCCR*xn*[CI] = 0), the timer generates an interrupt on every timer cycle.

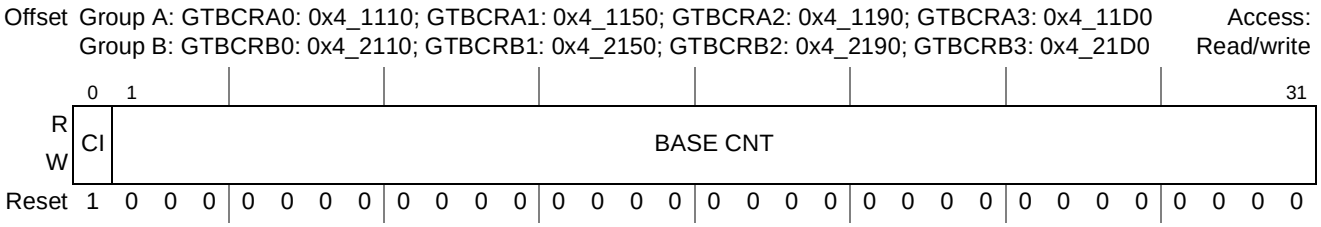


Figure 9-14. Global Timer Base Count Register (GTBCR*xn*)

Table 9-16 describes the GTBCR*xn* fields.

Table 9-16. GTBCR*xn* Field Descriptions

Bits	Name	Description
0	CI	Count inhibit. Always set following reset 0 Counting enabled 1 Counting inhibited
1–31	BASE CNT	Base count. When CI transitions from 1 to 0, this value is copied into the corresponding GTCCR*xn* and the toggle bit is cleared. If CI is already cleared (counting is in progress), the base count is copied to the GTCCR*xn* at the next zero crossing of the current count.

9.3.2.4 Global Timer Vector/Priority Registers (GTVPRA0–GTVPRA3, GTVPRB0–GTVPRB3)

The GTVPRs contain the interrupt vector and the interrupt priority values for the timers as shown in Figure 9-15. They also contain the mask and activity fields for all the timers.

Offset Group A: GTVPRA0: 0x4_1120; GTVPRA1: 0x4_1160; GTVPRA2: 0x4_11A0; GTVPRA3: 0x4_11E0; Access:
Group B: GTVPRB0: 0x4_2120; GTVPRB1: 0x4_2160; GTVPRB2: 0x4_21A0; GTVPRB3: 0x4_21E0 Read/Write

	0	1	2–11	12–15	16–31
R	MSK	A	—	PRIORITY	VECTOR
W					
Reset	1	0	0 0 0 0 0 0 0 0 0 0	0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Figure 9-15. Global Timer Vector/Priority Register (GTVPR*xn*)

Table 9-17 describes the GTVPR*xn* fields.

Table 9-17. GTVPR*xn* Field Descriptions

Bits	Name	Description
0	MSK	Mask. Mask interrupts from this source. MSK affects only interrupts routed to *int*. 0 An interrupt request is generated if the corresponding IPR bit is set. 1 Further interrupts from this source are disabled.
1	A	Activity. Indicates an interrupt has been requested or is in service. The VECTOR and PRIORITY values should not be changed while this bit is set. Affects only interrupts routed to *int*. 0 No current interrupt activity associated with this source. 1 The interrupt field for this source is set in the IPR or ISR.
2–11	—	Reserved, should be cleared.
12–15	PRIORITY	Priority. Specifies the interrupt priority. The lowest priority is 0 and the highest priority is 15. A priority level of 0 inhibits signalling of this interrupt to the core. Affects only interrupts routed to *int*.
16–31	VECTOR	Vector (Affects only interrupts routed to *int*). Contains the value returned when IACK is read and this interrupt resides in the corresponding interrupt request register (IRR) for that core, as shown in Figure 9-51.

9.3.2.5 Global Timer Destination Registers (GTDRA0–GTDRA3, GTDRB0–GTDRB3)

The GTDR*xn* registers, shown in Figure 9-16, control the destination (core) to which each timer's interrupt is directed. Note that GTDR*xn* bits can be set independently of each other and that either P1 or P0 or both can be set for this type of interrupt.

Offset Group A: GDTRA0: 0x4_1130; GDTRA1: 0x4_1170; GDTRA2: 0x4_11B0; GDTRA3: 0x4_11F0; Access:
Group B: GDTRB0: 0x4_2130; GDTRB1: 0x4_2170; GDTRB2: 0x4_21B0; GDTRB3: 0x4_21F0 Read/Write

	0																								29	30	31
R W	—																									P1[1]	P0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

1 Reserved in single-processor implementations.

Figure 9-16. Global Timer Destination Registers (GTDR*xn*)

Table 9-18 describes the GTDR*xn* fields.

Table 9-18. GTDR*xn* Field Descriptions

Bits	Name	Description
0–29	—	Reserved, should be cleared.
30	P1	Processor core 1. This interrupt is multicasting, so both P0 and P1 can be set. Reserved in single-processor implementations. 0 Processor core 1 does not receive this interrupt 1 Directs the timer interrupt to processor core 1.
31	P0	Processor core 0. Default destination after PIC is reset. Both P0 and P1 can be set. 0 Processor core 0 does not receive this interrupt. 1 Directs the timer interrupt to processor core 0.

9.3.2.6 Timer Control Registers (TCRA–TCRB)

The TCR registers, shown in Figure 9-18, provide various configuration options such as count frequency and roll-over behavior for the timers.

There are two choices for the clock source for the timers: a selectable frequency ratio from the MPX bus clock, or the RTC signal. TCRs can be cascaded to create timers larger than the default 31-bit global timers. Timer cascade fields allow configuration of up to two 63-bit timers, one 95-bit timer, or one 127-bit timer (within each group).

With one exception mentioned below, the value reloaded into a timer is determined by its roll-over control field, TCR*x*[ROVR]. Setting TCR*x*[ROVR] causes its GTCCR*xn* to roll over to all ones when the count reaches zero. This is equivalent to reloading the count register with 0xFFFF_FFFF instead of its base count value. Clearing a timer's associated ROVR bit ensures the timer always reloads with its base count value.

When timers are cascaded, the last (most significant) counter in the cascade also affects their roll-over behavior. Cascaded timers always reload their base count when the most significant counter has decremented to zero, regardless of the TCR*x*[ROVR] settings.

For example, timers 0–2 can be cascaded to generate one interrupt per hour. As shown in Table 9-19, given an MPX clock frequency of 333 MHz, letting the timer clock frequency default to 1/8th the system clock, (TCR*x*[CLKR] = 0 sets a clock ratio of 8), provides a basic input of 41.625 MHz to timer 0. Setting timer 0 to count 41,625,000 (0x27B_25A8) timer clock cycles generates one output per second. Setting both timers 1 and 2 to 59, and cascading all three timers, generates one interrupt every hour from timer 2.

Table 9-19. 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 x 10^6 (0x027B_25A8)	59[1] (0x0000_0036)	59 (0x0000_0036)

[1] Counting down from 59 through 0 requires 60 ticks.

$$(41.625 \text{ x } 106 \text{ ticks/sec}) \times (60 \text{ sec/min}) \times (60 \text{ min/hr}) = \text{total ticks/hr generating 1 interrupt/hr}$$

Figure 9-17. Example Calculation for Cascaded Timers

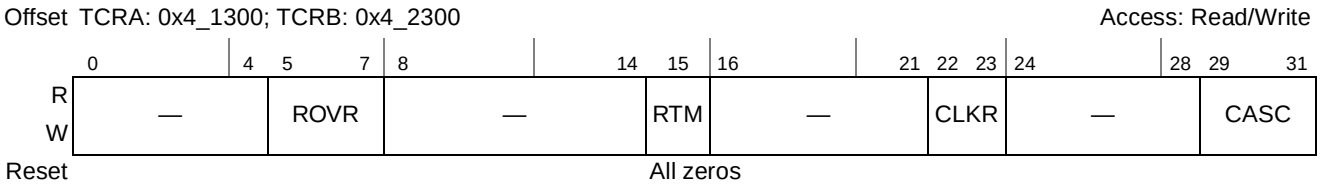


Figure 9-18. Timer Control Registers (TCR*x*)

Table 9-20 describes the TCR*x* fields.

Table 9-20. TCR*x* Field Descriptions

Bits	Name	Description
0–4	—	Reserved, should be cleared.
5–7	ROVR	Roll-over control for cascaded timers only. Specifies behavior when count reaches zero by identifying the source of the reload value. Cascaded timers are always reloaded with their base count value when the more significant timer in the cascade (the upstream timer) is zero. Bits 5–7 correspond to timers 2–0. Note that global timer 3 always reloads with its GTBCR*xn*. 0 The timer does not roll over. When the count reaches zero, GTCCR*xn* is reloaded with the GTBCR*xn* value. 1 Timer rolls over at zero to all ones. (When the count reaches zero, GTCCR*xn* is reloaded with 0xFFFF_FFFF.) 000 All timers reload with base count. 001 Timers 1 and 2 reload with base count, timer 0 rolls over (reloads with 0xFFFF_FFFF). 010 Timers 0 and 2 reload with base count, timer 1 rolls over (reloads with 0xFFFF_FFFF). 011 Timer 2 reloads with base count, timers 0 and 1 roll over (reload with 0xFFFF_FFFF). 100 Timers 0 and 1 reload with base count, timer 2 rolls over (reloads with 0xFFFF_FFFF). 101 Timer 1 reloads with base count, timers 0 and 2 roll over (reload with 0xFFFF_FFFF). 110 Timer 0 reloads with base count, timers 1 and 2 roll over (reload with 0xFFFF_FFFF). 111 Timers 0, 1, and 2 roll over (reload with 0xFFFF_FFFF).
8–14	—	Reserved, should be cleared.
15	RTM	Real time mode. Specifies the clock source for the PIC timers. 0 Timer clock frequency is a ratio of the frequency of the MPX 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.

Table 9-20. TCR*x* Field Descriptions (continued)

Bits	Name	Description
16–21	—	Reserved, should be cleared.
22–23	CLKR	Clock ratio. Specifies the ratio of the timer frequency to the MPX clock. The following are supported: 00 Default. Divide by 8 01 Divide by 16 10 Divide by 32 11 Divide by 64
24–28	—	Reserved, should be cleared.
29–31	CASC	Cascade timers. Specifies the output of particular global timers as input to others. 000 Default. Timers not cascaded 001 Cascade timers 0 and 1 010 Cascade timers 1 and 2 011 Cascade timers 0, 1, and 2 100 Cascade timers 2 and 3 101 Cascade timers 0 and 1; timers 2 and 3 110 Cascade timers 1, 2, and 3 111 Cascade timers 0, 1, 2, and 3

9.3.3 IRQ_OUT and Critical Interrupt Summary Registers

The summary registers indicate the specific interrupt sources routed to the $\overline{\text{IRQ\_OUT}}$ or *cint0/cint1*. PIC outputs. Summary register bits are cleared when the corresponding interrupt that caused a bit to be set is negated. Note that only level-sensitive interrupts can be routed to $\overline{\text{IRQ\_OUT}}$ or *cint0* and *cint1*.

The $\overline{\text{IRQ\_OUT}}$ summary registers, shown in Figure 9-20 through Figure 9-22 contain one bit for each interrupt source that can be routed to $\overline{\text{IRQ\_OUT}}$. The corresponding bit is set if the interrupt is active and is routed to $\overline{\text{IRQ\_OUT}}$ (that is, if the corresponding *x*IDR*n*[EP] is set).

The critical interrupt summary registers, shown in Figure 9-23 through Figure 9-25, contain one bit for each interrupt source that can be designated as a critical interrupt. The corresponding bit is set if the interrupt is active and is routed to either the *cint0* or *cint1* outputs of the PIC (if *x*IDR*n*[CI*n*] = 1 in its corresponding destination register).

9.3.3.1 External Interrupt Summary Register (ERQSR)

NOTE

ERQSR fields report only the current state of IRQ0–IRQ11. These fields were designed to work with level-sensitive interrupts; values returned for edge-sensitive interrupts may be unreliable.

Figure 9-19 shows the ERQSR fields.

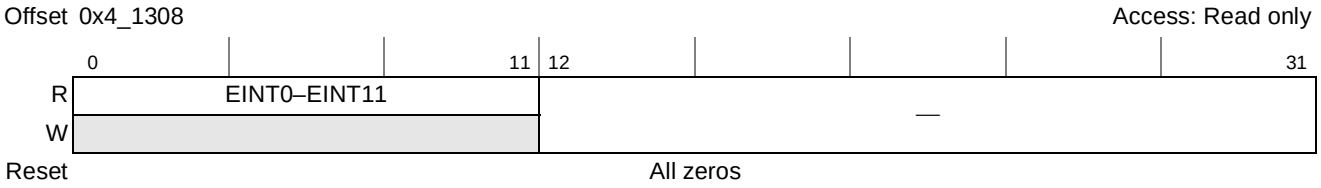


Figure 9-19. External Interrupt Summary Register (ERQSR)

Table 9-21 describes the ERQSR fields.

Table 9-21. ERQSR Field Descriptions

Bits	Name	Description
0–11	EINT*n*	External interrupts signal 0–11 status. Bit 0 represents EINT0. Bit 11 represents EINT11. 0 The corresponding external interrupt signal is not active. 1 The corresponding external interrupt signal is active.
12–31	—	Reserved, should be cleared.

9.3.3.2 IRQ_OUT Summary Register 0 (IRQSR0)

Figure 9-20 shows the IRQSR0 fields.

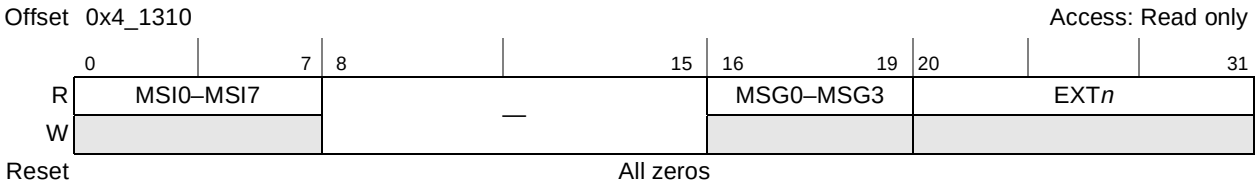


Figure 9-20. IRQ_OUT Summary Register 0 (IRQSR0)

Table 9-22 describes the IRQSR0 fields.

Table 9-22. IRQSR0 Field Descriptions

Bits	Name	Description
0–7	MSI*n*	Shared message signaled interrupt *n* status 0 Interrupt is not active or not routed to $\overline{\text{IRQ\_OUT}}$. 1 Interrupt is active and is routed to the $\overline{\text{IRQ\_OUT}}$ signal (that is, if the corresponding *x*IDR*n*[EP] is set).
8–15	—	Reserved, should be cleared.

Table 9-22. IRQSR0 Field Descriptions (continued)

Bits	Name	Description
16–19	MSG*n*	Message interrupt *n* status 0 Interrupt is not active or not routed to $\overline{\text{IRQ\_OUT}}$. 1 Interrupt is active and is routed to the $\overline{\text{IRQ\_OUT}}$ signal (that is, if the corresponding *x*IDR*n*[EP] is set).
20–31	EXT*n*	External interrupts 0–11. Each bit corresponds to a unique interrupt according to the following: Bit Interrupt 20 IRQ0 21 IRQ1 … 31 IRQ11 0 The corresponding interrupt is not active or not routed to $\overline{\text{IRQ\_OUT}}$. 1 The corresponding interrupt is active and routed to $\overline{\text{IRQ\_OUT}}$ (if the corresponding *x*IDR*n*[EP] is set).

9.3.3.3 IRQ_OUT Summary Register 1 (IRQSR1)

Figure 9-21 shows the IRQSR1 fields.

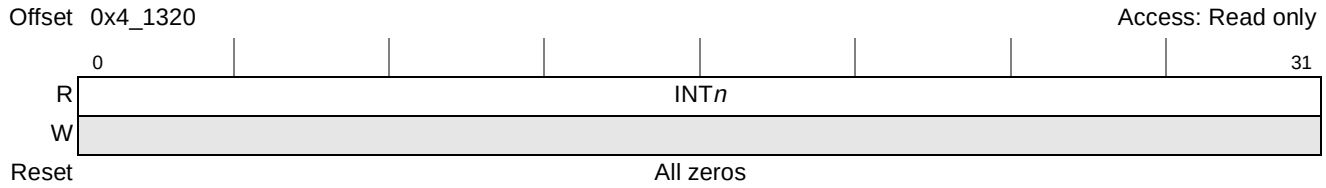


Figure 9-21. IRQ_OUT Summary Register 1 (IRQSR1)

Table 9-23 describes the IRQSR1 fields.

Table 9-23. IRQSR1 Field Descriptions

Bits	Name	Description
0–31	INT*n*	Internal interrupts 0–31 status. Bit 0 represents INT0. Bit 31 represents INT31. 0 The corresponding interrupt is not active or not routed to $\overline{\text{IRQ\_OUT}}$. 1 The corresponding interrupt is active and is routed to $\overline{\text{IRQ\_OUT}}$ (that is, if the corresponding *x*IDR*n*[EP] is set).

9.3.3.4 IRQ_OUT Summary Register 2 (IRQSR2)

Figure 9-22 shows the IRQSR2 fields.

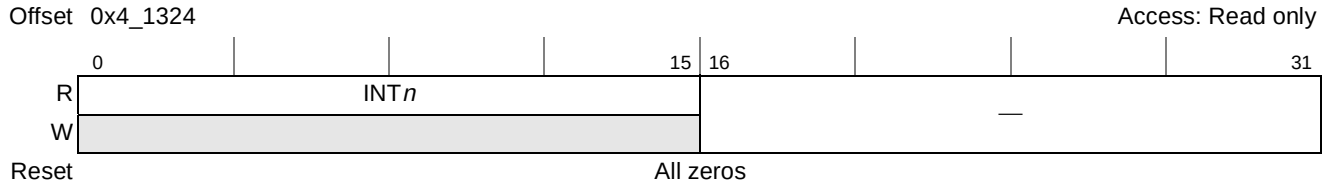


Figure 9-22. IRQ_OUT Summary Register 2 (IRQSR2)

Table 9-24 describes the IRQSR2 fields.

Table 9-24. IRQSR2 Field Descriptions

Bits	Name	Description
0–15	INT*n*	Internal interrupts 32–47 status. Bit 0 represents INT32. Bit 15 represents INT47. 0 The corresponding interrupt is not active or not routed to $\overline{\text{IRQ\_OUT}}$. 1 The corresponding interrupt is active and is routed to $\overline{\text{IRQ\_OUT}}$, if the corresponding *x*IDR*n*[EP] is set.
16–31	—	Reserved, should be cleared.

9.3.3.5 Critical Interrupt Summary Register 0 (CISR0)

Figure 9-23 shows CISR0.

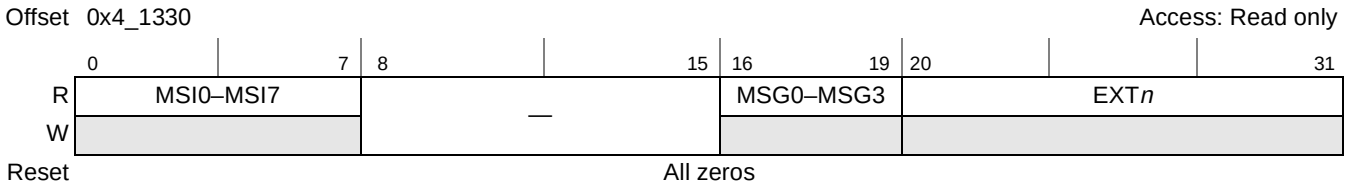


Figure 9-23. Critical Interrupt Summary Register 0 (CISR0)

Table 9-25 describes CISR0 fields.

Table 9-25. CISR0 Field Descriptions

Bits	Name	Description
0–7	MSI*n*	Shared message signaled interrupts 0–7. Bit 0 represents MSI0; bit 7 represents MSI7. 0 The corresponding interrupt is not active or not routed to *cint0* or *cint1*. 1 The corresponding interrupt is active and is routed to *cint0* or *cint1*, if the corresponding *x*IDR*n*[CI] is set.
8–15	—	Reserved, should be cleared.
16–19	MSG*n*	Message interrupts 0–3. Bit 16 represents MSG0; bit 19 represents MSG3. 0 The corresponding interrupt is not active or not routed to *cint0* or *cint1*. 1 The corresponding interrupt is active and is routed to *cint0* or *cint1* (if the corresponding *x*IDR*n*[CI] is set).
20–31	EXT*n*	External interrupts 0–11. Bit 20 represents IRQ0. Bit 31 represents IRQ11. 0 The corresponding interrupt is not active or not routed to *cint0* or *cint1*. 1 The corresponding interrupt is active and is routed to *cint0* or *cint1* (if the corresponding *x*IDR*n*[CI] is set).

9.3.3.6 Critical Interrupt Summary Register 1 (CISR1)

Figure 9-24 shows the CISR1.

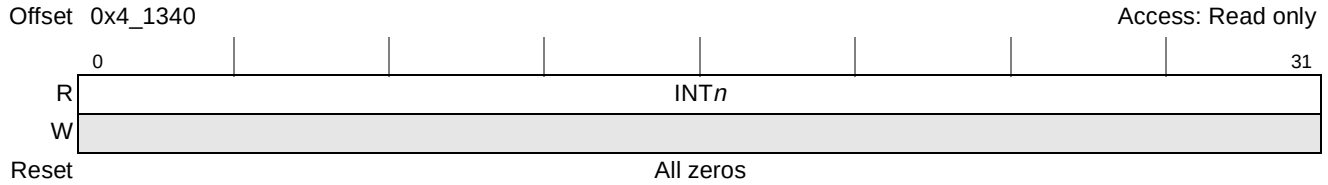


Figure 9-24. Critical Interrupt Summary Register 1 (CISR1)

Table 9-26 describes CISR1.

Table 9-26. CISR1 Field Descriptions

Bits	Name	Description
0–31	INT*n*	Internal interrupts 0–31. Bit 0 represents INT0. Bit 31 represents INT31. 0 Corresponding interrupt is not active or not routed to *cint0* or *cint1*. 1 The corresponding interrupt is active and is routed to the *cint0* or *cint1* (if the corresponding *x*IDR*n*[CI] is set).

9.3.3.7 Critical Interrupt Summary Register 2 (CISR2)

Figure 9-25 shows the CISR2.

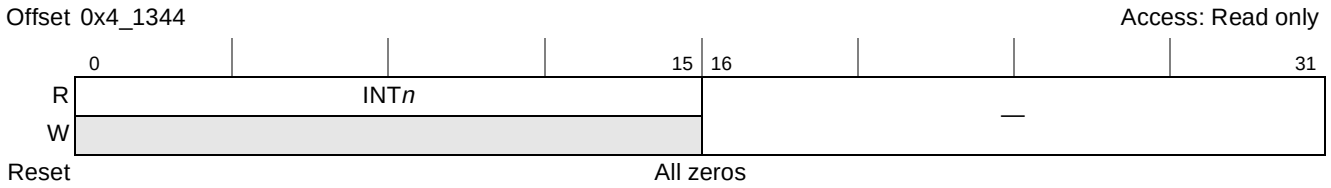


Figure 9-25. Critical Interrupt Summary Register 2 (CISR2)

Table 9-27 describes CISR2.

Table 9-27. CISR2 Field Descriptions

Bits	Name	Description
0–15	INT*n*	Internal interrupts 32–48. Bit 0 represents INT32. Bit 15 represents INT47. 0 Corresponding interrupt is not active or not routed to *cint0* or *cint1*. 1 The corresponding interrupt is active and is routed to the *cint0* or *cint1*, if the corresponding *x*IDR*n*[CI] is set.
16–31	—	Reserved, should be cleared.

9.3.4 Performance Monitor Mask Registers (PMMRs)

The twelve performance monitor mask registers consist of four sets of three 32-bit registers, PM*n*MR0, PM*n*MR1, and PM*n*MR2. Each set can be configured to select one interrupt source (interprocessor, timer, message, shared message signaled, external, or internal) to generate a performance monitor event. The performance monitor can be configured to track this event in the performance monitor local control registers. See Section 18.3.2.2, “Performance Monitor Local Control Registers (PMLCAn, PMLCBn).”

9.3.4.1 Performance Monitor Mask Registers 0 (PM0MR0–PM3MR0)

Each PM*n*MR0 register, shown in Figure 9-26, is matched with a PM*n*MR1 and a PM*n*MR2 register. Because each unreserved bit in the 96-bit vector (PM*n*MR0/1/2) specifies a different interrupt, only one bit in the 96-bit vector can be unmasked at a time. Unmasking more than one bit per set is considered a programming error and results in unpredictable behavior.

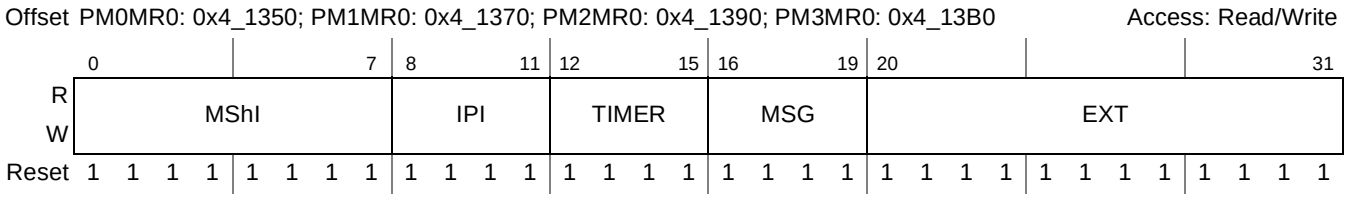


Figure 9-26. Performance Monitor Mask Registers 0 (PM*n*MR0)

Table 9-28 describes the PM*n*MR0 fields.

Table 9-28. PM*n*MR0 Field Descriptions

Bits	Name	Description
0–7	MShI	Shared message signaled interrupts 0–7 0 The corresponding interrupt source generates a performance monitor event when the interrupt occurs. 1 The corresponding interrupt does not generate a performance monitor event.
8–11	IPI	Interprocessor interrupts 0–3 0 The corresponding interrupt source generates a performance monitor event when the interrupt occurs. 1 The corresponding interrupt does not generate a performance monitor event.
12–15	TIMER	Timer interrupts 0–3 (Group A and Group B: Each bit represents an OR of the event for the correspondingly numbered timer in Group A and that in Group B). 0 The corresponding interrupt source generates a performance monitor event when the interrupt occurs. 1 The corresponding interrupt does not generate a performance monitor event.
16–19	MSG	Message interrupts 0–3 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.

9.3.4.2 Performance Monitor Mask Registers 1 (PM0MR1–PM3MR1)

Figure 9-27 shows the PM*n*MR1 registers.

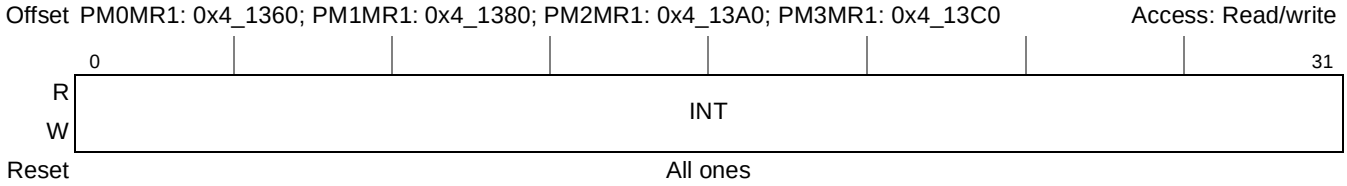


Figure 9-27. Performance Monitor Mask Registers 1 (PM*n*MR1)

Table 9-29 describes the PM*n*MR1 registers.

Table 9-29. 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.

9.3.4.3 Performance Monitor Mask Registers 2 (PM0MR2–PM3MR2)

Figure 9-28 shows the PM*n*MR2 registers.

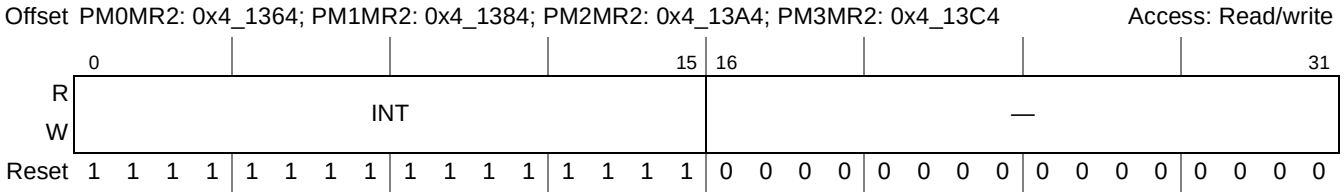


Figure 9-28. Performance Monitor Mask Registers 2 (PM*n*MR2)

Table 9-30 describes the PM*n*MR2 registers.

Table 9-30. PM*n*MR2 Field Descriptions

Bits	Name	Description
0–15	INT	Internal interrupts 32–47 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.
16–31	—	Reserved, should be cleared.

9.3.5 Message Registers

The following registers support the message register interrupts:

- Section 9.3.5.1, “Message Registers (MSGR0–MSGR3)”
- Section 9.3.5.2, “Message Enable Register (MER)”
- Section 9.3.5.3, “Message Status Register (MSR)”
- Section 9.3.7.5, “Messaging Interrupt Vector/Priority Registers (MIVPRn)”
- Section 9.3.7.6, “Messaging Interrupt Destination Registers (MIDR0–MIDR3)”

Writing to one of the four message registers (MSGR0–MSGR3) causes a messaging interrupt as directed by the other message registers listed above. Reading a message register clears the messaging interrupt. Note that a messaging interrupt can also be cleared by writing a one to the corresponding status field of the PIC message status register (MSR), shown in Figure 9-31.

9.3.5.1 Message Registers (MSGR0–MSGR3)

The message registers (MSGR0–MSGR3), shown in Figure 9-29, can contain a 32-bit message.

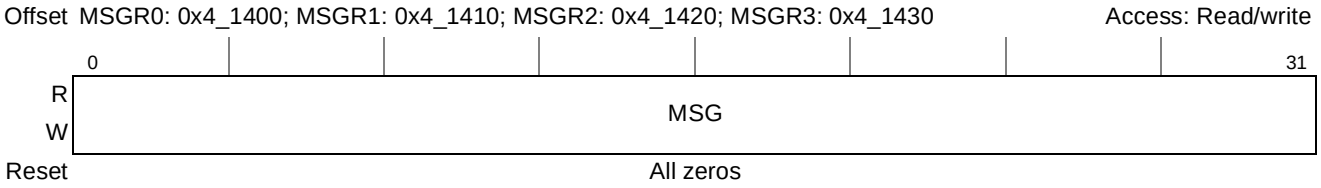


Figure 9-29. Message Registers (MSGRs)

Table 9-31 describes the MSGR registers.

Table 9-31. MSGR*n* Field Descriptions

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

9.3.5.2 Message Enable Register (MER)

The MER, shown in Figure 9-30, contains the enable bits for each message register. The enable bit must be set to enable interrupt generation when the corresponding message register is written.

When bits in MER are set to mask message interrupts, an interrupt is not generated if the message register is written while it is masked in MER and the MER bit is then cleared. To mask the interrupt without loss, set MIVPR*n*[MSK]. (See Section 9.3.7.5, “Messaging Interrupt Vector/Priority Registers (MIVPRn).”) MER should be set to 0x0000_000F at reset and then left unchanged during normal operation.

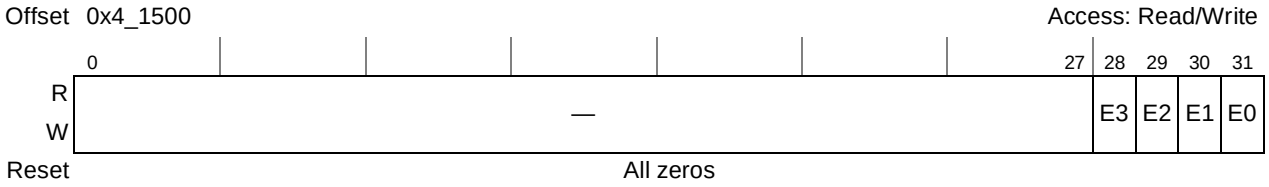


Figure 9-30. Message Enable Register (MER)

Table 9-32 describes the MER fields.

Table 9-32. MER Field Descriptions

Bits	Name	Description
0–27	—	Reserved, should be cleared.
28–32	E*n*	Enable 3–enable 0. Used to enable interrupt generation for MSGR*n* (where *n* = 0–3). 0 Interrupt generation for MSGR*n* disabled. 1 Interrupt generation for MSGR*n* enabled.

9.3.5.3 Message Status Register (MSR)

The message status register (MSR) shown in Figure 9-31 contains status bits for each message register. A status bit is set when the corresponding messaging interrupt is active. Writing a 1 to a status bit clears the corresponding message interrupt and the status bit.

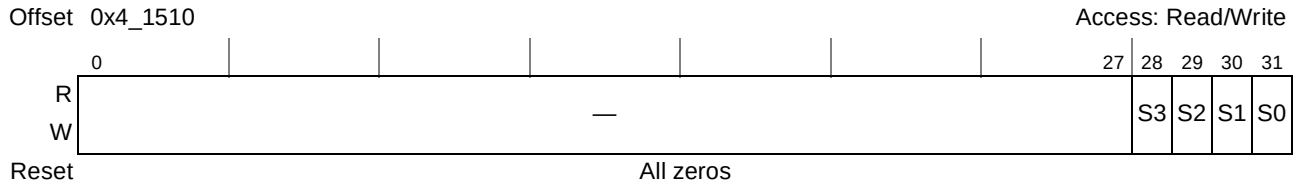


Figure 9-31. Message Status Register (MSR)

Table 9-33 describes the MSR fields.

Table 9-33. MSR Field Descriptions

Bits	Name	Description
0–27	—	Reserved, should be cleared.
28	S*n*	Status 3–status 0. Reports status of messaging interrupt *n*. Writing a 1 clears this field. 0 Messaging interrupt *n* is not active. 1 Messaging interrupt *n* is active.

9.3.6 Shared Message Signaled Registers

This section contains description the shared message signaled interrupt registers (MSIRs). The shared message signaled interrupt structure allows programs to interrupt each other by simply writing to these shared memory-mapped registers in the PIC. Each of the eight MSIRs can be thought of as collecting interrupts from 32 different memory-mapped writes that can cause interrupts.

9.3.6.1 Shared Message Signaled Interrupt Registers (MSIR0–MSIR7)

The eight MSIRs indicate which of the up to 32 interrupt sources sharing the message register have pending interrupts. These registers are cleared when read. A write to these registers has no effect.

Offset MSIR0: 0x4_1600; MSIR1: 0x4_1610; MSIR2: 0x4_1620; MSIR3: 0x4_1630; MSIR4: 0x4_1640; MSIR5: 0x4_1650; MSIR6: 0x4_1660; MSIR7: 0x4_1670 — Access: Read only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SH31	SH30	SH29	SH28	SH27	SH26	SH25	SH24	SH23	SH22	SH21	SH20	SH19	SH18	SH17	SH16
W																
Reset	All zeros															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SH15	SH14	SH13	SH12	SH11	SH10	SH9	SH8	SH7	SH6	SH5	SH4	SH3	SH2	SH1	SH0
W																
Reset	All zeros (Note: After soft reset, read the MSIRs to ensure that any interrupts previously pending are cleared).															

Figure 9-32. Message Signaled Interrupt Registers (MSIR*n*)

Table 9-34 describes the bits of the MSIRs.

Table 9-34. MSIR*n* Field Descriptions

Bits	Name	Description
n	SH*n*	Message sharer *n* has a pending interrupt.

9.3.6.2 Shared Message Signaled Interrupt Status Register (MSISR)

MSISR, shown in Figure 9-33, contains the status bits for the shared message signaled interrupts. A status bit is set when the corresponding MSIR has an active interrupt. The status bit is 0 if all the corresponding shared interrupt sources are cleared for that MSIR.

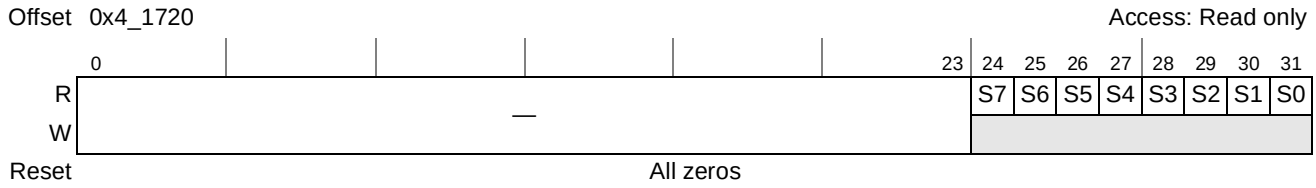


Figure 9-33. Shared Message Signaled Interrupt Status Register (MSISR)

Table 9-35 describes the bits of the MSISR.

Table 9-35. MSISR Field Descriptions

Bits	Name	Description
0–23	—	Reserved, should be cleared.
24–31	S*n*	Status *n*. 0 MSIR*n* is not active. 1 MSIR*n* has an active interrupt.

9.3.6.3 Shared Message Signaled Interrupt Index Register (MSIIR)

MSIIR, shown in Figure 9-33, provides the mechanism for setting an interrupt in the MSIRs. When MSIIR is written, MSIIR[SRS] selects the register in which an interrupt bit is to be set; MSIIR[IBS] selects the shared interrupt field in the selected MSIR register to be set. MSIIR is primarily intended to support PCI Express MSIs.

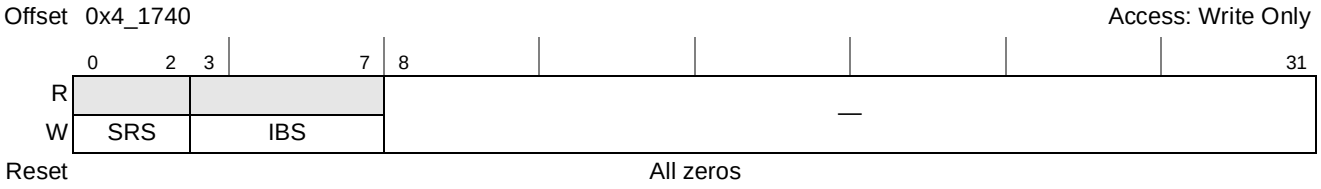


Figure 9-34. Shared Message Signaled Interrupt Index Register (MSIIR)

Table 9-36 describes the bits of the MSIIR.

Table 9-36. MSIIR Field Descriptions

Bits	Name	Description
0–2	SRS	Shared interrupt register select. Selects the MSIR to be written 000 MSIR 0 001 MSIR 1 010 MSIR 2 ... 111 MSIR 7
3–7	IBS	Interrupt bit select—Selects the bit to set in the MSIR 00000 Set field SH0 (bit 31) 00001 Set field SH1 (bit 30) 00010 Set field SH2 (bit 29) ... 11111 Set field SH31 (bit 0)
8–31	—	Reserved, should be cleared.

9.3.6.4 Shared Message Signaled Interrupt Vector/Priority Register (MSIVPRs)

The MSIVPRs, shown in Figure 9-35, have the same fields and format as the GTVPRs.

Offset MSIVPR0: 0x5_1C00; MSIVPR1: 0x5_1C20; MSIVPR2: 0x5_1C40; MSIVPR3: 0x5_1C60; MSIVPR4: 0x5_1C80; MSIVPR5: 0x5_1CA0; MSIVPR6: 0x5_1CC0; MSIVPR7: 0x5_1CE0 Access: Read/Write

	0	1										11	12			15	16															31
R	MSK	A	—										PRIORITY				VECTOR															
W																																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 9-35. Shared Message Signaled Interrupt Vector/Priority Register (MSIVPRs)

Table 9-37 describes the bits of the MSIVPRs.

Table 9-37. MSIVPRn Field Descriptions

Bits	Name	Description
0	MSK	Mask. Mask interrupts from this source. MSK affects only interrupts routed to *int*. 0 An interrupt request is generated if the corresponding IPR bit is set. 1 Further interrupts from this source are disabled.
1	A	Activity. Indicates an interrupt has been requested or is in service. The VECTOR and PRIORITY values should not be changed while this bit is set. Affects only interrupts routed to *int*. 0 No current interrupt activity associated with this source. 1 The interrupt field for this source is set in the IPR or ISR.
2–11	—	Reserved, should be cleared.
12–15	PRIORITY	Priority. Specifies the interrupt priority. The lowest priority is 0 and the highest priority is 15. A priority level of 0 inhibits signalling of this interrupt to the core. Affects only interrupts routed to *int*.
16–31	VECTOR	Vector (Affects only interrupts routed to *int*). Contains the value returned when IACK is read and this interrupt resides in the corresponding interrupt request register (IRR) for that core, as shown in Figure 9-50.

9.3.6.5 Shared Message Signaled Interrupt Destination Registers 0–7 (MSIDR*n*)

The MSIDRs, shown in Figure 9-36, contain the destination fields for shared message signaled interrupts. Only one destination bit may be set; otherwise, behavior is undefined.

Offset MSIDR0: 0x5_1C10; MSIDR1: 0x5_1C30; MSIDR2: 0x5_1C50; MSIDR3: 0x5_1C70; MSIDR4: 0x5_1C90; MSIDR5: 0x5_1CB0; MSIDR6: 0x5_1CD0; MSIDR7: 0x5_1CF0 Access: Read/Write

	0	1	2		29	30	31
R/W	EP	CI0	CI1[1]	—		P1[1]	P0
Reset	0	0	0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0	0	1

[1] Reserved in single-processor implementations.

Figure 9-36. Shared Message Signaled Interrupt Destination Registers (MSIDR*n*)

Table 9-38 describes MSIDR*n* fields.

Table 9-38. MSIDR*n* Field Descriptions

Bits	Name	Description
0	EP	External signal. Allows interrupt to be serviced externally. 0 Interrupt is not routed to $\overline{\text{IRQ\_OUT}}$. 1 Interrupt is routed to $\overline{\text{IRQ\_OUT}}$ for external servicing.
1	CI0	Critical interrupt 0. 0 Processor core 0 does not receive this interrupt. 1 Directs the shared message signaled interrupt to processor core 0 by causing the *cint0* output signal from the PIC to assert. See Section 9.1.3, “Interrupts to the Processor Core.”
2	CI1	Critical interrupt 1. Reserved in single-processor implementations. 0 Processor core 1 does not receive this interrupt. 1 Directs the shared message signaled interrupt to processor core 1 by causing the *cint1* output signal from the PIC to assert. See Section 9.1.3, “Interrupts to the Processor Core.”
3–29	—	Reserved, should be cleared.
30	P1	Processor core 1. Indicates whether processor core 1 receives the interrupt through *int.* Reserved in single-processor implementations. 0 Processor core 1 does not receive this interrupt. 1 Directs the interrupt to processor core 1 through the assertion of *int1*.
31	P0	Processor core 0. Indicates whether processor core 0 receives the interrupt. 0 Processor core 0 does not receive this interrupt. 1 Directs the interrupt to processor core 0 through the assertion of *int0*. The default destination is for processor core 0 to receive this shared message signaled interrupt after the PIC is reset.

9.3.7 Interrupt Source Configuration Registers

The interrupt source configuration registers control the source and destinations of interrupts, specifying parameters such as the interrupting event, signal polarity, and relative priority.

Figure 9-37 shows the destination register differences.

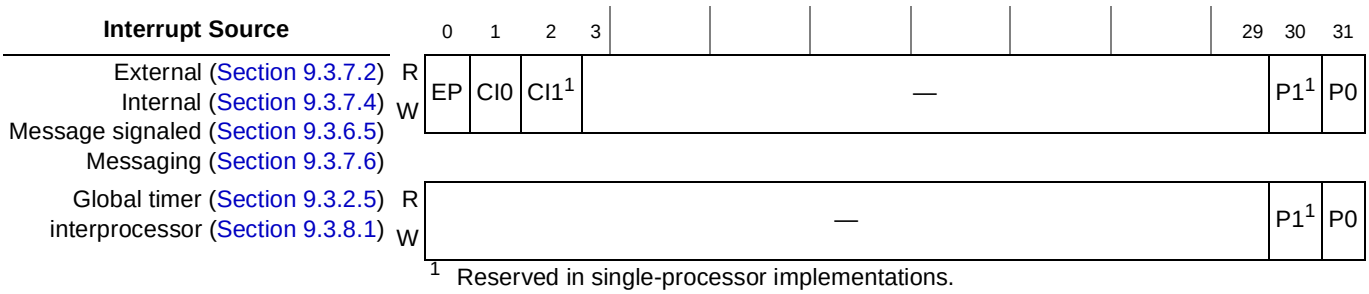


Figure 9-37. Destination Register Summary

Note the following:

- The global timer and interprocessor destination register support only the P0 and P1 options. That is, they cannot be routed to *cint* or to $\overline{\text{IRQ\_OUT}}$.
- Only the global timer and interprocessor interrupts are multicasting, so only these interrupts allow more than one destination bit to be specified.

Figure 9-37 shows the vector/priority register differences.

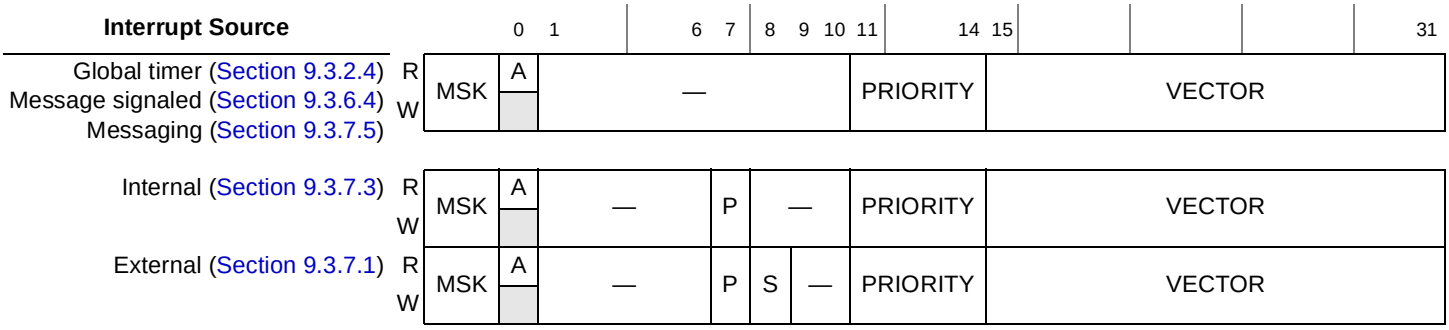


Figure 9-38. Vector/Priority Register Summary

Note the following:

- The MSK, A, PRIORITY, and VECTOR fields are defined by the OpenPIC specification and have meaning only for interrupts routed to the *int* signal.
- The polarity field, P, is provided to indicate whether the signals from the corresponding source are active high or low.
- The sense field, S, is provided to allow external interrupt sources to be configured as level-sensitive so they can be routed to either *cint* or $\overline{\text{IRQ\_OUT}}$.

9.3.7.1 External Interrupt Vector/Priority Registers (EIVPR0–EIVPR11)

The EIVPRs, shown in Figure 9-39, contain polarity and sense fields for the external interrupts, that is, those caused by the assertion of any of IRQ[0:11].

Offset EIVPR0: 0x5_0000; EIVPR1: 0x5_0020; EIVPR2: 0x5_0040; EIVPR3: 0x5_0060; EIVPR4: 0x5_0080; EIVPR5: 0x5_00A0; EIVPR6: 0x5_00C0; EIVPR7: 0x5_00E0; EIVPR8: 0x5_0100; EIVPR9: 0x5_0120; EIVPR10: 0x5_0140; EIVPR11: 0x5_0160 — Access: Read/write

	0	1	2–7	8	9	10–11	12–15	16–31
R	MSK	A	—	P	S	—	PRIORITY	VECTOR
W								
Reset	1	0	0 0 0 0 0 0	0	0	0 0	0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Figure 9-39. External Interrupt Vector/Priority Registers (EIVPR0–EIVPR11)

Table 9-39 describes the EIVPR fields.

Table 9-39. EIVPR*n* Field Descriptions

Bits	Name	Description
0	MSK	Mask. Mask interrupts from this source. MSK affects only interrupts routed to *int*. 0 An interrupt request is generated if the corresponding IPR bit is set. 1 Further interrupts from this source are disabled.
1	A	Activity. Indicates an interrupt has been requested or is in service. The VECTOR and PRIORITY values should not be changed while this bit is set. Affects only interrupts routed to *int*. 0 No current interrupt activity associated with this source. 1 The interrupt field for this source is set in the IPR or ISR.
2–7	—	Reserved, should be cleared.
8	P	Polarity. Specifies the polarity for the external interrupt. 0 Polarity is active-low or negative edge-triggered. 1 Polarity is active-high or positive edge-triggered.
9	S	Sense. Specifies the sense for external interrupts. 0 The external interrupt is edge sensitive. 1 The external interrupt is level sensitive. This setting must be used to direct the interrupt to $\overline{\text{IRQ\_OUT}}$ or *cint*. Note: If an IRQ*n* signal is used to receive INT*x* signals from one of the PCI Express ports as a root complex, S must be set to be level-sensitive.
10–11	—	Reserved, should be cleared.
12–15	PRIORITY	Priority. Specifies the interrupt priority. The lowest priority is 0 and the highest priority is 15. A priority level of 0 inhibits signalling of this interrupt to the core. Affects only interrupts routed to *int*.
16–31	VECTOR	Vector (Affects only interrupts routed to *int*). Contains the value returned when IACK is read and this interrupt resides in the corresponding interrupt request register (IRR) for that core, as shown in Figure 9-51.

9.3.7.2 External Interrupt Destination Registers (EIDR0–EIDR11)

The EIDRs, shown in Figure 9-40, control the destination of external interrupts caused by the assertion of any of IRQ[0:11]. Only one destination bit may be set; otherwise, behavior is undefined.

Offset EIDR0: 0x5_0010; IDR1: 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 — Access: Read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	EP	CI0	CI1[1]	—																											P1[1]	P0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

[1] Reserved in single-processor implementations.

Figure 9-40. External Interrupt Destination Registers (EIDRs)

Table 9-40 describes the EIDR fields.

Table 9-40. EIDR*n* Field Descriptions

Bits	Name	Description
0	EP	External signal. Allows interrupt to be serviced externally. EP should be set only for level-sensitive external interrupts (EIVPR*n*[S]= 1). Setting for edge-sensitive does not provide reliable interrupt response. 0 Interrupt is not routed to $\overline{\text{IRQ\_OUT}}$. 1 Interrupt is routed to $\overline{\text{IRQ\_OUT}}$ for external service.
1	CI0	Critical interrupt 0. Ci*n* fields should be set only for level-sensitive external interrupts (EIVPR*n*[S]= 1). Setting them for edge-sensitive does not provide reliable interrupt response. 0 Processor core 0 does not receive this interrupt. 1 Directs the external interrupt to processor core 0 by causing the *cint0* output signal from the PIC to assert. See Section 9.1.3, “Interrupts to the Processor Core.”
2	CI1	Critical interrupt 1. Ci*n* fields should be set only for level-sensitive external interrupts (EIVPR*n*[S]= 1). Setting them for edge-sensitive does not provide reliable interrupt response. Reserved in single-processor implementations. 0 Processor core 1 does not receive this interrupt. 1 Directs the external interrupt to processor core 1 by causing the *cint1* output signal from the PIC to assert. See Section 9.1.3, “Interrupts to the Processor Core.”
3–29	—	Reserved, should be cleared.
30	P1	Processor core 1. Indicates whether processor core 1 receives the interrupt through *int.* Reserved in single-processor implementations. 0 Processor core 1 does not receive this interrupt. 1 Directs the interrupt to processor core 1 through the assertion of *int1*.
31	P0	Processor core 0. Indicates whether processor core 0 receives the interrupt. 0 Processor core 0 does not receive this interrupt. 1 Directs the interrupt to processor core 0 through the assertion of *int0*. The default destination is for processor core 0 to receive this external interrupt after the PIC is reset.

9.3.7.3 Internal Interrupt Vector/Priority Registers (IIVPR*n*)

The IIVPRs, shown in Figure 9-41, have the same fields and format as the GTVPRs, except that they apply to the internal interrupt sources listed in Table 9-3. These interrupts are all level-sensitive.

NOTE

Because all internal interrupts are active-high, clearing the polarity field, IIVPR*n*[P], disables that interrupt. Care should be taken to ensure this field is set during initialization and that it is not inadvertently corrupted when loading or reloading IIVPRs with priority, mask, or vector data.

Offset IIVPR0–7 0x5_0200, 0x5_0220, 0x5_0240, 0x5_0260, 0x5_0280, 0x5_02A0, 0x5_02C0, 0x5_02E0 Access:
IIVPR8–15 0x5_0300, 0x5_0320, 0x5_0340, 0x5_0360, 0x5_0380, 0x5_03A0, 0x5_03C0, 0x5_03E0 Read/Write
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
IIVPR32–39 0x5_0600, 0x5_0620, 0x5_0640, 0x5_0660, 0x5_0680, 0x5_06A0, 0x5_06C0, 0x5_06E0
IIVPR40–47 0x5_0700, 0x5_0720, 0x5_0740, 0x5_0760, 0x5_0780, 0x5_07A0, 0x5_07C0, 0x5_07E0

	0	1	2					7	8	9		11	12			15	16															31
R	MSK	A	—						P	—			PRIORITY				VECTOR															
W																																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 9-41. Internal Interrupt Vector/Priority Registers (IIVPRs)

Table 9-41 describes the IIVPR fields.

Table 9-41. IIVPR*n* Field Descriptions

Bits	Name	Description
0	MSK	Mask. Mask interrupts from this source. MSK affects only interrupts routed to *int*. 0 An interrupt request is generated if the corresponding IPR bit is set. 1 Further interrupts from this source are disabled.
1	A	Activity. Indicates an interrupt has been requested or is in service. The VECTOR and PRIORITY values should not be changed while this bit is set. Affects only interrupts routed to *int*. 0 No current interrupt activity associated with this source. 1 The interrupt field for this source is set in the IPR or ISR.
2–7	—	Reserved, should be cleared.
8	P	Polarity. Specifies the polarity for the internal interrupt. Note: Because all internal interrupts are active-high, clearing this bit disables the interrupt. 0 Interrupt polarity is active-low. This value disables the interrupt. 1 Interrupt polarity is active-high.
9–11	—	Reserved, should be cleared.
12–15	PRIORITY	Priority. Specifies the interrupt priority. The lowest priority is 0 and the highest priority is 15. A priority level of 0 inhibits signalling of this interrupt to the core. Affects only interrupts routed to *int*.
16–31	VECTOR	Vector (Affects only interrupts routed to *int*). Contains the value returned when IACK is read and this interrupt resides in the corresponding interrupt request register (IRR) for that core, as shown in Figure 9-51.

9.3.7.4 Internal Interrupt Destination Registers (IIDR*n*)

The IIDRs, shown in Figure 9-42, have the same fields and format as EIVDRs, except that they apply to the internal interrupt sources listed in Table 9-3. Only one destination bit may be set; otherwise, behavior is undefined.

Offset IIDR0–70x5_0210, 0x5_0230, 0x5_0250, 0x5_0270, 0x5_0290, 0x5_02B0, 0x5_02D0, 0x5_02F0 Access:
IIDR8–15 0x5_0310, 0x5_0330, 0x5_0350, 0x5_0370, 0x5_0390, 0x5_03B0, 0x5_03D0, 0x5_03F0 Read/Write
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
IIDR32–39 0x5_0610, 0x5_0630, 0x5_0650, 0x5_0670, 0x5_0690, 0x5_06B0, 0x5_06D0, 0x5_06F0
IIDR40–47 0x5_0710, 0x5_0730, 0x5_0750, 0x5_0770, 0x5_0590, 0x5_07B0, 0x5_07D0, 0x5_07F0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	EP	CI0	CI1[1]	—																											P1[1]	P0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

[1] Reserved in single-processor implementations.

Figure 9-42. Internal Interrupt Destination Registers (IIDRs)

Table 9-42 describes the IIDR fields.

Table 9-42. IIDR*n* Field Descriptions

Bits	Name	Description
0	EP	External signal. Allows internal interrupt to be serviced externally. 0 Interrupt is not routed to $\overline{\text{IRQ\_OUT}}$. 1 Interrupt is routed to $\overline{\text{IRQ\_OUT}}$ for external service.
1	CI0	Critical interrupt 0. See Section 9.1.3, “Interrupts to the Processor Core,” for more information. 0 Processor core 0 does not receive this interrupt. 1 Directs the internal interrupt to processor core 0 by causing the *cint0* output signal from the PIC to assert.
2	CI1	Critical interrupt 1. See Section 9.1.3, “Interrupts to the Processor Core,” for more information. Reserved in single-processor implementations. 0 Processor core 1 does not receive this interrupt. 1 Directs the internal interrupt to processor core 1 by causing the *cint1* output signal from the PIC to assert.
3–29	—	Reserved, should be cleared.
30	P1	Processor core 1. Indicates whether processor core 1 receives the interrupt through *int.* Reserved in single-processor implementations. 0 Processor core 1 does not receive this interrupt. 1 Directs the interrupt to processor core 1 through the assertion of *int1*.
31	P0	Processor core 0. Indicates whether processor core 0 receives the interrupt. 0 Processor core 0 does not receive this interrupt. 1 Directs the interrupt to processor core 0 through the assertion of *int0*. The default destination is for processor core 0 to receive this external interrupt after the PIC is reset.

9.3.7.5 Messaging Interrupt Vector/Priority Registers (MIVPR*n*)

The MIVPRs have the same fields and format as the GTVPRs, except they apply to messaging interrupts.

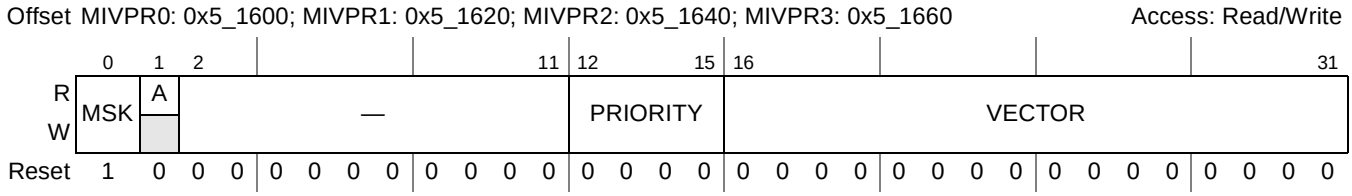


Figure 9-43. Messaging Interrupt Vector/Priority Registers (MIVPR*n*)

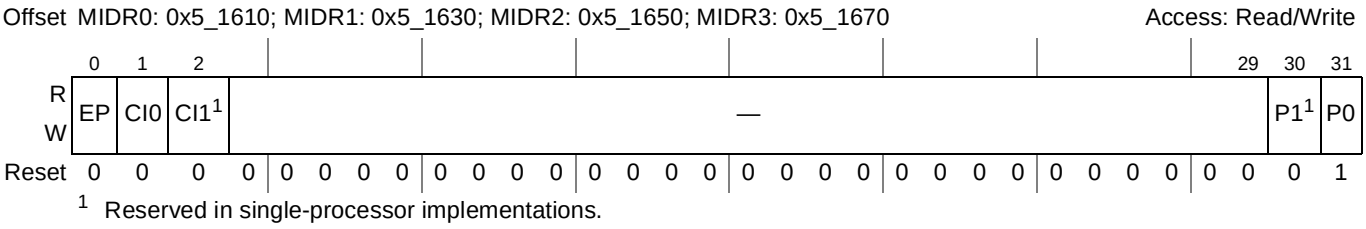
Table 9-43 describes the MIVPR*n* fields.

Table 9-43. MIVPR*n* Field Descriptions

Bits	Name	Description
0	MSK	Mask. Mask interrupts from this source. MSK affects only interrupts routed to *int*. 0 An interrupt request is generated if the corresponding IPR bit is set. 1 Further interrupts from this source are disabled.
1	A	Activity. Indicates an interrupt has been requested or is in service. The VECTOR and PRIORITY values should not be changed while this bit is set. Affects only interrupts routed to *int*. 0 No current interrupt activity associated with this source. 1 The interrupt field for this source is set in the IPR or ISR.
2–11	—	Reserved, should be cleared.
12–15	PRIORITY	Priority. Specifies the interrupt priority. The lowest priority is 0 and the highest priority is 15. A priority level of 0 inhibits signalling of this interrupt to the core. Affects only interrupts routed to *int*.
16–31	VECTOR	Vector (Affects only interrupts routed to *int*). Contains value returned when IACK is read and this interrupt resides in the corresponding interrupt request register (IRR) for that core, as shown in Figure 9-51.

9.3.7.6 Messaging Interrupt Destination Registers (MIDR0–MIDR3)

The messaging interrupt destination registers (MIDRs), shown in Figure 9-44, control the destination for the messaging interrupts. Only one destination bit may be set; otherwise, behavior is undefined.



[1] Reserved in single-processor implementations.

Figure 9-44. Messaging Interrupt Destination Registers (MIDR*n*)

Table 9-44 describes the MIDR*n* fields.

Table 9-44. MIDR*n* Field Descriptions

Bits	Name	Description
0	EP	External signal. Allows messaging interrupt to be serviced externally. 0 Interrupt is not routed to $\overline{\text{IRQ\_OUT}}$. 1 Message interrupt is routed to $\overline{\text{IRQ\_OUT}}$ for external service.
1	CI0	Critical interrupt 0. 0 Processor core 0 does not receive this interrupt. 1 Directs the interrupt to processor core 0 by causing the *cint0* output signal from the PIC to assert. See Section 9.1.3, "Interrupts to the Processor Core," for more information.
2	CI1	Critical interrupt 1. Reserved in single-processor implementations. 0 Processor core 1 does not receive this interrupt. 1 Directs the interrupt to processor core 1 by causing the *cint1* output signal from the PIC to assert. See Section 9.1.3, "Interrupts to the Processor Core," for more information.
3–29	—	Reserved, should be cleared.
30	P1	Processor core 1. Indicates whether processor core 1 receives the interrupt through *int*. Reserved in single-processor implementations. 0 Processor core 1 does not receive this interrupt. 1 Directs the interrupt to processor core 1 through the assertion of *int1*.
31	P0	Processor core 0. Indicates whether processor core 0 receives the interrupt. 0 Processor core 0 does not receive this interrupt. 1 Directs the interrupt to processor core 0 through the assertion of *int0*. The default destination is for processor core 0 to receive this external interrupt after the PIC is reset.

9.3.8 Per-CPU (Private Access) Registers

The OpenPIC programming model supports multiprocessor systems of up to 32 separate processors. As such, the OpenPIC interface specification provides for coordinating both the requesting and servicing of interrupts among several processor cores within a single integrated device. To comply with the OpenPIC specification, the PIC incorporates several of these multiprocessor capabilities.

NOTE

Note that these registers are meaningful only for interrupts routed to *int*.

The registers in Table 9-45 are called per-CPU registers because they are duplicated for each core in a multi-core device. The OpenPIC interface specifies that a copy of these registers be available to each core at the same physical address by using the ID of the processor core that initiates the transaction to determine the set of per-CPU registers to access.

Table 9-45. Per-CPU Registers—Private Access Address Offsets

Register Name	Offset
Interprocessor 0 dispatch register (IPIDR0)	0x4_0040
Interprocessor 1 dispatch register (IPIDR1)	0x4_0050
Interprocessor 2 dispatch register (IPIDR2)	0x4_0060
Interprocessor 3 dispatch register (IPIDR3)	0x4_0070

Table 9-45. Per-CPU Registers—Private Access Address Offsets

Register Name	Offset
Current task priority register (CTPR)	0x4_0080
Who am I register (WHOAMI0)	0x4_0090
Interrupt acknowledge register (IACK)	0x4_00A0
End of interrupt register (EOI)	0x4_00B0

These addresses, shown in Table 9-45, appear in the memory map at the same offset for every processor in what is called the private access space. This duplication allows user code to execute correctly in an multiprocessor environment without needing to know which core it is running on. On a single-core device, each register has two addresses, one in the normal address space and one in the private access space. It is included on even single-core devices to simplify the porting of such code.

Figure 9-45 shows how the duplicated registers are addressed in a four-core device Note that when accessing a register normally, each core sources a different address. However, when accessing the same register using the per-CPU address space, each core sources the same address.

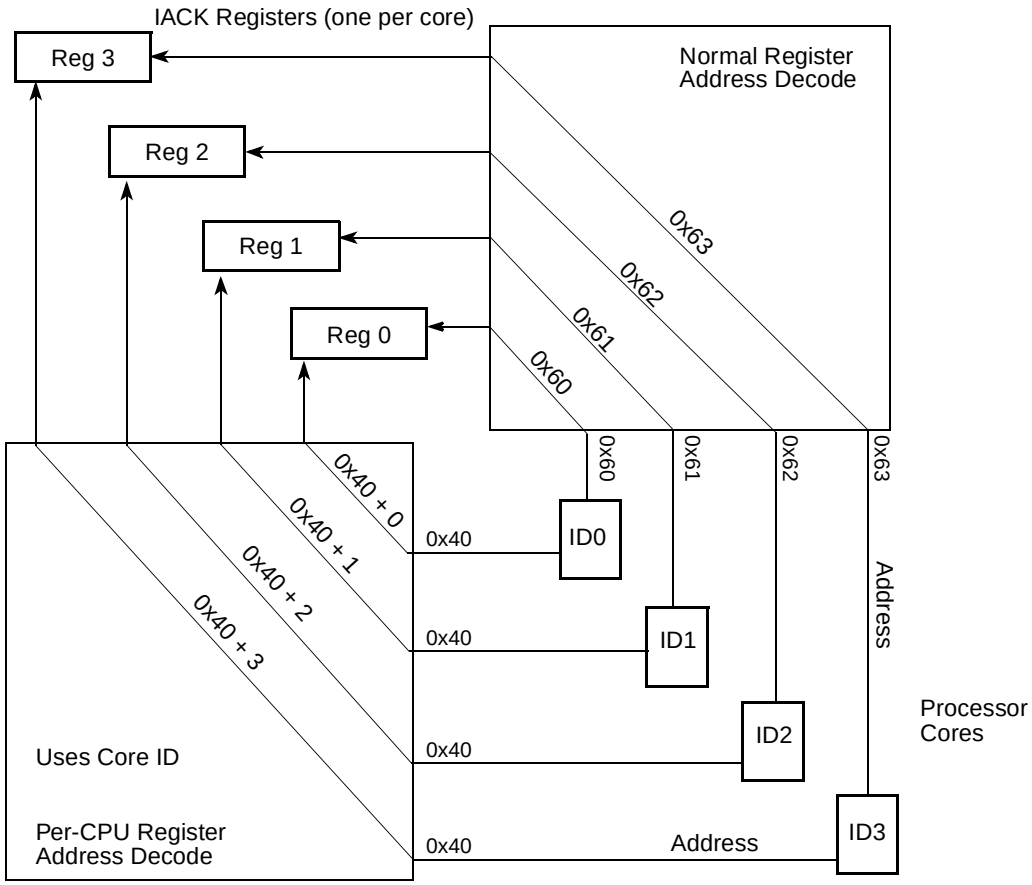


Figure 9-45. Per-CPU Register Address Decoding in a Four-Core Device

9.3.8.1 Interprocessor Interrupt Dispatch Register (IPIDR0–IPIDR3)

Figure 9-46 shows the four IPIDRs, one for each interprocessor interrupt channel. Writing to an IPIDR with a bit set causes a self interrupt for a single-core device. Because external bus masters can write to these registers, this feature can serve as a doorbell type interrupt.

Offset Processor core 0: IPIDR0: 0x6_0040; IPIDR1: 0x6_0050; IPIDR2: 0x6_0060; IPIDR3: 0x6_0070 Access: Write Only
Processor core 1[1]:IPIDR0: 0x6_1040; IPIDR1: 0x6_1050; IPIDR2: 0x6_1060; IPIDR3: 0x6_1070
Pre-CPU offsets: IPIDR0: 0x4_0040; IPIDR1: 0x4_0050; IPIDR2: 0x4_0060; IPIDR3: 0x4_0070

1 Reserved in single-processor implementations.

Figure 9-46. Interprocessor Interrupt Dispatch Registers (IPIDR0–IPIDR3)

Table 9-41 describes the IPIDR*n* fields.

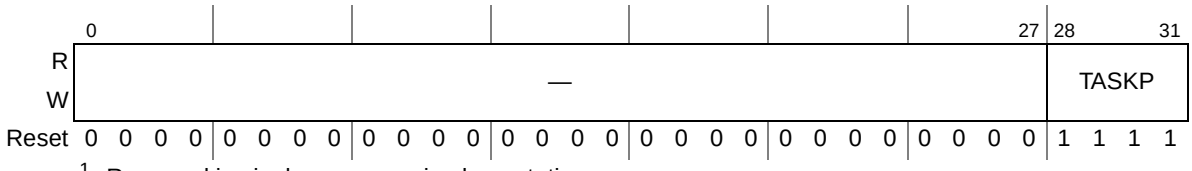
Table 9-46. IPIDR*n* Field Descriptions

Bits	Name	Description
0–29	—	Reserved, should be cleared.
30	P1	Processor core 1. Specifies if processor core 1 receives the interrupt. Reserved in single-processor implementations. This interrupt is multicasting, so both P0 and P1 can be set. 0 Processor core 1 does not receive the interrupt 1 Directs the interrupt to processor core 1
31	P0	Processor core 0. Determines if processor core 0 receives the interrupt. 0 Processor core 0 does not receive the interrupt. 1 Directs the interrupt to processor core 0.

9.3.8.2 Processor Core Current Task Priority Registers 0–1 (CTPR0–CTPR1)

There is one CTPR per processor core on this device as shown in Figure 9-47.

Offset CTPR0: 0x6_0080; CTPR1[1] 0x6_1080; Pre-CPU offset: 0x4_0080 Access: Read/Write



1 Reserved in single-processor implementations.

Figure 9-47. Processor Core Current Task Priority Registers (CTPR*n*)

NOTE

CTPR has meaning only for interrupts routed to *int*.

Software must write the priority of the current processor core task in the CTPR for each core. The PIC uses this value for comparison with the priority of incoming interrupts. Given several concurrent incoming interrupts, the highest priority interrupt is asserted to that core if the following apply:

- The interrupt is not masked.
- The priority of the interrupt is higher than the values in the corresponding CTPR[TASKP] and ISR.

Table 9-47 describes the CTPR task priority field.

Table 9-47. CTPR*n* Field Descriptions

Bits	Name	Description
0–27	—	Reserved, should be cleared.
28–31	TASKP	Task priority. Indicates the threshold that individual interrupt priorities must exceed for the interrupt request to be serviced. 0000–1111 *x*VPR*n*[PRIORITY] must exceed this value for the interrupt request to be serviced. Note the following special cases: 0000 Lowest priority. All interrupts except those whose priority are 0 can be serviced. 1111 Highest priority. No interrupts are signaled to that processor core. Hardware selects this value on a device hard reset or when the corresponding PIR[P*n*] is set.

9.3.8.3 Who Am I Registers 0–1 (WHOAMI0–WHOAMI1)

The processor core WHOAMI*n* register, shown in Figure 9-48, can be read by a processor core to determine its physical connection to the PIC. The value returned when reading this register may be used to determine the value for the destination masks used for dispatching interrupts.

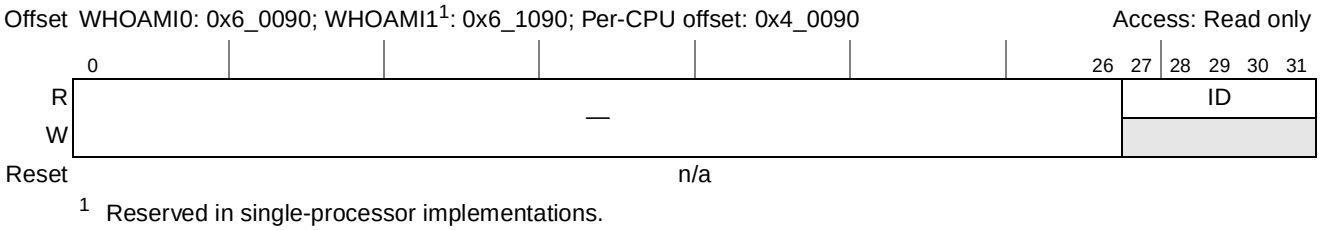


Figure 9-48. Processor Core Who Am I Registers (WHOAMI*n*)

Table 9-48 describes the WHOAMI*n* fields.

Table 9-48. WHOAMI*n* Field Descriptions

Bits	Name	Description
0–26	—	Reserved, should be cleared.
27–31	ID	Returns the ID of the processor core reading this register. 0_0000 Processor core 0 0_0001 Processor core 1. (Value not supported in single-processor implementations.) 1_1111 Other devices

9.3.8.4 Processor Core Interrupt Acknowledge Registers 0–1 (IACK0–IACK1)

NOTE

IACK has meaning only for interrupts routed to *int* and should not be accessed for interrupts routed to *cint* or $\overline{\text{IRQ\_OUT}}$.

In systems based on processors built on Power Architecture™ technology, the interrupt acknowledge function occurs as an explicit read operation to a memory-mapped interrupt acknowledge register (IACK), shown in Figure 9-49. Each processor core has an IACK register assigned to it. Reading IACK returns the interrupt vector corresponding to the highest priority pending interrupt. Reading IACK also has the following side effects:

- The associated field in the corresponding interrupt pending register (IPR) is cleared for edge-sensitive interrupts.
- The corresponding in-service register (ISR) is updated.
- The corresponding *int* output signal from the PIC is negated.

Reading IACK when no interrupt is pending returns the spurious vector value, as described in Section 9.3.1.9, "Spurious Vector Register (SVR)."

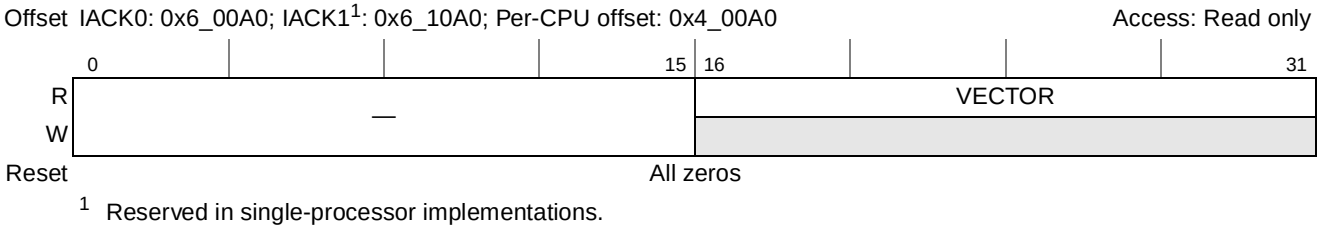


Figure 9-49. Processor Core Interrupt Acknowledge Registers (IACK*n*)

Table 9-49 describes the IACK*n* fields.

Table 9-49. IACK*n* Field Descriptions

Bits	Name	Description
0–15	—	Reserved, should be cleared.
16–31	VECTOR	Interrupt vector. Vector of the highest pending interrupt (read only)

9.3.8.5 Processor Core End of Interrupt Registers (EOI0–EOI1)

NOTE

EOI has meaning only for interrupts routed to *int* and should not be accessed for interrupts routed to *cint* or $\overline{\text{IRQ\_OUT}}$.

Each core is assigned an EOI register, shown in Figure 9-50. Writing to EOI signals the end of processing for the highest-priority interrupt (routed to *int*) currently in service. It also updates the corresponding ISR*n* by retiring the highest priority interrupt. Data values written to EOI are ignored, and zero is assumed.

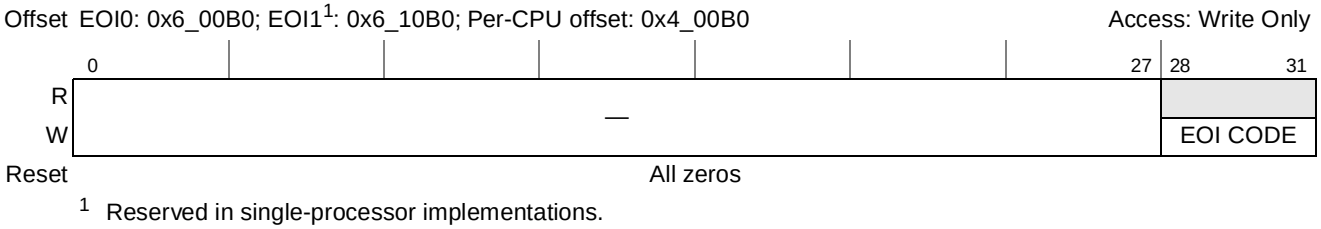


Figure 9-50. End of Interrupt Registers (EOI*n*)

Table 9-50 describes the EOI*n* fields.

Table 9-50. EOI*n* Field Descriptions

Bits	Name	Description
0–27	—	Reserved, should be cleared.
28–31	EOI CODE	0000 (write only)

9.4 Functional Description

This section is a functional description of the PIC.

9.4.1 Flow of Interrupt Control

Figure 9-51 shows the flow of interrupts directed by the PIC to the *int*, *cint*, and $\overline{\text{IRQ\_OUT}}$ outputs. Note that this diagram describes a conceptual model of an PIC on a single processor. This logic is replicated for each implemented processor. This conceptual diagram does not fully represent all internal circuitry of the implementation

This figure focusses especially on the OpenPIC-defined logic and shows how the PIC controls interrupt requests that target the *int* signal. The flow in Figure 9-51 is from the bottom to the top, and shows at the bottom how the destination register associated with each source determines the path.

9.4.1.1 Interrupts Routed to *cint* or $\overline{\text{IRQ\_OUT}}$

Interrupt requests routed to *cint* or $\overline{\text{IRQ\_OUT}}$ bypass the logic that is dedicated to interrupt sources that compete for *int*. That is, if *x*IDR*n*[CI*n*] or *x*IDR*n*[EP] = 1, corresponding *x*IVPR field settings have no hardware effects; however, an interrupt handler may be able to make use of some of those fields.

cint signals are connected to the respective core's machine check input.

NOTES

Because interrupt sources routed to *cint* or $\overline{\text{IRQ\_OUT}}$ must be level sensitive, EIVPR[S] should be set. See Section 9.3.7.1, "External Interrupt Vector/Priority Registers (EIVPR0–EIVPR11)."

Because these interrupts bypass the OpenPIC logic, it is especially important that handlers do not read IACK. Doing so causes a spurious interrupt. Likewise, they should not write EOI.

9.4.1.2 Interrupts Routed to *int*

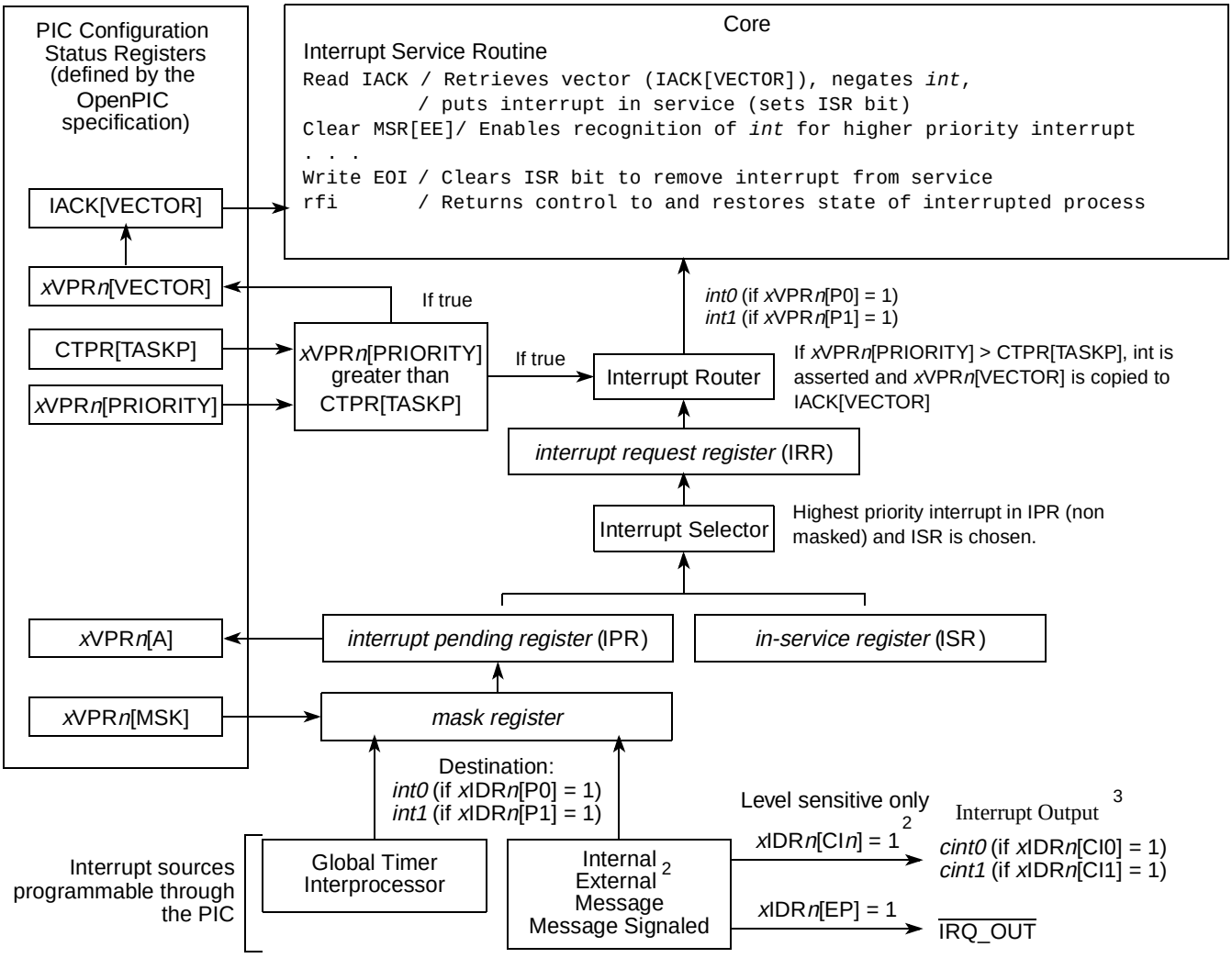
As shown in Figure 9-51, the PIC receives interrupt requests from external and internal sources and from within the PIC itself. As Figure 9-51 shows, all of these interrupt sources can be routed to *int*; the global timer and timer processor interrupts can be directed only to *int*.

The sources' mask bits (*x*VPR*n*[MSK]) are tracked in the internal mask register. If a source's MSK bit is set, the mask register prevents the PIC from asserting *int* on its behalf.

Unmasked interrupt requests are qualified and latched in the interrupt pending register (IPR), an internal interrupt summary register with a bit for each source. If an interrupt request is multi-cast, a bit is set in the IPR for each targeted processor. Although the IPR cannot be read by software, when an IPR bit is set, the corresponding source's activity bit (*x*VPR*n*[A]) is automatically set.

The interrupt selector monitors the IPR and the in-service register (ISR), which tracks previously taken interrupts that were superseded by a higher-priority interrupt before the interrupt handler finished. The interrupt selector recognizes the highest priority unmasked interrupt request and latches it into the interrupt request register (IRR). The source's vector (*x*VPR*n*[VECTOR]) is copied to IACK[VECTOR], which the interrupt handler retrieves by reading IACK.

If the priority (*x*VPR*n*[PRIORITY]) of an interrupt latched in the IRR is higher than the value in the target processor's CTPR[TASKP], the interrupt router asserts the external interrupt signal (*int*), causing that processor core to vector to its external interrupt handler.



1 If *cint* or $\overline{\text{IRQ\_OUT}}$ is the destination, EIVPR*n*[S] must be set to configure the source as level sensitive.

2 If multiple destination register bits are set, PIC behavior is undefined.

3 Although setting CI*n* directs the interrupt request to the critical interrupt output (*cint0*/*cint1*), integrated logic may connect this signal to a different interrupt input to the core.

Figure 9-51. PIC Interrupt Processing Flow Diagram for Each Core (*n*)

The interrupt handler must acknowledge the interrupt by explicitly reading the corresponding IACK register, described in Section 9.3.8.4, “Processor Core Interrupt Acknowledge Registers 0–1 (IACK0–IACK1).” The PIC interprets this read as an interrupt acknowledge (IACK) cycle. The IACK cycle not only returns the source’s vector, it also negates the *int* signal to the processor (making it possible for a higher priority interrupt to assert *int*) and sets the source’s bit in the ISR, indicating that this interrupt has been put in service. An interrupt remains in service from the time until the corresponding end-of-interrupt register (EOI) is written, generating what the PIC considers an EOI signal.

Figure 9-51 shows required elements in the interrupt handler

9.4.1.2.1 Nesting of Interrupts

While an interrupt is being handled, if an interrupt request arrives with a higher *x*VPR*n*[PRIORITY] value, the interrupt being serviced is superseded. As described in Section 9.4.1.2, “Interrupts Routed to *int*,” the PIC asserts *int*, and the newer, higher priority interrupt is handled. This happens even if software, as part of its interrupt service routine, updates the corresponding CTPR with a lower value.

Thus, although several interrupts can be in service simultaneously (and tracked by the ISR), the highest priority interrupt by that processor is always the one actively handled. When the interrupt routine completes, it performs a write EOI cycle, a side effect of which is to take the current highest priority interrupt out of service (removes it from the ISR). At this point, the interrupt selector chooses the new highest priority interrupt request, and, assuming CTPR[TASKP] has not been updated to a value higher than the new interrupt, the PIC asserts *int* on its behalf.

The next write EOI cycle takes the current highest priority interrupt out of service. An interrupt with lower priority than those in service is not started until all higher priority interrupts complete even if its priority is greater than the CTPR value.

9.4.1.2.2 Interrupt Source Priority

Each interrupt source routed to *int* is assigned a value through its *x*VPR*n*[PRIORITY] field. Priority values range from 0 to 15, where 15 is the highest. Interrupts are delivered only when the priority of the source is greater than the destination processor’s CTPR[TASKP]. Therefore, setting *x*VPR*n*[PRIORITY] to zero inhibits that interrupt. Likewise, setting TASKP to 15 prevents the PIC from delivering interrupts to that core through the *int* signal. Note that this is the reset value, preventing the PIC from asserting *int* before the PIC is configured.

The PIC services simultaneous interrupts occurring with the same priority according to the following order:

1. MSG0–MSG3
2. MSI0–MSI7
3. IPI0–IPI3
4. Group A timer0–timer3
5. Group B timer0–timer3
6. IRQ[0:11]/PCI INT*x*
7. Internal0–internal47

For example, if MSG0, MSG2, and IPI0 are all assigned the same priority and receive simultaneous interrupts, they are serviced in the following order:

1. MSG0
2. MSG2
3. IPI0

9.4.1.2.3 Interrupt Acknowledge

When the PIC causes *int* to be asserted, the external interrupt service routine acknowledges the request by reading that core's IACK register, which at this point holds the 16-bit vector value for the interrupt source that generated the request. This is the value programmed in that source's *x*VPR*n*[VECTOR]. Reading IACK has the following effects:

- The *int* signal for that core is negated, making it possible for another interrupt source to signal an external interrupt to the core, and more particularly, allowing the PIC to signal a higher-priority interrupt, as described in Section 9.4.1.2.1, "Nesting of Interrupts."
- The source that caused that resource is represented in the internal in-service register (ISR).

The interrupt is then considered to be in service. It remains so until the processor core performs a write to the corresponding EOI. Writing to EOI is referred to as an EOI cycle.

9.4.1.2.4 Spurious Vector Generation

Under certain circumstances, the PIC has no valid vector to return to a processor core during an interrupt acknowledge cycle. In these cases, the spurious vector from the spurious vector register is returned. The following cases cause a spurious vector fetch:

- *int* is asserted in response to an externally or internally-sourced interrupt which is activated with level-sensitive logic, and the asserted level is negated before the interrupt is acknowledged.
- *int* is asserted for an interrupt source that is later masked (using the mask bit in the vector/priority register corresponding to that source) before the interrupt is acknowledged.
- *int* is asserted for an interrupt source that is later masked by an increase in the task priority level before the interrupt is acknowledged.
- An interrupt acknowledge cycle is performed by the processor core in spite of the fact that the int signal has not been asserted by the PIC.

In all cases, a spurious vector is returned only if no pending interrupt has sufficient priority to signal an interrupt, otherwise, the vector for that interrupt source is returned.

NOTE

EOI should not be written in response to a spurious vector. Otherwise, a previously accepted interrupt might be cleared unintentionally.

9.4.2 Interprocessor Interrupts

Processors 0 and 1 can generate interprocessor interrupts that target either or both processors. A self interrupt occurs when a core dispatches an interprocessor interrupt event to itself. Interrupts are initiated

by writing either or both of the PO*n* bits in an interprocessor interrupt dispatch register (IPIDR0–IPIDR3) of one of the four IPI channels. If subsequent interprocessor interrupts from a given channel to a given target processor are initiated before the first is acknowledged, only one interrupt is generated.

9.4.3 Message Interrupts

The four MSGRs, described in Section 9.3.5.1, "Message Registers (MSGR0–MSGR3)," can be used to send 32-bit messages to one or more processors. A messaging interrupt is generated by writing an MSGR if the corresponding MER bit is set and the interrupt is not masked. Reading a MSGR or writing a 1 to the status bit clears the interrupt.

9.4.4 Shared Message Signaled Interrupts

There are eight shared MSIRs, described in Section 9.3.6.1, "Shared Message Signaled Interrupt Registers (MSIR0–MSIR7)," that indicate which of the interrupt sources sharing the MSI register have pending interrupts. Up to 32 sources can share any individual MSI register. A shared message signaled interrupt is generated by writing to Shared Message Signaled Interrupt Index Register (MSIIR) fields SRS and IBS. This register is primarily intended to support inbound PCI Express message signaled interrupts (MSIs) when the PCI Express controller is configured as a root complex (RC).

MSIIR[SRS] selects the associated MSIR and MSIIR[IBS] selects the interrupt flag/bit in that register that is to be set. The corresponding interrupt needs to be unmasked for the interrupt to occur. A read to an MSIR clears the all of its flags.

9.4.5 PCI Express INTx

Whenever the PCI Express controller is in root complex mode and it receives an inbound INTx asserted or negated message transaction, it asserts or negates an equivalent internal INTx signal to the PIC. This INTx virtual-wire interrupt signaling mechanism replaces the PCI standard sideband interrupts (INTA, INTB, INTC, and INTD) that historically were connected to the IRQ*n* external interrupt inputs. The internal INTx signals from the PCI Express controller are logically combined with the interrupt request (IRQ*n*) signals so that they share the same OpenPIC external interrupt controlled by the associated EIVPR*n* and EIDR*n* registers.

Table 9-51details the association of INTx signals to IRQ*n* signals.

Table 9-51. PCI Express INTx/IRQ*n* Sharing

PCI Express Number	INTx	IRQ*n*
PCI Express 1	INTA	IRQ0
	INTB	IRQ1
	INTC	IRQ2
	INTD	IRQ3

Table 9-51. PCI Express INTx/IRQ*n* Sharing (continued)

PCI Express Number	INTx	IRQ*n*
PCI Express 2	INTA	IRQ4
	INTB	IRQ5
	INTC	IRQ6
	INTD	IRQ7

In general, these signals should be considered mutually exclusive. If a PCI Express INTx signal is being used, the PIC must be configured so that external interrupts are level sensitive (EIVPRn[S] = 1). If an IRQ*n* signal is being used as edge-triggered (EIVPRn[S] = 0), the system must not allow inbound PCI Express INTx transactions.

Note that it is possible to share IRQ*n* and INTx if the external interrupt is level sensitive; however, if an interrupt occurs, the interrupt service routine must poll both the external sources connected to the IRQ*n* input and the PCI Express INTx sources to determine from which path the external interrupt came. In any case, IRQ*n* should be pulled to the negated state as determined by the associated polarity setting in EIVPR*n*[P].

9.4.6 Global Timers

There are appropriate clock prescalers and synchronizers to provide a time base for the internal PIC timers. These 8 timers are organized as 2 groups of 4 timers each. The timers can be individually programmed to generate a processor core interrupt when they count down to zero and can be used to generate regular periodic interrupts. Each timer has the following four configuration and control registers:

- Global timer current count register (GTCCR*xn*)
- Global timer base count register (GTBCR*xn*)
- Global timer vector-priority register (GTVPR*xn*)
- Global timer destination register (GTDR*xn*)

The timer frequency should be written to the TFRR*xn*, described in Section 9.3.2.1, “Timer Frequency Reporting Register (TFRRA–TFRRB).”

Timer interrupts are all edge-triggered interrupts. If a timer period expires while a previous interrupt from the same source is pending or in service, the subsequent interrupt is lost.

The timer control register (TCR) provides users with the ability to create timers larger than the 31-bit global timers. The timer frequency can also be changed by setting the appropriate TCR fields, as described in Section 9.3.2.6, “Timer Control Registers (TCRA–TCRB).”

9.4.7 Resets

This section describes the behavior of the PIC at reset and the PIC’s ability to initiate processor resets.

9.4.8 Resetting the PIC

The PIC is reset by a device power-on reset (POR) or by software that sets GCR[RST], either of which causes the following:

- All pending and in-service interrupts are cleared.
- All interrupt mask bits are set.
- Polarity, sense, external signal, critical interrupt, and activity fields are reset to default values.
- PIR, TFRR, TCR, MER, MSR, and MSGR0–MSGR3 are cleared.
- MSG and timer destination fields are set.
- The interprocessor dispatch registers are cleared.
- All timer base count values are reset to zero with count inhibited.
- CTPR[TASKP] is reset to 0x000F, disabling delivery of interrupts that target *int*.
- The spurious interrupt vector resets to 0xFFFF.
- The PMMRs are reset to 0xFFFF.
- The PIC defaults to the pass-through mode (GCR[M] = 0).
- All other registers remain at their pre-reset programmed values.

GCR[RST] is automatically cleared when the reset sequence is complete.

9.4.8.1 Processor Core Resetting

A processor core can be reset by writing to the processor core reset register (PRR). This causes the assertion of the *core0_hreset* and/or *core1_hreset* output signals from the PIC. When this occurs, the corresponding CTPR also gets written to 0x000F to disable the delivery of any interrupts to that core.

9.4.8.2 Processor Core Initialization

A software reset can be routed to either of the cores by writing to the processor core initialization register (PIR). This causes the assertion of the corresponding *sreset* output signal from the PIC. When this occurs, the corresponding CTPR also gets written to 0x000F to prevent delivery of any interrupts to *int*.

9.5 Initialization/Application Information

This section contains initialization and application information for the PIC.

9.5.1 Programming Guidelines

The following subsections contain information about programming PIC registers.

9.5.1.1 PIC Registers

Most PIC control and status registers are readable and return the last value written. The exceptions to this rule are as follows:

- Interprocessor dispatch and EOI registers, which return zeros on reads.

- Activity bits (A) of the vector/priority registers reflect the status of the corresponding interrupt source.
- IACK, which returns the vector of the highest priority pending interrupt or the spurious vector (SVR[VECTOR]) if none is pending.
- Reserved fields always return 0.

When the PIC is in mixed mode (GCR[M] = 1), the following guidelines are recommended:

- All PIC registers must be located in a cache-inhibited, guarded area (configured through the core's MMU).
- The PIC portion of the address map must be set up appropriately.

In addition, the following initialization sequence is recommended:

1. Write the vector, priority, and polarity values in each interrupt's vector/priority register, leaving their MSK (mask) bit set. This is required only if interrupts are used.
2. Clear CTPR (CTPR = 0x0000_0000).
3. Program the PIC to mixed mode by setting GCR[M].
4. Clear the MSK bit in the vector/priority registers to be used.
5. Perform a software loop to clear all pending interrupts:
   — Load counter with FRR[NIRQ].
   — While counter > 0, read IACK and write EOI to guarantee all the IPR and ISR bits are cleared.
6. Set the processor core CTPR values to the desired values.
7. Read the MSIRs to clear any pending message signaled interrupts that may have been pending before a soft reset.
8. Set MER to 0x0000_000F. See Section 9.3.5.2, "Message Enable Register (MER)," for more information.

Depending on the interrupt system configuration, the PIC may generate spurious interrupts to clear interrupts latched during power-up. A spurious or non-spurious vector is returned for an interrupt acknowledge cycle in this case. See the programming note below for the non-spurious case.

NOTE

Because the default polarity/sense for external interrupts is edge-sensitive, and edge-sensitive interrupts are not cleared until they are acknowledged, it is possible for the PIC to store spurious edges detected during power-up as pending external interrupts. If software permanently configures an external interrupt source to be edge-sensitive, it may receive the vector for the interrupt source and not a spurious interrupt vector when software clears the mask bit. This can occur once for any edge-sensitive interrupt when its mask bit is first cleared and the PIC is in mixed mode.

To avoid a false interrupt for this case, software can clear the IPR of these spurious edge detections by first configuring the polarity/sense of external interrupt sources to be level-sensitive: high-level if the input is a positive-edge source and low-level if it is a negative-edge source (while the mask bit remains set). After this is complete, configuring the external interrupt source as edge-sensitive does not cause a false interrupt.

9.5.1.2 Changing Interrupt Source Configuration

To change the vector, priority, polarity, sense, or destination of an active (unmasked) interrupt source, the following steps should be taken:

1. Mask the source using the mask (MSK) bit in the vector/priority register.
2. Wait for the activity (A) bit for that source to be cleared.
3. Make the desired changes.
4. Unmask the source.

Note that changing the destination from *int* to *cint* or $\overline{\text{IRQ\_OUT}}$ makes the A, MSK, and PRIORTY fields meaningless.

Chapter 10
I²C Interfaces

This chapter describes the two inter-IC (IIC or I²C) bus interfaces implemented on this device.

10.1 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 10-1 shows a block diagram of the two I²C interfaces.

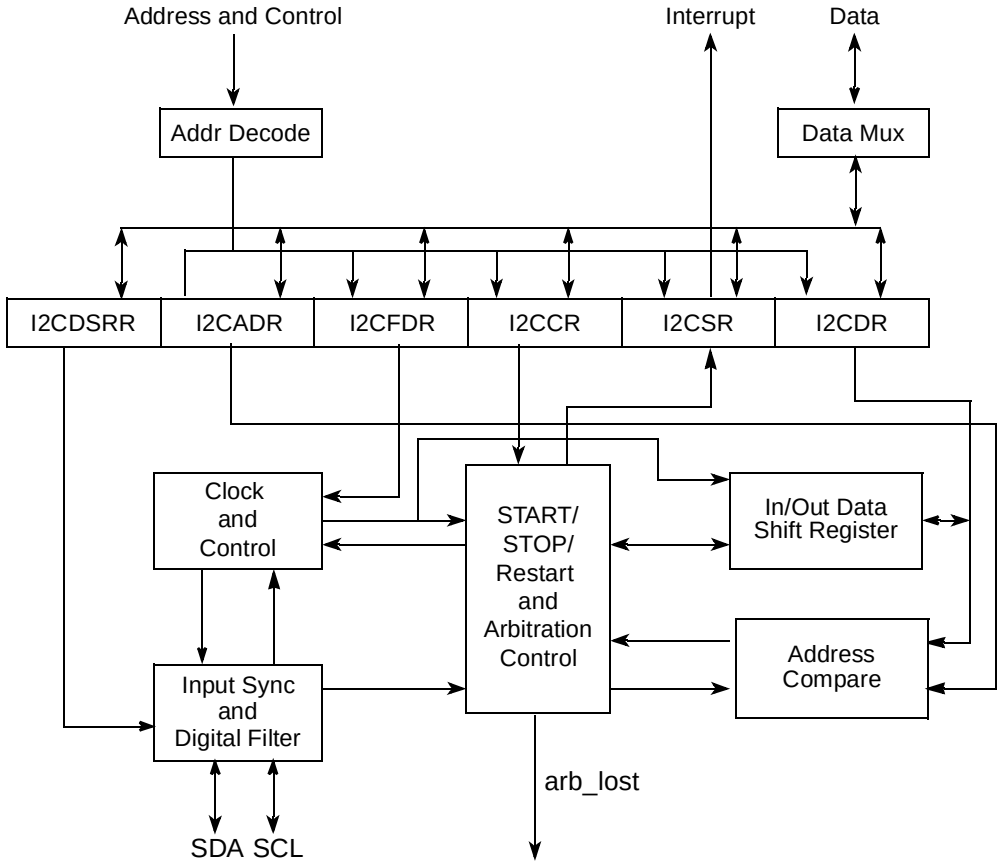


Figure 10-1. I²C Block Diagram

10.1.1 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.

10.1.2 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

10.1.3 Modes of Operation

The I²C units on this device can operate in one of the following modes:

- Master mode—The I²C is the driver of the SDA line. It cannot use its own slave address as a calling address. The I²C cannot be a master and a slave simultaneously.
- Slave mode—The I²C is not the driver of the SDA line. The module must be enabled before a START condition from a non-I²C master is detected.
- Interrupt-driven byte-to-byte data transfer—When successful slave addressing is achieved (and SCL returns to zero), the data transfer can proceed on a byte-to-byte basis in the direction specified by the R/$\overline{W}$ bit sent by the calling master. Each byte of data must be followed by an acknowledge bit, which is signaled from the receiving device. Several bytes can be transferred during a data transfer session.
- Boot sequencer mode—This mode can be used to initialize the configuration registers in the device after the I²C1 module is initialized. Note that the device powers up with boot sequencer mode disabled as a default, but this mode can be selected with the cfg_boot_seq[0:1] power-on reset (POR) configuration signals that are located on the LGPL3 and LGPL5 signals.

Additionally, the following three I²C-specific states are defined for the I²C interface:

- START condition—This condition denotes the beginning of a new data transfer (each data transfer contains several bytes of data) and awakens all slaves.
- Repeated START condition—A START condition that is generated without a STOP condition to terminate the previous transfer.
- STOP condition—The master can terminate the transfer by generating a STOP condition to free the bus.

10.2 External Signal Descriptions

The following sections give an overview of signals and provide detailed signal descriptions.

10.2.1 Signal Overview

The I²C interface uses the SDA and SCL signals, described in Table 10-1, for data transfer. Note that the signal patterns driven on SDA represent address, data, or read/write information at different stages of the protocol.

Table 10-1. I²C Interface Signal Descriptions

Signal Name	Idle State	I/O	State Meaning
Serial Clock (IIC*n*_SCL)	HIGH	I	When the I²C module is idle or acts as a slave, SCL defaults as an input. The unit uses SCL to synchronize incoming data on SDA. The bus is assumed to be busy when SCL is detected low.
		O	As a master, the I²C module drives SCL along with SDA when transmitting. As a slave, the I²C module drives SCL low for data pacing.
Serial Data (IIC*n*_SDA)	HIGH	I	When the I²C module is idle or in a receiving mode, SDA defaults as an input. The unit receives data from other I²C devices on SDA. The bus is assumed to be busy when SDA is detected low.
		O	When writing as a master or slave, the I²C module drives data on SDA synchronous to SCL.

10.2.2 Detailed Signal Descriptions

SDA and SCL, described in Table 10-2, serve as a communication interconnect with other devices. All devices connected to these two signals must have open-drain or open-collector outputs. The logic AND function is performed on both of these signals with external pull-up resistors. Refer to the device hardware specifications for the electrical characteristics of these signals.

Table 10-2. I²C Interface Signal—Detailed Signal Descriptions

Signal	I/O	Description	
IIC*n*_SCL	I/O	Serial clock. Performs as an input when the device is programmed as an I²C slave. SCL also performs as an output when the device is programmed as an I²C master.	
	O	As outputs for the bidirectional serial clock, these signals operate as described below.	
		State Meaning	Asserted/Negated—Driven along with SDA as the clock for the data.
	I	As inputs for the bidirectional serial clock, these signals operate as described below.	
		State Meaning	Asserted/Negated—The I²C unit uses this signal to synchronize incoming data on SDA. The bus is assumed to be busy when this signal is detected low.
IIC*n*_SDA	I/O	Serial data. Performs as an input when the device is in a receiving mode. SDA also performs as an output signal when the device is transmitting (as an I²C master or a slave).	
	O	As outputs for the bidirectional serial data, these signals operate as described below.	
		State Meaning	Asserted/Negated— Data is driven.
	I	As inputs for the bidirectional serial data, these signals operate as described below.	
		State Meaning	Asserted/Negated—Used to receive data from other devices. The bus is assumed to be busy when SDA is detected low.

10.3 Memory Map/Register Definition

Table 10-3 lists the I²C-specific registers and their offsets. It lists the offset, name, and a cross-reference to the complete description of each register. Note that the full register address is comprised of CCSRBAR together with the block base address and offset listed in Table 10-3. The offsets to the memory map table are defined for both I²C interfaces. That is, I²C1 starts at address offset 0x000, and I²C2 starts at address offset 0x100. The registers for I²C1 are listed in Table 10-3, but the registers for I²C2 are not. Note that the registers are the same for I²C2 except that the offsets change from 0x0*nn* to 0x1*nn*.

All I²C registers are one byte wide. Reads and writes to these registers must be byte-wide operations.

Table 10-3. I²C Memory Map

I²C Controller 1—Block Base Address 0x0_3000 **I²C Controller 2—Block Base Address 0x0_3100**				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
I²C1 Registers				
0x000	I2CADR—I²C address register	R/W	0x00	10.3.1.1/10-5
0x004	I2CFDR—I²C frequency divider register	R/W	0x00	10.3.1.2/10-6

Table 10-3. I²C Memory Map (continued)

I²C Controller 1—Block Base Address 0x0_3000 I²C Controller 2—Block Base Address 0x0_3100				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0x008	I2CCR—I²C control register	Mixed	0x00	10.3.1.3/10-7
0x00C	I2CSR—I²C status register	Mixed	0x81	10.3.1.4/10-9
0x010	I2CDR—I²C data register	R/W	0x00	10.3.1.5/10-10
0x014	I2CDFSRR—I²C digital filter sampling rate register	R/W	0x10	10.3.1.6/10-11
I²C2 Registers				
0x100–0x114	I²C2 Registers[1]			

1 I²C2 has the same memory-mapped registers that are described for I²C1 from 0x000 to 0x014, except the offsets range from 0x100 to 0x114.

10.3.1 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).

10.3.1.1 I²C Address Register (I2CADR)

Figure 10-2 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.

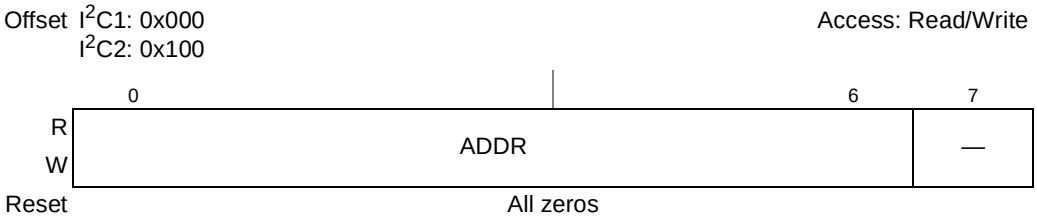


Figure 10-2. I²C Address Register (I2CADR)

Table 10-4 describes the fields of I2CADR.

Table 10-4. 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

10.3.1.2 I²C Frequency Divider Register (I2CFDR)

Figure 10-3 shows the bits of the I²C frequency divider register. Refer to application note AN2919, *Determining the I²C Frequency Divider Ratio for SCL*, for additional guidance regarding the proper use of I2CFDR and I2CDFSRR.

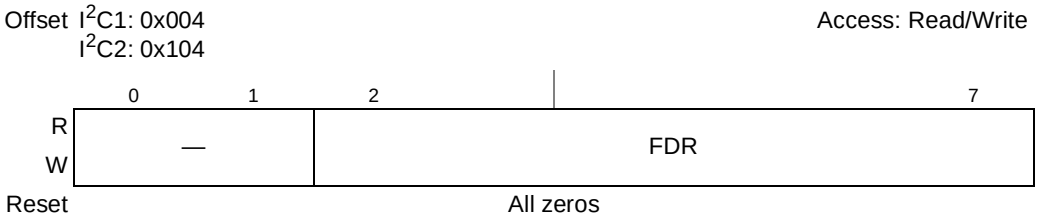


Figure 10-3. I²C Frequency Divider Register (I2CFDR)

Table 10-5 describes the bit settings of I2CFDR. It also maps the I2CFDR[FDR] field to the clock divider values.

Table 10-5. I2CFDR Field Descriptions

Bits	Name	Description
0–1	—	Reserved
2–7	FDR	Frequency divider ratio. Used to prescale the clock for bit rate selection. The serial bit clock frequency of SCL is equal to one half the platform (MPX) clock divided by the designated divider. Note that the frequency divider value can be changed at any point in a program. The serial bit clock frequency divider selections are described as follows:

FDR	Divider (Decimal)	FDR	Divider (Decimal)	FDR	Divider (Decimal)
0x00	384	0x16	12288	0x2B	1024
0x01	416	0x17	15360	0x2C	1280
0x02	480	0x18	18432	0x2D	1536
0x03	576	0x19	20480	0x2E	1792
0x04	640	0x1A	24576	0x2F	2048
0x05	704	0x1B	30720	0x30	2560
0x06	832	0x1C	36864	0x31	3072
0x07	1024	0x1D	40960	0x32	3584
0x08	1152	0x1E	49152	0x33	4096
0x09	1280	0x1F	61440	0x34	5120
0x0A	1536	0x20	256	0x35	6144
0x0B	1920	0x21	288	0x36	7168
0x0C	2304	0x22	320	0x37	8192
0x0D	2560	0x23	352	0x38	10240
0x0E	3072	0x24	384	0x39	12288
0x0F	3840	0x25	448	0x3A	14336
0x10	4608	0x26	512	0x3B	16384
0x11	5120	0x27	576	0x3C	20480
0x12	6144	0x28	640	0x3D	24576
0x13	7680	0x29	768	0x3E	28672
0x14	9216	0x2A	896	0x3F	32768
0x15	10240				

10.3.1.3 I²C Control Register (I2CCR)

Figure 10-4 shows the I²C control register, I2CCR.

Offset I²C1: 0x008
I²C2: 0x108

Access: Mixed

	0	1	2	3	4	5	6	7
R	MEN	MIEN	MSTA	MTX	TXAK		—	BCST
W						RSTA		
Reset	All zeros							

Figure 10-4. I²C Control Register (I2CCR)

Table 10-6 describes the bit settings of the I2CCR.

Table 10-6. I2CCR Field Descriptions

Bits	Name	Description
0	MEN	Module enable. This bit controls the software reset of the I²C module. 0 The module is reset and disabled. When low, the interface is held in reset but the registers can still be accessed. 1 The I²C module is enabled. This bit must be set before any other control register bits have any effect. All I²C registers for slave receive or master START can be initialized before setting this bit.
1	MIEN	Module interrupt enable 0 Interrupts from the I²C module are disabled. This does not clear any pending interrupt conditions. 1 Interrupts from the I²C module are enabled. An interrupt occurs provided I2CSR[MIF] is also set.
2	MSTA	Master/slave mode START 0 When this bit is changed from one to zero, a STOP condition is generated and the mode changes from master to slave. 1 Cleared without generating a STOP condition when the master loses arbitration. When this bit is changed from zero to one, a START condition is generated on the bus, and master mode is selected.
3	MTX	Transmit/receive mode select. This bit selects the direction of the master and slave transfers. When configured as a slave, this bit should be set by software according to I2CSR[SRW]. In master mode, the bit should be set according to the type of transfer required. Therefore, for address cycles, this bit is always high. The MTX bit is cleared when the master loses arbitration. 0 Receive mode 1 Transmit mode
4	TXAK	Transfer acknowledge. This bit specifies the value driven onto the SDA line during acknowledge cycles for both master and slave receivers. The value of this bit only applies when the I²C module is configured as a receiver, not a transmitter. It also does not apply to address cycles; when the device is addressed as a slave, an acknowledge is always sent. 0 An acknowledge signal (low value on SDA) is sent out to the bus at the 9th clock after receiving one byte of data. 1 No acknowledge signal response (high value on SDA) is sent.
5	RSTA	Repeated START. Setting this bit always generates a repeated START condition on the bus, provides the device with the current bus master. Attempting a repeated START at the wrong time (or if the bus is owned by another master), results in loss of arbitration. Note that this bit is not readable, which means if a read is performed to I2CCR[RSTA], a zero value is returned. 0 No START condition is generated 1 Generates repeated START condition
6	—	Reserved
7	BCST	Broadcast 0 Disables the broadcast accept capability 1 Enables the I²C to accept broadcast messages at address zero

10.3.1.4 I²C Status Register (I2CSR)

The I²C status register, shown in Figure 10-5, is read only with the exception of the MIF and MAL bits, which can be cleared by software. The MCF and RXAK bits are set at reset; all other I2CSR bits are cleared on reset.

Offset I²C1: 0x00C
I²C2: 0x10C

Access: Mixed

	0	1	2	3	4	5	6	7
R	MCF	MAAS	MBB	MAL	BCSTM	SRW	MIF	RXAK
W								
Reset	1	0	0	0	0	0	0	1

Figure 10-5. I²C Status Register (I2CSR)

Table 10-7 describes the bit settings of the I2CSR.

Table 10-7. I2CSR Field Descriptions

Bits	Name	Description
0	MCF	Data transfer. When one byte of data is transferred, the bit is cleared. It is set by the falling edge of the 9th clock of a byte transfer. 0 Byte transfer in progress. MCF is cleared under the following conditions: •When I2CDR is read in receive mode or •When I2CDR is written in transmit mode 1 Byte transfer is completed
1	MAAS	Addressed as a slave. When the value in I2CDR matches with the calling address, this bit is set. The processor is interrupted, if I2CCR[MIEN] is set. Next, the processor must check the SRW bit and set I2CCR[MTX] accordingly. Writing to the I2CCR automatically clears this bit. 0 Not addressed as a slave 1 Addressed as a slave
2	MBB	Bus busy. Indicates the status of the bus. When a START condition is detected, MBB is set. If a STOP condition is detected, it is cleared. 0 I²C bus is idle 1 I²C bus is busy
3	MAL	Arbitration lost. Automatically set when the arbitration procedure is lost. Note that the device does not automatically retry a failed transfer attempt. 0 Arbitration is not lost. Can only be cleared by software 1 Arbitration is lost
4	BCSTM	Broadcast match 0 There has not been a broadcast match. 1 The calling address matches with the broadcast address instead of the programmed slave address. This also sets 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 10-7. I2CSR Field Descriptions (continued)

Bits	Name	Description
6	MIF	Module interrupt. The MIF bit is set when an interrupt is pending, causing a processor interrupt request (provided I2CCR[MIEN] is set). The interrupts for I²C1 and I²C2 are combined into one interrupt, which is sourced by the dual I²C controller. 0 No interrupt is pending. Can be cleared only by software. 1 Interrupt is pending. MIF is set when one of the following events occurs: •One byte of data is transferred (set at the falling edge of the 9th clock). •The value in I2CADR matches with the calling address in slave-receive mode. •Arbitration is lost.
7	RXAK	Received acknowledge. The value of SDA during the reception of acknowledge bit of a bus cycle. If the received acknowledge bit (RXAK) is low, it indicates that an acknowledge signal has been received after the completion of eight bits of data transmission on the bus. If RXAK is high, it means no acknowledge signal has been detected at the 9th clock. 0 Acknowledge received 1 No acknowledge received

10.3.1.5 I²C Data Register (I2CDR)

The I2C data register is shown in Figure 10-6.

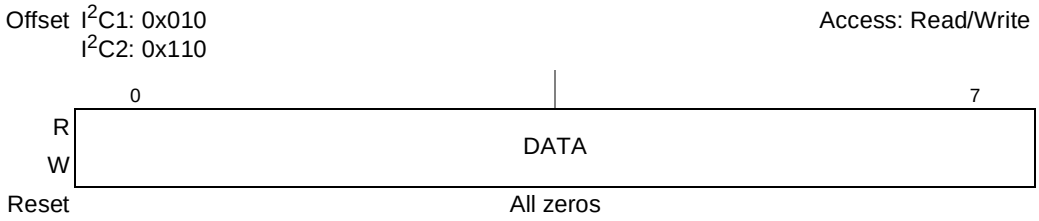


Figure 10-6. I²C Data Register (I2CDR)

Table 10-8 shows the bit descriptions for I2CDR.

Table 10-8. I2CDR Field Descriptions

Bits	Name	Description
0–7	DATA	Transmission starts when an address and the R/W bit are written to the data register and the I²C interface performs as the master. A data transfer is initiated when data is written to the I2CDR. The most significant bit is sent first in both cases. In master receive mode, reading the data register allows the read to occur, but also allows the I²C module to receive the next byte of data on the I2C interface. In slave mode, the same function is available after it is addressed. Note that in both master receive and slave receive modes, the very first read is always a dummy read.

10.3.1.6 Digital Filter Sampling Rate Register (I2CDFSRR)

The digital filter sampling rate register (I2CDFSRR) is shown in Figure 10-7. Refer to application note AN2919, *Determining the I2C Frequency Divider Ratio for SCL*, for additional guidance regarding the proper use of I2CFDR and I2CDFSRR.

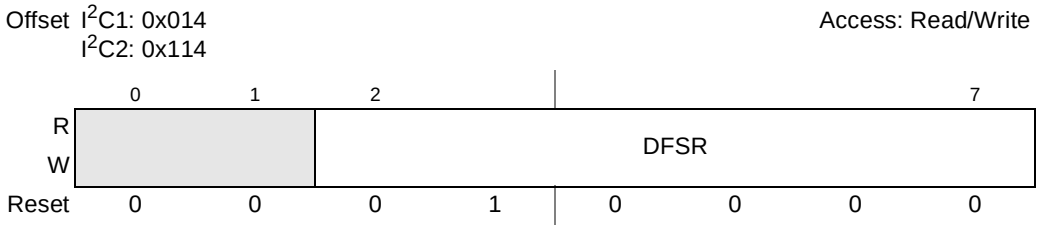


Figure 10-7. I2C Digital Filter Sampling Rate Register (I2CDFSRR)

Table 10-9 shows the field descriptions for I2CDFSRR.

Table 10-9. I2CDFSRR Field Descriptions

Bits	Name	Description
0–1	—	Reserved
2–7	DFSR	Digital filter sampling rate. To assist in filtering out signal noise, the sample rate is programmed. This field is used to prescale the frequency at which the digital filter takes samples from the I2C bus. The resulting sampling rate is calculated by dividing one half the platform (MPX clock) frequency by the non-zero value of DFSR.

10.4 Functional Description

The I2C unit always performs as a slave receiver as a default, unless explicitly programmed to be a master or slave transmitter. After the boot sequencer has completed (when powered up in boot sequencer mode), the I2C interface performs as a slave receiver.

Note that the boot sequencer only functions from the I2C1 interface; the I2C2 interface cannot be used for this purpose.

10.4.1 Transaction Protocol

A standard I2C transfer consists of the following:

- START condition
- Slave target address transmission
- Data transfer
- STOP condition

Figure 10-8 shows the interaction of these four parts with the calling address, data byte, and new calling address components of the I²C protocol. The details of the protocol are described in the following sections.

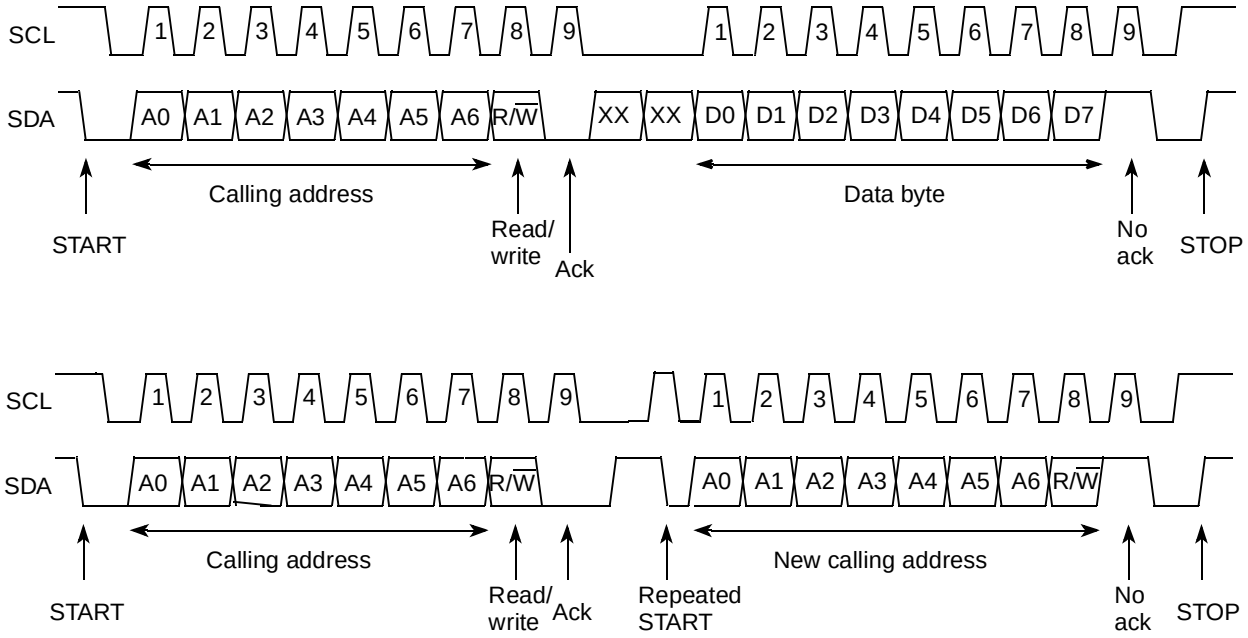


Figure 10-8. I²C Interface Transaction Protocol

10.4.1.1 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 10-8, 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].

10.4.1.2 Slave Address Transmission

The first byte of data is transferred by the master immediately after the START condition is the slave address. This is a seven-bit calling address followed by a R/$\overline{W}$ bit, which indicates the direction of the data being transferred to the slave. Each slave in the system has a unique address. In addition, when the I²C module is operating as a master, it must not transmit an address that is the same as its slave address. An I²C device cannot be master and slave at the same time; if this is attempted, the results are boundedly undefined.

Only the slave with a calling address that matches the one transmitted by the master responds by returning an acknowledge bit (pulling the SDA signal low at the 9th clock) as shown in Figure 10-8. If no slave acknowledges the address, the master should generate a STOP condition or a repeated START condition.

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

The I2C module responds to a general call (broadcast) command when I2CCR[BCST] is set. A broadcast address is always zero; however the I2C module does not check the R/$\overline{W}$ bit. The second byte of the broadcast message is the master address. Because the second byte is automatically acknowledged by hardware, the receiver device software must verify that the broadcast message is intended for itself by reading the second byte of the message. If the master address is for another receiver device and the third byte is a write command, software can ignore the third byte during the broadcast. If the master address is for another receiver device and the third byte is a read command, software must write 0xFF to I2CDR with I2CCR[TXAK] = 1, so that it does not interfere with the data written from the addressed device.

Each data byte is 8 bits long. Data bits can be changed only while SCL is low and must be held stable while SCL is high, as shown in Figure 10-8. 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.

10.4.1.3 Repeated START Condition

Figure 10-8 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.

10.4.1.4 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 10-8. 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 10.4.1.3, “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.

10.4.1.5 Protocol Implementation Details

The following sections give details of how aspects of the protocol are implemented in this I2C module.

10.4.1.5.1 Transaction Monitoring—Implementation Details

The different conditions of the I²C data transfers are monitored as follows:

- START conditions are detected when an SDA fall occurs while SCL is high.
- STOP conditions are detected when an SDA rise occurs while SCL is high.
- Data transfers in progress are canceled when a STOP condition is detected or if there is a slave address mismatch. Cancellation of data transactions resets the clock module.
- The bus is detected to be busy upon the detection of a START condition, and idle upon the detection of a STOP condition.

10.4.1.5.2 Control Transfer—Implementation Details

The I²C module contains logic that controls the output to the serial data (SDA) and serial clock (SCL) lines of the I²C. The SCL output is pulled low as determined by the internal clock generated in the clock module. The SDA output can only change at the midpoint of a low cycle of the SCL, unless it is performing a START, STOP, or restart condition. Otherwise, the SDA output is held constant.

The SDA signal is pulled low when one or more of the following conditions are true in either master or slave mode:

- Master mode
  - Data bit (transmit)
  - Ack bit (receive)
  - START condition
  - STOP condition
  - Restart condition
- Slave mode
  - Acknowledging address match
  - Data bit (transmit)
  - Ack bit (receive)

The SCL signal corresponds to the internal SCL signal when one or more of the following conditions are true in either master or slave mode:

- Master mode
  - Bus owner
  - Lost arbitration
  - START condition
  - STOP condition
  - Restart condition begin
  - Restart condition end
- Slave mode
  - Address cycle
  - Transmit cycle

— Ack cycle

10.4.1.6 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

10.4.2 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.

10.4.2.1 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.

10.4.3 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.

10.4.4 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

10.4.4.1 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.

10.4.4.2 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.

10.4.4.2.1 Input Signal Synchronization

The input synchronization block synchronizes the input SCL and SDA signals to the system clock and detects transitions of these signals.

10.4.4.2.2 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.

10.4.4.3 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.

10.4.5 Boot Sequencer Mode

If boot sequencer mode is selected on POR (by the settings on the cfg_boot_seq[0:1] reset configuration signals, as described in Section 4.4.3.11, “Boot Sequencer Configuration”), the I²C1 module communicates with one or more EEPROMs through the I²C interface on IIC1_SCL and IIC1_SDA. The boot sequencer accesses the I²C1 serial ROM device at a serial bit clock frequency equal to the platform (MPX) clock frequency divided by 2560. The EEPROM(s) can be programmed to initialize one or more configuration registers of this integrated device.

If the boot sequencer is enabled for normal I²C addressing mode, the I²C interface initiates the following sequence during reset:

1. Generate RESET sequence (START then 9 SCL cycles) to the EEPROM twice. This clears any transactions that may have been in progress prior to the reset.
2. Generate START
3. Transmit 0xA0 which is the 7-bit calling address (0b101_0000) with a write command appended (0 as the least significant bit).
4. Transmit 0x00 which is the 8-bit starting address
5. Generate a repeated START
6. Transmit 0xA1 which is the 7-bit calling address (0b101_0000) with a read command appended (1 as the least significant bit).
7. Receive 256 bytes of data from the EEPROM (unless the CONT bit is cleared in the data structure).
8. Generate a repeated START
9. Transmit 0xA2 which is the 7-bit calling address of the second target (0b101_0001) with a write command appended (0 as the least significant bit).
10. Transmit 0x00 which is the 8-bit starting address for the second target.
11. Generate a repeated START

12. Transmit 0xA3 which is the 7-bit calling address (0b101_0001) with a read command appended (1 as the least significant bit).
13. Receive another 256 bytes of data from the second EEPROM (unless the CONT bit is cleared in the data structure).

The sequence repeats with successive targets until the CONT bit in the data structure is cleared and the CRC check is executed. If the last register is not detected (that is, the CONT bit is never cleared) before wrapping back to the first address, an error condition is detected, causing the device to hang and the $\overline{\text{HRESET\_REQ}}$ signal to assert externally. The I²C module continues to read from the EEPROM(s) as long as the continue (CONT) bit is set in the EEPROM(s). The CONT bit resides in the address/attributes field that is transferred from the EEPROM, as described in Section 10.4.5.1, "EEPROM Calling Address." There should be no other I²C traffic when the boot sequencer is active.

The boot sequencer mode also supports an extension of the standard I²C interface that uses more address bits to allow for EEPROM devices that have more than 256 bytes, and this extended addressing mode is selectable during POR with a different encoding on the cfg_boot_seq[0:1] reset configuration signals. In this mode, only one EEPROM device may be used, and the maximum number of registers is limited by the size of the EEPROM. If the boot sequencer is enabled for extended I²C addressing mode, the I²C interface initiates the following sequence during reset:

1. Generate RESET sequence (START then 9 SCL cycles) to the EEPROM twice. This clears any transactions that may have been in progress prior to the reset.
2. Generate START
3. Transmit 0xA0 which is the 7-bit calling address (0b101_0000) with a write command appended (0 as the least significant bit).
4. Transmit 0x00 which is the high-order starting address
5. Transmit 0x00 which is the low-order starting address
6. Generate a repeated START
7. Transmit 0xA1 which is the 7-bit calling address (0b101_0000) with a read command appended (1 as the least significant bit).
8. Receive data continuously from the EEPROM until the CONT bit is cleared and the CRC check is executed. See Section 10.4.5.2, "EEPROM Data Format," for more information.

Note that as described in Section 4.4.3.11, "Boot Sequencer Configuration," the default value for the cfg_boot_seq[0:1] reset configuration pins is 0b11, which corresponds to the I²C boot sequencer being disabled at power-up.

10.4.5.1 EEPROM Calling Address

The MPC8641uses 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.

10.4.5.2 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 10-9. The first 3 bytes hold the attributes and address offset, as follows. The attributes contained are alternate configuration space (ACS), byte enables, and continue (CONT). The boot sequencer expects the address offset to be a 32-bit (word) offset, that is, the 2 low-order bits are not included in the boot sequencer command. For example, to access LAWBAR0 (byte offset of 0x00C08), the boot sequencer ADDR[0:17] should be set to 0x00302.

After the first 3 bytes, 4 bytes of data should hold the desired value of the configuration register, regardless of the size of the transaction. Byte enables should be asserted for any byte that is written to the configuration register, and they should be asserted contiguously, creating a 1-, 2-, or 4-byte write to a register. The boot sequencer assumes that a big-endian address is stored in the EEPROM. In addition, byte enable bit 0 (bit 1 of the byte) corresponds to the most-significant byte of data (data[0:7]), and byte enable bit 3 (bit 4 of the byte) corresponds to the LSB of data (data[24:31]).

By setting ACS, an alternate configuration space address is prepended to the write request from the boot sequencer. Otherwise, CCSRBAR is prepended to the EEPROM address.

If CONT is cleared, the first 3 bytes, including ACS, the byte enables, and the address, must also be cleared. Also, the data contains the final cyclic redundancy check (CRC). A CRC-32 algorithm is used to check the integrity of the data. The polynomial used is:

$$1 + x^1 + x^2 + x^4 + x^5 + x^7 + x^8 + x^{10} + x^{11} + x^{12} + x^{16} + x^{22} + x^{23} + x^{26} + x^{32}$$

CRC values are calculated using the above polynomial with a start value of 0xFFFF_FFFF and an XOR with 0x0000_0000. The CRC should cover all bytes stored in the EEPROM prior to the CRC. This includes the preamble, all register preloads, and the first 3 bytes of the last 7-byte preload (which should be all zeros). If a preamble or CRC fail is detected, the device hangs and the external $\overline{\text{HRESET\_REQ}}$ signal asserts. If there is a preamble fail, the boot sequencer may continue to pull I²C pins low until a hard reset occurs.

0	1 – 4	5	6 – 7
ACS	BYTE_EN	CONT	ADDR[0–1]
ADDR[2–9]			
ADDR[10–17]			
DATA[0–7]			
DATA[8–15]			
DATA[16–23]			
DATA[24–31]			

Figure 10-9. EEPROM Data Format for One Register Preload Command

Figure 10-10 shows an example of the EEPROM contents, including the preamble, data format, and CRC.

0	1	2	3	4	5	6	7	
1	0	1	0	1	0	1	0	Preamble
0	1	0	1	0	1	0	1	
1	0	1	0	1	0	1	0	
ACS	BYTE_EN				1	ADDR[0–1]		First Configuration Preload Command
ADDR[2–9]								
ADDR[10–17]								
DATA[0–7]								
DATA[8–15]								
DATA[16–23]								
DATA[24–31]								
ACS	BYTE_EN				1	ADDR[0–1]		Second Configuration Preload Command
ADDR[2–9]								
ADDR[10–17]								
DATA[0–7]								
DATA[8–15]								
DATA[16–23]								
DATA[24–31]								
. . .								
ACS	BYTE_EN				1	ADDR[0–1]		Last Configuration Preload Command
ADDR[2–9]								
ADDR[10–17]								
DATA[0–7]								
DATA[8–15]								
DATA[16–23]								
DATA[24–31]								
0	0	0	0	0	0	0	0	End Command
0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	

Figure 10-10. EEPROM Contents

CRC[0–7]	Cyclic Redundancy Check
CRC[8–15]	
CRC[16–23]	
CRC[24–31]	

Figure 10-10. EEPROM Contents (continued)

10.5 Initialization/Application Information

This section describes some programming guidelines recommended for the I²C interface. Figure 10-11 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.

10.5.1 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 MPX (platform) clock. Note that the platform frequency must first be divided by two; see Section 10.3.1.2, “I²C Frequency Divider Register (I2CFDR),” for more details.
3. Update I2CADR to define the slave address for this device.
4. Modify I2CCR to select master/slave mode, transmit/receive mode, and interrupt-enable or disable.
5. Set the I2CCR[MEN] to enable the I²C interface.

10.5.2 Generation of START

After initialization, the following sequence can be used to generate START:

1. If the device is connected to a multimaster I²C system, test the state of I2CSR[MBB] to check whether the serial bus is free (I2CSR[MBB] = 0) before switching to master mode.
2. Select master mode (set I2CCR[MSTA]) to transmit serial data and select transmit mode (set I2CCR[MTX]) for the address cycle.

3. Write the slave address being called into I2CDR. The data written to I2CDR[0–6] comprises the slave calling address. I2CCR[MTX] indicates the direction of transfer (transmit/receive) required from the slave.

The scenario above assumes that the I²C interrupt bit (I2CSR[MIF]) is cleared. If MIF is set at any time, an I²C interrupt is generated (provided interrupt reporting is enabled with I2CCR[MIEN] =1) so that the I²C interrupt handler can handle the interrupt. Note that the interrupts for I²C1 and I²C2 are combined into one interrupt, which is sourced by the dual I²C controller.

10.5.3 Post-Transfer Software Response

Transmission or reception of a byte automatically sets the data transferring bit (I2CSR[MCF]), which indicates that one byte has been transferred. The I²C interrupt bit (I2CSR[MIF]) is also set and an interrupt is generated to the processor if the interrupt function is enabled during the initialization sequence (I2CCR[MIEN] is set). In the interrupt handler, software must take the following steps:

1. Clear I2CSR[MIF]
2. Read the contents of the I²C data register (I2CDR) in receive mode or write to I2CDR in transmit mode. Note that this causes I2CSR[MCF] to be cleared. See Section 10.5.8, “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 10.5.8, “Interrupt Service Routine Flowchart.”

If the interrupt function is disabled, software can service the I2CDR in the main program by monitoring I2CSR[MIF]. In this case, I2CSR[MIF] must be polled rather than I2CSR[MCF] because MCF behaves differently when arbitration is lost. Note that interrupt or other bus conditions may be detected before the I²C signals have time to settle. Thus, when polling I2CSR[MIF] (or any other I2CSR bits), software delays may be needed in order to give the I²C signals sufficient time to settle.

During slave-mode address cycles (I2CSR[MAAS] is set), I2CSR[SRW] should be read to determine the direction of the subsequent transfer and I2CCR[MTX] should be programmed accordingly. For slave-mode data cycles (MAAS is cleared), I2CSR[SRW] is not valid and I2CCR[MTX] must be read to determine the direction of the current transfer. See Section 10.5.8, “Interrupt Service Routine Flowchart,” for more details.

10.5.4 Generation of STOP

A data transfer ends with a STOP condition generated by the master device. A master transmitter can generate a STOP condition after all the data has been transmitted.

If a master receiver wants to terminate a data transfer, it must inform the slave transmitter by not acknowledging the last byte of data (by setting the transmit acknowledge bit (I2CCR[TXAK])) before reading the next-to-last byte of data. At this time, the next-to-last byte of data has already been transferred on the I²C interface, so the last byte does not receive the data acknowledge (because I2CCR[TXAK] is set). Before the interrupt service routine reads the last byte of data, a STOP condition must first be generated.

The I²C controller automatically generates a STOP if I2CCR[TXAK] is set. Therefore, I2CCR[TXAK] must be set before allowing the I²C module to receive the last data byte on the I²C bus. Eventually, I2CCR[TXAK] needs to be cleared again for subsequent I²C transactions. This can be accomplished when setting up the I2CCR for the next transfer.

10.5.5 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].

10.5.6 Generation of SCL When SDA Low

It is sometimes necessary to force the I²C module to become the I²C bus master out of reset and drive SCL (even though SDA may already be driven, which indicates that the bus is busy). This can occur when a system reset does not cause all I²C devices to be reset. Thus, SDA can be driven low by another I²C device while this I²C module is coming out of reset and stays low indefinitely. The following procedure can be used to force this I²C module to generate SCL so that the device driving SDA can finish its transaction:

1. Disable the I²C module and set the master bit by setting I2CCR to 0x20
2. Enable the I²C module by setting I2CCR to 0xA0
3. Read the I2CDR
4. Return the I²C module to slave mode by setting I2CCR to 0x80

10.5.7 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.

10.5.7.1 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 10.5.8, “Interrupt Service Routine Flowchart.”

10.5.7.2 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 10.4.2.1, “Arbitration Control,” for more information.

10.5.8 Interrupt Service Routine Flowchart

Figure 10-11 shows an example algorithm for an I2C interrupt service routine. Deviation from the flowchart may result in unpredictable I2C bus behavior. However, in the slave receive mode 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 I2C register read or write to guarantee in-order instruction execution.

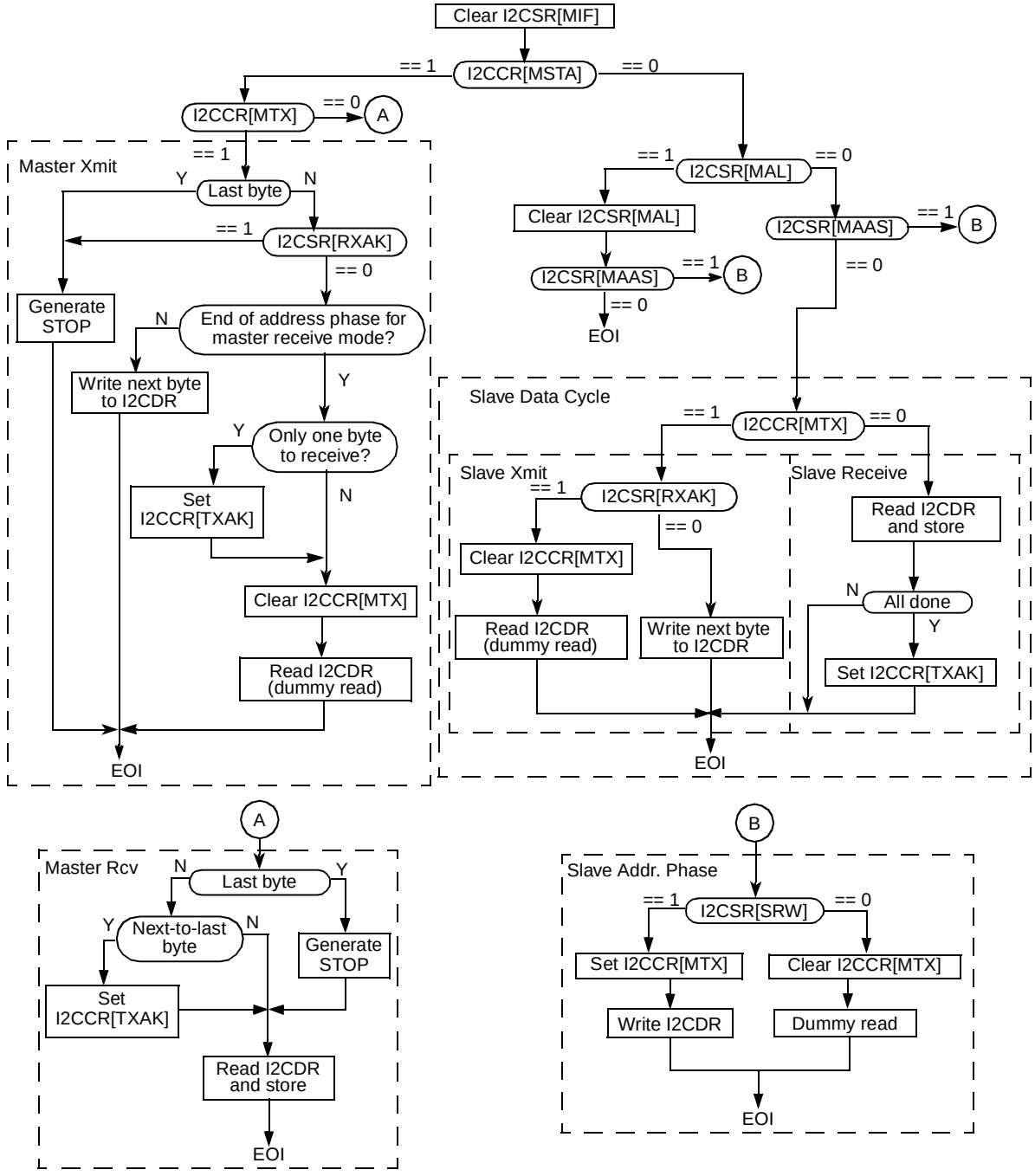


Figure 10-11. Example I²C Interrupt Service Routine Flowchart

Chapter 11
DUART

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

11.1 Overview

The DUART consists of two universal asynchronous receiver/transmitters (UARTs). The UARTs act independently; all references to UART refer to one of these receiver/transmitters. Each UART is clocked by the platform (MPX) 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 11-1, 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

11.1.1 Features

The DUART includes these distinctive features:

- Full-duplex operation
- Programming model compatible with the original PC16450 UART and the PC16550D (an improved version of the PC16450 that also operates in FIFO mode)
- PC16450 register reset values
- FIFO mode for both transmitter and receiver, providing 16-byte FIFOs
- Serial data encapsulation and decapsulation with standard asynchronous communication bits (START, STOP, and parity)
- Maskable transmit, receive, line status, and modem status interrupts
- Software-programmable baud generators that divide the platform clock by 1 to ($2^{16} - 1$) and generate a 16x clock for the transmitter and receiver engines

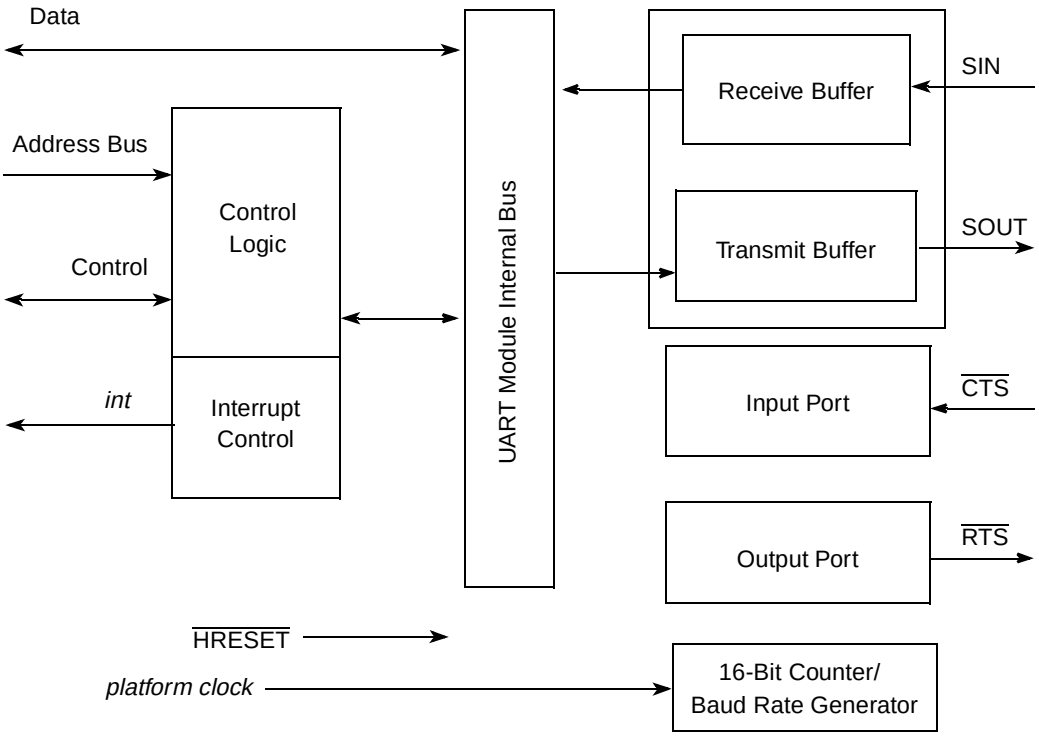


Figure 11-1. UART Block Diagram

- Clear to send ($\overline{\text{CTS}}$) and ready to send ($\overline{\text{RTS}}$) modem control functions
- Software-selectable serial interface data format (data length, parity, 1/1.5/2 STOP bit, baud rate)
- Line and modem status registers
- Line-break detection and generation
- Internal diagnostic support, local loopback, and break functions
- Prioritized interrupt reporting
- Overrun, parity, and framing error detection

11.1.2 Modes of Operation

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

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

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

11.1.2.1 DUART Signal Mode Selection

11.2 External Signal Descriptions

This section contains a signal overview and detailed signal descriptions.

11.2.1 Signal Overview

The DUART signals are described in Table 11-1. Note that although the actual device signal names are prepended with the UART_ prefix as shown in the table, the functional (abbreviated) signal names are often used throughout this chapter.

11.2.2 Detailed Signal Descriptions

Table 11-1 provided detailed description of the external DUART signals.

Table 11-1. DUART Signals—Detailed Signal Descriptions

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

11.3 Memory Map/Register Definition

Table 11-2 lists the DUART registers and their offsets. It lists the address, name, and a cross-reference to the complete description of each register. Note that the full register address is comprised of CCSRBAR together with the block base address and offset listed in Table 11-2.

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

The registers in each UART interface are used for configuration, control, and status. The divisor latch access bit, ULCR[DLAB], is used to access the divisor latch least- and most-significant bit registers and the alternate function register. Refer to Section 11.3.1.7, “Line Control Registers (ULCRn),” for more information on ULCR[DLAB].

All the DUART registers are one byte wide. Reads and writes to these registers must be byte-wide operations. Table 11-2 provides a register summary with references to the section and page that contains detailed information about each register. Undefined byte address spaces within offset 0x000–0xFFF are reserved.

Table 11-2. DUART Register Summary

DUART—Block Base Address 0x0_4000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
UART0 Registers				
0x500	URBR—ULCR[DLAB] = 0 UART0 receiver buffer register	R	0x00	11.3.1.1/11-5
0x500	UTHR—ULCR[DLAB] = 0 UART0 transmitter holding register	W	0x00	11.3.1.2/11-5
0x500	UDLB—ULCR[DLAB] = 1 UART0 divisor least significant byte register	R/W	0x00	11.3.1.3/11-6
0x501	UIER—ULCR[DLAB] = 0 UART0 interrupt enable register	R/W	0x00	11.3.1.4/11-8
0x501	UDMB—ULCR[DLAB] = 1 UART0 divisor most significant byte register	R/W	0x00	11.3.1.3/11-6
0x502	UIIR—ULCR[DLAB] = 0 UART0 interrupt ID register	R	0x01	11.3.1.5/11-8
0x502	UFCR—ULCR[DLAB] = 0 UART0 FIFO control register	W	0x00	11.3.1.6/11-10
0x502	UAFR—ULCR[DLAB] = 1 UART0 alternate function register	R/W	0x00	11.3.1.12/11-16
0x503	ULCR—ULCR[DLAB] = x UART0 line control register	R/W	0x00	11.3.1.7/11-11
0x504	UMCR—ULCR[DLAB] = x UART0 modem control register	R/W	0x00	11.3.1.8/11-13
0x505	ULSR—ULCR[DLAB] = x UART0 line status register	R	0x60	11.3.1.9/11-14
0x506	UMSR—ULCR[DLAB] = x UART0 modem status register	R	0x00	11.3.1.10/11-15
0x507	USCR—ULCR[DLAB] = x UART0 scratch register	R/W	0x00	11.3.1.11/11-16
0x510	UDSR—ULCR[DLAB] = x UART0 DMA status register	R	0x01	11.3.1.13/11-17

Table 11-2. DUART Register Summary (continued)

DUART—Block Base Address 0x0_4000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
UART1 Registers				
0x600–0x610	UART1 Registers[1]			

[1] UART1 has the same memory-mapped registers that are described for UART0 from 0x500 to 0x510, except the offsets range from 0x600 to 0x610.

11.3.1 Register Descriptions

The following sections describe the UART*n* registers.

11.3.1.1 Receiver Buffer Registers (URBR*n*) (ULCR[DLAB] = 0)

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

Except for the case when there is an overrun, URBR returns the data in the order it was received from the transmitter. Refer to the ULSR[OE] description, Section 11.3.1.9, "Line Status Registers (ULSRn)." Figure 11-3 shows the receiver buffer registers. Note that these registers have same offset as the UTHRs.

Figure 11-2 shows the bits in the URBRs.

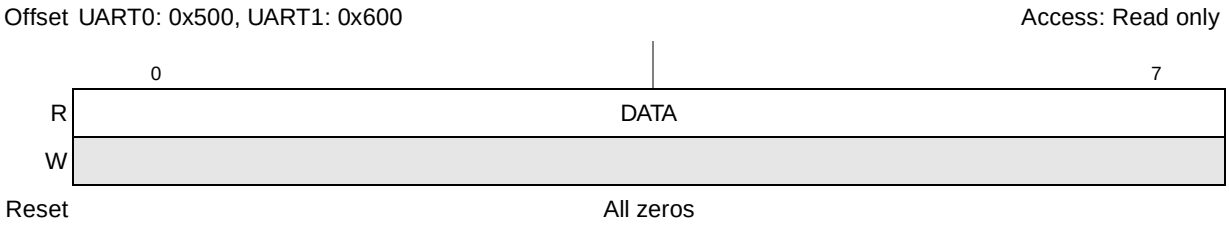


Figure 11-2. Receiver Buffer Registers (URBR*n*)

Table 11-3 describes the fields of URBR.

Table 11-3. URBR Field Descriptions

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

11.3.1.2 Transmitter Holding Registers (UTHR*n*) (ULCR[DLAB] = 0)

A write to these 8-bit registers causes the UART devices to transfer 5–8 data bits on the UART bus in the format set up in the ULCR (line control register). In FIFO mode, data written to UTHR is placed into the FIFO. The data written to UTHR is the data sent onto the UART bus, and the first byte written to UTHR

is the first byte onto the bus. UDSR[$\overline{\text{TXRDY}}$] indicates when the FIFO is full. Refer to the Table 11-20 and the Table 11-21 for more details.

Figure 11-3 shows the bits in the UTHRs.

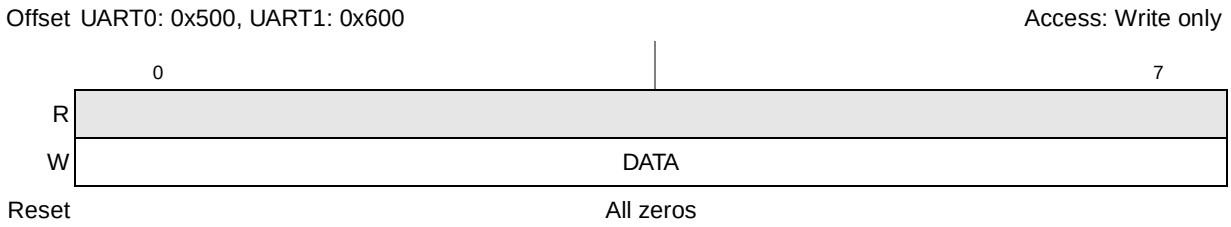


Figure 11-3. Transmitter Holding Registers (UTHR*n*)

Table 11-4 describes the fields of UTHR.

Table 11-4. UTHR Field Descriptions

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

11.3.1.3 Divisor Most and Least Significant Byte Registers (UDMB and UDLB) (ULCR[DLAB] = 1)

The divisor least significant byte register (UDLB) is concatenated with the divisor most significant byte register (UDMB) to create the divisor used to divide the input clock into the DUART. The output frequency of the baud generator is 16 times the baud rate; therefore the desired baud rate = platform clock frequency / (16 × [UDMB||UDLB]). Equivalently, [UDMB||UDLB:0b0000] = platform clock frequency / desired baud rate. Baud rates that can be generated by specific input clock frequencies are shown in Table 11-7.

Figure 11-4 shows the bits in the UDMBs.

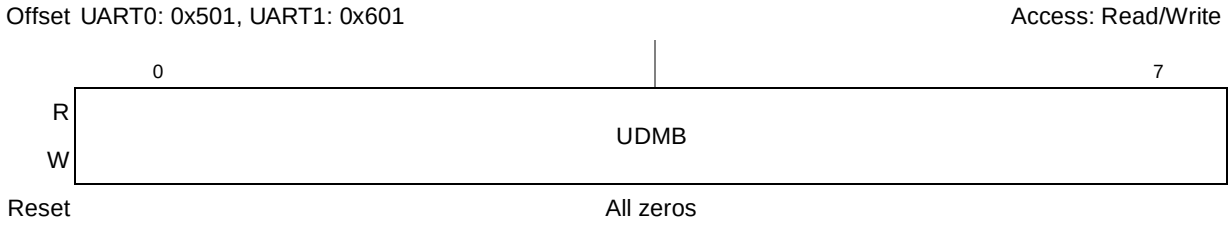


Figure 11-4. Divisor Most Significant Byte Registers (UDMB0, UDMB1)

Table 11-5 describes the fields of UDMB registers.

Table 11-5. UDMB Field Descriptions

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

Figure 11-5 shows the bits in the UDLBs.

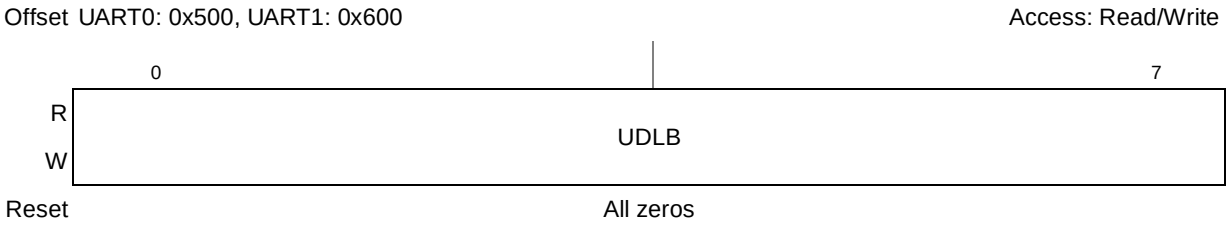


Figure 11-5. Divisor Least Significant Byte Registers (UDLB*n*)

Table 11-6 describes the fields of UDLB registers.

Table 11-6. UDLB Field Descriptions

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

Table 11-7 shows examples of baud rate generation based on common input clock frequencies. Many other target baud rates are also possible. Note that because only integer values can be used as divisors, the actual baud rate differs slightly from the desired (target) baud rate; for this reason, both target and actual baud rates are given, along with the percentage of error.

Table 11-7. Baud Rate Examples

Target Baud Rate (Decimal)	Divisor		Platform Clock (MPX) Frequency (MHz)	Actual Baud Rate (Decimal)	Percent Error (Decimal)
	Decimal	Hex			
9,600	2170	87A	333	9600.61444	0.0064
19,200	1085	43D	333	19,201.22888	0.0064
38,400	543	21F	333	38,367.09638	0.0858
57,600	362	16A	333	57,550.64457	0.0857
115,200	181	B5	333	115,101.28913	0.0857
230,400	90	5A	333	231,481.48148	0.4694
9,600	3906	F42	600	9600.6144	0.0064
19,200	1953	7A1	600	19,201.2288	0.0064
38,400	977	3D1	600	38382.8045	0.0448
57,600	651	28B	600	57603.6866	0.0064
115,200	326	146	600	115030.6748	0.1470
230,400	163	A3	600	230061.3497	0.1470

11.3.1.4 Interrupt Enable Register (UIER) (ULCR[DLAB] = 0)

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

Figure 11-6 shows the bits in the UIER.

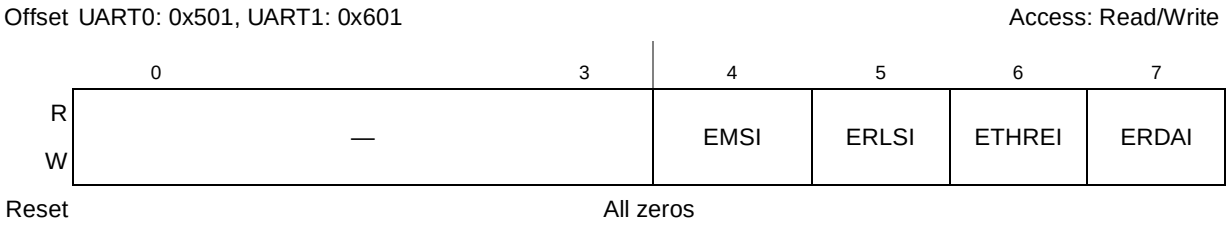


Figure 11-6. Interrupt Enable Register (UIER)

Table 11-8 describes the fields of UIER.

Table 11-8. UIER Field Descriptions

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

11.3.1.5 Interrupt ID Registers (UIIR*n*) (ULCR[DLAB] = 0)

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

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

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

4. Modem status

See Table 11-10 for more details.

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

Figure 11-7 shows the bits in the UIIR.

Figure 11-7. Interrupt ID Registers (UIIR)

Table 11-9 describes the fields of the UIIR.

Table 11-9. UIIR Field Descriptions

Bits	Name	Description
0–1	FE	FIFOs enabled. Reflects the setting of UFCR[FEN]
2–3	—	Reserved
4	IID3	Interrupt ID bits identify the highest priority interrupt that is pending as indicated in Table 11-10. IID3 is set along with IID2 only when a timeout interrupt is pending for FIFO mode.
5–6	IID2–1	Interrupt ID bits identify the highest priority interrupt that is pending as indicated in Table 11-10.
7	IID0	IID0 indicates when an interrupt is pending. 0 The UART has an active interrupt ready to be serviced. 1 No interrupt is pending.

The bits contained in the UIIR registers are described in Table 11-10.

Table 11-10. UIIR IID Bits Summary

IID Bits IID[3–0]	Priority Level	Interrupt Type	Interrupt Description	How To Reset Interrupt
0b0001	—	—	—	—
0b0110	Highest	Receiver line status	Overrun error, parity error, framing error, or break interrupt	Read the line status register.
0b0100	Second	Received data available	Receiver data available or trigger level reached in FIFO mode	Read the receiver buffer register or interrupt is automatically reset if the number of bytes in the receiver FIFO drops below the trigger level.

Table 11-10. UIIR IID Bits Summary (continued)

IID Bits IID[3–0]	Priority Level	Interrupt Type	Interrupt Description	How To Reset Interrupt
0b1100	Second	Character time-out	No characters have been removed from or input to the receiver FIFO during the last 4 character times and there is at least one character in the receiver FIFO during this time.	Read the receiver buffer register.
0b0010	Third	UTHR empty	Transmitter holding register is empty	Read the UIIR or write to the UTHR.
0b0000	Fourth	Modem status	$\overline{\text{CTS}}$ input value changed since last read of UMSR	Read the UMSR.

11.3.1.6 FIFO Control Registers (UFCR*n*) (ULCR[DLAB] = 0)

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

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

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

Figure 11-8 shows the bits in the UFCRs.

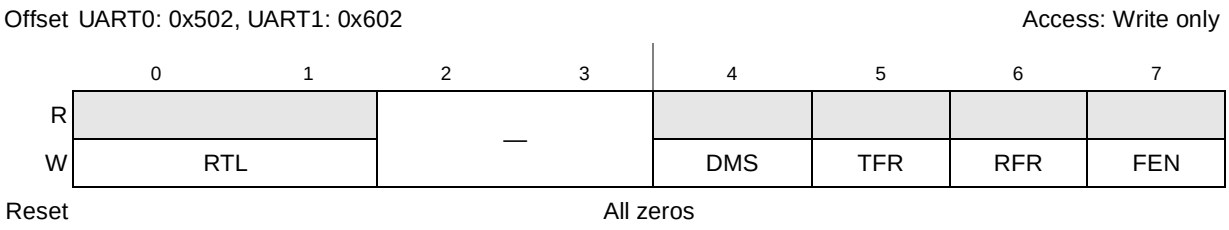


Figure 11-8. FIFO Control Registers (UFCR*n*)

Table 11-11 describes the fields of the UFCRs.

Table 11-11. UFCR Field Descriptions

Bits	Name	Description
0–1	RTL	Receiver trigger level. A received data available interrupt occurs when UIER[ERDAI] is set and the number of bytes in the receiver FIFO equals the designated interrupt trigger level as follows: 00 1 byte 01 4 bytes 10 8 bytes 11 14 bytes
2–3	—	Reserved

Table 11-11. UFCR Field Descriptions (continued)

Bits	Name	Description
4	DMS	DMA mode select. See Section 11.4.5.2, "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

11.3.1.7 Line Control Registers (ULCR*n*)

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

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

The stick parity bit, ULCR[SP], assigns a set parity value for the parity bit time slot sent on the UART bus. The set value is defined as mark parity (logic 1) or space parity (logic 0). ULCR[PEN] and ULCR[EPS] help determine the set parity value. See Table 11-13 for more information. ULCR[NSTB], defines the number of STOP bits to be sent at the end of the data transfer. The receiver only checks the first STOP bit, regardless of the number of STOP bits selected. The word length select bits (1 and 0) define the number of data bits that are transmitted or received as a serial character. The word length does not include START, parity, and STOP bits.

Figure 11-9 shows the bits in the ULCRs.

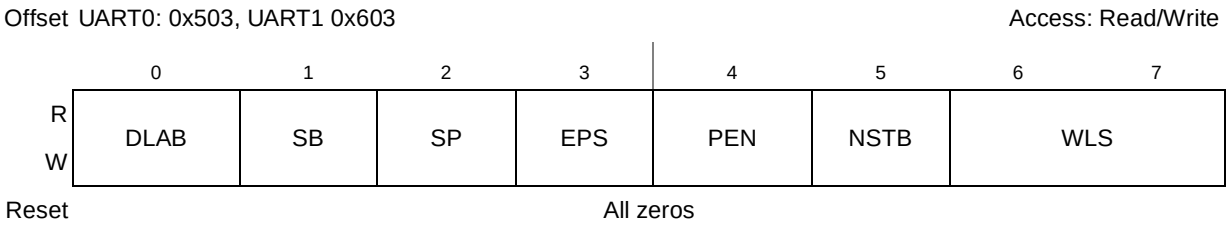


Figure 11-9. Line Control Register (ULCR)

Table 11-12 describes the fields of the ULCRs.

Table 11-12. ULCR Field Descriptions

Bits	Name	Description
0	DLAB	Divisor latch access bit. 0 Access to all registers except UDLB, UAFR, and UDMB 1 Ability to access divisor latch least and most significant byte registers and alternate function register (UAFR)
1	SB	Set break. 0 Send normal UTHR data onto the serial output (SOUT) signal 1 Force logic 0 to be on the SOUT signal. Data in the UTHR is not affected
2	SP	Stick parity. 0 Stick parity is disabled. 1 If PEN = 1 and EPS = 1, space parity is selected. And if PEN = 1 and EPS = 0, mark parity is selected.
3	EPS	Even parity select. See Table 11-13 for more information. 0 If PEN = 1 and SP = 0, odd parity is selected. 1 If PEN = 1 and SP = 0, even parity is selected.
4	PEN	Parity enable. 0 No parity generation and checking 1 Generate parity bit as a transmitter, and check parity as a receiver
5	NTSB	Number of STOP bits. 0 One STOP bit is generated in the transmitted data. 1 When a 5-bit data length is selected, 1∫ STOP bits are generated. When either a 6-, 7-, or 8-bit word length is selected, two STOP bits are generated.
6–7	WLS	Word length select. Number of bits that comprise the character length. The word length select values are as follows: 00 5 bits 01 6 bits 10 7 bits 11 8 bits

Table 11-13. Parity Selection Using ULCR[PEN], ULCR[SP], and ULCR[EPS]

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

11.3.1.8 Modem Control Registers (UMCR*n*)

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

Figure 11-10 shows the bits in the UMCRs

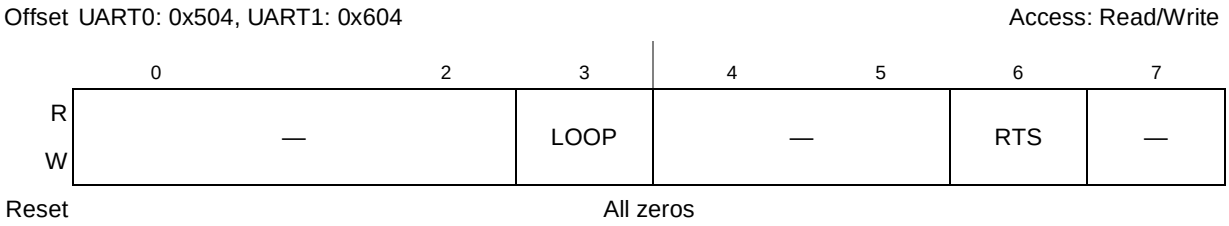


Figure 11-10. Modem Control Register (UMCR)

Table 11-14 describes the fields of UMCRs.

Table 11-14. UMCR Field Descriptions

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

11.3.1.9 Line Status Registers (ULSR*n*)

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

Figure 11-11 shows the bits in the ULSRs.

Offset UART0: 0x505, UART1: 0x605 Access: Read only

	0	1	2	3	4	5	6	7
R	RFE	TEMT	THRE	BI	FE	PE	OE	DR
W								
Reset	0	1	1	0	0	0	0	0

Figure 11-11. Line Status Register (ULSR)

Table 11-15 describes the fields of the ULSRs.

Table 11-15. ULSR Field Descriptions

Bits	Name	Description
0	RFE	Receiver FIFO error. 0 This bit is cleared when there are no errors in the receiver FIFO or on a read of the ULSR with no remaining receiver FIFO errors. 1 Set to one when one of the characters in the receiver FIFO encounters an error (framing, parity, or break interrupt)
1	TEMT	Transmitter empty. 0 Either or both the UTHR or the internal transmitter shift register has a data character. In FIFO mode, a data character is in the transmitter FIFO or the internal transmitter shift register. 1 Both the UTHR and the internal transmitter shift register are empty. In FIFO mode, both the transmitter FIFO and the internal transmitter shift register are empty.
2	THRE	Transmitter holding register empty. 0 The UTHR is not empty. 1 A data character has transferred from the UTHR into the internal transmitter shift register. In FIFO mode, the transmitter FIFO contains no data character.
3	BI	Break interrupt. 0 This bit is cleared when the ULSR is read or when a valid data transfer is detected (that is, STOP bit is received). 1 Received data of logic 0 for more than START bit + Data bits + Parity bit + one STOP bits length of time. A new character is not loaded until SIN returns to the mark state (logic 1) and a valid START is detected. In FIFO mode, a zero character is encountered in the FIFO (the zero character is at the top of the FIFO). In FIFO mode, only one zero character is stored.
4	FE	Framing error. 0 This bit is cleared when ULSR is read or when a new character is loaded into the URBR from the receiver shift register. 1 Invalid STOP bit for receive data (only the first STOP bit is checked). In FIFO mode, this bit is set when the character that detected a framing error is encountered in the FIFO (that is the character at the top of the FIFO). An attempt to resynchronize occurs after a framing error. The UART assumes that the framing error (due to a logic 0 being read when a logic 1 (STOP) was expected) was due to a STOP bit overlapping with the next START bit, so it assumes this logic 0 sample is a true START bit and then receives the following new data.

Table 11-15. ULSR Field Descriptions (continued)

Bits	Name	Description
5	PE	Parity error. 0 This bit is cleared when ULSR is read or when a new character is loaded into the URBR. 1 Unexpected parity value encountered when receiving data. In FIFO mode, the character with the error is at the top of the FIFO.
6	OE	Overrun error. 0 This bit is cleared when ULSR is read. 1 Before the URBR is read, the URBR was overwritten with a new character. The old character is loss. In FIFO mode, the receiver FIFO is full (regardless of the receiver FIFO trigger level setting) and a new character has been received into the internal receiver shift register. The old character was overwritten by the new character. Data in the receiver FIFO was not overwritten.
7	DR	Data ready. 0 This bit is cleared when URBR is read or when all of the data in the receiver FIFO is read. 1 A character has been received in the URBR or the receiver FIFO.

11.3.1.10 Modem Status Registers (UMSR*n*)

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

Figure 11-12 shows the bits in the UMSRs.

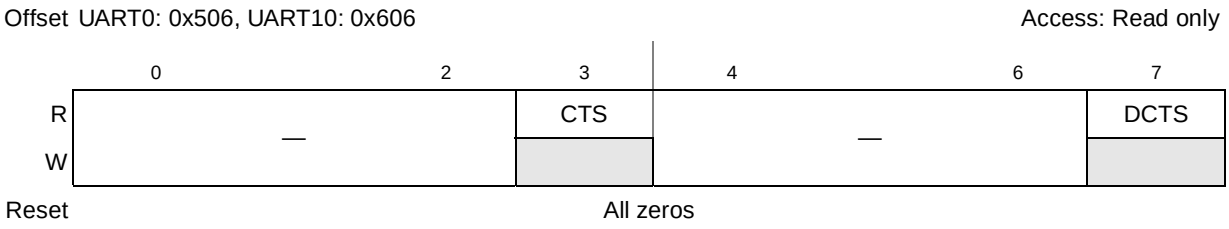


Figure 11-12. Modem Status Register (UMSR)

Table 11-16 describes the fields of the UMSRs.

Table 11-16. UMSR Field Descriptions

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

11.3.1.11 Scratch Registers (USCR*n*)

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

Figure 11-13 shows the bits in USCRs.

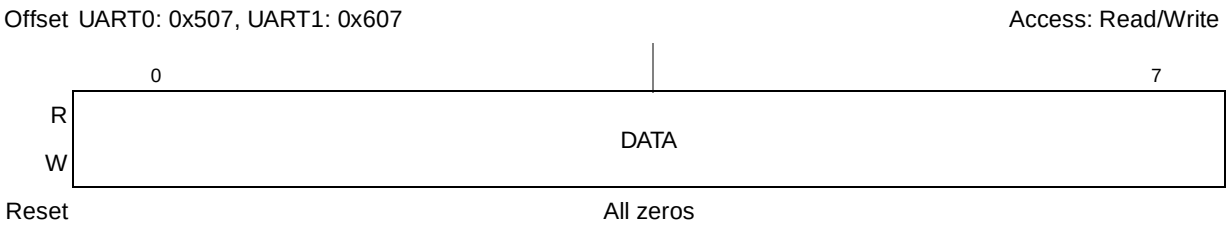


Figure 11-13. Scratch Register (USCR)

Table 11-17 describes the fields of the USCRs.

Table 11-17. USCR Field Descriptions

Bits	Name	Description
0–7	DATA	Data

11.3.1.12 Alternate Function Registers (UAFR*n*) (ULCR[DLAB] = 1)

The UAFRs give software the ability to gate off the baud clock and write to both UART0/UART1 registers simultaneously with the same write operation.

Figure 11-14 shows the bits in the UAFRs.

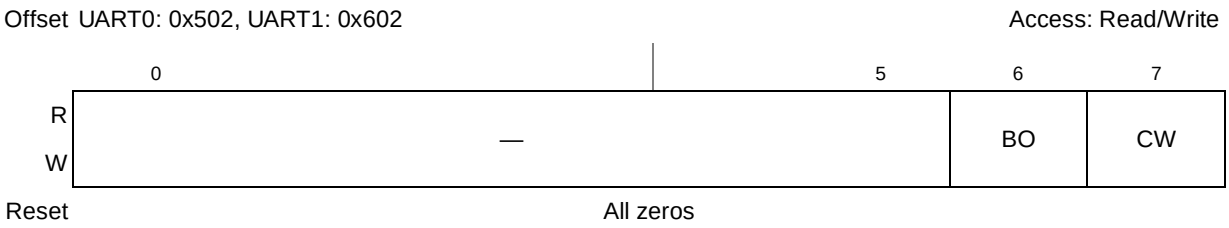


Figure 11-14. Alternate Function Register (UAFR)

Table 11-18 describes the fields of the UAFRs.

Table 11-18. UAFR Field Descriptions

Bits	Name	Description
0–5	—	Reserved.
6	BO	Baud clock select. 0 The baud clock is not gated off. 1 The baud clock is gated off.
7	CW	Concurrent write enable. 0 Disables writing to both UART0 and UART1 1 Enables concurrent writes to corresponding UART registers. A write to a register in UART0 is also a write to the corresponding register in UART1 and vice versa. The user needs to ensure that the LCR[DLAB] of both UARTs are in the same state before executing a concurrent write to register addresses 0x*n*00, 0x*n*01 and 0x*n*02, where *n* is the offset of the corresponding UART.

11.3.1.13 DMA Status Registers (UDSR*n*)

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

Figure 11-15 shows the bits in UDSRs.

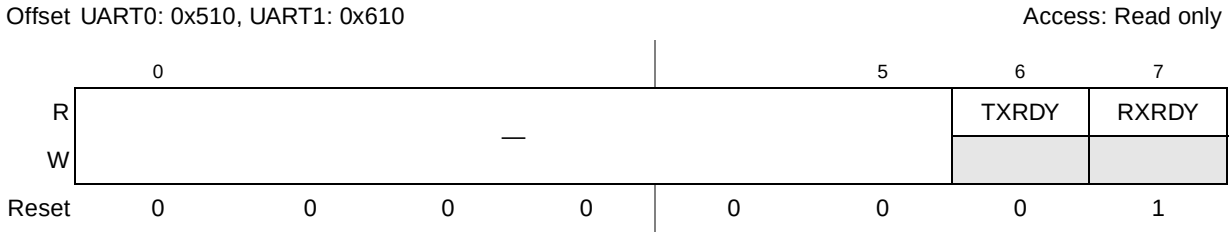


Figure 11-15. DMA Status Register (UDSR)

Table 11-19 describes the fields of the UDSRs.

Table 11-19. UDSR Field Descriptions

Bits	Name	Description
0–5	—	Reserved
6	TXRDY	Transmitter ready. This read-only bit reflects the status of the transmitter FIFO or the UTHR. The status depends on the DMA mode selected, which is determined by the DMS and FEN bits in the UFCR. 0 The bit is cleared, as shown in Table 11-21. 1 This bit is set, as shown in Table 11-20.
7	RXRDY	Receiver ready. This read-only bit reflects the status of the receiver FIFO or URBR. The status depends on the DMA mode selected, which is determined by the DMS and FEN bits in the UFCR. 0 The bit is cleared, as shown in Table 11-23. 1 This bit is set, as shown in Table 11-22.

Table 11-20. UDSR[TXRDY] Set Conditions

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

Table 11-21. UDSR[TXRDY] Cleared Conditions

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

Table 11-22. UDSR[RXRDY] Set Conditions

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

Table 11-23. UDSR[RXRDY] Cleared Conditions

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

11.4 Functional Description

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

The transmitter accepts parallel data with a write access to the transmitter holding register (UTHR). In FIFO mode, the data is placed directly into an internal transmitter shift register, or into the transmitter FIFO—see Section 11.4.5, “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.

11.4.1 Serial Interface

The UART bus is a serial, full-duplex, point-to-point bus as shown in Figure 11-16. Therefore, only two devices are attached to the same signals and there is no need for address or arbitration bus cycles.

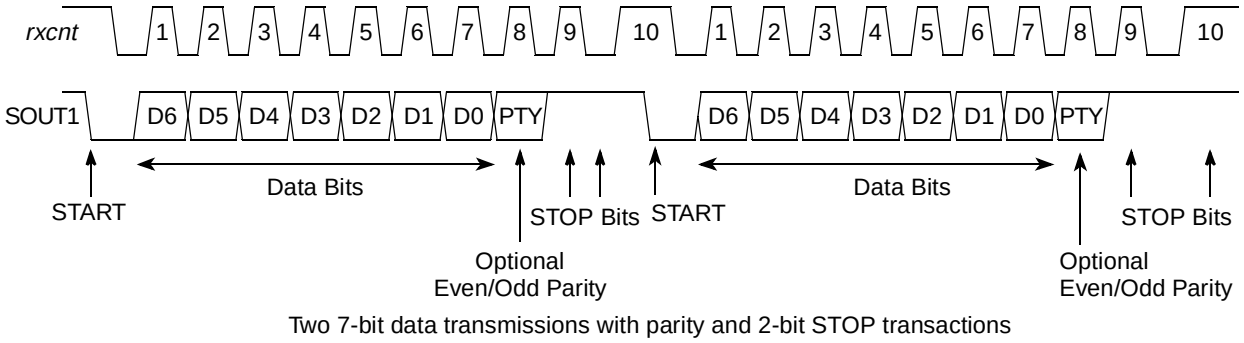


Figure 11-16. UART Bus Interface Transaction Protocol Example

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

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

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

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

11.4.1.1 START Bit

A write to the transmitter holding register (UTHR) generates a START bit on the SOUT signal. Figure 11-16 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.

11.4.1.2 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.

11.4.1.3 Parity Bit

The user has the option of using even, odd, no parity, or stick parity (see Section 11.3.1.7, “Line Control Registers (ULCRn).” Both the receiver and transmitter parity definition must agree before attempting to transfer data. When receiving data a parity error can occur if an unexpected parity value is detected. (See Section 11.3.1.9, “Line Status Registers (ULSRn).”)

11.4.1.4 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.

11.4.2 Baud-Rate Generator Logic

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

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

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

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

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

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

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

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

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

11.4.3 Local Loopback Mode

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

11.4.4 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.

11.4.4.1 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.

11.4.4.2 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.

11.4.4.3 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.

11.4.5 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.

11.4.5.1 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 11.3.1.4, “Interrupt Enable Register (UIER) (ULCR[DLAB] = 0),” for more details on interrupt enables.

The interrupt ID register (UIIR) indicates if the FIFOs are enabled. Interrupt ID3 UIIR[IID3] bit is only set for FIFO mode interrupts. The character time-out interrupt occurs when no characters have been removed from or input to the receiver FIFO during the last four character times and there is at least one character in the receiver FIFO during this time. The character time-out interrupt (controlled by UIIR[IID]) is cleared when the URBR is read. See Section 11.3.1.5, “Interrupt ID Registers (UIIRn) (ULCR[DLAB] = 0),” for more information.

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

11.4.5.2 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 11.3.1.13, “DMA Status Registers (UDSRn),” for a complete description of the USDR[RXRDY] and USDR[TXRDY] bits.

11.4.5.3 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 11.3.1.4, “Interrupt Enable Register (UIER) (ULCR[DLAB] = 0).”

When the interrupts are disabled in UIER, polling software cannot 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.

11.5 DUART Initialization/Application Information

The following requirements must be met for DUART accesses:

- All DUART registers must be mapped to a cache-inhibited and guarded area. (That is, the WIMG setting in the MMU needs to be 0b01X1.)
- All DUART registers are 1 byte wide. Reads and writes to these registers must be byte-wide operations.

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

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

Chapter 12
Local Bus Controller

This chapter describes the local bus controller (LBC) block. It describes the external signals and the memory-mapped registers as well as a functional description of the general-purpose chip-select machine (GPCM), SDRAM machine, and user-programmable machines (UPMs) of the LBC. Finally, it includes an initialization and applications information section with many specific examples of its use.

12.1 Introduction

Figure 12-1 is a functional block diagram of the LBC, which supports three interfaces: GPCM, UPM, and SDRAM controller.

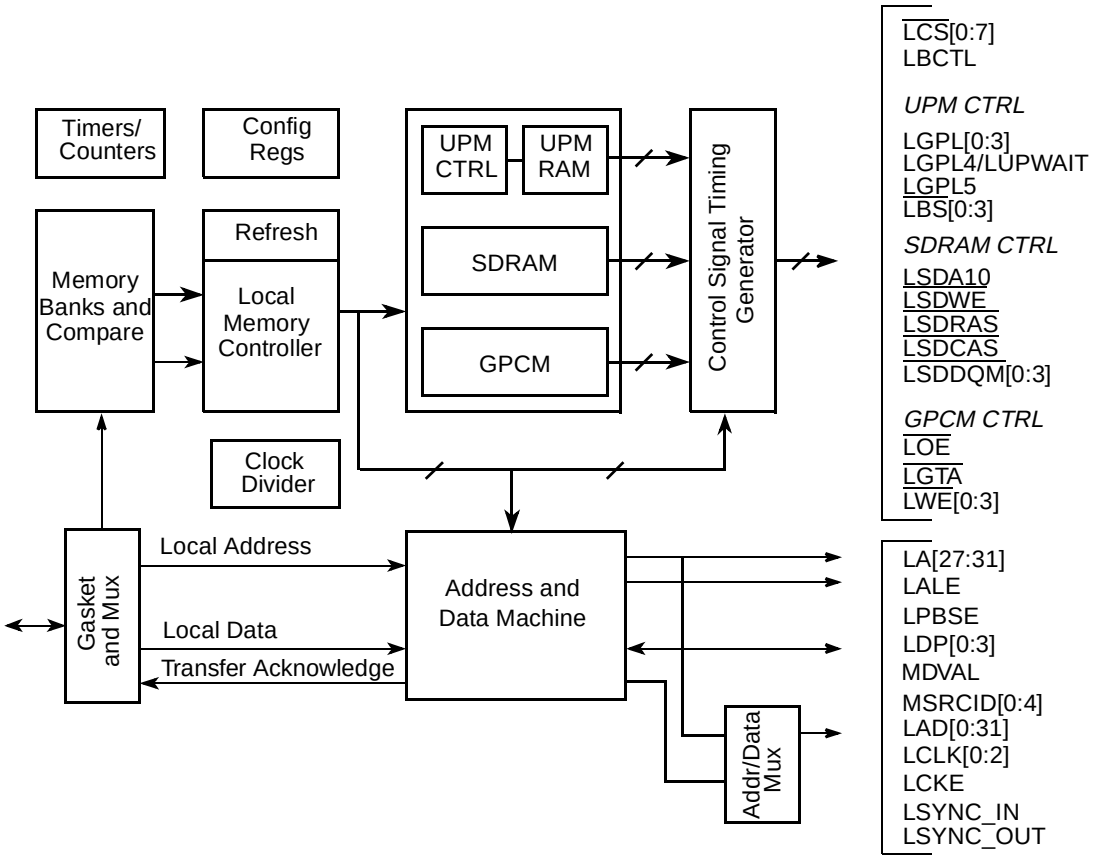


Figure 12-1. Local Bus Controller Block Diagram

12.1.1 Overview

The main component of the LBC is its memory controller, which provides a seamless interface to many types of memory devices and peripherals. The memory controller is responsible for controlling eight memory banks shared by a high performance SDRAM machine, a GPCM, and up to three UPMs. As such, it supports a minimal glue logic interface to synchronous DRAM (SDRAM), SRAM, EPROM, Flash EPROM, burstable RAM, regular DRAM devices, extended data output DRAM devices, and other peripherals. The external address latch signal (LALE) allows multiplexing of addresses with data signals to reduce the device signal count.

The LBC also includes a number of data checking and protection features such as data parity generation and checking, write protection and a bus monitor to ensure that each bus cycle is terminated within a user-specified period.

12.1.2 Features

The LBC main features are as follows:

- Memory controller with eight memory banks
  - 34-bit[1] 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

1. Refers to the logical address space of the LBC. Once the address is decoded by the LBC, the 34-bit address becomes the right-most 34 bits of the 36-bit physical address space.

  - 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 phase-locked loop (PLL) with software-configurable bypass for low frequency bus clocks

12.1.3 Modes of Operation

The LBC provides one GPCM, one SDRAM machine, and three UPMs for the local bus, with no restriction on how many of the eight banks (chip selects) can be programmed to operate with any given machine. When a memory transaction is dispatched to the LBC, the memory address is compared with the address information of each bank (chip select). The corresponding machine assigned to that bank (GPCM, SDRAM, or UPM) then takes ownership of the external signals that control the access and maintains control until the transaction ends. Thus, with the LBC in GPCM, SDRAM, or UPM mode, only one of the eight chip selects is active at any time for the duration of the transaction.

12.1.3.1 LBC Bus Clock and Clock Ratios

The LBC supports ratios of 4, 8, and 16 between the faster system (MPX) 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 signals LCLK[0:2] to allow the clock load to be shared equally across a pair of signal nets, thereby enhancing the edge rates of the bus clock.

12.1.3.2 Source ID Debug Mode

In debug mode, the LBC provides the ID of a transaction source on external device signals. This mode is enabled on power-on reset, as described in Section 19.4.3, "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 signals. The reserved value of 0x1F, which indicates invalid address or data, appears on the source ID signals at all other times. The combination of a valid source ID (any value except 0x1F) and the value of external address latch enable (LALE) and data valid (MDVAL) facilitate capturing useful debug data as follows:

- If a valid source ID is detected on MSRCID[0:4] and LALE is asserted, a valid full 32-bit address may be latched from LAD[0:31]. Note that in SDRAM mode the address vector contains the full address as {row, bank, column, lsbs} where row corresponds to the same row address for the given column address and lsbs are the unconnected lsbs of the address for a given port size.
- If a valid source ID is detected on MSRCID[0:4] and MDVAL is asserted, valid data may be latched from LAD[0:31].

12.1.4 Power-Down Mode

The LBC can enter a power-down mode when the system stops the internal (system) clock to the block 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 PLL sufficient time to recover following the reapplication of the system clock.

Once the LBC has been put into power-down mode, the only way to exit from this mode is through $\overline{\text{HRESET}}$.

12.2 External Signal Descriptions

Table 12-1 contains a list of external signals related to the LBC and summarizes their function. I/O impedance of designated local bus signals is determined by PORIMPSCR, as described in Section 17.4.1.3, "POR I/O Impedance Control Register (PORIMPCR)."

Table 12-1. Signal Properties—Summary

Name	Number of Signals	Direction	Function
LALE	1	Output	External address latch enable
$\overline{\text{LCS}}$	8	Output	Chip selects
$\overline{\text{LWE}}n$/ $\overline{\text{LSDDQM}}n$/ $\overline{\text{LBS}}n$	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

Table 12-1. Signal Properties—Summary (continued)

Name	Number of Signals	Direction	Function
$\overline{\text{LOE}}$/ $\overline{\text{LSDRAS}}$/ 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 lsbs
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	PLL synchronize input
LSYNC_OUT	1	Output	PLL synchronize output
MDVAL	1	Output	In LBC debug mode: local bus data valid
MSRCID	5	Output	In LBC debug mode: local bus source ID

Table 12-2 contains detailed external signal descriptions for the LBC.

Table 12-2. Local Bus Controller Detailed Signal Descriptions

Signal	I/O	Description	
LALE	O	External address latch enable. The local bus memory controller provides control for an external address latch, which allows address and data to be multiplexed on the device signals.	
		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.

Table 12-2. Local Bus Controller Detailed Signal Descriptions (continued)

Signal	I/O	Description	
$\overline{\text{LWE}}$[0:3]/ $\overline{\text{LSDDQM}}$[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, $\overline{\text{LSDDQM}}$[0:3] function as the DQM or data mask signals provided by JEDEC-compliant SDRAM devices, with one DQM provided per byte lane. $\overline{\text{LSDDQM}}$[0:3] are driven high when the LBC wishes to mask a write or disable read data output from the SDRAM. $\overline{\text{LBS}}$[0:3] are programmable byte-select signals in UPM mode. See Section 12.4.4.4, "RAM Array," for programming details about $\overline{\text{LBS}}$[0:3].
		Timing	Assertion/Negation—See Section 12.4.2, "General-Purpose Chip-Select Machine (GPCM)," for details regarding the timing of $\overline{\text{LWE}}$[0:3].
LSDA10/ LGPL0	O	SDRAM A10/General-purpose line 0	
		State Meaning	Asserted/Negated—For SDRAM accesses, represents address bit 10. When the row address is driven, it drives the value of address bit 10. When the column address is driven, it forms part of the SDRAM command. One of six general-purpose signals when in UPM mode; it drives a value programmed in the UPM array.
$\overline{\text{LSDWE}}$/ LGPL1	O	SDRAM write enable/General-purpose line 1	
		State Meaning	Asserted/Negated—Should be connected to the SDRAM device WE input. Acts as the SDRAM write enable when accessing SDRAM. One of six general-purpose signals when in UPM mode, and drives a value programmed in the UPM array.
$\overline{\text{LOE}}$/ $\overline{\text{LSDRAS}}$/ LGPL2	O	GPCM output enable/SDRAM $\overline{\text{RAS}}$/General-purpose line 2	
		State Meaning	Asserted/Negated—Controls the output buffer of memory when accessing memory/devices in GPCM mode. For SDRAM accesses, it is the row address strobe ($\overline{\text{RAS}}$). One of six general-purpose lines when in UPM mode; it drives a value programmed in the UPM array.
$\overline{\text{LSDCAS}}$/ LGPL3	O	SDRAM $\overline{\text{CAS}}$/General-purpose line 3	
		State Meaning	Asserted/Negated—In SDRAM mode, drives the column address strobe ($\overline{\text{CAS}}$). One of six general-purpose signals when in UPM mode, and drives a value programmed in the UPM array.
$\overline{\text{LGTA}}$/ 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 $\overline{\text{LBS}}$[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 signal that is an internal AND of the four (active low) byte selects, allowing glueless, faster connection to RMW-parity devices.

Table 12-2. Local Bus Controller Detailed Signal Descriptions (continued)

Signal	I/O	Description	
LGPL5	O	General-purpose line 5	
		State Meaning	Asserted/Negated—One of six general-purpose signals when in UPM mode, and drives a value programmed in the UPM array.
LBCTL	O	Data buffer control. The memory controller activates LBCTL for the local bus when a GPCM- 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 signal normally functions as a write/$\overline{\text{read}}$ control for a bus transceiver connected to the LAD lines. Note that an external data buffer must not drive the LAD lines in conflict with the LBC when LBCTL is high, because LBCTL remains high after reset and during address phases.
LA[27:31]	O	Local bus non-multiplexed 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 signals, LA[27:31]. These may be used, unlatched, in place of LAD[27:31] to connect the five lsbs of the address for address phases. For some RAM devices, such as fast-page DRAM, LA[27:31] serve as the column address offset during a burst access.
LAD[0:31]	I/O	Multiplexed address/data bus. For configuration of a port size in BR*n*[PS] as 32 bits, all of LAD[0:31] must be connected to the external RAM data bus, with LAD[0:7] occupying the most significant byte lane (at address offset 0). For a port size of 16 bits, LAD[0:7] connect to the most significant byte lane (at address offset 0), while LAD[8:15] connect to the least-significant byte lane (at address offset 1); LAD[16:31] are unused for 16-bit port sizes. For a port size of 8 bits, only LAD[0:7] are connected to the external RAM.	
		State Meaning	Asserted/Negated—LAD[0:31] is the shared 32-bit address/data bus through which external RAM devices transfer data and receive addresses.
		Timing	Assertion/Negation—During assertion of LALE, LAD[0:31] are driven with the RAM address for the access to follow. External logic should propagate the address on LAD[0:31] while LALE is asserted, and latch the address upon negation of LALE. After LALE is negated, LAD[0:31] are either driven by write data or are made high impedance by the LBC in order to sample read data driven by an external device. Following the last data transfer of a write access, LAD[0:31] are again taken into a high-impedance state.
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.

Table 12-2. Local Bus Controller Detailed Signal Descriptions (continued)

Signal	I/O	Description	
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 PLL is enabled (see LCRR[PBYP], Figure 12-19 on page 12-30), the bus clock phase is shifted earlier than transitions on other LBC signals (such as LAD[0:31] and $\overline{\text{LCS}n}$) by a time delay matching the delay of the PLL timing loop set up between LSYNC_OUT and LSYNC_IN.
LSYNC_OUT	O	PLL synchronization out	
		State Meaning	Asserted/Negated—A replica of the bus clock, appearing on LSYNC_OUT, should be propagated through a passive timing loop and returned to LSYNC_IN for achieving correct PLL lock.
		Timing	Assertion/Negation—The time delay of the timing loop should be such that it compensates for the round-trip flight time of LCLK[2] and clocked drivers in the system. No load other than a timing loop should be placed on LSYNC_OUT.
LSYNC_IN	I	PLL synchronization in	
		State Meaning	Asserted/Negated—See description of LSYNC_OUT.
MDVAL	O	Local bus data valid (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] signals are driven high unless MSRCID[0:4] is driving a debug source ID for identifying the internal system device controlling the LBC.	
		State Meaning	Asserted/Negated—Remain high until the last bus cycle of the assertion of LALE, in which case the source ID of the address is indicated, or until MDVAL is asserted, in which case the source ID relating to the data transfer is indicated. In case of address debug, MSRCID[0:4] is valid only when the address on LAD[0:31] consists of all physical address bits—with optional padding—for reconstructing the system address presented to the LBC. For example, MSRCID[0:4] is valid only during $\overline{\text{CAS}}$ phases of SDRAM accesses, because the column, bank select, and (normally unused) row address bits are all present on LAD[0:31] during a $\overline{\text{CAS}}$ cycle

12.3 Memory Map/Register Definition

Table 12-3 shows the memory mapped registers of the LBC and their offsets. It lists the offset, name, and a cross-reference to the complete description of each register. Note that the full register address is comprised of CCSRBAR together with the block base address and offset listed in Table 12-3. Undefined 4-byte address spaces within offset 0x000–0xFFF are reserved.

Table 12-3. Local Bus Controller Memory Map

Local Bus Controller—Block Base Address 0x0_5000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0x000	BR0—Base register 0	R/W	0x0000_*nn*01[1]	12.3.1.1/12-10
0x008	BR1—Base register 1		0x0000_0000	
0x010	BR2—Base register 2			
0x018	BR3—Base register 3			
0x020	BR4—Base register 4			
0x028	BR5—Base register 5			
0x030	BR6—Base register 6			
0x038	BR7—Base register 7			
0x004	OR0—Options register 0	R/W	0x0000_0FF7	12.3.1.2/12-12
0x00C	OR1—Options register 1		0x0000_0000	
0x014	OR2—Options register 2			
0x01C	OR3—Options register 3			
0x024	OR4—Options register 4			
0x02C	OR5—Options register 5			
0x034	OR6—Options register 6			
0x03C	OR7—Options register 7			
0x068	MAR—UPM address register	R/W	0x0000_0000	12.3.1.3/12-17
0x070	MAMR—UPMA mode register	R/W	0x0000_0000	12.3.1.4/12-17
0x074	MBMR—UPMB mode register	R/W	0x0000_0000	12.3.1.4/12-17
0x078	MCMR—UPMC mode register	R/W	0x0000_0000	12.3.1.4/12-17
0x084	MRTPR—Memory refresh timer prescaler register	R/W	0x0000_0000	12.3.1.5/12-20
0x088	MDR—UPM data register	R/W	0x0000_0000	12.3.1.6/12-20
0x094	LSDMR—SDRAM mode register	R/W	0x0000_0000	12.3.1.7/12-21
0x0A0	LURT—UPM refresh timer	R/W	0x0000_0000	12.3.1.8/12-23
0x0A4	LSRT—SDRAM refresh timer	R/W	0x0000_0000	12.3.1.9/12-23
0x0B0	LTESR—Transfer error status register	w1c	0x0000_0000	12.3.1.10/12-24
0x0B4	LTEDR—Transfer error disable register	R/W	0x0000_0000	12.3.1.11/12-25
0x0B8	LTEIR—Transfer error interrupt register	R/W	0x0000_0000	12.3.1.12/12-26
0x0BC	LTEATR—Transfer error attributes register	R/W	0x0000_0000	12.3.1.13/12-27
0x0C0	LTEAR—Transfer error address register	R/W	0x0000_0000	12.3.1.14/12-28

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

Local Bus Controller—Block Base Address 0x0_5000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0x0D0	LBCR—Configuration register	R/W	0x0000_0000	12.3.1.15/12-29
0x0D4	LCRR—Clock ratio register	R/W	0x8000_0008	12.3.1.16/12-30

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

12.3.1 Register Descriptions

This section provides a detailed description of the LBC configuration, status, and control registers with detailed bit and field descriptions.

Address offsets in the LBC address range that are not defined in Table 12-3 should not be accessed for reading or writing. Similarly, only zero should be written to reserved bits of defined registers, as writing ones can have unpredictable results in some cases.

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

12.3.1.1 Base Registers (BR0–BR7)

The base registers (BR*n*), shown in Figure 12-2, contain the base address and address types for each memory bank. The memory controller uses this information to compare the address bus value with the current address accessed. Each register (bank) includes a memory attribute and selects the machine for memory operation handling. Note that after system reset, BR0[V] is set, BR1[V]–BR7[V] are cleared, and the value of BR0[PS] reflects the initial port size configured by the boot ROM location signals.

Offset 0x000 (BR0), 0x008 (BR1), 0x010 (BR2), 0x018 (BR3), 0x020 (BR4), 0x028 (BR5), 0x030 (BR6), 0x038 (BR7) — Access: Read/Write

	0																16	17	18	19	20	21	22	23	24		26	27	28	29	30	31
R W	BA																XBA			PS	DECC			WP	MSEL			—	ATOM		—	V
Reset BR0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	*n*	*n*	0	0	0	0	0	0	0	0	0	0	1[1]
Reset BR1–BR7	All zeros																														

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 signals during reset. All other base registers have all bits cleared to zero during reset.

Figure 12-2. Base Registers (BR*n*)

Table 12-4 describes BR*n* fields.

Table 12-4. BR*n* Field Descriptions

Bits	Name	Description
0–16	BA	Base address. The upper 17 bits of each base register are compared to the address on the address bus to determine if the bus master is accessing a memory bank controlled by the memory controller. Used with the address mask bits OR*n*[AM].
17–18	XBA	Extended base address. Extends BA by a further 2 bits such that 19 bits of each base register are compared to the 34-bit transaction address. XBA provides the extra 2 msb's (that is, {XBA,BA} represents the full base address). Used with the extended address mask bits OR*n*[XAM].
19–20	PS	Port size. Specifies the port size of this memory region. For BR0, PS is configured from the boot ROM location signals during reset. For all other banks the value is reset to 00 (port size not defined). 00 Reserved 01 8-bit 10 16-bit 11 32-bit
21–22	DECC	Specifies the method for data error checking. 00 Data error checking disabled, but normal parity generation 01 Normal parity generation and checking 10 Read-modify-write parity generation and normal parity checking (32-bit port size only) 11 Reserved
23	WP	Write protect 0 Read and write accesses are allowed. 1 Only read accesses are allowed. The memory controller does not assert $\overline{\text{LCS}n}$ on write cycles to this memory bank. LTESR[WP] is set (if WP is set) if a write to this memory bank is attempted, and a local bus error interrupt is generated (if enabled), terminating the cycle.
24–26	MSEL	Machine select. Specifies the machine to use for handling memory operations. 000 GPCM (reset value) 001 Reserved 010 Reserved 011 SDRAM 100 UPMA 101 UPMB 110 UPMC 111 Reserved
27	—	Reserved
28–29	ATOM	Atomic operation. Writes (reads) to the address space handled by the memory controller bank reserve the selected memory bank for the exclusive use of the accessing device. The reservation is released when the device performs a read (write) operation to this memory controller bank. If a subsequent read (write) request to this memory controller bank is not detected within 256 bus clock cycles of the last write (read), the reservation is released and an atomic error is reported (if enabled). 00 The address space controlled by this bank is not used for atomic operations. 01 Read-after-write-atomic (RAWA) 10 Write-after-read-atomic (WARA) 11 Reserved
30	—	Reserved
31	V	Valid bit. Indicates that the contents of the BR*n* and OR*n* pair are valid. $\overline{\text{LCS}n}$ does not assert unless V is set (an access to a region that has no valid bit set may cause a bus time-out). After a system reset, only BR0[V] is set. 0 This bank is invalid. 1 This bank is valid.

12.3.1.2 Option Registers (OR0–OR7)

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

- GPCM mode
- UPM mode
- SDRAM mode

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

12.3.1.2.1 Address Mask

The address mask fields of the option registers (OR*n*[XAM,AM]) mask up to 19 corresponding BR*n*[BA,XBA] fields. The 15 lsbs of the 34-bit internal address do not participate in bank address matching in selecting a bank for access. Masking address bits independently allows external devices of different size address ranges to be used. Address mask bits can be set or cleared in any order in the field, allowing a resource to reside in more than one area of the address map. Table 12-5 shows memory bank sizes from 32 Kbytes to 4 Gbytes. Note that the use of the OR*n*[XAM] fields provides 34-bits of internal addressing which allows aliasing across 32-bit external banks.

Table 12-5. Memory Bank Sizes in Relation to Address Mask

Address Mask	Memory Bank Size	Address Mask	Memory Bank Size
0000_0000_0000_0000_0	4 Gbytes	1111_1111_1000_0000_0	8 Mbytes
1000_0000_0000_0000_0	2 Gbytes	1111_1111_1100_0000_0	4 Mbytes
1100_0000_0000_0000_0	1 Gbyte	1111_1111_1110_0000_0	2 Mbytes
1110_0000_0000_0000_0	512 Mbytes	1111_1111_1111_0000_0	1 Mbyte
1111_0000_0000_0000_0	256 Mbytes	1111_1111_1111_1000_0	512 Kbytes
1111_1000_0000_0000_0	128 Mbytes	1111_1111_1111_1100_0	256 Kbytes
1111_1100_0000_0000_0	64 Mbytes	1111_1111_1111_1110_0	128 Kbytes
1111_1110_0000_0000_0	32 Mbytes	1111_1111_1111_1111_0	64 Kbytes
1111_1111_0000_0000_0	16 Mbytes	1111_1111_1111_1111_1	32 Kbytes

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

Figure 12-3 shows the bit fields for OR*n* when the corresponding BR*n*[MSEL] selects the GPCM machine.

Offset 0x004 (OR0) 0x024 (OR4) Access: Read/Write
0x00C (OR1) 0x02C (OR5)
0x014 (OR2) 0x034 (OR6)
0x01C (OR3) 0x03C (OR7)

	0				16	17	18	19	20	21	22	23	24	27	28	29	30	31
R W	AM					XAM		BCTLD	CSNT	ACS		XACS	SCY		SETA	TRLX	EHTR	EAD
Reset OR0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1[1]
Reset OR1–7	All zeros																	

Figure 12-3. Option Registers (OR*n*) in GPCM Mode

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.

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

Table 12-6. OR*n*—GPCM Field Descriptions

Bits	Name	Description
0–16	AM	GPCM address mask. Masks corresponding BR*n* bits. Masking address bits independently allows external devices of different size address ranges to be used. Address mask bits can be set or cleared in any order in the field, allowing a resource to reside in more than one area of the address map. 0 Corresponding address bits are masked. 1 Corresponding address bits are used in the comparison between base and transaction addresses.
17–18	XAM	Extended address mask. Masks the corresponding XBA bits in the BR*n* register, effectively extending the address mask (AM) by 2 bits.
19	BCTLD	Buffer control disable. Disables assertion of LBCTL during access to the current memory bank. 0 LBCTL is asserted upon access to the current memory bank. 1 LBCTL is not asserted upon access to the current memory bank.
20	CSNT	Chip select negation time. Determines when $\overline{\text{LCS}n}$ and $\overline{\text{LWE}}$ are negated during an external memory write access handled by the GPCM, provided that ACS ≠ 00 (when ACS = 00, only $\overline{\text{LWE}}$ is affected by the setting of CSNT). This helps meet address/data hold times for slow memories and peripherals. 0 $\overline{\text{LCS}n}$ and $\overline{\text{LWE}}$ are negated normally. 1 $\overline{\text{LCS}n}$ and $\overline{\text{LWE}}$ are negated one quarter of a bus clock cycle earlier.
21–22	ACS	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.

Value	Meaning
00	$\overline{\text{LCS}n}$ is output at the same time as the address lines. Note that this overrides the value of CSNT such that CSNT=0.
01	Reserved.
10	$\overline{\text{LCS}n}$ is output a quarter bus clock cycle after the address lines.
11	$\overline{\text{LCS}n}$ is output a half bus clock cycle after the address lines.

Table 12-6. OR*n*—GPCM Field Descriptions (continued)

Bits	Name	Description
23	XACS	Extra address to chip-select setup. Setting this bit increases the delay of the $\overline{\text{LCS}n}$ assertion relative to the address change when the external memory access is handled by the GPCM. After a system reset, OR0[XACS] = 1. 0 Address to chip-select setup is determined by ORx[ACS]. 1 Address to chip-select setup is extended (see Table 12-23 and Table 12-24).
24–27	SCY	Cycle length in bus clocks. Determines the number of wait states inserted in the bus cycle, when the GPCM handles the external memory access. Thus it is the main parameter for determining cycle length. The total cycle length depends on other timing attribute settings. After a system reset, OR0[SCY] = 1111. 0000 No wait states 0001 1 bus clock cycle wait state ... 1111 15 bus clock cycle wait states
28	SETA	External address termination 0 Access is terminated internally by the memory controller unless the external device asserts $\overline{\text{LGTA}}$ earlier to terminate the access. 1 Access is terminated externally by asserting the $\overline{\text{LGTA}}$ external signal. (Only $\overline{\text{LGTA}}$ can terminate the access).
29	TRLX	Timing relaxed. Modifies the settings of timing parameters for slow memories or peripherals. 0 Normal timing is generated by the GPCM. 1 Relaxed timing on the following parameters: • Adds an additional cycle between the address and control signals (only if ACS ≠ 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.
30	EHTR	Extended hold time on read accesses. Indicates with TRLX how many cycles are inserted between a read access from the current bank and the next access.
31	EAD	External address latch delay. Allow extra bus clock cycles when using external address latch (LALE). 0 No additional bus clock cycles (LALE asserted for one bus clock cycle only) 1 Extra bus clock cycles are added (LALE is asserted for the number of bus clock cycles specified by LCRR[EADC]).

TRLX	EHTR	Meaning
0	0	The memory controller generates normal timing. No additional cycles are inserted.
0	1	1 idle clock cycle is inserted.
1	0	4 idle clock cycles are inserted.
1	1	8 idle clock cycles are inserted.

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

Figure 12-4 shows the bit fields for OR*n* when the corresponding BR*n*[MSEL] selects a UPM machine.

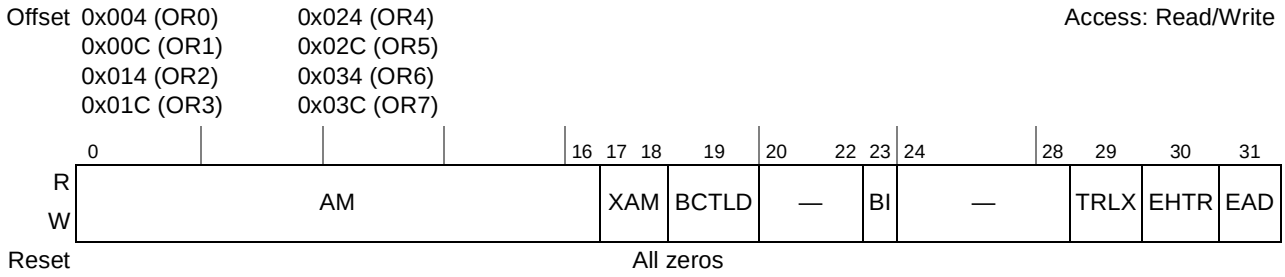


Figure 12-4. Option Registers (OR*n*) in UPM Mode

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

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

Bits	Name	Description
0–16	AM	UPM address mask. Masks corresponding BR*n* bits. Masking address bits independently allows external devices of different size address ranges to be used. Address mask bits can be set or cleared in any order in the field, allowing a resource to reside in more than one area of the address map. 0 Corresponding address bits are masked. 1 The corresponding address bits are used in the comparison with address signals.
17–18	XAM	Extended address mask. Masks the corresponding XBA bits in the BR*n* register, effectively extending the address mask (AM) by 2 bits.
19	BCTLD	Buffer control disable. Disables assertion of LBCTL during access to the current memory bank. 0 LBCTL is asserted upon access to the current memory bank. 1 LBCTL is not asserted upon access to the current memory bank.
20–22	—	Reserved
23	BI	Burst inhibit. Indicates if this memory bank supports burst accesses. 0 The bank supports burst accesses. 1 The bank does not support burst accesses. The selected UPM executes burst accesses as a series of single accesses.
24–28	—	Reserved
29	TRLX	Timing relaxed. Works in conjunction with EHTR to extend hold time on read accesses.
30	EHTR	Extended hold time on read accesses. Indicates with TRLX how many cycles are inserted between a read access from the current bank and the next access.
31	EAD	External address latch delay. Allow extra bus clock cycles when using external address latch (LALE). 0 No additional bus clock cycles (LALE asserted for one bus clock cycle only) 1 Extra bus clock cycles are added (LALE is asserted for the number of bus clock cycles specified by LCRR[EADC]).

TRLX	EHTR	Meaning
0	0	The memory controller generates normal timing. No additional cycles are inserted.
0	1	1 idle clock cycle is inserted.
1	0	4 idle clock cycles are inserted.
1	1	8 idle clock cycles are inserted.

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

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

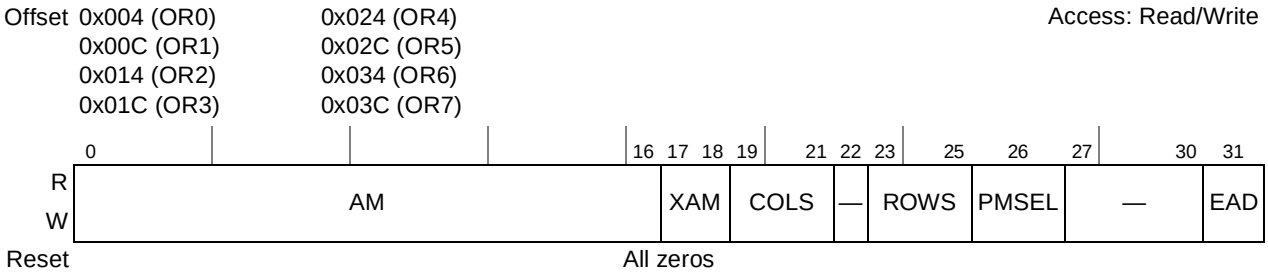


Figure 12-5. Option Registers (OR*n*) in SDRAM Mode

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

Table 12-8. OR*n*—SDRAM Field Descriptions

Bits	Name	Description
0–16	AM	SDRAM address mask. Masks corresponding BR*n* bits. Masking address bits independently allows external devices of different size address ranges to be used. Address mask bits can be set or cleared in any order in the field, allowing a resource to reside in more than one area of the address map. AM can be read or written at any time. 0 Corresponding address bits are masked. 1 The corresponding address bits are used in the comparison with address signals.
17–18	XAM	Extended address mask. Masks the corresponding BR*n*[XBA] bits, effectively extending the address mask (AM) by 2 bits.
19–21	COLS	Number of column address lines. Sets the number of column address lines in the SDRAM device. 000 7 100 11 001 8 101 12 010 9 110 13 011 10 111 14
22	—	Reserved
23–25	ROWS	Number of row address lines. Sets the number of row address lines in the SDRAM device. 000 9 100 13 001 10 101 14 010 11 110 15 011 12 111 Reserved
26	PMSEL	Page mode select. Selects page mode for the SDRAM connected to the memory controller bank. 0 Back-to-back page mode (normal operation). Page is closed when the bus becomes idle. 1 Page is kept open until a page miss or refresh occurs.
27–30	—	Reserved
31	EAD	External address latch delay. Allow extra bus clock cycles when using external address latch (LALE). 0 No additional bus clock cycles (LALE asserted for one bus clock cycle only) 1 Extra bus clock cycles are added (LALE is asserted for the number of bus clock cycles specified by LCRR[EADC]).

12.3.1.3 UPM Memory Address Register (MAR)

Figure 12-6 shows the fields of the UPM memory address register (MAR).

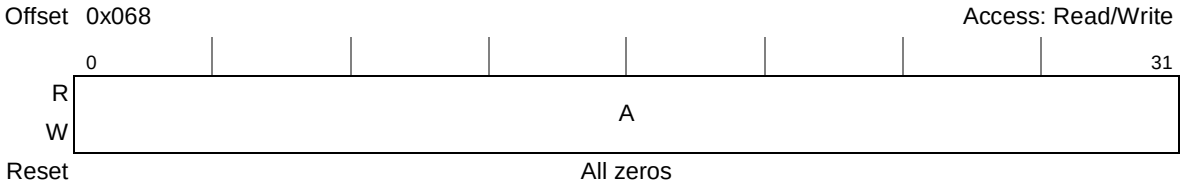


Figure 12-6. UPM Memory Address Register (MAR)

Table 12-9 describes the MAR fields.

Table 12-9. MAR Field Descriptions

Bits	Name	Description
0–31	A	Address that can be output to the address signals under control of the AMX bits in the UPM RAM word.

12.3.1.4 UPM Mode Registers (M*x*MR)

The UPM machine mode registers (MAMR, MBMR and MCMR), shown in Figure 12-7, contain the configuration for the three UPMs.

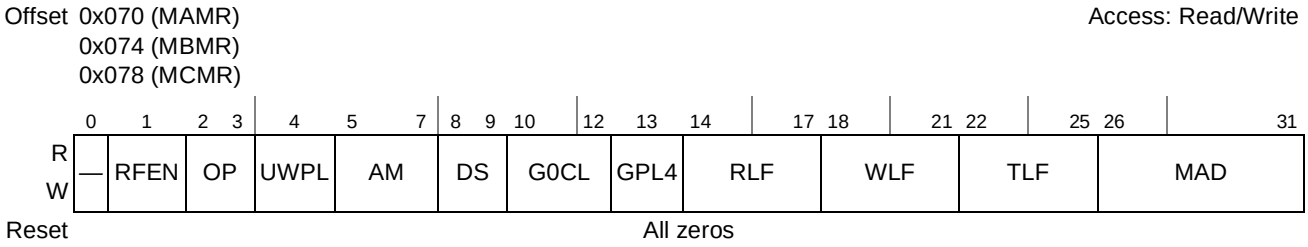


Figure 12-7. UPM Mode Registers (M*x*MR)

Table 12-10 describes UPM mode fields.

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

Bits	Name	Description
0	—	Reserved
1	RFEN	Refresh enable. Indicates that the UPM needs refresh services. This bit must be set for UPMA (refresh executor) if refresh services are required on any UPM assigned chip selects. If MAMR[RFEN] = 0, no refresh services can be provided, even if UPMB and/or UPMC have their RFEN bit set. 0 Refresh services are not required 1 Refresh services are required

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

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>2–3</td><td>OP</td><td>Command opcode. Determines the command executed by the UPMn when a memory access hits a UPM assigned bank. See Section 12.4.4.2, “Programming the UPMs,” for important programming considerations.
00 Normal operation
01 Write to UPM array. On the next memory access that hits a UPM assigned bank, write the contents of the MDR into the RAM location pointed to by MAD. After the access, MAD is automatically incremented.
10 Read from UPM array. On the next memory access that hits a UPM assigned bank, read the contents of the RAM location pointed to by MAD into the MDR. After the access, MAD is automatically incremented.
11 Run pattern. On the next memory access that hits a UPM assigned bank, run the pattern written in the RAM array. The pattern run starts at the location pointed to by MAD and continues until the LAST bit is set in the RAM word.</td></tr>
<tr><td>4</td><td>UWPL</td><td>LUPWAIT polarity active low. Sets the polarity of the LUPWAIT signal when in UPM mode.
0 LUPWAIT is active high.
1 LUPWAIT is active low.</td></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 signals. This field is needed when interfacing with devices requiring row and column addresses multiplexed on the same signals.
<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>
</table>

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

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></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 signal 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 signal is controlled by the corresponding bits in the UPMn array. See Table 12-28.
<table>
<tr><th rowspan="2">Value</th><th rowspan="2">LGPL4/LUPWAIT Signal Function</th><th colspan="2">Interpretation of UPM Word Bits</th></tr>
<tr><th>G4T1/DLT3</th><th>G4T3/WAEN</th></tr>
<tr><td>0</td><td>LGPL4 (output)</td><td>G4T1</td><td>G4T3</td></tr>
<tr><td>1</td><td>LUPWAIT (input)</td><td>DLT3</td><td>WAEN</td></tr>
</table></td></tr>
<tr><td>14–17</td><td>RLF</td><td>Read loop field. Determines the number of times a loop defined in the UPMn is executed for a burst- or single-beat read pattern or when MxMR[OP] = 11 (RUN command)
0000 16
0001 1
0010 2
0011 3
...
1110 14
1111 15</td></tr>
<tr><td>18–21</td><td>WLF</td><td>Write loop field. Determines the number of times a loop defined in the UPMn is executed for a burst- or single-beat write pattern.
0000 16
0001 1
0010 2
0011 3
...
1110 14
1111 15</td></tr>
<tr><td>22–25</td><td>TLF</td><td>Refresh loop field. Determines the number of times a loop defined in the UPMn is executed for a refresh service pattern.
0000 16
0001 1
0010 2
0011 3
...
1110 14
1111 15</td></tr>
<tr><td>26–31</td><td>MAD</td><td>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 UPMn.</td></tr>
</table>

12.3.1.5 Memory Refresh Timer Prescaler Register (MRTPR)

The refresh timer prescaler register (MRTPR), shown in Figure 12-8, is used to divide the system clock to provide the SDRAM and UPM refresh timers clock.

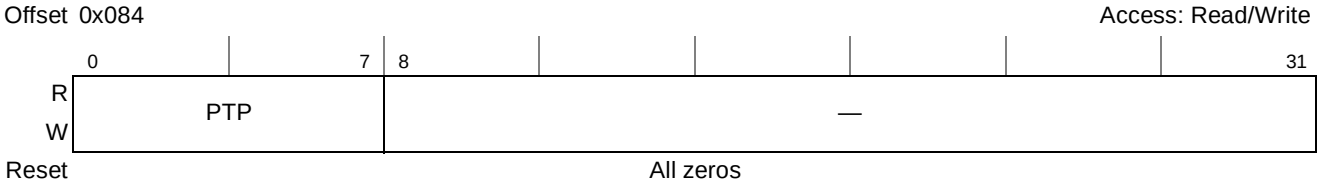


Figure 12-8. Memory Refresh Timer Prescaler Register (MRTPR)

Table 12-11 describes MRTPR fields.

Table 12-11. MRTPR Field Descriptions

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

12.3.1.6 UPM Data Register (MDR)

The memory data register (MDR), shown in Figure 12-9, contains data written to or read from the RAM array for UPM read or write commands. MDR must be set up before issuing a write command to the UPM.

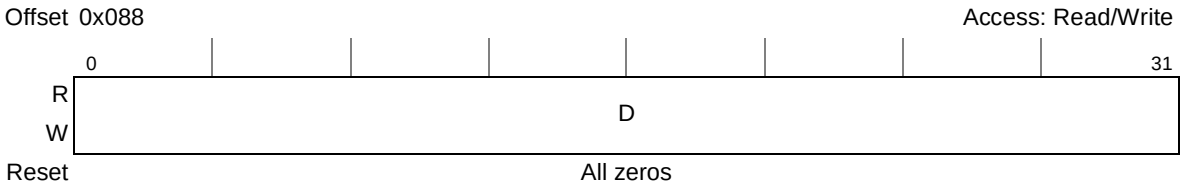


Figure 12-9. UPM Data Register (MDR)

Table 12-12 describes MDR[D].

Table 12-12. MDR Field Descriptions

Bits	Name	Description
0–31	D	The data to be read or written into the RAM array when a write or read command is supplied to the UPM (M*x*MR[OP] = 01 or M*x*MR[OP] = 10).

12.3.1.7 SDRAM Machine Mode Register (LSDMR)

The local bus SDRAM mode register (LSDMR), shown in Figure 12-10, is used to configure operations pertaining to SDRAM.

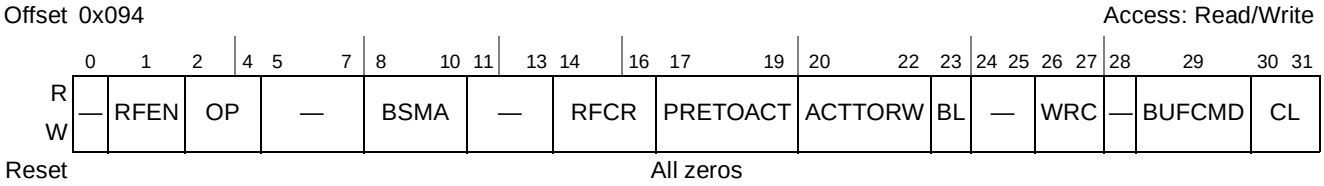


Figure 12-10. SDRAM Machine Mode Register (LSDMR)

Table 12-12 describes LSDMR fields.

Table 12-13. LSDMR Field Descriptions

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>0</td><td>—</td><td>Reserved</td></tr>
<tr><td>1</td><td>RFEN</td><td>Refresh enable. Indicates that the SDRAM requires refresh services.
0 Refresh services are not required.
1 Refresh services are required.</td></tr>
<tr><td>2–4</td><td>OP</td><td>SDRAM operation. Selects the operation that occurs when the SDRAM device is accessed.
<table>
<tr><th>Value</th><th>Meaning</th><th>Use</th></tr>
<tr><td>000</td><td>Normal operation</td><td>Normal operation</td></tr>
<tr><td>001</td><td>Auto refresh</td><td>Initialization</td></tr>
<tr><td>010</td><td>Self refresh</td><td>Power-down mode or debug</td></tr>
<tr><td>011</td><td>Mode Register write</td><td>Initialization</td></tr>
<tr><td>100</td><td>Precharge bank</td><td>Debug</td></tr>
<tr><td>101</td><td>Precharge all banks</td><td>Initialization</td></tr>
<tr><td>110</td><td>Activate bank</td><td>Debug</td></tr>
<tr><td>111</td><td>Read/write without valid data transfer</td><td>Debug</td></tr>
</table></td></tr>
<tr><td>5–7</td><td>—</td><td>Reserved</td></tr>
<tr><td>8–10</td><td>BSMA</td><td>Bank select multiplexed address line. Selects which address signals 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>
</table>

Table 12-13. LSDMR Field Descriptions (continued)

Bits	Name	Description
14–16	RFCR	Refresh recovery. Sets the refresh recovery interval in bus clock cycles. Defines the earliest timing for an ACTIVATE or REFRESH command after a REFRESH command. 000 Reserved 001 3 clocks 010 4 clocks 011 5 clocks 100 6 clocks 101 7 clocks 110 8 clocks 111 16 clocks
17–19	PRETOACT	Defines the earliest timing for ACTIVATE or REFRESH command after a PRECHARGE command (number of bus clock cycle wait states). 000 8 001 1 010 2 011 3 100 4 101 5 110 6 111 7
20–22	ACTTORW	Defines the earliest timing for READ/WRITE command after an ACTIVATE command (number of bus clock cycle wait states). 000 8 001 Reserved 010 2 011 3 100 4 101 5 110 6 111 7
23	BL	Sets the burst length for SDRAM accesses. 0 SDRAM burst length is 4. Use this value if the device port size is16. 1 SDRAM burst length is 8. Use this value if the device port size is 32 or 8.
24–25	—	Reserved
26–27	WRC	Write recovery time. Defines the earliest timing for PRECHARGE command after the last data is written to the SDRAM. 00 4 01 Reserved 10 2 11 3
28	—	Reserved
29	BUFCMD	Control line assertion timing. If external buffers are placed on the control lines going to both the SDRAM and address lines, setting BUFCMD causes all SDRAM control lines except $\overline{\text{LCS}n}$, LCKE, LALE, and $\overline{\text{LSDDQM}}$[0:3] to be asserted for LCRR[BUFCMDC] cycles, instead of one. 0 Normal timing for the control lines 1 All control lines except $\overline{\text{LCS}n}$ are asserted for the number of cycles specified by LCRR[BUFCMDC].
30–31	CL	CAS latency. Defines the timing for first read data after SDRAM samples a column address. 00 Extended CAS latency. According to LCRR[ECL]. See Table 12-22. 01 1 10 2 11 3

12.3.1.8 UPM Refresh Timer (LURT)

The UPM refresh timer (LURT), shown in Figure 12-11, generates a refresh request for all valid banks that selected a UPM machine and are refresh-enabled (M*x*MR[RFEN] = 1). Each time the timer expires, a qualified bank generates a refresh request using the selected UPM. The qualified banks rotate their requests.

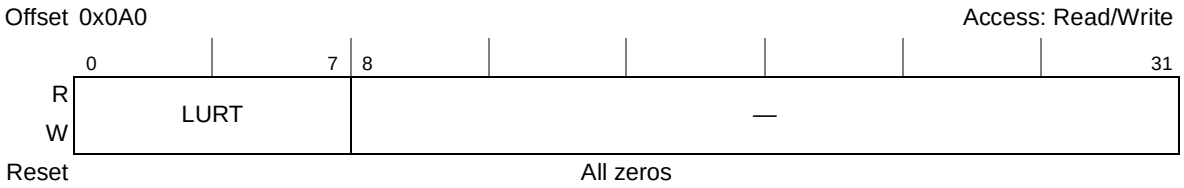


Figure 12-11. UPM Refresh Timer (LURT)

Table 12-14 describes LURT fields.

Table 12-14. LURT Field Descriptions

Bits	Name	Description
0–7	LURT	UPM refresh timer period. Determines, along with the timer prescaler (MRTPR), the timer period according to the following equation: $\text{TimerPeriod} = \frac{\text{LURT}}{\left(\frac{\text{Fsystemclock}}{\text{MRTPR[PTP]}}\right)}$ **Example**: For a 266-MHz system clock and a required service rate of 15.6 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 × MRTPR[PTP] system clock cycles.
8–31	—	Reserved

12.3.1.9 SDRAM Refresh Timer (LSRT)

The SDRAM refresh timer (LSRT), shown in Figure 12-12, generates a refresh request for all valid banks that selected a SDRAM machine and are refresh-enabled (LSDMR[RFEN] = 1). Each time the timer expires, all qualifying banks generate a bank staggering auto-refresh request using the SDRAM machine.

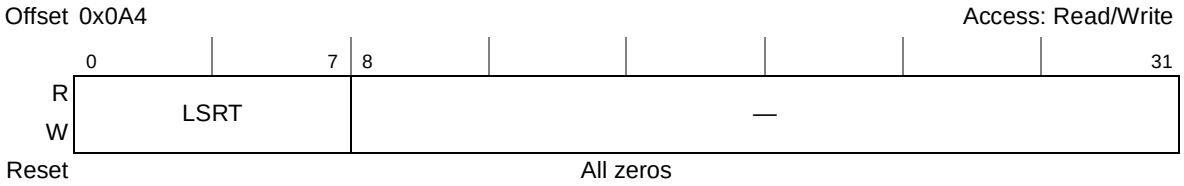


Figure 12-12. LSRT SDRAM Refresh Timer (LSRT)

Table 12-15 describes LSRT fields.

Table 12-15. LSRT Field Descriptions

Bits	Name	Description
0–7	LSRT	SDRAM refresh timer period. Determines, along with the timer prescaler (MRTPR), the timer period according to the following equation: $\text{TimerPeriod} = \frac{\text{LSRT}}{\left(\frac{\text{Fsystemclock}}{\text{MRTPR[PTP]}}\right)}$ **Example:** For a 266-MHz system clock and a required service rate of 15.6 µs, given PTP = 32, the LSRT value should be 128 decimal. 128/(266 MHz/32) = 15.4 µs, which is less than the required service period of 15.6 µs. Note that the reset value, 0x00, sets the maximum period to 256 × MRTPR[PTP] system clock cycles.
8–31	—	Reserved

12.3.1.10 Transfer Error Status Register (LTESR)

The LBC has the following registers for error management:

- The transfer error status register (LTESR) indicates the cause of an error.
- The transfer error check disable register (LTEDR) is used to enable (and disable) error checking.
- The transfer error check interrupt register (LTEIR) enables reporting of errors through an interrupt.
- The transfer error attributes register (LTEATR) captures source attributes of an error.
- The transfer error address register (LTEAR) captures the address of a transaction that caused an error.

LTESR, shown in Figure 12-13, is a write-one-to-clear register. Reading LTESR occurs normally; however, write operations can clear but not set bits. A bit is cleared whenever the register is written and the data in the corresponding bit location is a 1. For example, to clear only the write protect error bit (LTESR[WP]) without affecting other LTESR bits, 0x0400_0000 should be written to the register. Note that LTEATR[V] bit has to be cleared to register subsequent errors in LTESR.

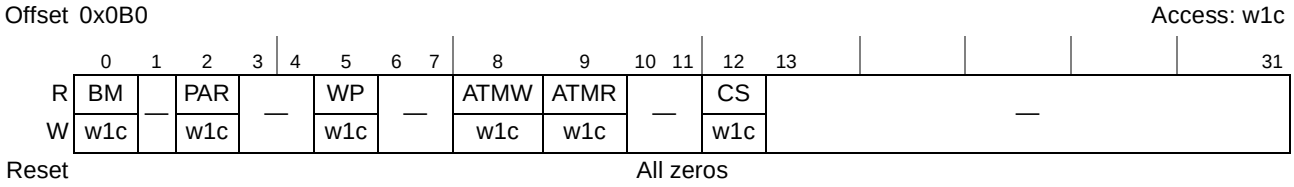


Figure 12-13. Transfer Error Status Register (LTESR)

Table 12-16 describes LTESR fields.

Table 12-16. LTESR Field Descriptions

Bits	Name	Description
0	BM	Bus monitor time-out 0 No local bus monitor time-out occurred. 1 Local bus monitor time-out occurred. No data beat was acknowledged on the bus within LBCR[BMT] × 8 bus clock cycles from the start of a transaction.
1	—	Reserved
2	PAR	Parity 0 No local bus parity error 1 Local bus parity error. LTEATR[PB] indicates the byte lane that caused the error and LTEATR[BNK] indicates which memory controller bank was accessed.
3–4	—	Reserved
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 occurs (as the cycle is not automatically terminated).
6–7	—	Reserved
8	ATMW	Atomic error write 0 No atomic write error occurred. 1 The subsequent write (WARA) to a memory bank did not occur within 256 bus clock cycles.
9	ATMR	Atomic error read 0 No atomic read error occurred. 1 The subsequent read (RAWA) to a memory bank did not occur within 256 bus clock cycles.
10–11	—	Reserved
12	CS	Chip select error 0 No chip select error occurred. 1 A transaction was sent to the LBC that did not hit any memory bank.
13–31	—	Reserved

12.3.1.11 Transfer Error Check Disable Register (LTEDR)

The transfer error check disable register (LTEDR), shown in Figure 12-14, is used to disable error checking. Note that control of error checking is independent of control of reporting of errors (LTEIR) through the interrupt mechanism.

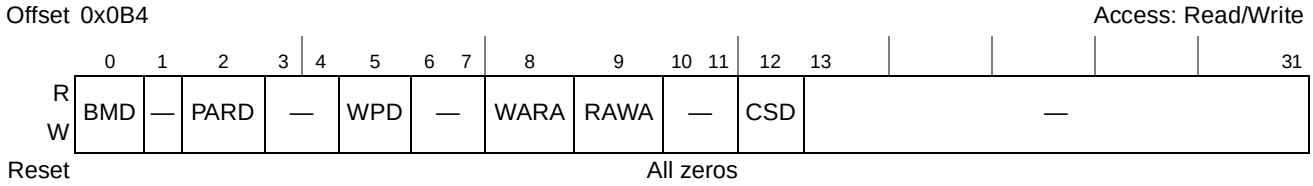


Figure 12-14. Transfer Error Check Disable Register (LTEDR)

Table 12-17 describes LTEDR fields.

Table 12-17. LTEDR Field Descriptions

Bits	Name	Description
0	BMD	Bus monitor disable 0 Bus monitor is enabled 1 Bus monitor is disabled
1	—	Reserved
2	PARD	Parity error checking disabled. 0 Parity error checking is enabled. 1 Parity error checking is disabled.
3–4	—	Reserved
5	WPD	Write protect error checking disable 0 Write protect error checking is enabled. 1 Write protect error checking is disabled.
6–7	—	Reserved
8	WARA	Write-after-read atomic (WARA) error checking disable 0 WARA error checking is enabled. 1 WARA error checking is disabled.
9	RAWA	Read-after-write atomic (RAWA) error checking disable 0 RAWA error checking is enabled. 1 RAWA error checking is disabled.
10–11	—	Reserved
12	CSD	Chip select error checking disable 0 Chip select error checking is enabled. 1 Chip select error checking is disabled.
13–31	—	Reserved

12.3.1.12 Transfer Error Interrupt Enable Register (LTEIR)

The transfer error interrupt enable register (LTEIR), shown in Figure 12-15, is used to send or block error reporting through the LBC internal interrupt mechanism. Software should clear pending errors in LTESR before enabling interrupts. After an interrupt has occurred, clearing relevant LTESR error bits negates the interrupt.

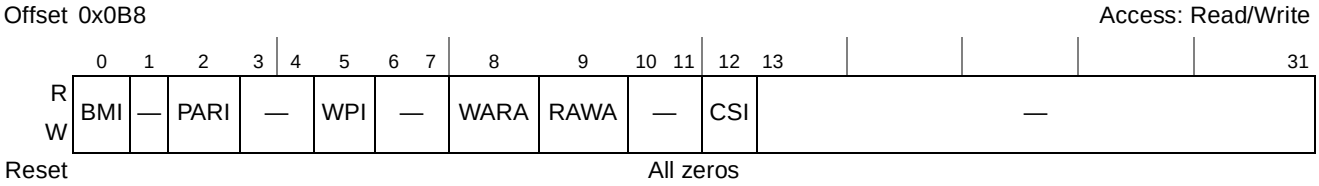


Figure 12-15. Transfer Error Interrupt Enable Register (LTEIR)

Table 12-18 describes LTEIR fields.

Table 12-18. LTEIR Field Descriptions

Bits	Name	Description
0	BMI	Bus monitor error interrupt enable 0 Bus monitor error reporting is disabled. 1 Bus monitor error reporting is enabled.
1	—	Reserved
2	PARI	Parity error interrupt enable. 0 Parity error reporting is disabled. 1 Parity error reporting is enabled.
3–4	—	Reserved
5	WPI	Write protect error interrupt enable 0 Write protect error reporting is disabled. 1 Write protect error reporting is enabled.
6–7	—	Reserved
8	WARA	Write-after-read atomic (WARA) error interrupt enable 0 WARA error reporting is disabled. 1 WARA error reporting is enabled.
9	RAWA	Read-after-write atomic (RAWA) error interrupt enable 0 RAWA error reporting is disabled. 1 RAWA error reporting is enabled.
10–11	—	Reserved
12	CSI	Chip select error interrupt enable 0 Chip select error reporting is disabled. 1 Chip select error reporting is enabled.
13–31	—	Reserved

12.3.1.13 Transfer Error Attributes Register (LTEATR)

Figure 12-16 shows the LTEATR. After LTEATR[V] has been set, software must clear this bit to allow LBC error registers to update following any subsequent errors.

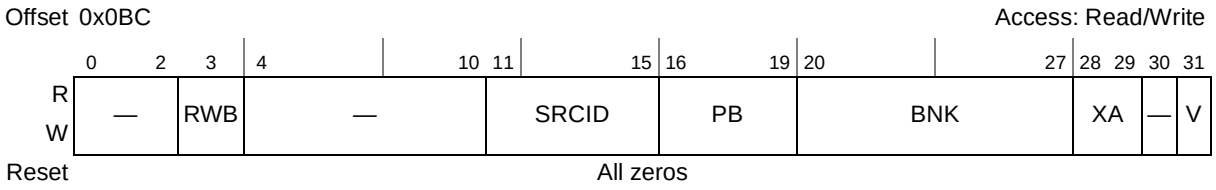


Figure 12-16. Transfer Error Attributes Register (LTEATR)

Table 12-19 describes LTEATR fields.

Table 12-19. LTEATR Field Descriptions

Bits	Name	Description
0–2	—	Reserved
3	RWB	Transaction type for the error 0 The transaction for the error was a write transaction. 1 The transaction for the error was a read transaction.
4–10	—	Reserved
11–15	SRCID	Transaction source for the error 00000 PCI Express interface 1 00001 PCI Express interface 2/RapidIO 00011–01001 Reserved 01010 Boot sequencer 01011–01111 Reserved 10000 Core 0 (instruction/data) 10001 Reserved 10010 Core 1 (instruction/data) 10011–10100 Reserved 10101 DMA 10110–10111 Reserved 11000 eTSEC 1 11001 eTSEC 2 11010 eTSEC 3 11011 eTSEC 4 11100 RapidIO message unit 11101 RapidIO doorbell unit 11110 RapidIO port-write unit 11111 Reserved
16–19	PB	Parity error on byte. There are four parity error status bits, one per byte lane. A bit is set for the byte that had a parity error (bit 16 represents byte 0, the most significant byte lane).
20–27	BNK	Memory controller bank. There is one error status bit per memory controller bank (bit 20 represents bank 0). A bit is set for the local bus memory controller bank that had an error. Note that BNK is invalid if the error was not caused by parity checks.
28–29	XA	Extended address for the error. These bits capture the two msbs of the 34-bit address of the transaction resulting in an error.
30	—	Reserved
31	V	Error attribute capture is valid. Indicates that the captured error information is valid 0 Captured error attributes and address are not valid 1 Captured error attributes and address are valid

12.3.1.14 Transfer Error Address Register (LTEAR)

The transfer error address register (LTEAR) is shown in Figure 12-17.

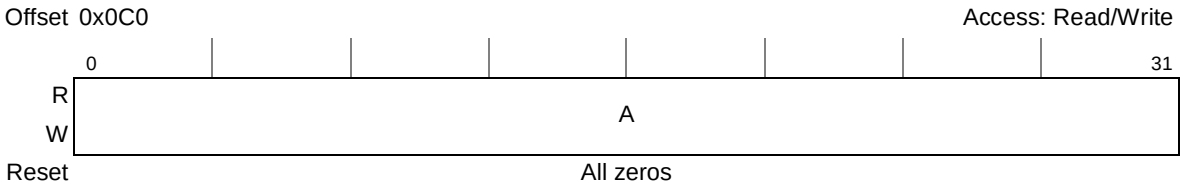


Figure 12-17. Transfer Error Address Register (LTEAR)

Table 12-20 describes LTEAR fields.

Table 12-20. LTEAR Field Descriptions

Bits	Name	Description
0–31	A	Transaction address for the error. Holds the 32 lsbs of the 34-bit address of the transaction resulting in an error.

12.3.1.15 Local Bus Configuration Register (LBCR)

The local bus configuration register (LBCR) is shown in Figure 12-18.

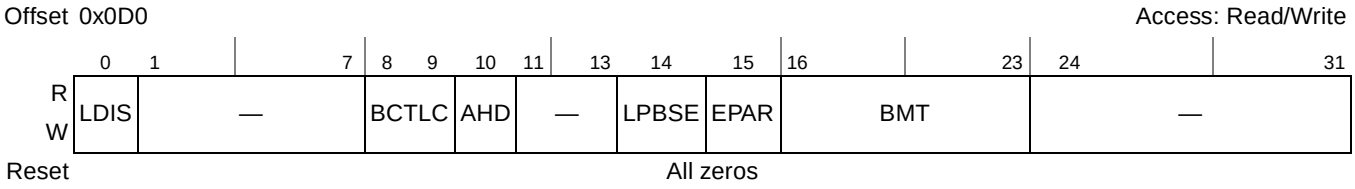


Figure 12-18. Local Bus Configuration Register

Table 12-21 describes LBCR fields.

Table 12-21. LBCR Field Descriptions

Bits	Name	Description
0	LDIS	Local bus disable 0 Local bus is enabled 1 Local bus is disabled. No internal transactions are acknowledged.
1–7	—	Reserved
8–9	BCTLC	Defines the use of LBCTL 00 LBCTL is used as W/$\overline{R}$ control for GPCM or UPM accesses (buffer control). 01 LBCTL is used as $\overline{LOE}$ for GPCM accesses only. 10 LBCTL is used as $\overline{LWE}$ for GPCM accesses only. 11 Reserved.
10	AHD	Address hold disable. Removes part of the hold time for LAD with respect to LALE in order to lengthen the LALE pulse 0 During address phases on the local bus, the LALE signal negates two platform clock periods prior to the address being invalidated. At 666 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 one platform clock period prior to the address being invalidated. This halves the address hold time, but extends the latch enable duration. This may be necessary for very high frequency designs.
11–13	—	Reserved
14	LPBSE	Enables parity byte select on $\overline{LGTA}$/LGPL4/LUPWAIT/LPBSE signal. 0 Parity byte select is disabled. $\overline{LGTA}$/LGPL4/LUPWAIT/LPBSE signal is available for memory control as LGPL4 (output) or $\overline{LGTA}$/LUPWAIT (input). 1 Parity byte select is enabled. $\overline{LGTA}$/LGPL4/LUPWAIT/LPBSE signal is dedicated as the parity byte select output, and $\overline{LGTA}$/LUPWAIT is disabled.

Table 12-21. LBCR Field Descriptions (continued)

Bits	Name	Description
15	EPAR	Determines odd or even parity. Writing the memory with EPAR = 1 and reading the memory with EPAR = 0 generates parity errors for testing. 0 Odd parity 1 Even parity
16–23	BMT	Bus monitor timing. Defines the bus monitor time-out period. Clearing BMT (reset value) selects the maximum count of 2048 bus clock cycles. For non-zero values of BMT, the number of LCLK clock cycles to count down before a time-out error is generated is given by: bus cycles = BMT x 8. Apart from BMT = 0x00, the minimum value of BMT is 5, corresponding with 40 bus cycles. Shorter time-outs may result in spurious errors during SDRAM operation.
24–31	—	Reserved

12.3.1.16 Clock Ratio Register (LCRR)

The clock ratio register sets the system (MPX) clock to LBC bus frequency ratio. It also provides configuration bits for extra delay cycles for address and control signals.

Offset 0x0D4 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	PBYP	—	BUFCMDC		—		ECL		—						EADC		—												CLKDIV			
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

Figure 12-19. Clock Ratio Register (LCRR)

Table 12-22 describes LCRR fields.

Table 12-22. LCRR Field Descriptions

Bits	Name	Description
0	PBYP	PLL bypass. This bit should be set when using low bus clock frequencies if the PLL is unable to lock. When in PLL bypass mode, incoming data is captured in the middle of the bus clock cycle.It is recommended that PLL bypass mode be used at frequencies of 83 MHz or less. 0 The PLL is enabled. 1 The PLL 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

Table 12-22. LCRR Field Descriptions (continued)

Bits	Name	Description
6–7	ECL	Extended CAS latency. Determines the extended CAS latency for SDRAM accesses when LSDMR[CL] = 00. 00 4 01 5 10 6 11 7
8–13	—	Reserved
14–15	EADC	Additional external address delay cycles. Defines the number of additional 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 (MPX) clock divider. Sets the frequency ratio between the system (MPX) clock and the memory bus clock. Only the values shown in the table below are allowed. **Note:** It is critical that no transactions are being executed through the local bus while CLKDIV is being modified. As such, prior to modification, the user must ensure that code is not executing out of the local bus. Once LCRR[CLKDIV] is written, the register should be read, and then an isync should be executed. 0000–0001 Reserved 0010 4 0011 Reserved 0100 8 0101–0111 Reserved 1000 16 1001–1111 Reserved

12.4 Functional Description

The LBC allows the implementation of memory systems with very specific timing requirements.

- The SDRAM machine provides an interface to SDRAMs using bank interleaving and back-to-back page mode to achieve high performance through a multiplexed address/data bus. An internal PLL for bus clock generation ensures improved data set-up margins for board designs.
- The GPCM provides interfacing for simpler, lower-performance memories and memory-mapped devices. It has inherently lower performance because it does not support bursting. For this reason, GPCM-controlled banks are used primarily for boot-loading and access to low-performance memory-mapped peripherals.
- The UPM supports refresh timers, address multiplexing of the external bus and generation of programmable control signals for row address and column address strobes, to allow for a minimal glue logic interface to DRAMs, burstable SRAMs, and almost any other kind of peripheral. The UPM can be used to generate flexible, user-defined timing patterns for control signals that govern a memory device. These patterns define how the external control signals behave during a read, write, burst-read, or burst-write access. Refresh timers are also available to periodically initiate user-defined refresh patterns.

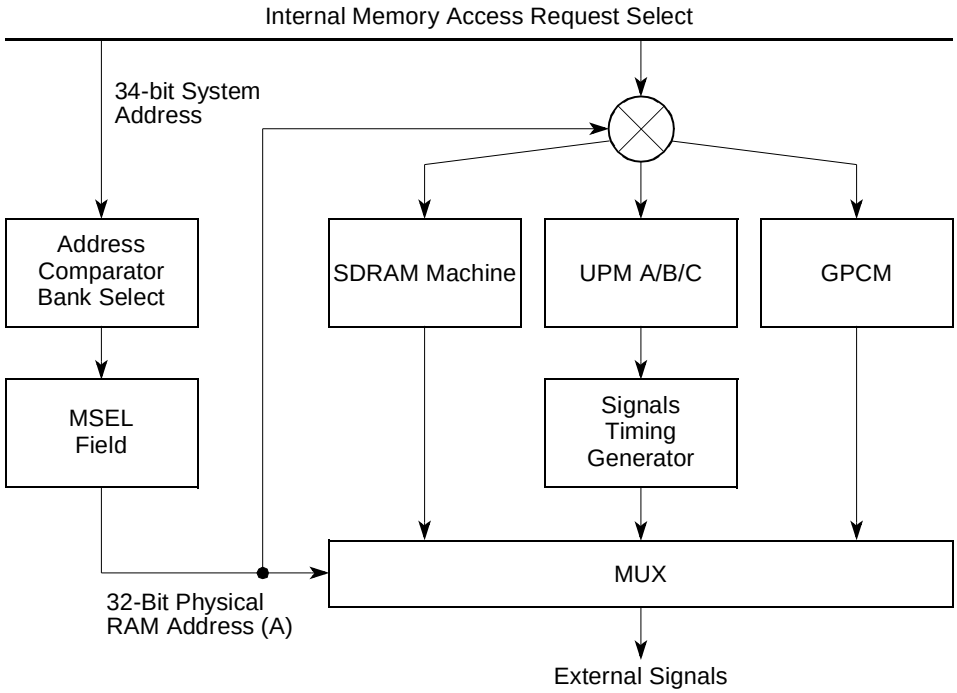


Figure 12-20. Basic Operation of Memory Controllers in the LBC

Each memory bank (chip select) can be assigned to any one of these three type of machines through the machine select bits of the base register for that bank (BR*n*[MSEL]), as illustrated in Figure 12-20. If a bank match occurs, the corresponding machine (GPCM, SDRAM or UPM) then takes ownership of the external signals that control the access and maintains control until the transaction ends.

12.4.1 Basic Architecture

The following sections describe the basic architecture of the LBC.

12.4.1.1 Address and Address Space Checking

The defined base addresses are written to the BR*n* registers, while the corresponding address masks are written to the OR*n* registers. Each time a local bus access is requested, the internal transaction address is compared with each bank. Addresses are decoded by comparing the 19 msbs of the address, masked by OR*n*[XAM] and OR*n*[AM], with the base address for each bank (BR*n*[XBA] and BR*n*[BA]). If a match is found on a memory controller bank, the attributes defined in the BR*n* and OR*n* for that bank are used to control the memory access. If a match is found in more than one bank, the lowest-numbered bank handles the memory access (that is, bank 0 has priority over bank 1).

12.4.1.2 External Address Latch Enable Signal (LALE)

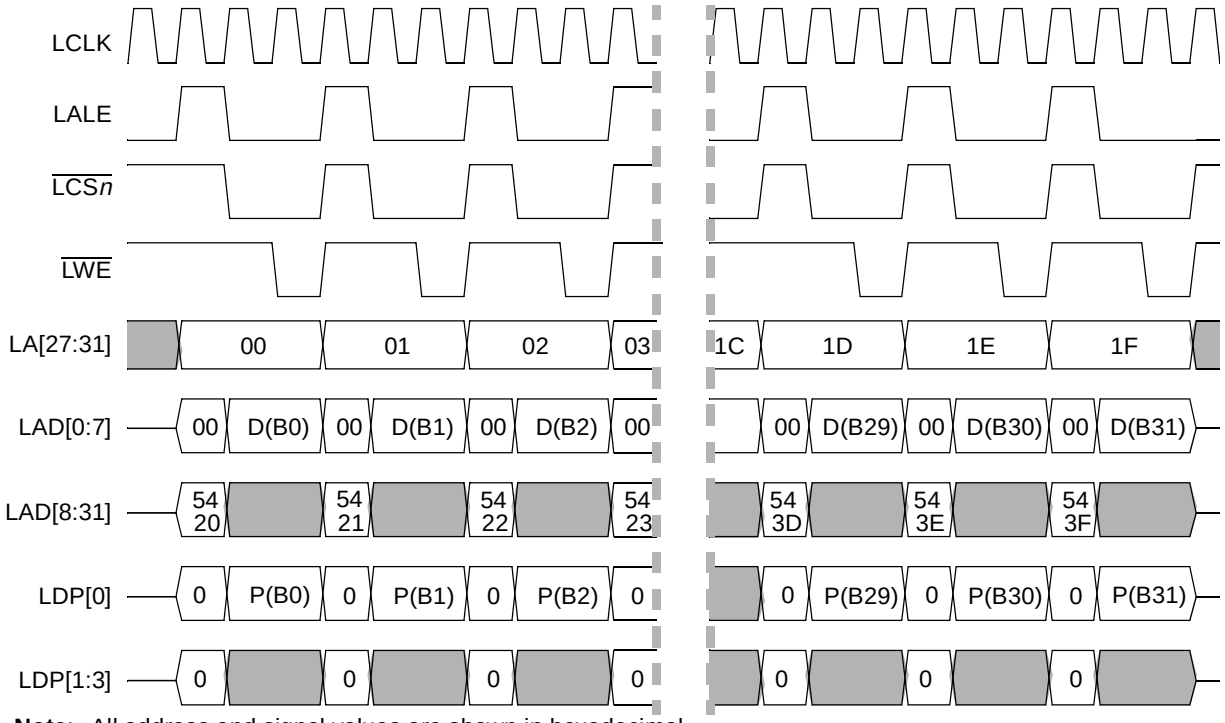
The local bus uses a multiplexed address/data bus. Therefore, the LBC must distinguish between address and data phases, which take place on the same bus (LAD[0:31] signals). The LALE signal, when asserted, signifies an address phase during which the LBC drives the memory address on the LAD[0:31] signals.

An external address latch uses this signal to capture the address and provide it to the address signals of the memory or peripheral device. When LALE is negated, LAD[0:31] then serves as the (bidirectional) 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 two platform clock periods (which, divided by LCRR[CLKDIV], yields the local bus clock). For example, if the LBC is operating at 666 MHz internally, then an additional 3 ns of address hold time is introduced. However, when LCRR[CLKDIV] = 2 (clock ratio of 4) and the LCLK frequency exceeds 100 MHz, the duration of the shortened LALE pulse may not meet the minimum latch enable pulse width specifications of some latches. In such cases, setting LBCR[AHD] = 1 increases the LALE pulse width by one platform clock cycle, and decreases the address hold time by the same amount. At 666 MHz and with LCRR[CLKDIV] = 2 (clock ratio of 4), the duration of LALE would then be 4.5 ns, with 1.5 ns of hold time. If both longer hold time and longer LALE pulse duration are needed, then the address phase can be extended using the ORn[EAD] and LCRR[EADC] fields, and the LBCR[AHD] bit can be left at 0. However, this adds 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, on commencement, but it is possible to program UPM to assert LALE several times, and to change the values of LA[27:31] with and without LALE being involved. In general, when using the GPCM and SDRAM controllers it is not necessary to use LA[27:31] if a sufficiently wide latch is used to capture the entire address during LALE phases. UPM may require LA[27:31] if the LBC is generating its own burst address sequence.

To illustrate how a large transaction is handled by the LBC, Figure 12-21 shows LBC signals for GPCM performing a 32-byte write starting at address 0x5420. Note that during each of the 32 assertions of LALE, LA[27:31] exactly mirror LAD[27:31], but during data phases, only LAD[0:7] and LDP[0] are driven with valid data and parity, respectively.



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

Figure 12-21. Example of 8-Bit GPCM Writing 32 Bytes to Address 0x5420

12.4.1.3 Data Transfer Acknowledge (TA)

The three memory controllers in the LBC generate an internal transfer acknowledge signal, TA, to allow data on LAD[0:31] to be either sampled (for reads) or changed (on writes). The data sampling/data change always occurs at the end of the bus cycle in which the LBC asserts TA internally. In LBC debug mode, TA is also visible externally on the MDVAL signal. GPCM and SDRAM controllers automatically generate TA according to the timing parameters programmed for them in option and mode registers; a UPM generates TA only when a UPM pattern has the UTA RAM word bit set.

Figure 12-22 shows LALE, TA (internal), and $\overline{\text{LCS}n}$. Note that TA and LALE are never asserted together, and that for the duration of LALE, $\overline{\text{LCS}n}$ (or any other control signal) remains negated or frozen.

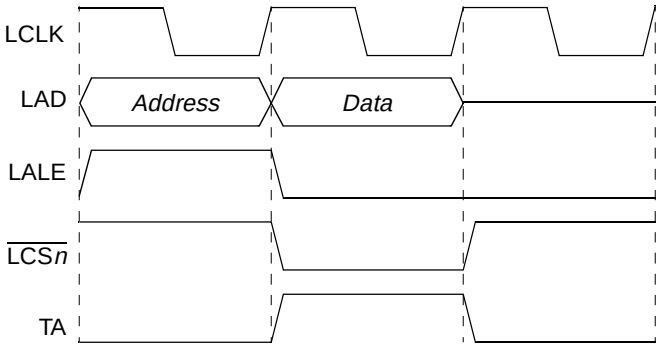


Figure 12-22. Basic LBC Bus Cycle with LALE, TA, and $\overline{\text{LCS}n}$

12.4.1.4 Data Buffer Control (LBCTL)

The memory controller provides a data buffer control signal for the local bus (LBCTL). This signal is activated when a GPCM or UPM controlled bank is accessed. LBCTL can be disabled by setting OR*n*[BCTLD]. Access to an SDRAM machine controlled bank does not activate the LBCTL control. LBCTL can be further configured by LBCR[BCTLC] to act as an extra $\overline{\text{LWE}}$ or an extra $\overline{\text{LOE}}$ signal when in GPCM mode.

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

If an external bus transceiver is used, LBCTL should be used to signify the write direction when high. Note that the default (reset and bus idle) value of LBCTL is also high.

12.4.1.5 Atomic Operation

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

- Read-after-write atomic (RAWA). When a write access hits a memory bank in which ATOM = 01, the LBC reserves the selected memory bank for the exclusive use of the accessing master.

  While the bank is reserved, no other device can be granted access to this bank. The reservation is released when the master that created it accesses the same bank with a read transaction. Additional write transactions prior to the releasing read do not change reservation status, but are otherwise processed normally. If the master fails to release the reservation within 256 bus clock cycles, the reservation is released and an atomic error is reported (if enabled). This feature is intended for CAM operations.
- Write-after-read atomic (WARA). When a read access hit a memory bank in which ATOM = 10, the LBC reserves the bus for the exclusive use of the accessing master.

  During the reservation period, no other device can be granted access to the atomic bank. The reservation is released when the device that created it accesses the same bank with a write transaction. Additional read transactions prior to the releasing write are otherwise processed normally and do not change the reservation status. If the device fails to release the reservation within 256 bus clock cycles, the reservation is released and an atomic error is reported (if enabled).

12.4.1.6 Parity Generation and Checking (LDP)

Parity can be configured for any bank by programming BR*n*[DECC]. Parity is generated and checked on a per-byte basis using LDP[0:3] for the bank if BR*n*[DECC] = 01 (normal parity) or BR*n*[DECC] = 10 for read-modify-write (RMW) parity. Byte lane parity on LDP[0:3] is generated regardless of the BR*n*[DECC] setting. Note that RMW parity can be used only for 32-bit port size banks. LBCR[EPAR] determines the global type of parity (odd or even).

12.4.1.7 Bus Monitor

A bus monitor is provided to ensure that each bus cycle is terminated within a reasonable (user defined) period. When a transaction starts, the bus monitor starts counting down from the time-out value (LBCR[BMT]) until a data beat is acknowledged on the bus. It then reloads the time-out value and resumes the countdown until the data tenure completes and then idles if there is no pending transaction. Setting LTEDR[BMD] disables bus monitor error checking (i.e,. the LTESR[BM] bit is not set by a bus monitor time-out); however, the bus monitor is still active and can generate a UPM exception (as noted in Section 12.4.4.1.4, “Exception Requests”) or terminate a GPCM access.

It is very important to ensure that the value of LBCR[BMT] is not set too low; otherwise spurious bus time-outs may occur during normal operation—particularly for SDRAMs— resulting in incomplete data transfers. Accordingly, apart from the reset value of 0x00 (corresponding with the maximum time-out of 2048 bus cycles), LBCR[BMT] must not be set below 0x05 (or 40 bus cycles for time-out) under any circumstances.

12.4.2 General-Purpose Chip-Select Machine (GPCM)

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

Figure 12-23 shows a simple connection between an 8-bit port size SRAM device and the LBC in GPCM mode. Byte-write enable signals ($\overline{\text{LWE}}$) are available for each byte written to memory. Also, the output enable signal ($\overline{\text{LOE}}$) is provided to minimize external glue logic. On system reset, a global (boot) chip-select is available that provides a boot ROM chip-select ($\overline{\text{LCS0}}$) prior to the system being fully configured.

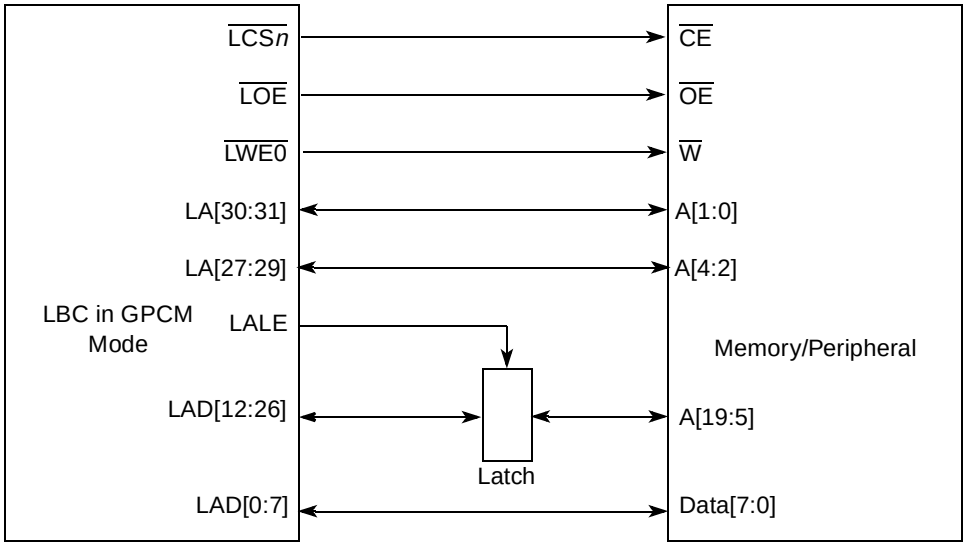


Figure 12-23. Local Bus to GPCM Device Interface

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

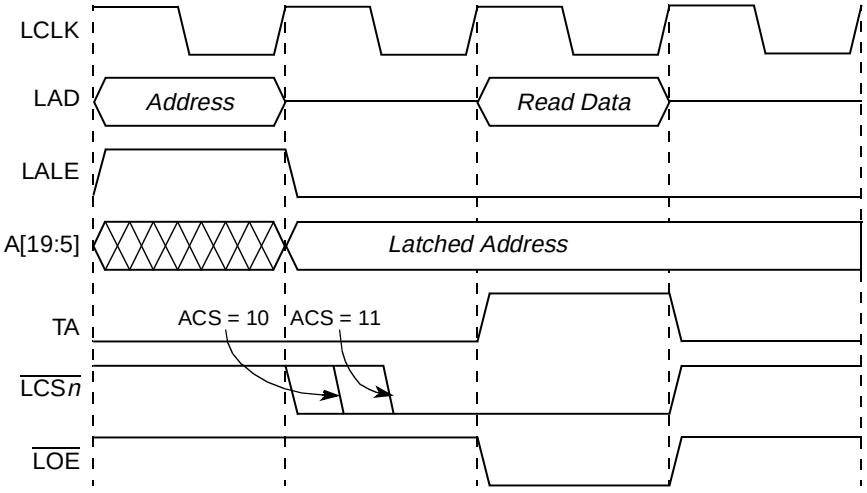


Figure 12-24. GPCM Basic Read Timing (XACS = 0, ACS = 1x, TRLX = 0)

12.4.2.1 Timing Configuration

If BR*n*[MSEL] selects the GPCM, the attributes for the memory cycle are taken from OR*n*. These attributes include the CSNT, ACS, XACS, SCY, TRLX, EHTR, and SETA fields.

Table 12-23 shows signal behavior and system response for a write access.

Table 12-23. GPCM Write Control Signal Timing

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

Table 12-23. GPCM Write Control Signal Timing (continued)

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]
1	0	10	0	1+1/4	0	0	4+2*SCY
1	0	11	0	1+1/2	0	0	4+2*SCY
1	1	00	0	0	0	0	3+2*SCY
1	1	10	0	2	0	0	4+2*SCY
1	1	11	0	3	0	0	5+2*SCY
1	0	00	1	0	0	–1–1/4	4+2*SCY
1	0	10	1	1+1/4	–1–1/4	–1–1/4	5+2*SCY
1	0	11	1	1+1/2	–1–1/4	–1–1/4	5+2*SCY
1	1	00	1	0	0	–1–1/4	4+2*SCY
1	1	10	1	2	–1–1/4	–1–1/4	5+2*SCY
1	1	11	1	3	–1–1/4	–1–1/4	6+2*SCY

1 Total cycles when LALE is asserted for one cycle only (OR*n*[EAD] = 0; OR*n*[EAD] = 1 and LCRR[EADC] = 01). Asserting LALE for more than one cycle increases the total cycle count accordingly.

Table 12-24 shows the signal behavior and system response for a read access.

Table 12-24. GPCM Read Control Signal Timing

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

Table 12-24. GPCM Read Control Signal Timing (continued)

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]
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.

12.4.2.2 Chip-Select Assertion Timing

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

- Simultaneous with the latched memory address. (This refers to the externally latched address, not the address timing on LAD[0:31]. That is, chip select does not assert during LALE).
- One quarter of a clock cycle later.
- One half of a clock cycle later
- One clock cycle later (for LCRR[CLKDIV] = 2 (clock ratio of 4)) when OR*n*[XACS] = 1.
- Two clock cycles later, when OR*n*[XACS] = 1.
- Three clock cycles later, when OR*n*[XACS] = 1 and OR*n*[TRLX] = 1.

The timing diagram in Figure 12-24 shows two chip-select assertion timings.

12.4.2.2.1 Programmable Wait State Configuration

The GPCM supports internal generation of transfer acknowledge. It allows between zero and 30 wait states to be added to an access by programming OR*n*[SCY] and OR*n*[TRLX]. Internal generation of transfer acknowledge is enabled if OR*n*[SETA] = 0. If $\overline{\text{LGTA}}$ is asserted externally two bus clock cycles or more before the wait state counter has expired (to allow for synchronization latency), the current memory cycle is terminated by $\overline{\text{LGTA}}$; otherwise it is terminated by the expiration of the wait state counter. Regardless of the setting of OR*n*[SETA], wait states prolong the assertion duration of both $\overline{\text{LOE}}$ and $\overline{\text{LWE}n}$ in the same manner. When TRLX = 1, the number of wait states inserted by the memory controller is doubled from OR*n*[SCY] cycles to 2 × OR*n*[SCY] cycles, allowing a maximum of 30 wait states.

12.4.2.2.2 Chip-Select and Write Enable Negation Timing

Figure 12-23 shows a basic connection between the local bus and a static memory device. In this case, $\overline{\text{LCS}n}$ is connected directly to $\overline{\text{CE}}$ of the memory device. The $\overline{\text{LWE}}$[0:3] signals are connected to the respective $\overline{\text{WE}}$[3:0] signals on the memory device where each $\overline{\text{LWE}}$[0:3] signal corresponds to a different data byte.

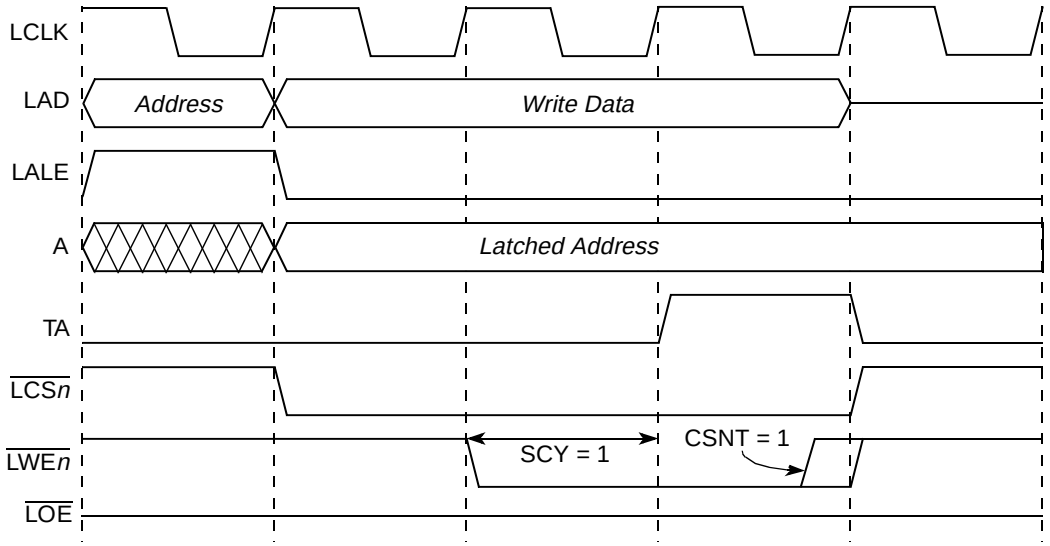


Figure 12-25. GPCM Basic Write Timing (XACS = 0, ACS = 00, CSNT = 1, SCY = 1, TRLX = 0)

As Figure 12-25 shows, the timing for $\overline{\text{LCS}n}$ is the same as for the latched address. The strobes for the transaction are supplied by $\overline{\text{LOE}}$ or $\overline{\text{LWE}n}$, depending on the transaction direction—read or write (write case shown in Figure 12-25). OR*n*[CSNT] controls the timing for the appropriate strobe negation in write cycles. When this attribute is asserted, the strobe is negated one quarter of a clock before the normal case. For example, when ACS = 00 and CSNT = 1, $\overline{\text{LWE}n}$ is negated one quarter of a clock earlier, as shown in Figure 12-25.

12.4.2.2.3 Relaxed Timing

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

- An additional bus cycle is added between the address and control signals (but only if ACS ≠ 00).
- The number of wait states specified by SCY is doubled, providing up to 30 wait states.
- The extended hold time on read accesses (EHTR) is extended further.
- $\overline{\text{LCS}n}$ signals are negated 1 cycle earlier during writes (but only if ACS ≠ 00).
- $\overline{\text{LWE}}$[0:3] signals are negated 1 cycle earlier during writes.

Figure 12-26 and Figure 12-27 show relaxed timing read and write transactions. The example in Figure 12-27 also shows address and data multiplexing on LAD[0:31] for a pair of writes issued consecutively.

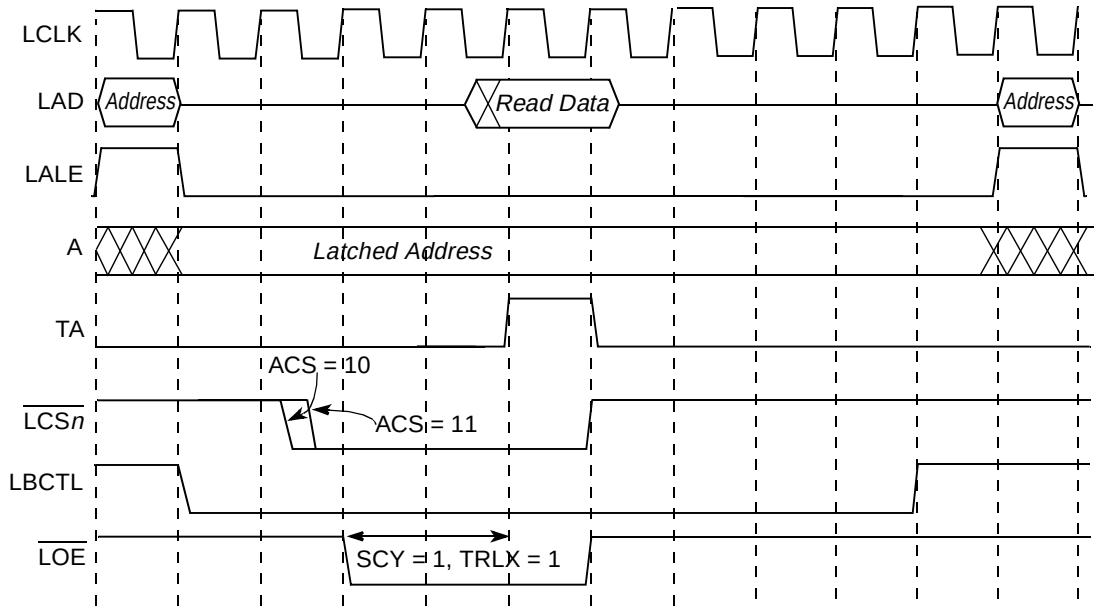


Figure 12-26. GPCM Relaxed Timing Read (XACS = 0, ACS = 1x, SCY = 1, EHTR = 0, TRLX = 1)

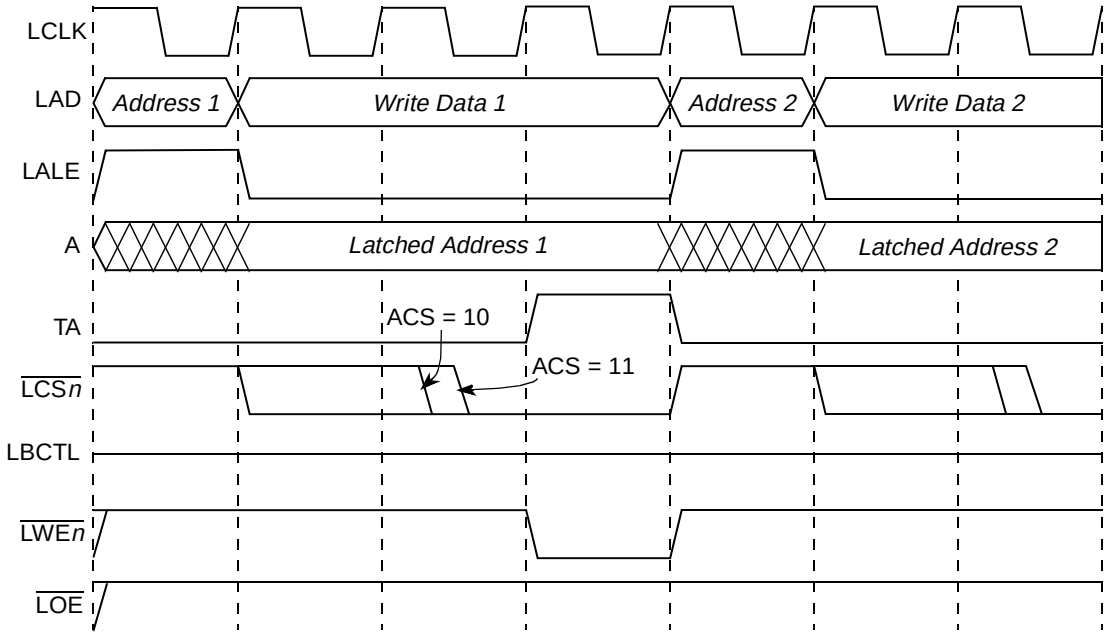


Figure 12-27. GPCM Relaxed Timing Back-to-Back Writes (XACS = 0, ACS = 1x, SCY = 0, CSNT = 0, TRLX = 1)

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

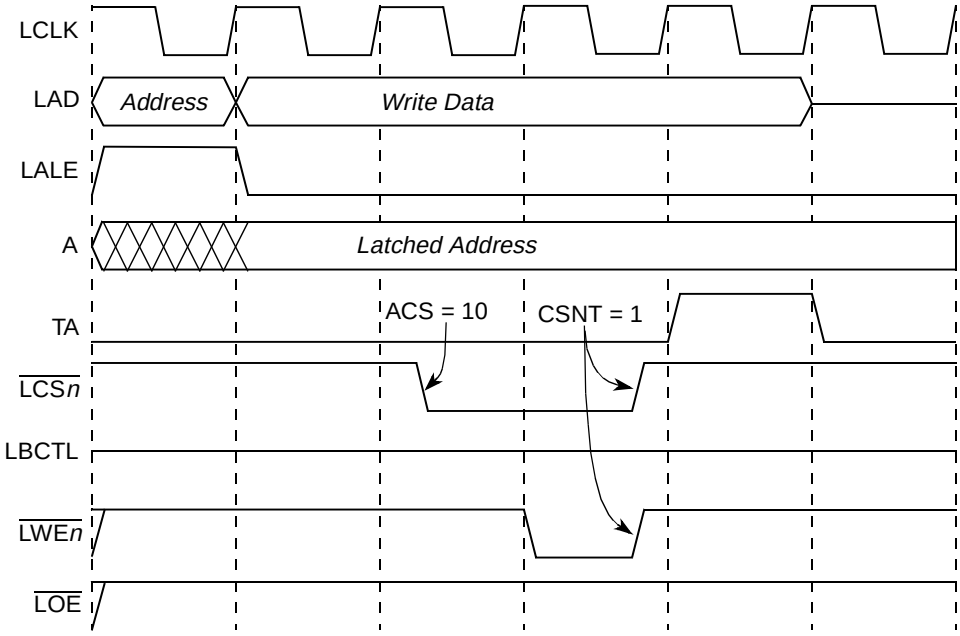


Figure 12-28. GPCM Relaxed Timing Write (XACS = 0, ACS = 10, SCY = 0, CSNT = 1, TRLX = 1)

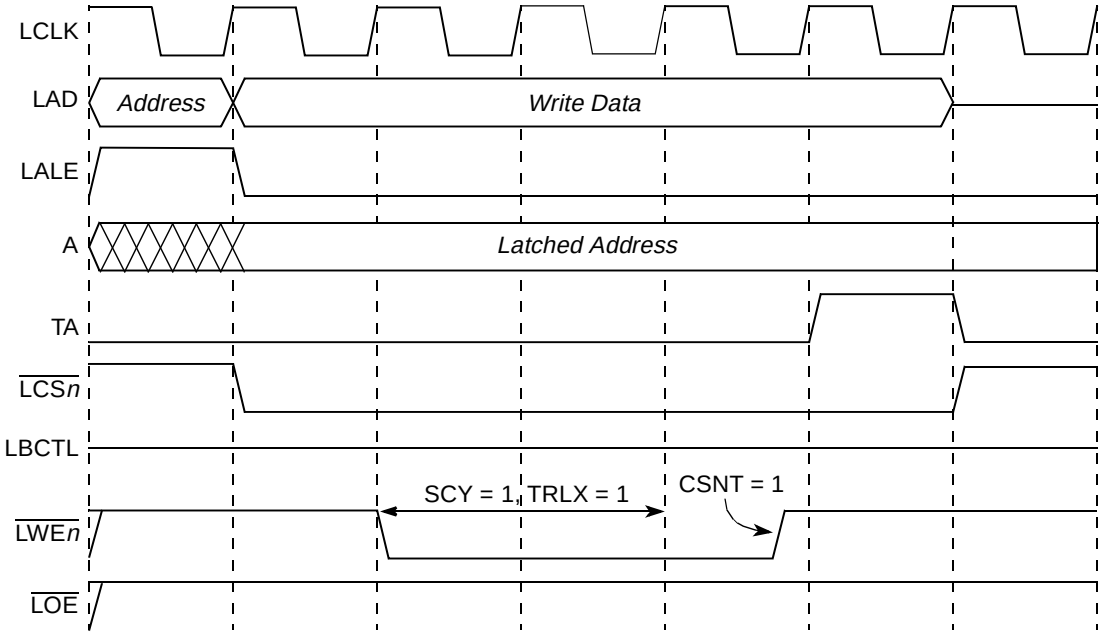


Figure 12-29. GPCM Relaxed Timing Write (XACS = 0, ACS = 00, SCY = 1, CSNT = 1, TRLX = 1)

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

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

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

12.4.2.2.5 Extended Hold Time on Read Accesses

Slow memory devices that take a long time to disable their data bus drivers on read accesses should choose some combination of OR*n*[TRLX,EHTR]. Any access following a read access to the slower memory bank is delayed by the number of clock cycles specified by the configuration of OR*n*[TRLX,EHTR], as described in Section 12.3.1.2.2, "Option Registers (ORn)—GPCM Mode," in addition to any existing bus turnaround cycle. The final bus turnaround cycle is automatically inserted by the LBC for reads, regardless of the setting of OR*n*[EHTR]. Figure 12-30, Figure 12-31, and Figure 12-32 present various GPCM timing examples.

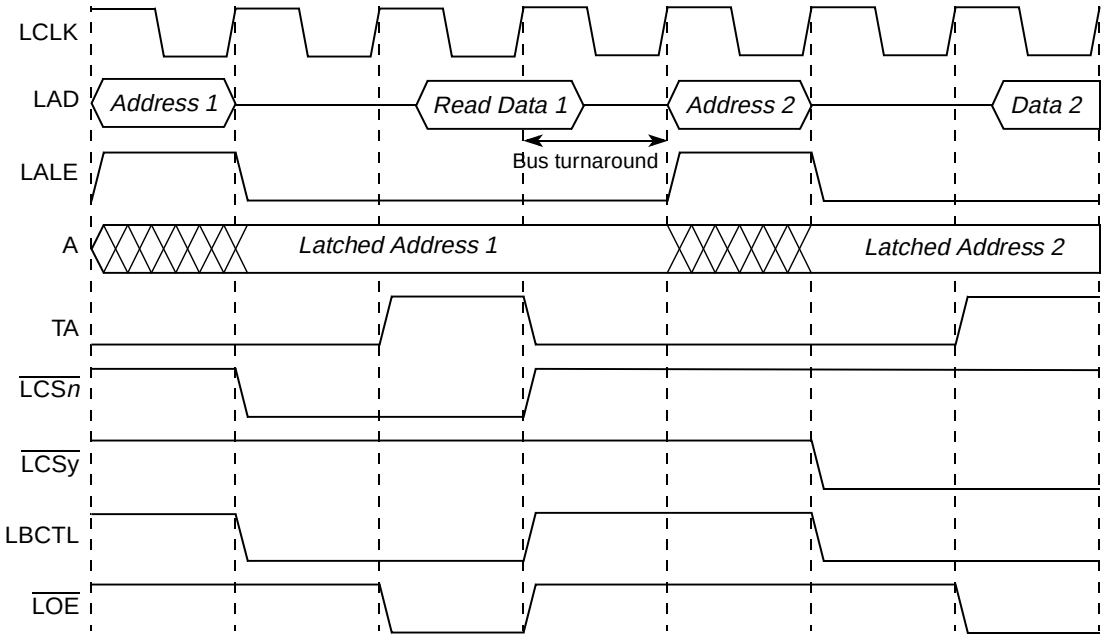


Figure 12-30. GPCM Read Followed by Read (TRLX = 0, EHTR = 0, Fastest Timing)

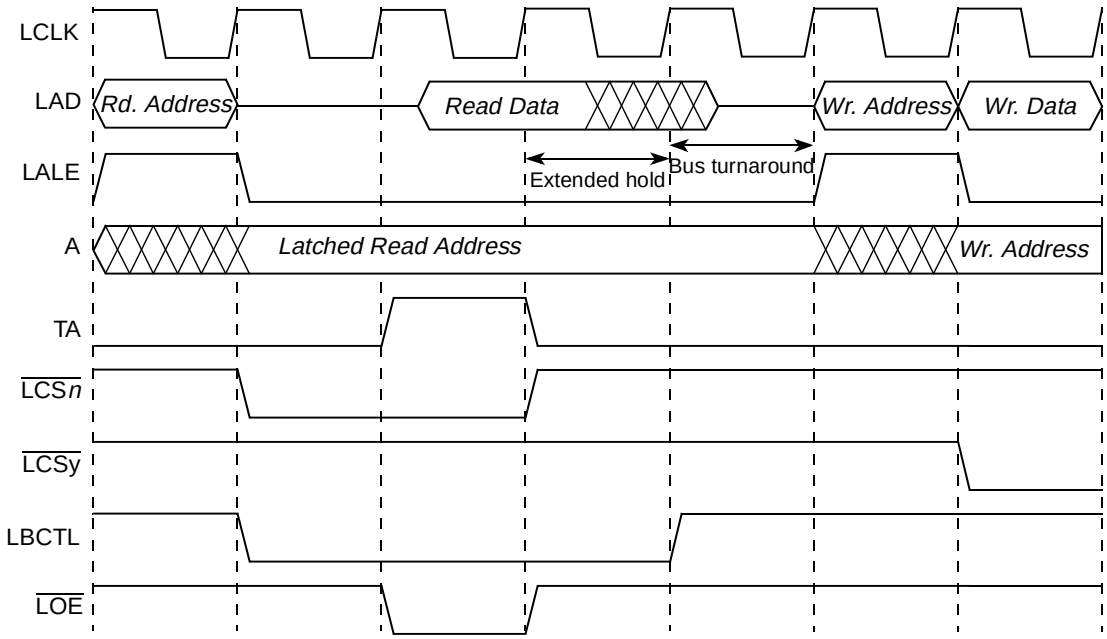


Figure 12-31. GPCM Read Followed by Write (TRLX = 0, EHTR = 1, 1-Cycle Extended Hold Time on Reads)

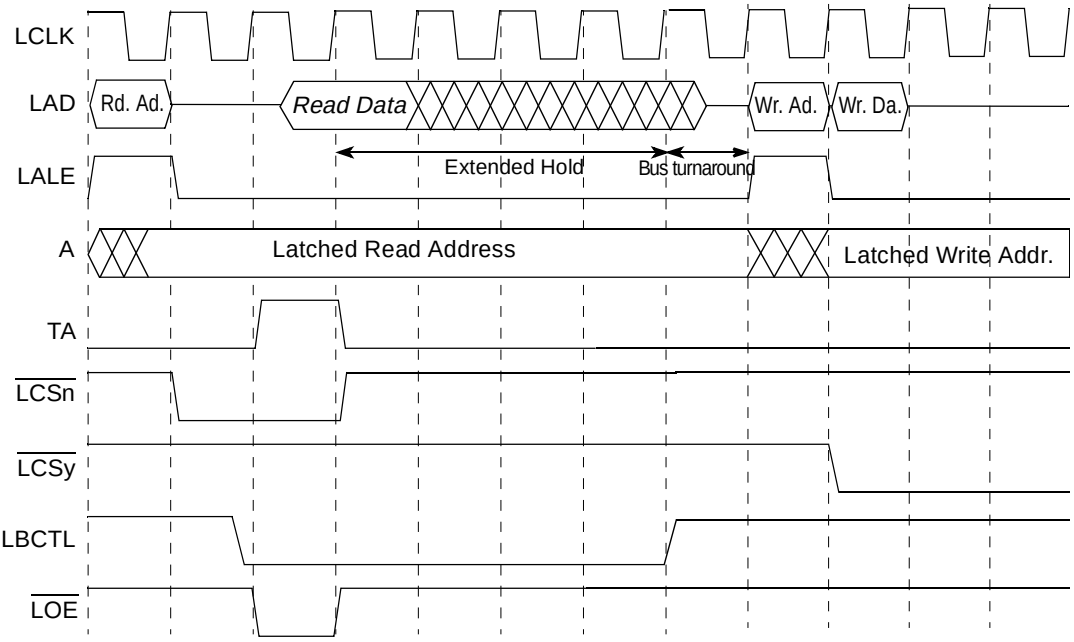


Figure 12-32. GPCM Read Followed by Write(TRLX = 1, EHTR = 0, 4-Cycle Extended Hold Time on Reads)

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

External access termination is supported by the GPCM using the asynchronous $\overline{\text{LGTA}}$ input signal, which is synchronized and sampled internally by the local bus. If, during assertion of $\overline{\text{LCS}n}$, the sampled $\overline{\text{LGTA}}$ signal is asserted, it is converted to an internal generation of transfer acknowledge, which terminates the current GPCM access (regardless of the setting of OR*n*[SETA]). $\overline{\text{LGTA}}$ should be asserted for at least one bus cycle to be effective. Note that because $\overline{\text{LGTA}}$ is synchronized, bus termination occurs two cycles after $\overline{\text{LGTA}}$ assertion, so in the case of a read cycle, the device still must drive data as long as $\overline{\text{LOE}}$ is asserted.

The user selects whether transfer acknowledge is generated internally or externally ($\overline{\text{LGTA}}$) by programming OR*n*[SETA]. Asserting $\overline{\text{LGTA}}$ always terminates an access, even if OR*n*[SETA] = 0 (internal transfer acknowledge generation), but it is the only means by which an access can be terminated if OR*n*[SETA] = 1. The timing of $\overline{\text{LGTA}}$ is illustrated by the example in Figure 12-33.

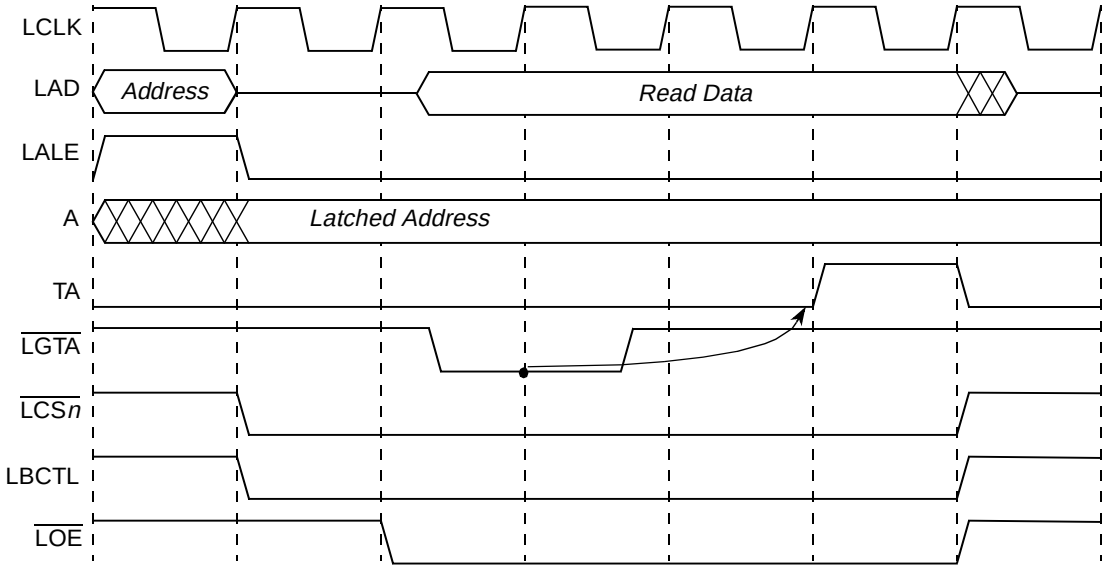


Figure 12-33. External Termination of GPCM Access

12.4.2.4 Boot Chip-Select Operation

Boot chip-select operation allows address decoding for a boot ROM before system initialization. $\overline{\text{LCS0}}$ is the boot chip-select output; its operation differs from other external chip-select outputs after a system reset. When the core begins accessing memory after system reset, $\overline{\text{LCS0}}$ is asserted for every local bus access until BR0 or OR0 is reconfigured.

The boot chip-select also provides a programmable port size, which is configured during reset. The boot chip-select does not provide write protection. $\overline{\text{LCS0}}$ operates this way until the first write to OR0 and it can be used as any other chip-select register after the preferred address range is loaded into BR0. After the first write to OR0, the boot chip-select can be restarted only with a hardware reset. Table 12-25 describes the initial values of the boot bank in the memory controller.

Table 12-25. Boot Bank Field Values After Reset

Register	Field	Setting	Register	Field	Setting
BR0	BA	0000_0000_0000_0000_0	OR0	AM	0000_0000_0000_0000_0
	XBA	00		XAM	00
	PS	From signal during reset.		BCTLD	0
	DECC	00		CSNT	1
	WP	0		ACS	11
	MSEL	000		XACS	1
	ATOM	00		SCY	1111
	V	1		SETA	0
				TRLX	1
				EHTR	1
				EAD	1

12.4.3 SDRAM Machine

The LBC provides an SDRAM interface (machine) for the local bus. The machine provides the control functions and signals for Intel PC133 and JEDEC-compliant SDRAM devices. Each bank can control an SDRAM device on the local bus.

12.4.3.1 Supported SDRAM Configurations

The memory controller supports any SDRAM configuration with the restrictions that all SDRAM devices that reside on the bus should have the same port size and timing parameters (as defined in LSDMR). Figure 12-34 shows an example connection between the LBC and a 32-bit SDRAM device with 12 address lines. Note that address signals A[2:0] of the SDRAM connect directly to LA[27:29], address signal A10 connects to the LBCs dedicated LSDA10 signal, while the remaining address bits (except A10) are latched from LAD[20:26].

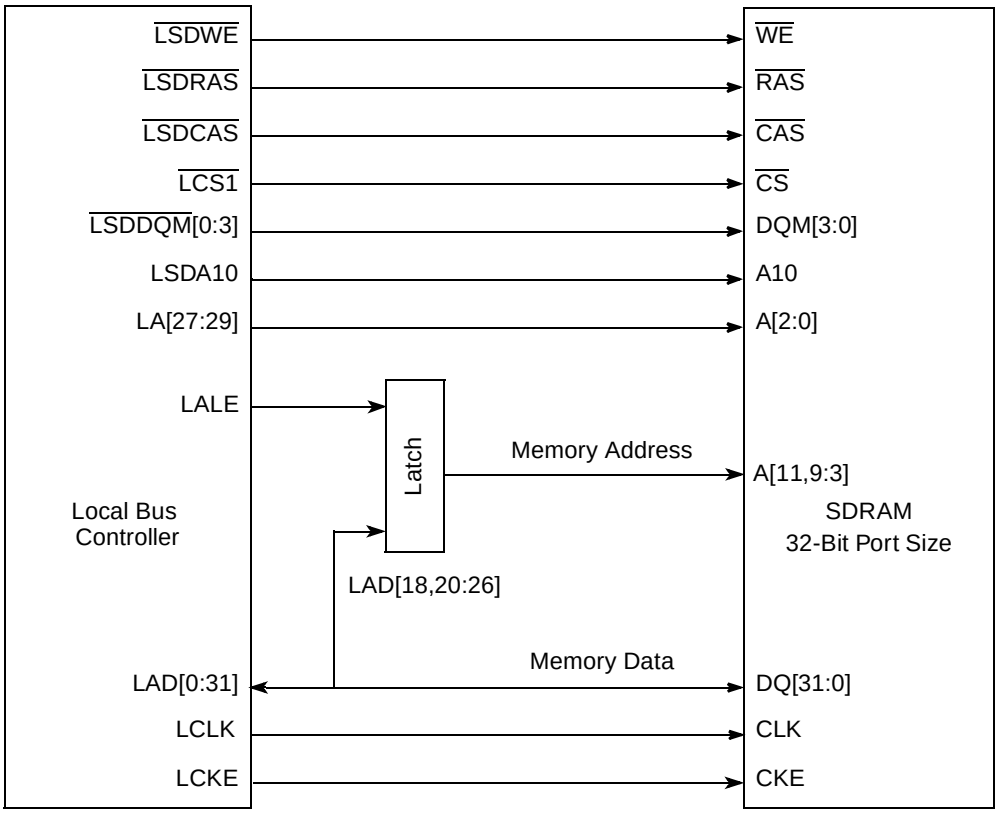


Figure 12-34. Connection to a 32-Bit SDRAM with 12 Address Lines

12.4.3.2 SDRAM Power-On Initialization

Following a system reset, initialization software must set up the programmable parameters in the memory controller banks registers (OR*n*, BR*n*, LSDMR). After all memory parameters are configured, system software should execute the following initialization sequence for each SDRAM device.

- Issue a PRECHARGE-ALL-BANKS command
- Issue eight AUTO-REFRESH commands
- Issue a MODE-SET command to initialize the mode register

The initial commands are executed by setting LSDMR[OP] and accessing the SDRAM with any write that hits the relevant bank. Since the result of any update to the LSDMR must be in effect before accessing the SDRAM with any write, a write to LSDMR should be followed immediately by a read from LSDMR, which must complete prior to an initial write to SDRAM. Further, the first write to SDRAM should be followed immediately by an SDRAM read, which must complete prior to additional LSDMR updates. This enforces a proper ordering between updates to the LSDMR and write accesses to the SDRAM. If the initialization is being done by the e600, this described protocol is guaranteed only if the SDRAM is mapped as cache-inhibited and guarded, as the CCSR memory region containing LSDMR should be. If the initialization is from an external host, said host must ensure completion of LSDMR and SDRAM reads prior to subsequent writes, as described above.

Note that software should ensure that no memory operations begin until this process completes.

NOTE

In general (not only during power-on reset) the LSDMR/SDRAM access ordering protocol should be observed for proper operation.

12.4.3.3 Intel PC133 and JEDEC-Standard SDRAM Interface Commands

The SDRAM machine performs all accesses to SDRAM using Intel PC133 and JEDEC-standard SDRAM interface commands. The SDRAM device samples the command and data inputs on the rising edge of the bus clock. Data at the output of the SDRAM device is sampled on the rising edge of the bus clock.

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

Table 12-26. 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. $\overline{\text{LSDDQM}}$[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 signal from the LBC is negated. This command can be issued at any time. Normal operation can be resumed only by setting LSDMR[OP] = 000, and waiting a minimum of 200 bus cycles before issuing reads or writes to the LBC.

12.4.3.4 Page Hit Checking

The SDRAM machine supports page-mode operation. Each time a page is activated on the SDRAM device, the SDRAM machine stores its address in a page register. The page information, which the user writes to the OR*n* register, is used along with the bank size to compare page bits of the address to the page register each time a bus-cycle access is requested. If a match is found, together with a bank match, the bus cycle is defined as a page hit. An open page is automatically closed by the SDRAM machine if the bus becomes idle, unless OR*n*[PMSEL] = 1.

12.4.3.5 Page Management

The LBC can manage at most four open pages (one page per SDRAM bank) for a single SDRAM device. After a page is opened, it remains open unless:

- The next access is to a page in a different SDRAM device, in which case all open pages on the current device are closed with a PRECHARGE-ALL-BANKS command.
- The next access is to a page in an SDRAM bank that has a different page open on it, in which case the old page is closed with a PRECHARGE-SINGLE-BANK command.
- The current SDRAM device requires refresh services, in which case all open pages on the current device are closed with a PRECHARGE-ALL-BANKS command.
- The bus becomes idle and OR*n*[PMSEL] = 0, in which case all open pages in the current device are closed with a PRECHARGE-ALL-BANKS command.

12.4.3.6 SDRAM Address Multiplexing

The lower address bus bits are connected to the memory device's address port with the memory controller multiplexing the row/column and the internal bank select lines. The position of the bank select lines are set according to LSDMR[BSMA]. Figure 12-35 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 signals specified by LSDMR[BSMA], supporting page-based interleaving. The lsb of the logical row address (A*n* in Figure 12-35) is aligned with the connected lsb of LAD (bits 29, 30, and 31 for port sizes of 32, 16, and 8 bits, respectively).

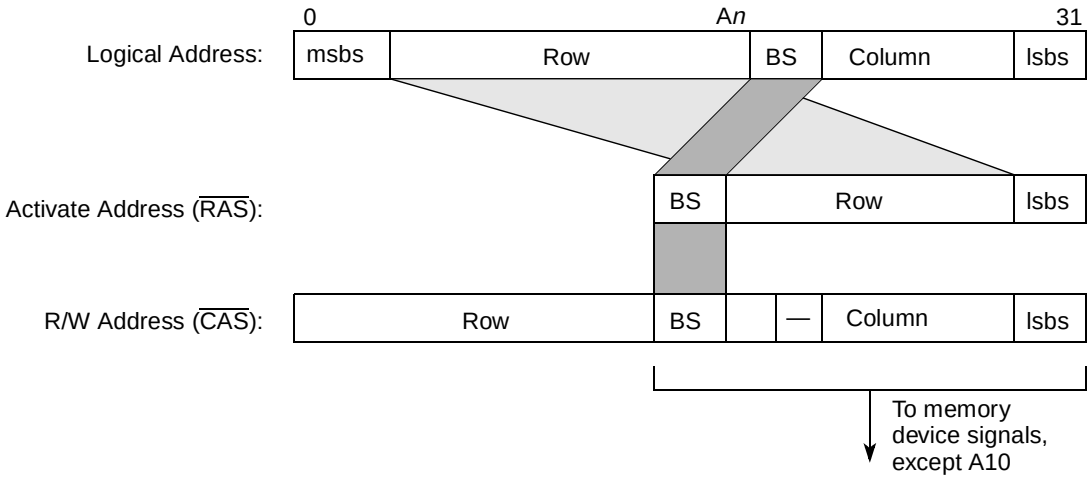


Figure 12-35. SDRAM Address Multiplexing

Note that during normal operation (read/write), a full 32-bit address that includes row and column is generated on LAD[0:31]. However, address/data signal multiplexing implies that the address must be latched by an external latch that is controlled by LALE. All SDRAM device address signals need to be connected to the latched address bits and burst address bits (LA[27:31]) of the LBC, with the exception of A10, which has a dedicated connection on LSDA10. LSDA10 is driven with the appropriate row address bit for SDRAM commands that require A10 to be an address.

12.4.3.7 SDRAM Device-Specific Parameters

The software is responsible for setting correct values for device-specific parameters that can be extracted from the device's data sheet. The values are stored in the OR*n* and LSDMR registers. These parameters include the following:

- Precharge to activate interval (LSDMR[PRETOACT])
- Activate to read/write interval (LSDMR[ACTTORW])
- CAS latency, column address to first data out (LSDMR[CL] and LCRR[ECL])
- Write recovery, last data in to precharge (LSDMR[WRC])
- Refresh recovery interval (LSDMR[RFRC])
- External buffers on the control lines present (LSDMR[BUFCMD] and LCRR[BUFCMDC])

In addition, the LBC hardware ensures a default activate to precharge interval of 10 bus cycles. The following sections describe SDRAM parameters programmed in LSDMR.

12.4.3.7.1 Precharge-to-Activate Interval

The precharge-to-activate interval parameter, controlled by LSDMR[PRETOACT], defines the earliest timing for an ACTIVATE or REFRESH command after a PRECHARGE command to the same SDRAM bank.

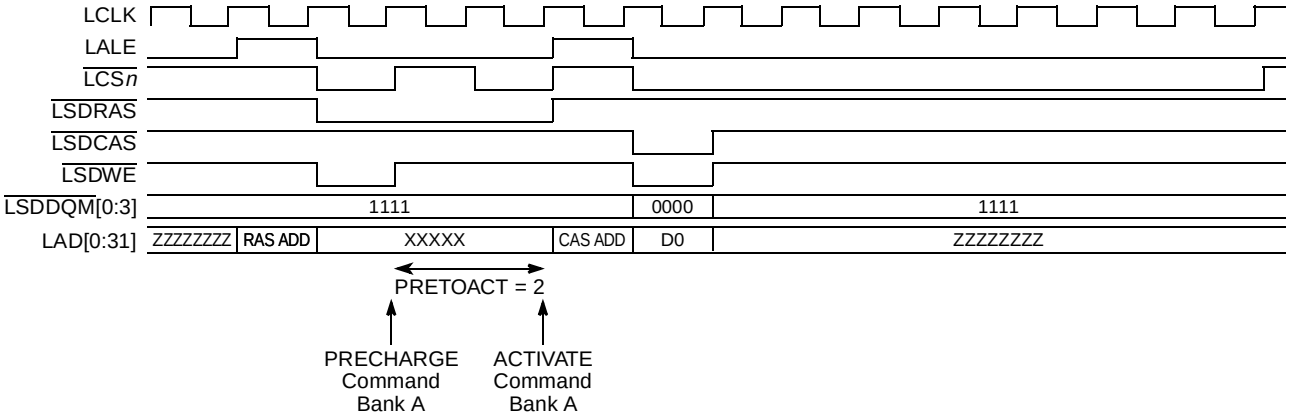


Figure 12-36. PRETOACT = 2 (2 Clock Cycles)

12.4.3.7.2 Activate-to-Read/Write Interval

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

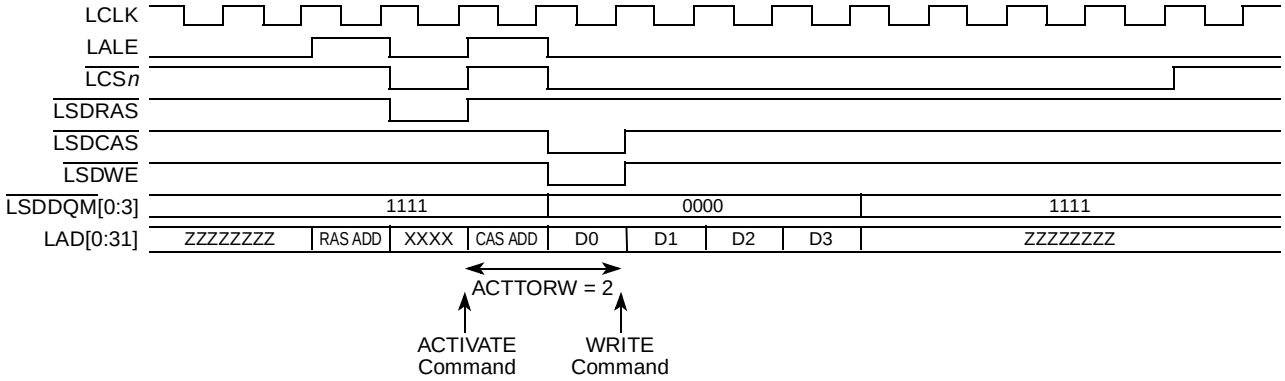


Figure 12-37. ACTTORW = 2 (2 Clock Cycles)

12.4.3.7.3 Column Address to First Data Out—CAS Latency

This parameter, controlled by LSDMR[CL] for latency of 1, 2, or 3 and by LCRR[ECL] for latency of more than 3, defines the timing for first read data after a column address is sampled by the SDRAM.

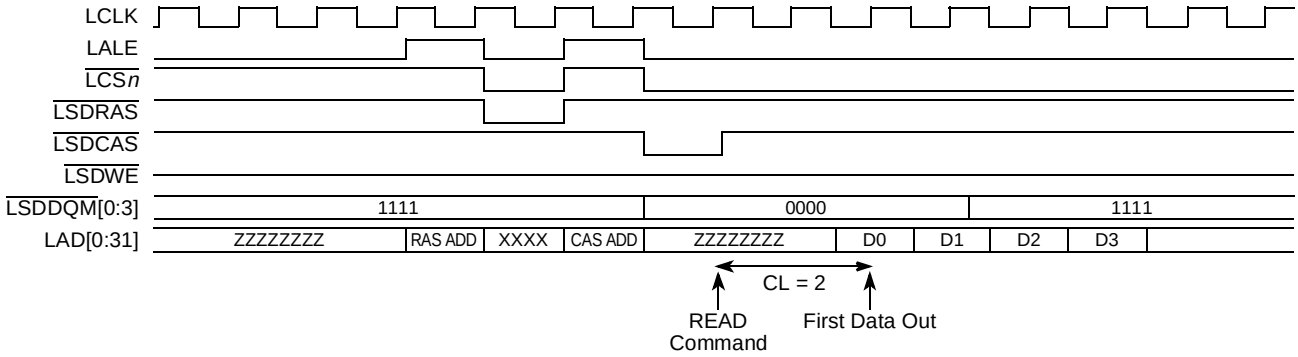


Figure 12-38. CL = 2 (2 Clock Cycles)

12.4.3.7.4 Last Data In to Precharge—Write Recovery

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

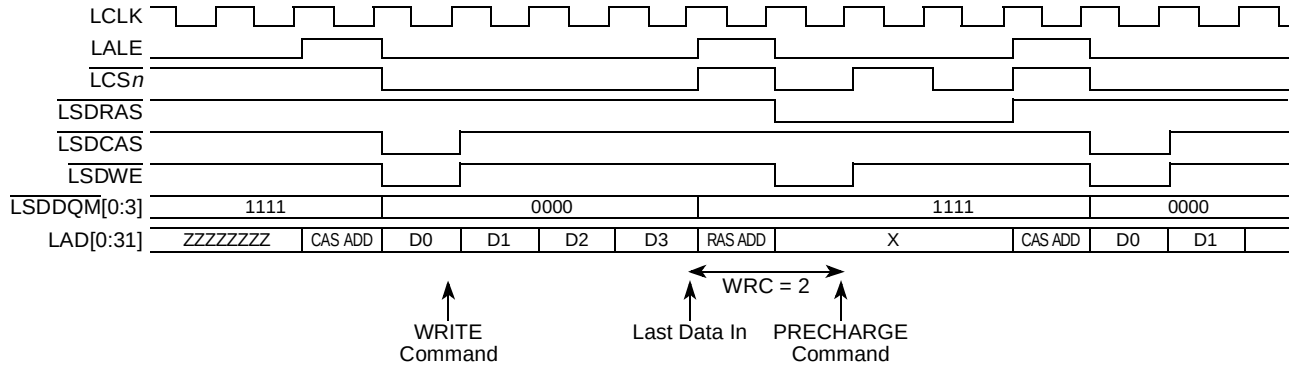


Figure 12-39. WRC = 2 (2 Clock Cycles)

12.4.3.7.5 Refresh Recovery Interval (RFRC)

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

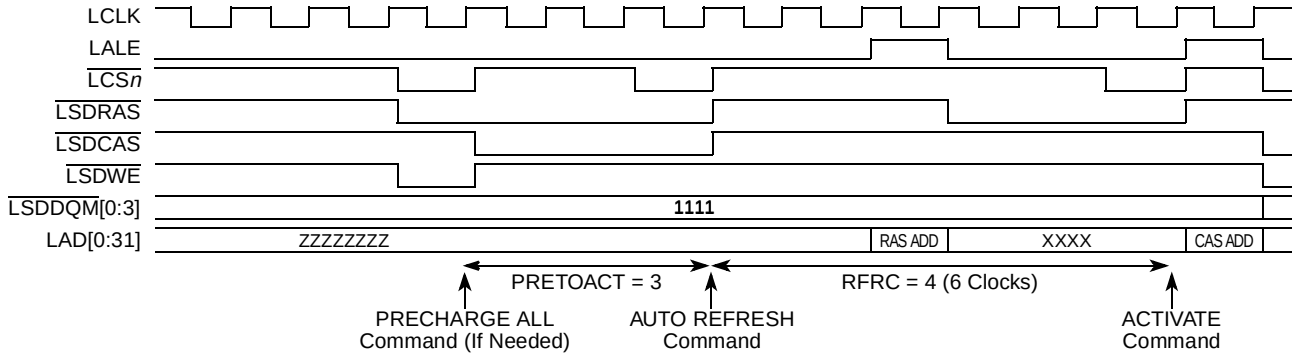


Figure 12-40. RFRC = 4 (6 Clock Cycles)

12.4.3.7.6 External Address and Command Buffers (BUFCMD)

If the additional delay of any buffers placed on the command strobes ($\overline{\text{LSDRAS}}$, $\overline{\text{LSDCAS}}$, $\overline{\text{LSDWE}}$, and LSDA10), is endangering the device setup time, LSDMR[BUFCMD] should be set. Setting this bit causes the memory controller to add LCRR[BUFCMDC] extra bus cycles to the assertion of SDRAM control signals ($\overline{\text{LSDRAS}}$, $\overline{\text{LSDCAS}}$, $\overline{\text{LSDWE}}$, and LSDA10) for each SDRAM command.

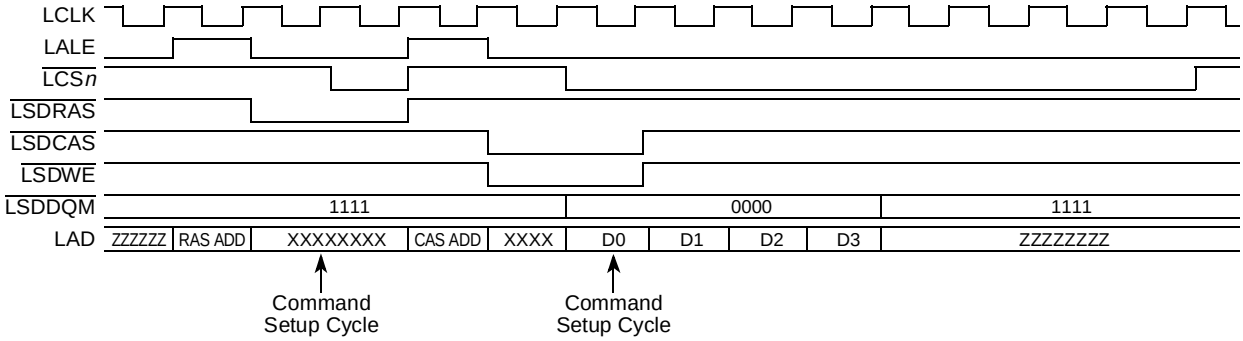


Figure 12-41. BUFCMD = 1, LCRR[BUFCMDC] = 2

12.4.3.8 SDRAM Interface Timing

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

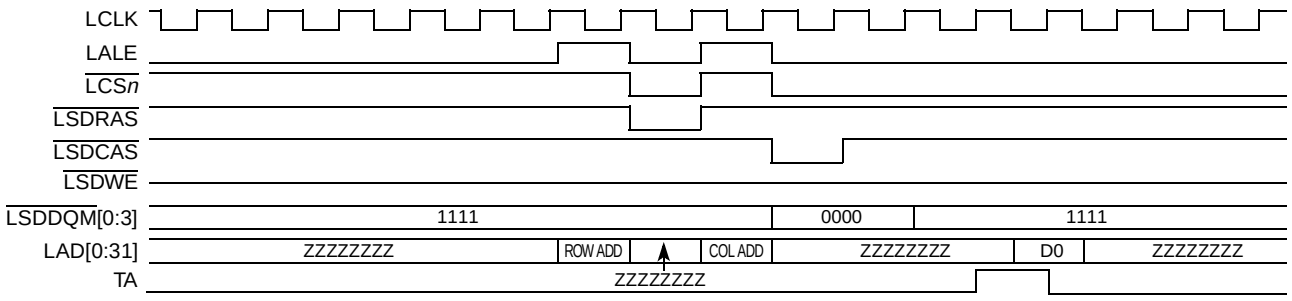


Figure 12-42. SDRAM Single-Beat Read, Page Closed, CL = 3

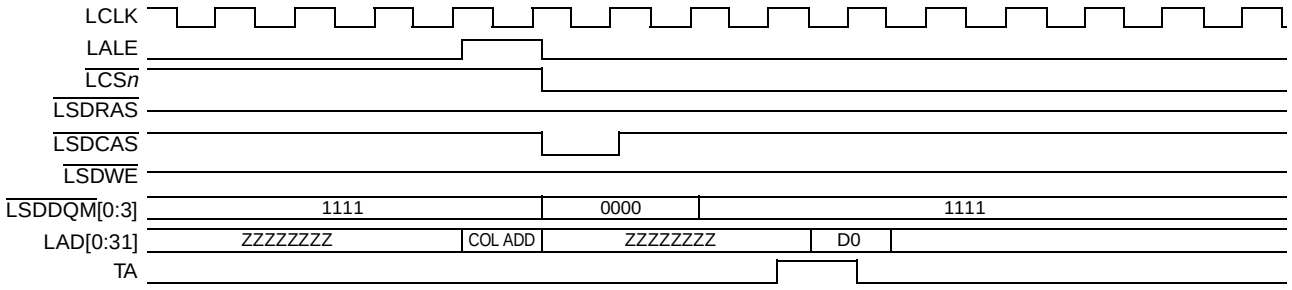


Figure 12-43. SDRAM Single-Beat Read, Page Hit, CL = 3

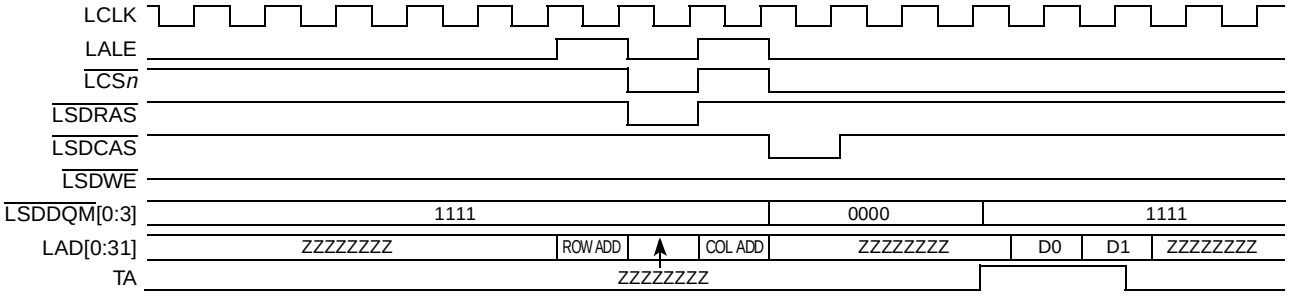


Figure 12-44. SDRAM Two-Beat Burst Read, Page Closed, CL = 3

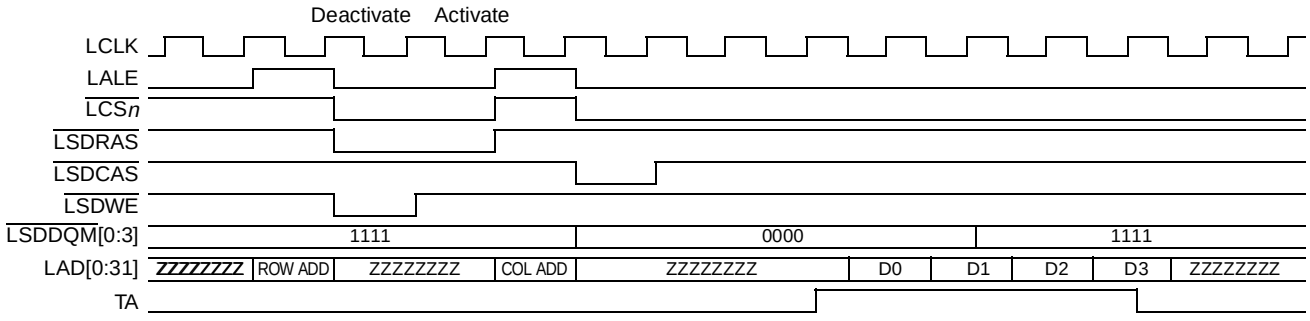


Figure 12-45. SDRAM Four-Beat Burst Read, Page Miss, CL = 3

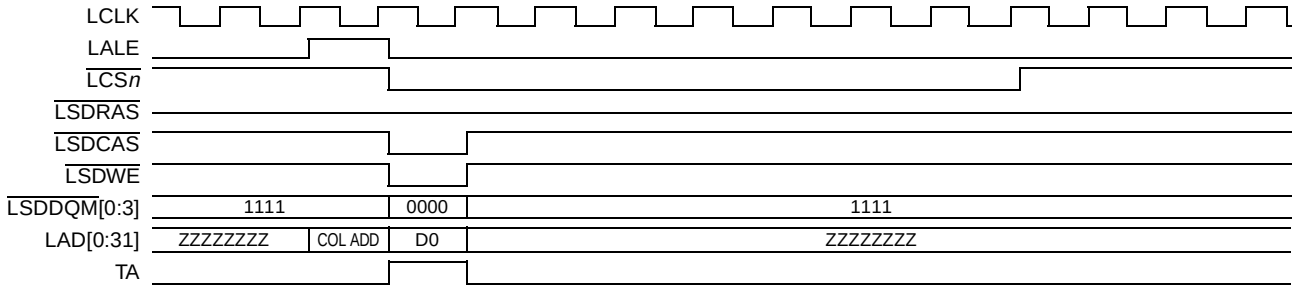


Figure 12-46. SDRAM Single-Beat Write, Page Hit

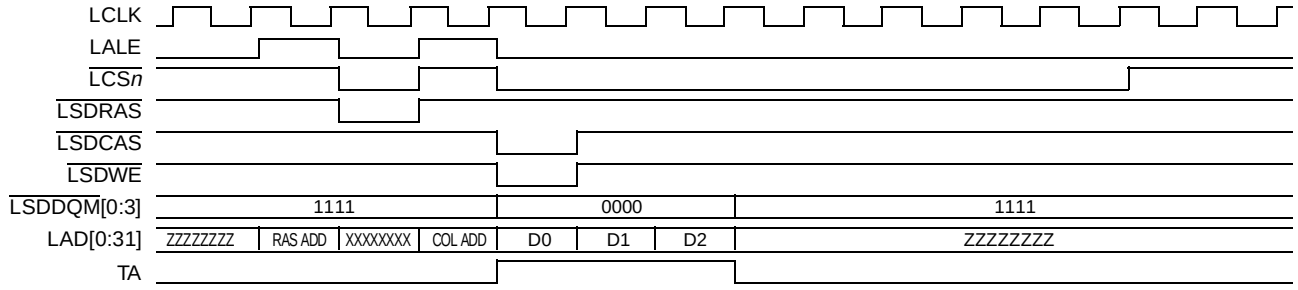


Figure 12-47. SDRAM Three-Beat Write, Page Closed

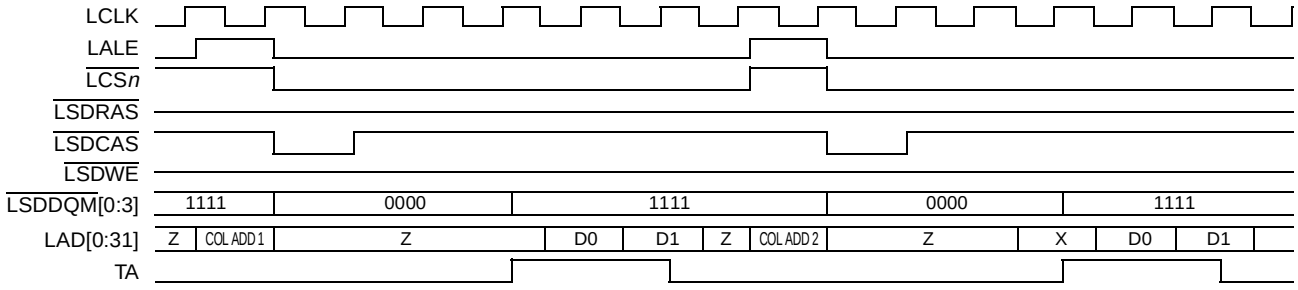


Figure 12-48. SDRAM Read-After-Read Pipelined, Page Hit, CL = 3

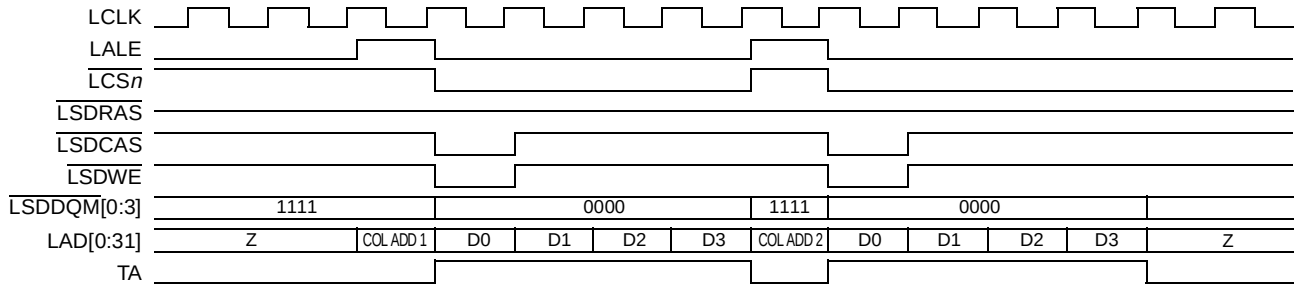


Figure 12-49. SDRAM Write-After-Write Pipelined, Page Hit

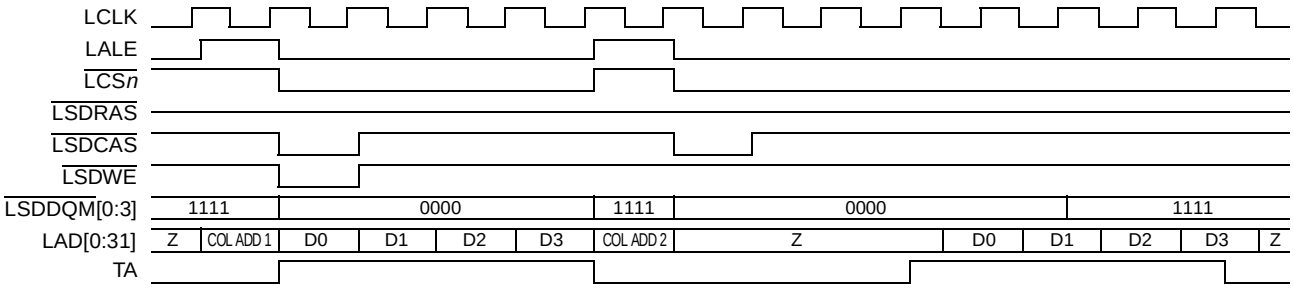


Figure 12-50. SDRAM Read-After-Write Pipelined, Page Hit

12.4.3.9 SDRAM Read/Write Transactions

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

12.4.3.10 SDRAM MODE-SET Command Timing

The LBC transfers mode register data (CAS latency and burst length) stored in the LSDMR register to the SDRAM device by issuing the MODE-SET command, as shown in Figure 12-51. In this case, the latched address carries the mode bits for the command.

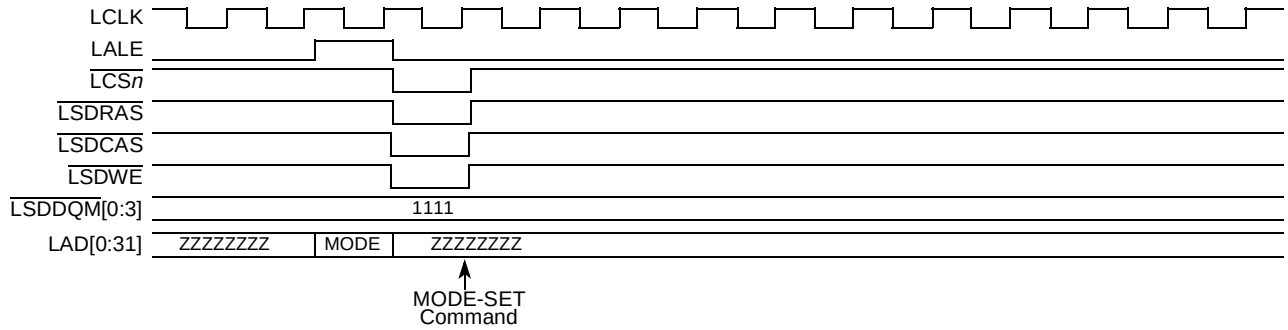


Figure 12-51. SDRAM MODE-SET Command

12.4.3.11 SDRAM Refresh

The memory controller supplies AUTO-REFRESH commands to any connected SDRAM device according to the interval specified in LSRT (and prescaled by MRTPR[PTP]). This represents the time period required between refreshes. The values of LSRT and MRTPR depend on the specific SDRAM devices used and the system clock frequency of the LBC. This value should allow for a potential collision

between memory accesses and refresh cycles. The period of the refresh interval must be greater than the access time to ensure that read and write operations complete successfully.

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

12.4.3.11.1 SDRAM Refresh Timing

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

After a refresh request is granted, the memory controller begins issuing an AUTO-REFRESH command to each device associated with the refresh timer. After a refresh command is issued to an SDRAM device, the memory controller waits for the number of bus clock cycles programmed in the S

DRAM machine's mode register (LSDMR[RFCR]) before issuing any subsequent ACTIVATE command to the same device. To avoid violating SDRAM device timing constraints, the user should ensure that the refresh request interval, defined by LSRT and MRTPR, is greater than the refresh recovery interval, defined by LSDMR[RFCR].

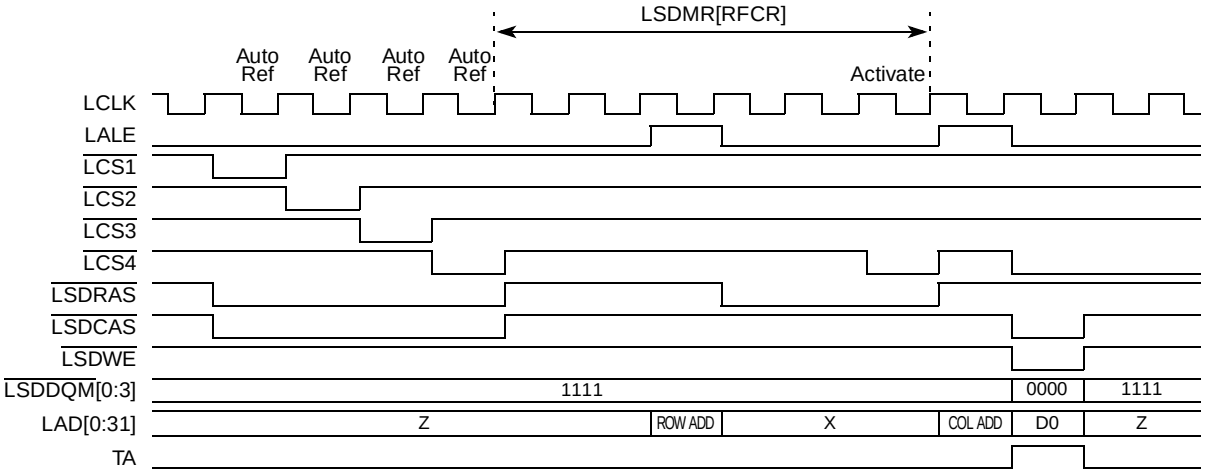


Figure 12-52. SDRAM Bank-Staggered Auto-Refresh Timing

12.4.4 User-Programmable Machines (UPMs)

UPMs are flexible interfaces that connect to a wide range of memory devices. At the heart of each UPM is an internal RAM array that specifies the logical value driven on the external memory control signals ($\overline{\text{LCS}}n$, $\overline{\text{LBS}}$[0:3], and LGPL[0:5]) for a given clock cycle. Each word in the RAM array provides bits that allow a memory access to be controlled with a resolution of up to one quarter of the external bus clock period on the byte select and chip select lines.

NOTE

If the LGPL4/$\overline{\text{LGTA}}$/LUPWAIT/LPBSE signal is used as both an input and an output, a weak pull-up is required. Refer to the hardware specification for details regarding termination options.

Figure 12-53 shows the basic operation of each UPM.

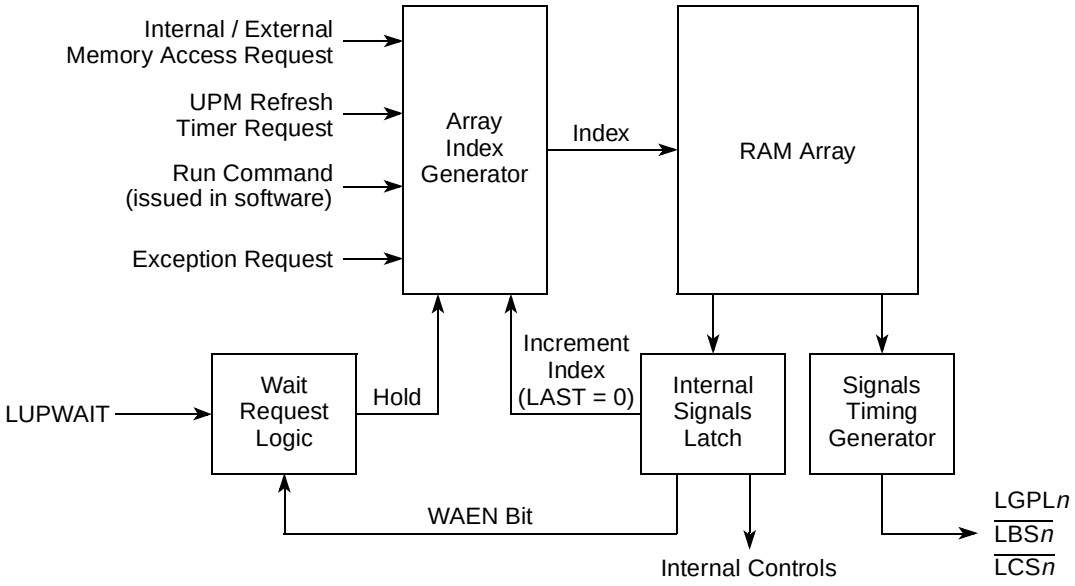


Figure 12-53. User-Programmable Machine Functional Block Diagram

The following events initiate a UPM cycle:

- Any internal device requests an external memory access to an address space mapped to a chip-select serviced by the UPM
- A UPM refresh timer expires and requests a transaction, such as a DRAM refresh
- A bus monitor time-out error during a normal UPM cycle redirects the UPM to execute an exception sequence

The RAM array contains 64 words of 32-bits each. The signal timing generator loads the RAM word from the RAM array to drive the general-purpose lines, byte-selects, and chip-selects. If the UPM reads a RAM word with WAEN set, the external LUPWAIT signal is sampled and synchronized by the memory controller and the current request is frozen.

12.4.4.1 UPM Requests

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

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

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

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

Figure 12-54 and Table 12-27 show the start addresses of these patterns in the UPM RAM, according to cycle type. RUN commands (M*x*MR[OP] = 11), however, can initiate patterns starting at any of the 64 UPM RAM words.

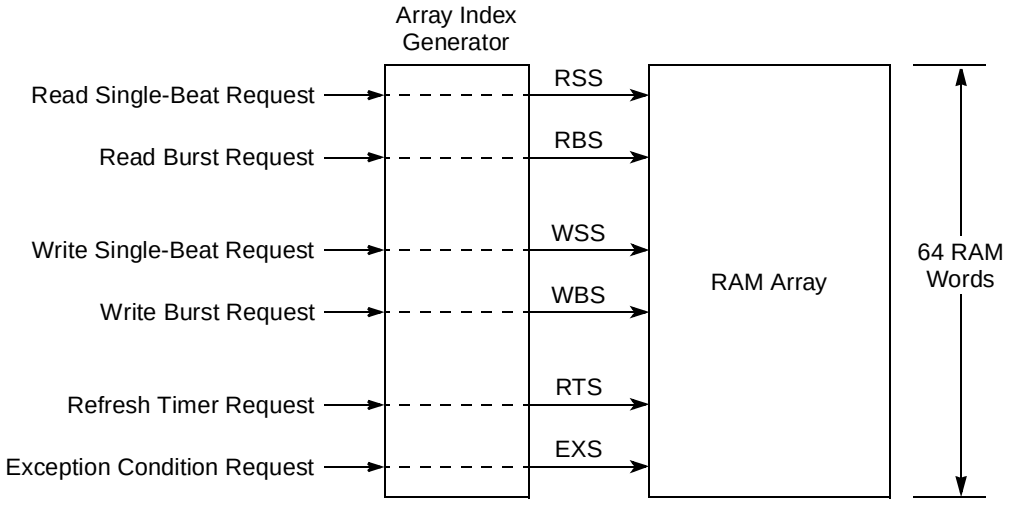


Figure 12-54. RAM Array Indexing

Table 12-27. UPM Routines Start Addresses

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

12.4.4.1.1 Memory Access Requests

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

The UPM supports two types of memory reads and writes:

- A single-beat transfer transfers one operand consisting of up to a single word (dependent on port size). A single-beat cycle starts with one transfer start and ends with one transfer acknowledge.
- A burst transfer transfers exactly 4 double words regardless of port size. For 32-bit accesses, the burst cycle starts with one transfer start but ends after eight transfer acknowledges, whereas an 8-bit device requires 32 transfer acknowledges.

The user must ensure that patterns for single-beat transfers contain one, and only one, transfer acknowledge (UTA bit in RAM word set high) and for a burst transfer, contain the exact number of transfer acknowledges required.

Any transfers that do not naturally fit single or burst transfers are synthesized as a series of single transfers. These accesses are treated by the UPM as back-to-back, single-beat transfers. Burst transfers can also be inhibited by setting OR*n*[BI]. Burst performance can be achieved by ensuring that UPM transactions are 32-byte aligned with a transaction size being some multiple of 32-bytes, which is a natural fit for cache-line transfers, for example.

12.4.4.1.2 UPM Refresh Timer Requests

Each UPM contains a refresh timer that can be programmed to generate refresh service requests of a particular pattern in the RAM array. Figure 12-55 shows the clock division hardware associated with memory refresh timer request generation. The UPM refresh timer register (LURT) defines the period for the timers associated with all three UPMs.

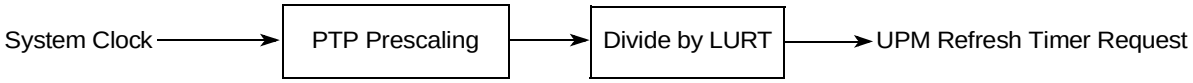


Figure 12-55. Memory Refresh Timer Request Block Diagram

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

Note that the UPM refresh timer request should not be used in a system with SDRAM refresh enabled. The system designer must choose to use either SDRAM refresh or UPM refresh. Using both may result in missing refresh periods to memory.

12.4.4.1.3 Software Requests—RUN Command

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

For these special cycles, the user creates a special RAM pattern that can be stored in any unused areas in the UPM RAM. Then a RUN command is used to run the cycle. The UPM runs the pattern beginning at the specified RAM location until it encounters a RAM word with its LAST bit set. The RUN command is issued by setting M*x*MR[OP] = 11 and accessing UPM*n* memory region with any write transaction that hits the corresponding UPM machine. M*x*MR[MAD] determines the starting address in the RAM array for the pattern.

Note that transfer acknowledges (UTA bit in the RAM word) are ignored for software (RUN command) requests, and hence the LAD signals remain high-impedance unless the normal initial LALE occurs or the RUN pattern causes assertion of LALE to occur on changes to the RAM word AMX field.

12.4.4.1.4 Exception Requests

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

12.4.4.2 Programming the UPMs

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

1. Set up BR*n* and OR*n* registers.
2. Write patterns into the RAM array.
3. Program MRTPR, LURT and MAMR[RFEN] if refresh is required.
4. Program M*x*MR.

Patterns are written to the RAM array by setting M*x*MR[OP] = 01 and accessing the UPM with any write transaction that hits the relevant chip select. The entire array is thus programmed by an alternating series of writes: to MDR (RAM word to be written) each time followed by a read from MDR and then followed by a (dummy) write transaction to the relevant UPM assigned bank. A read from MDR is required to ensure that the MDR update has occurred prior to the (dummy) write transaction.

RAM array contents may also be read for debug purposes, for example, by alternating dummy read transactions, each time followed by reads of MDR (when M*x*MR[OP] = 10).

NOTE

M*x*MR/MDR registers should not be updated while dummy read/write accesses are still in progress. Dummy transaction completion is indicated by incremented M*x*MR[MAD]. In order to enforce proper ordering between updates to the M*x*MR register and the dummy accesses to the UPM memory region, two rules must be followed:

1. Since the result of any update to the M*x*MR/MDR register must be in effect before the dummy read or write to the UPM region, a write to M*x*MR/MDR should be followed immediately by a read of M*x*MR/MDR.
2. The UPM memory region should have the same MMU settings as the memory region containing the M*x*MR configuration register; both should be mapped by the MMU as cache-inhibited and guarded. This prevents the e600 core from re-ordering a read of the UPM memory around the read of M*x*MR. Once the programming of the UPM array is complete the MMU setting for the associated address range can be set to the proper mode for normal operation, such as cacheable and copyback.

12.4.4.2.1 UPM Programming Example (Two Sequential Writes to the RAM Array)

The following example further illustrates the steps required to perform two writes to the RAM array at non-sequential addresses assuming that the relevant BR*n* and OR*n* registers have been previously setup.

1. Program M*x*MR for the first write (with desired RAM array address).
2. Write pattern/data to MDR to ensure that the M*x*MR has already been updated with the desired configuration.
3. Read MDR to ensure that the MDR has already been updated with the desired pattern. (Or, read M*x*MR if step 2 is not performed.)
4. Preform a dummy write transaction. (Write transaction can now be performed.)
5. Read/check M*x*MR[MAD]. If incremented, then the previous dummy write transaction is completed; proceed to step 6. Repeat step 5 until incremented.
6. Program M*x*MR for the second write with the desired RAM array address.
7. Write pattern/data to MDR to ensure that the M*x*MR has already been updated with the desired configuration.
8. Read MDR to ensure that the MDR has already been updated with the desired pattern.
9. Perform a dummy write transaction.(Write transaction can now be performed.)
10. Read/check M*x*MR[MAD]. If incremented, then the previous dummy write transaction is completed.

Note that if step 1 (or 6) and 2 (or 7) are reversed, then step 3 (or 8) is replaced by the following:

- Read M*x*MR to ensure that the M*x*MR has already been updated with the desired configuration.

12.4.4.2.2 UPM Programming Example (Two Sequential Reads from the RAM Array)

RAM array contents may also be read for debug purposes, for example, by alternating dummy read transactions, each time followed by reads of MDR (when M*x*MR[OP] = 0b10). The following example further illustrates the steps required to perform two reads from the RAM array at non-sequential addresses assuming that the relevant BR*n* and OR*n* registers have been previously setup.

1. Program M*x*MR for the first read with the desired RAM array address.
2. Read M*x*MR to ensure that the M*x*MR has already been updated with the desired configuration, such as RAM array address.
3. Perform a dummy read transaction.(Read transaction can now be performed.)
4. Read/check M*x*MR[MAD]. If incremented, then the previous dummy read transaction is completed; proceed to step 5. Repeat step 4 until incremented.
5. Read MDR.
6. Program M*x*MR for the second read with the desired RAM array address.
7. Read M*x*MR to ensure that the M*x*MR has already been updated with the desired configuration, such as RAM array address.
8. Perform a dummy read transaction.(Read transaction can now be performed.)

9. Read/check M*x*MR[MAD]. If incremented, then the previous dummy read transaction is completed; proceed to step 10. Repeat step 9 until incremented.
10. Read MDR.

12.4.4.3 UPM Signal Timing

RAM word fields specify the value of the various external signals at a granularity of up to four values for each bus clock cycle. The signal timing generator causes external signals to behave according to timing specified in the current RAM word. Each bit in the RAM word relating to $\overline{\text{LCS}n}$ and $\overline{\text{LBS}}$ timing specifies the value of the corresponding external signal at each quarter phase of the bus clock.

The division of UPM bus cycles into phases is shown in Figure 12-56.

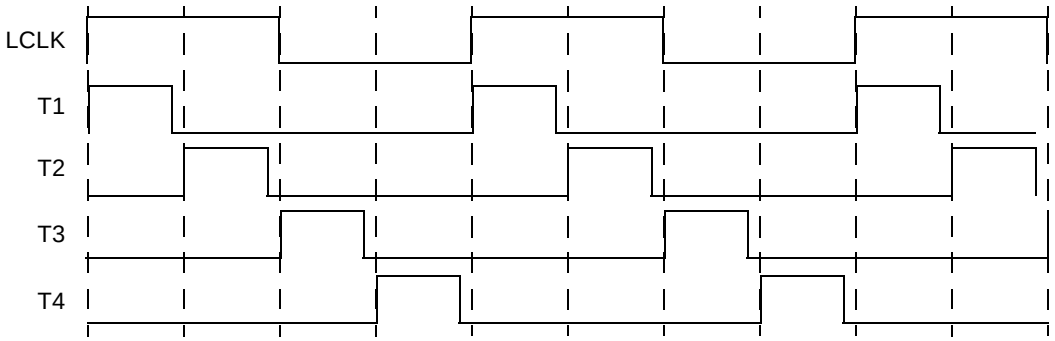


Figure 12-56. UPM Clock Scheme

12.4.4.4 RAM Array

The RAM array for each UPM is 64 locations deep and 32 bits wide, as shown in Figure 12-57. The signals at the bottom of the figure are UPM outputs. The selected $\overline{\text{LCS}n}$ is for the bank that matches the current address. The selected $\overline{\text{LBS}}$ is for the byte lanes read or written by the access.

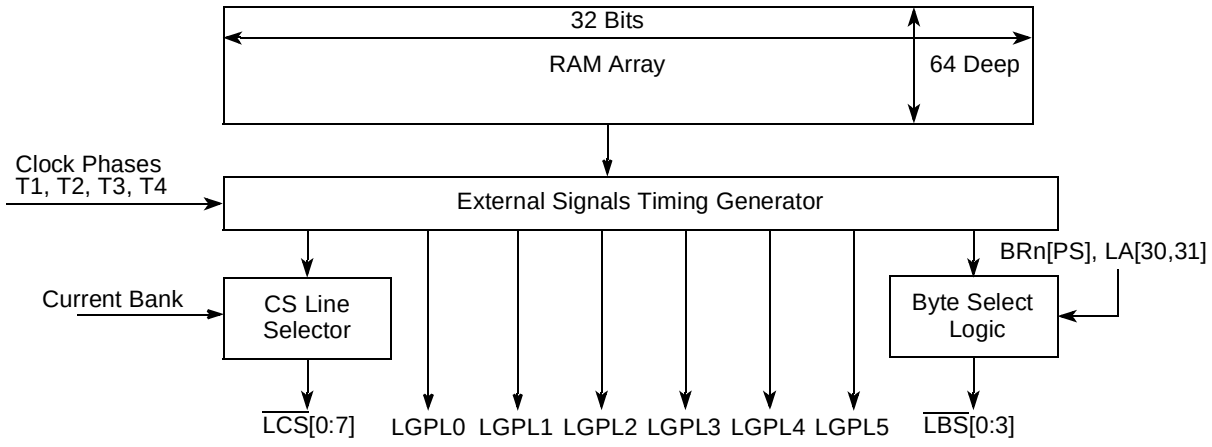


Figure 12-57. RAM Array and Signal Generation

12.4.4.4.1 RAM Words

The RAM word is a 32-bit microinstruction stored in one of 64 locations in the RAM array. It specifies timing for external signals controlled by the UPM. Figure 12-58 shows the RAM word fields. 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.

Access: Read/Write

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

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

Figure 12-58. RAM Word Field Descriptions

Table 12-28 describes RAM word fields.

Table 12-28. RAM Word Field Descriptions

Bits	Name	Description
0	CST1	Chip select timing 1. Defines the state (0 or 1) of $\overline{\text{LCS}}$*n* during bus clock quarter phase 1.
1	CST2	Chip select timing 2. Defines the state (0 or 1) of $\overline{\text{LCS}}$*n* during bus clock quarter phase 2.
2	CST3	Chip select timing 3. Defines the state (0 or 1) of $\overline{\text{LCS}}$*n* during bus clock quarter phase 3.
3	CST4	Chip select timing 4. Defines the state (0 or 1) of $\overline{\text{LCS}}$*n* during bus clock quarter phase 4.
4	BST1	Byte select timing 1. Defines the state (0 or 1) of $\overline{\text{LBS}}$ during bus clock quarter phase 1.
5	BST2	Byte select timing 2:. Defines the state (0 or 1) of $\overline{\text{LBS}}$ during bus clock quarter phase 2.
6	BST3	Byte select timing 3. Defines the state (0 or 1) of $\overline{\text{LBS}}$ during bus clock quarter phase 3.
7	BST4	Byte select timing 4. Defines the state (0 or 1) of $\overline{\text{LBS}}$ during bus clock quarter phase 4.
8–9	G0L	General-purpose line 0 lower. Defines the state of LGPL0 during the bus clock quarter phases 1 and 2 (first half phase). 00 Value defined by M*x*MR[G0CL] 01 Reserved 10 0 11 1
10–11	G0H	General-purpose line 0 higher. Defines the state of LGPL0 during the bus clock quarter phases 3 and 4 (second half phase). 00 Value defined by M*x*MR[G0CL] 01 Reserved 10 0 11 1
12	G1T1	General-purpose line 1 timing 1. Defines the state (0 or 1) of LGPL1 during bus clock quarter phases 1 and 2 (first half phase).

Table 12-28. RAM Word Field Descriptions (continued)

Bits	Name	Description
13	G1T3	General-purpose line 1 timing 3. Defines the state (0 or 1) of LGPL1 during bus clock quarter phases 3 and 4 (second half phase)
14	G2T1	General-purpose line 2 timing 1. Defines state (0 or 1) of LGPL2 during bus clock quarter phases 1 and 2 (first half phase).
15	G2T3	General-purpose line 2 timing 3. Defines the state (0 or 1) of LGPL2 during bus clock quarter phases 3 and 4 (second half phase).
16	G3T1	General-purpose line 3 timing 1. Defines the state (0 or 1) of LGPL3 during bus clock quarter phases 1 and 2 (first half phase).
17	G3T3	General-purpose line 3 timing 3. Defines the state (0 or 1) of LGPL3 during bus clock quarter phases 3 and 4 (second half phase).
18	G4T1/DLT3	General-purpose line 4 timing 1/delay time 3. The function of this bit is determined by M*x*MR[GPL4]. If M*x*MR[GPL4] = 0 and LGPL4/LUPWAIT signal 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 signal functions as an output (LGPL4), G4T3/WAEN defines the state (0 or 1) of LGPL4 during bus clock quarter phases 3 and 4 (second half phase). If M*x*MR[GPL4] = 1 and LGPL4/LUPWAIT functions as an input (LUPWAIT), G4T3/WAEN is used to enable the wait mechanism: 0 LUPWAIT detection is disabled. 1 LUPWAIT is enabled. If LUPWAIT is detected as being asserted, a freeze in the external signals logical values occurs until LUPWAIT is detected as being negated.
20	G5T1	General-purpose line 5 timing 1. Defines the state (0 or 1) of LGPL5 during bus clock quarter phases 1 and 2 (first half phase).
21	G5T3	General-purpose line 5 timing 3. Defines the state (0 or 1) of LGPL5 during bus clock quarter phases 3 and 4 (second half phase).
22–23	REDO	Redo current RAM word. Defines the number of times to execute the current RAM word. 00 Once (normal operation) 01 Twice 10 Three times 11 Four times
24	LOOP	Loop start/end. The first RAM word in the RAM array where LOOP is 1 is recognized as the loop start word. The next RAM word where LOOP is 1 is the loop end word. RAM words between, and including the start and end words, are defined as part of the loop. The number of times the UPM executes this loop is defined in the corresponding loop fields of the M*x*MR. 0 The current RAM word is not the loop start word or loop end word. 1 The current RAM word is the start or end of a loop. **Note:** AMX must not be changed from its previous value in any RAM word which begins a loop.

Table 12-28. 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 $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ 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. **Note:** AMX must not be changed from its previous value in any RAM word which begins a loop.
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 $\overline{\text{RAS}}$ precharge time. TODT turns the timer on to prevent another UPM access to the same bank until the timer expires.The disable timer period is determined in M*x*MR[DS*n*]. The disable timer does not affect memory accesses to different banks. Note that TODT must be set together with LAST, otherwise it is ignored. 0 The disable timer is turned off. 1 The disable timer for the current bank is activated preventing a new access to the same bank (when controlled by the UPMs) until the disable timer expires. For example, precharge time.
31	LAST	Last word. When LAST is read in a RAM word, the current UPM pattern terminates and control signal timing set in the RAM word is applied to the current (and last) cycle. However, if the disable timer is activated and the next access is to the same bank, execution of the next UPM pattern is held off and the control signal values specified in the last word are extended in duration for the number of clock cycles specified in M*x*MR[DS*n*]. 0 The UPM continues executing RAM words. 1 Indicates the last RAM word in the program. The service to the UPM request is done after this cycle concludes.

12.4.4.4.2 Chip-Select Signal Timing (CST*n*)

If BR*n*[MSEL] of the accessed bank selects a UPM on the currently requested cycle, the UPM manipulates the $\overline{\text{LCS}}n$ for that bank with timing as specified in the UPM RAM word CST*n* fields. The selected UPM

affects only the assertion and negation of the appropriate $\overline{\text{LCS}n}$ signal. The state of the selected $\overline{\text{LCS}n}$ signal of the corresponding bank depends on the value of each CST*n* bit. Figure 12-59 shows how UPMs control $\overline{\text{LCS}n}$ signals.

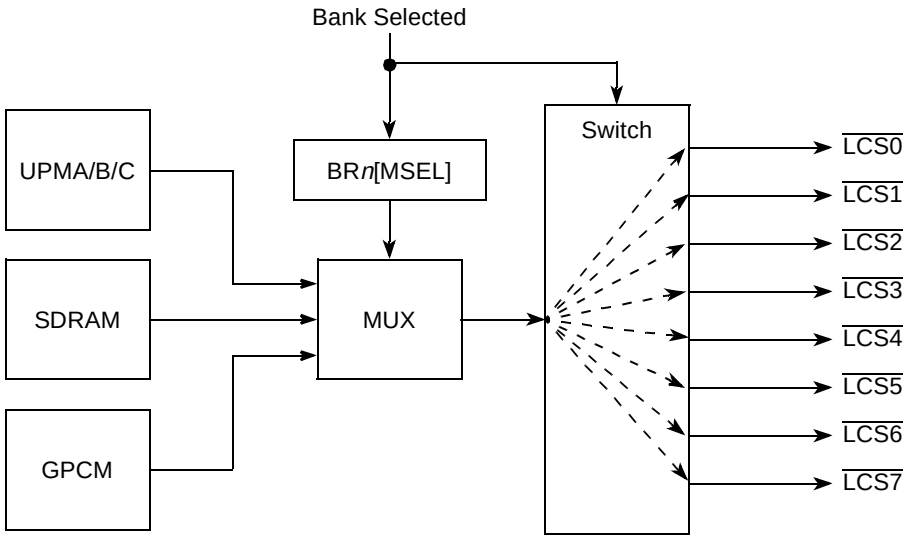


Figure 12-59. $\overline{\text{LCS}n}$ Signal Selection

12.4.4.4.3 Byte Select Signal Timing (BST*n*)

If BR*n*[MSEL] of the accessed memory bank selects a UPM on the currently requested cycle, the selected UPM affects the assertion and negation of the appropriate $\overline{\text{LBS}}$[0:3] signal. The timing of all four byte-select signals is specified in the RAM word. However, $\overline{\text{LBS}}$[0:3] are also controlled by the port size of the accessed bank, the number of bytes to transfer, and the address accessed.

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

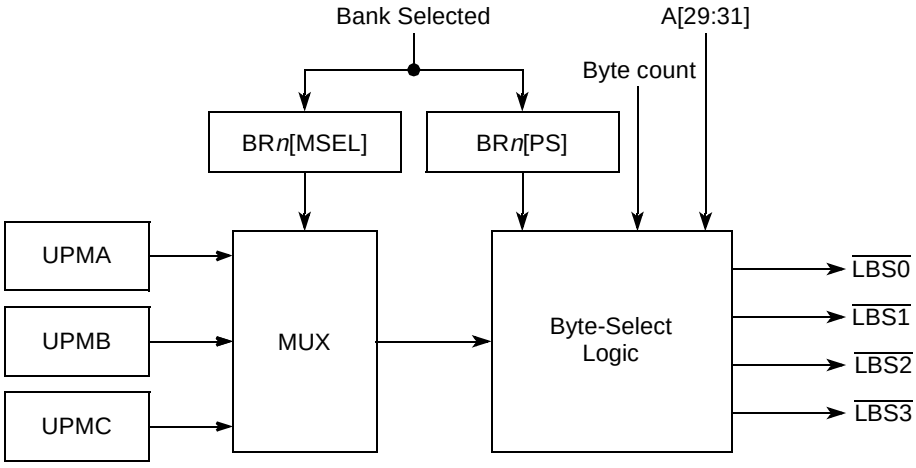


Figure 12-60. $\overline{\text{LBS}}$ Signal Selection

The uppermost byte select ($\overline{\text{LBS0}}$), when asserted, indicates that LAD[0:7] contains valid data during a cycle. Likewise, $\overline{\text{LBS1}}$ indicates that LAD[8:15] contains valid data, $\overline{\text{LBS2}}$ indicates that LAD[16:23] contains valid data, and $\overline{\text{LBS3}}$ indicates that LAD[24:31] contains valid data. For a UPM refresh timer

request, all $\overline{\text{LBS}}$[0:3] signals are asserted/negated by the UPM according to the refresh pattern only. Following any internal bus monitor exception, $\overline{\text{LBS}}$[0:3] signals are negated regardless of the exception handling provided by any UPM exception pattern to prevent spurious writes to external RAM.

12.4.4.4.4 General-Purpose Signals (G*n*T*n*, GO*n*)

The general-purpose signals (LGPL[0:5]) each have two bits in the RAM word that define the logical value of the signal to be changed at the rising edge of the bus clock and/or at the falling edge of the bus clock. LGPL0 offers enhancements beyond the other LGPL*n* lines.

GPL0 can be controlled by an address line specified in M*x*MR[G0CL]. To use this feature, G0H and G0L should be set in the RAM word. For example, for a SIMM with multiple banks, this address line can be used to switch between internal memory device banks.

12.4.4.4.5 Loop Control (LOOP)

The LOOP bit in the RAM word specifies the beginning and end of a set of UPM RAM words that are to be repeated. The first time LOOP = 1, the memory controller recognizes it as a loop start word and loads the memory loop counter with the corresponding contents of the loop field shown in Table 12-29. The next RAM word for which LOOP = 1 is recognized as a loop end word. When it is reached, the loop counter is decremented by one.

Continued loop execution depends on the loop counter. If the counter is not zero, the next RAM word executed is the loop start word. Otherwise, the next RAM word executed is the one after the loop end word. Loops can be executed sequentially but cannot be nested. Also, special care must be taken if LAST and LOOP must not be set together.

Table 12-29. M*x*MR Loop Field Use

Request Serviced	Loop Field
Read single-beat cycle	RLF
Read burst cycle	RLF
Write single-beat cycle	WLF
Write burst cycle	WLF
Refresh timer expired	TLF
RUN command	RLF

12.4.4.4.6 Repeat Execution of Current RAM Word (REDO)

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

Special care must be taken in the following cases:

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

12.4.4.4.7 Address Multiplexing (AMX)

The address lines can be controlled by the pattern the user provides in the UPM. The address multiplex bits can choose between driving the transaction address, driving it according to the multiplexing specified by the M*x*MR[AM] field, or driving the MAR contents on the address signals. In all cases, LA[27:31] of the LBC are driven by the five lsbs of the address selected by AMX, regardless of whether the NA bit of the RAM word is used to increment the current address. The effect of NA = 1 is visible only when AMX = 00 chooses the column address.

Table 12-30 shows how M*x*MR[AM] settings affect address multiplexing when the RAM word AMX = 10. The 16 msbs of the LAD[0:31] bus during an address phase are driven with zero in the AMX = 10 case.

Table 12-30. 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 signal 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.

NOTE

AMX must not be changed from its previous value in any RAM word which begins a loop.

12.4.4.4.8 Data Valid and Data Sample Control (UTA)

When a read access is handled by the UPM, and the UTA bit is 1 (data is to be sampled by the LBC), the value of the DLT3 bit in the same RAM word, in conjunction with M*x*MR[GPL*n*4DIS], determines when the data input is sampled by the LBC as follows:

- If M*x*MR[GPL*n*4DIS] = 1 (G4T4/DLT3 functions as DLT3) and DLT3 = 1 in the RAM word, data is latched on the falling edge of the bus clock instead of the rising edge. The LBC samples the data on the next falling edge of the bus clock, which is during the middle of the current bus cycle. This feature should be used only in systems without external synchronous bus devices that require mid-cycle sampling.
- If GPL*n*4DIS = 0 (G4T4/DLT3 functions as G4T4), or if GPL*n*4DIS = 1 but DLT3 = 0, data is latched on the rising edge of the bus clock, which occurs at the end of the current bus clock cycle (normal operation).

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

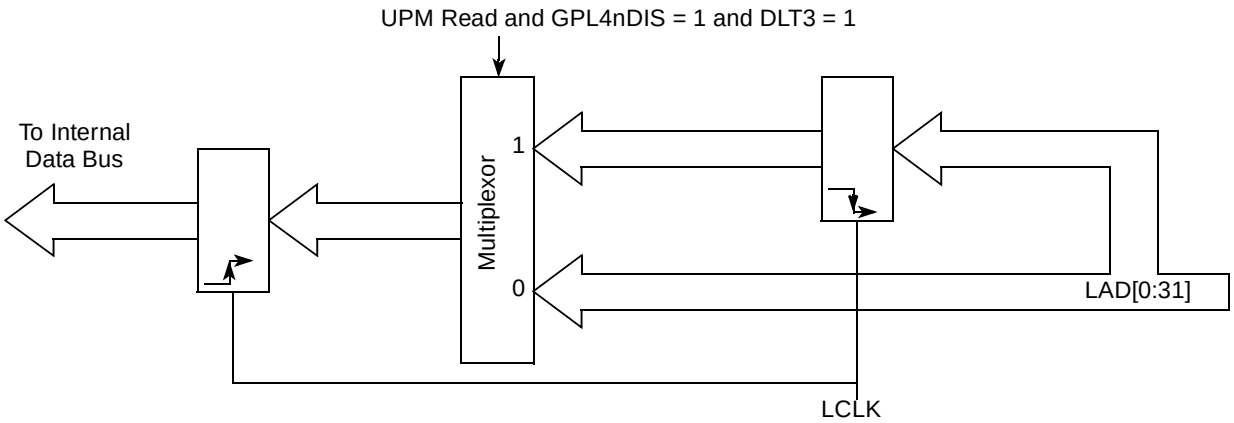


Figure 12-61. UPM Read Access Data Sampling

12.4.4.4.9 LGPL[0:5] Signal Negation (LAST)

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

12.4.4.4.10 Wait Mechanism (WAEN)

The WAEN bit in the RAM array word can be used to enable the UPM wait mechanism in selected UPM RAM words. If the UPM reads a RAM word with WAEN set, the external LUPWAIT signal is sampled and synchronized by the memory controller as if it were an asynchronous signal. The WAEN bit is ignored if LAST = 1 in the same RAM word.

Synchronization of LUPWAIT starts at the rising edge of the bus clock and takes at least 1 bus cycle to complete. If LUPWAIT is asserted and WAEN = 1 in the current UPM word, the UPM is frozen until LUPWAIT is negated. The value of external signals driven by the UPM remains as indicated in the

previous RAM word. When LUPWAIT is negated, the UPM continues normal functions. Note that during WAIT cycles, the UPM does not handle data.

Figure 12-62 shows how the WAEN bit in the word read by the UPM and the LUPWAIT signal are used to hold the UPM in a particular state until LUPWAIT is negated. As the example shows, the $\overline{\text{LCS}n}$ and LGPL1 states and the WAEN value are frozen until LUPWAIT is recognized as negated. WAEN is typically set before the line that contains UTA = 1. Note that if WAEN and NA are both set in the same RAM word, NA causes the burst address to increment once as normal regardless of whether the UPM freezes.

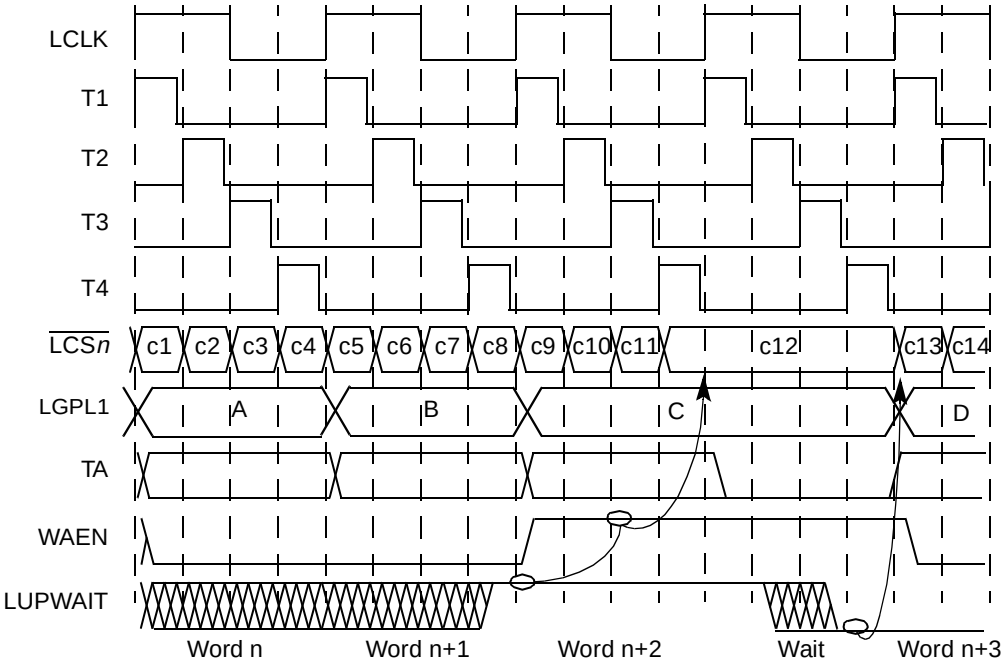


Figure 12-62. Effect of LUPWAIT Signal

12.4.4.5 Synchronous Sampling of LUPWAIT for Early Transfer Acknowledge

If LUPWAIT is to be considered an asynchronous signal, which can be asserted/negated at any time, no UPM RAM word must contain both WAEN = 1 and UTA = 1 simultaneously.

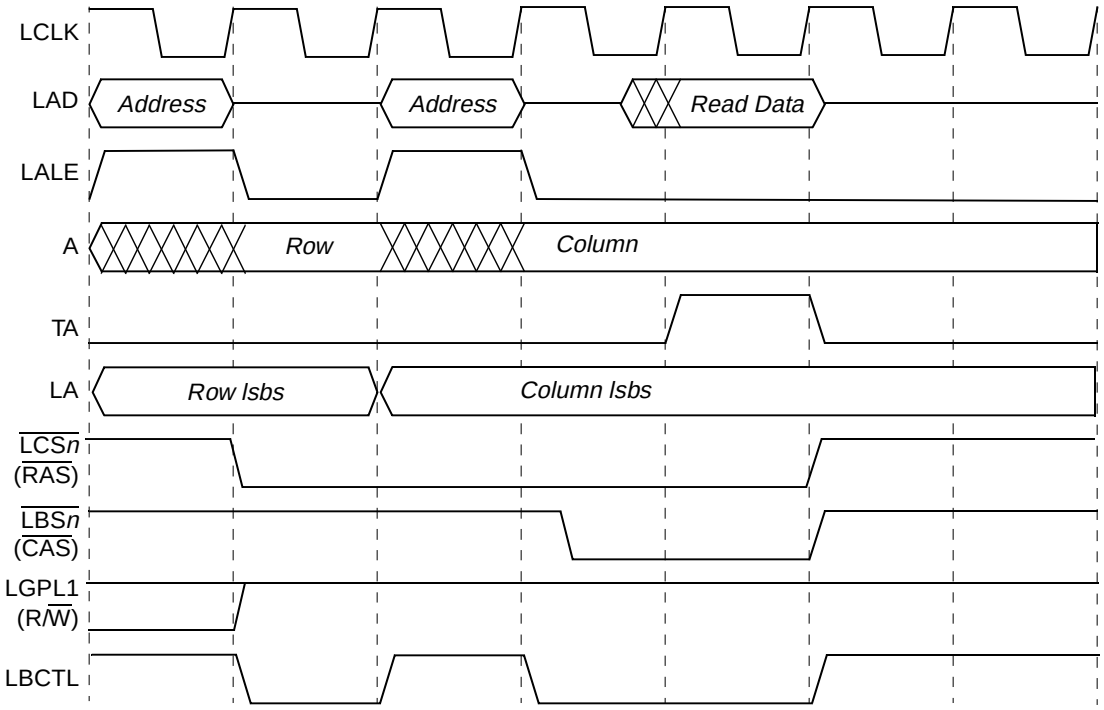
However, programming WAEN = 1 and UTA = 1 in the same RAM word allows UPM to treat LUPWAIT as a synchronous signal, which must meet set-up and hold times in relation to the rising edge of the bus clock. In this case, as soon as UPM samples LUPWAIT negated on the rising edge of the bus clock, it immediately generates an internal transfer acknowledge, which allows a data transfer one bus clock cycle later. The generation of transfer acknowledge is early because LUPWAIT is not re-synchronized, and the acknowledge occurs regardless of whether UPM was already frozen in WAIT cycles or not. This feature allows the synchronous negation of LUPWAIT to affect a data transfer, even if UTA, WAEN, and LAST are set simultaneously.

12.4.4.6 Extended Hold Time on Read Accesses

Slow memory devices that take a long time to turn off their data bus drivers on read accesses should choose some non-zero combination of OR*n*[TRLX] and OR*n*[EHTR]. The next accesses after a read access to the slow memory device is delayed by the number of clock cycles specified in the OR*n* register in addition to any existing bus turn around cycle.

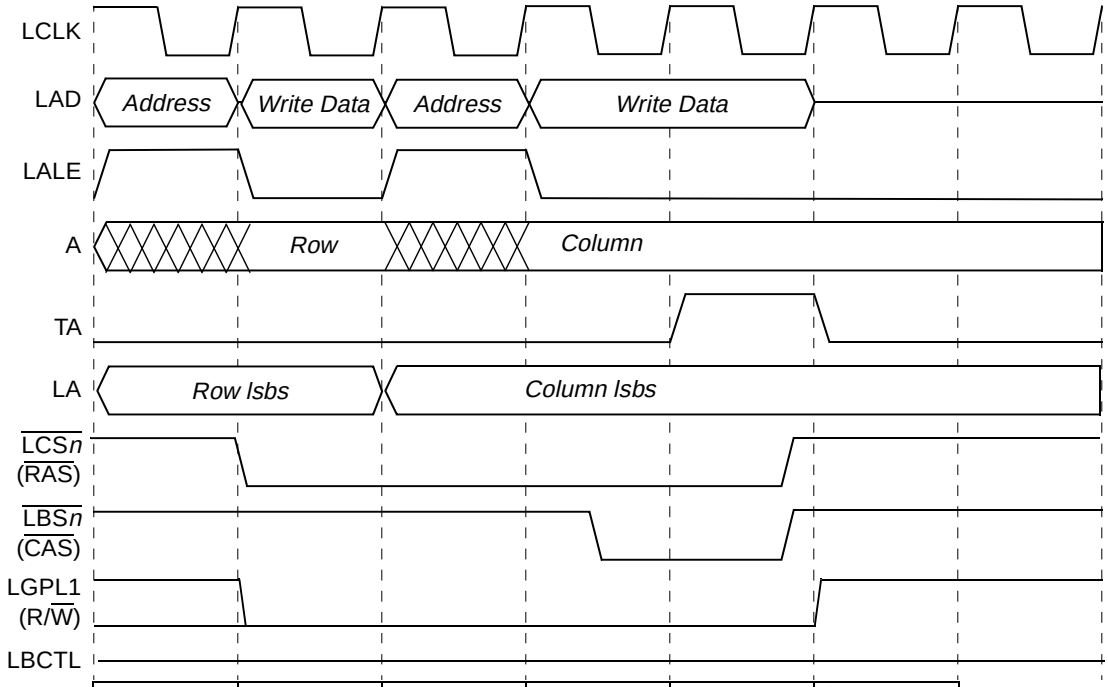
12.4.4.7 Memory System Interface Example Using UPM

Connecting the local bus UPM controller to a DRAM device requires a detailed examination of the timing diagrams representing the possible memory cycles that must be performed when accessing this device. This section shows timing diagrams for various UPM configurations, using fast-page mode DRAM as an example, with LCRR[CLKDIV] = 4(clock ratio of 8) or 8(clock ratio of 16). These illustrative examples may not represent the timing necessary for any specific device used with the LBC. Here, LGPL1 is programmed to drive R/$\overline{W}$ of the DRAM, although any LGPL*n* signal may be used for this purpose.



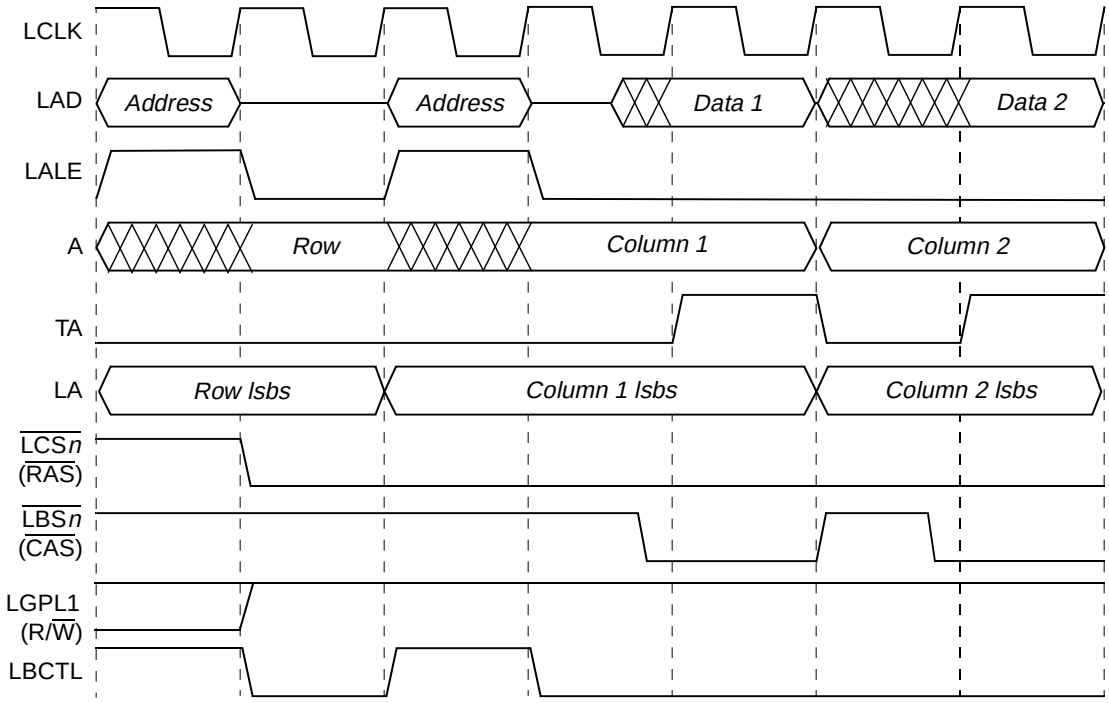
cst1	0	LALE pause (due to change in AMX)	0	0	Bit 0
cst2	0		0	0	Bit 1
cst3	0		0	0	Bit 2
cst4	0		0	0	Bit 3
bst1	1		1	0	Bit 4
bst2	1		0	0	Bit 5
bst3	1		0	0	Bit 6
bst4	1		0	0	Bit 7
g0l0					Bit 8
g0l1					Bit 9
g0h0					Bit 10
g0h1					Bit 11
g1t1	1		1	1	Bit 12
g1t3	1		1	1	Bit 13
g2t1					Bit 14
g2t3					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 12-63. Single-Beat Read Access to FPM DRAM



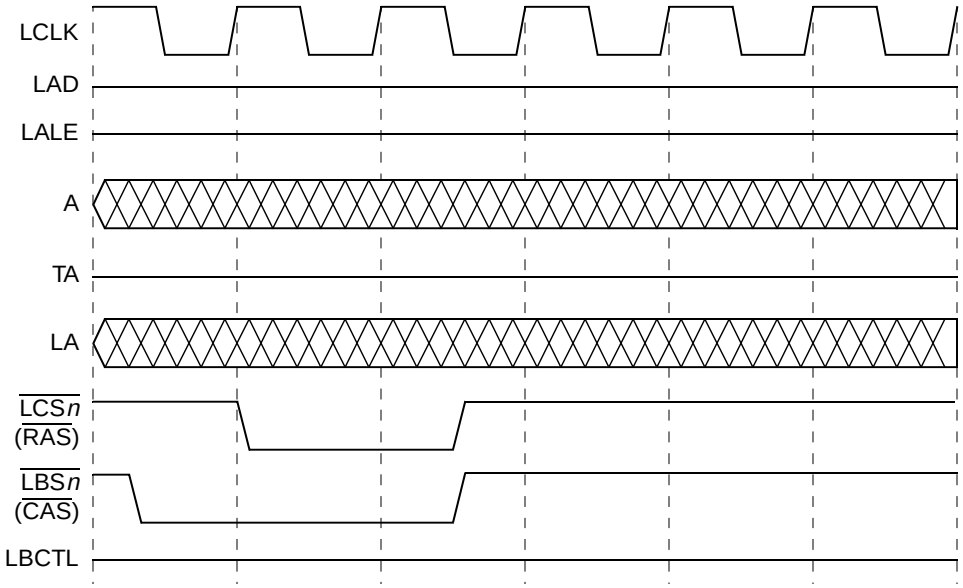
cst1 cst2 cst3 cst4	0 0 0 0	LALE pause (due to change in AMX)	0 0 0 0	0 0 0 1	Bit 0 Bit 1 Bit 2 Bit 3
bst1 bst2 bst3 bst4	1 1 1 1		1 1 0 0	0 0 0 1	Bit 4 Bit 5 Bit 6 Bit 7
g0l0 g0l1 g0h0 g0h1					Bit 8 Bit 9 Bit 10 Bit 11
g1t1 g1t3	0 0		0 0	0 0	Bit 12 Bit 13
g2t1 g2t3					Bit 14 Bit 15
g3t1 g3t3					Bit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

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



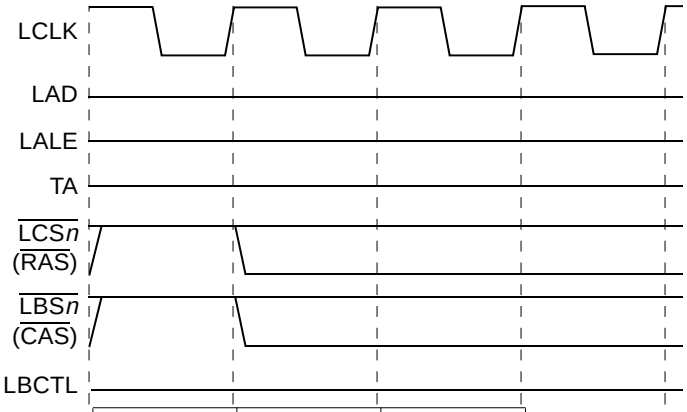
cst1 cst2 cst3 cst4	0 0 0 0	LALE pause (due to change in AMX)	0 0 0 0	0 0 0 0	1 1 1 1	Bit 0 Bit 1 Bit 2 Bit 3
bst1 bst2 bst3 bst4	1 1 1 1		1 1 1 0	0 0 0 0	1 1 1 1	Bit 4 Bit 5 Bit 6 Bit 7
g0l0 g0l1 g0h0 g0h1						Bit 8 Bit 9 Bit 10 Bit 11
g1t1 g1t3	1 1		1 1	1 1	1 1	Bit 12 Bit 13
g2t1 g2t3						Bit 14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

Figure 12-65. Burst Read Access to FPM DRAM Using LOOP (Two Beats Shown)



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

Figure 12-66. Refresh Cycle (CBR) to FPM DRAM



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

Figure 12-67. Exception Cycle

12.5 Initialization/Application Information

12.5.1 Interfacing to Peripherals

12.5.1.1 Multiplexed Address/Data Bus and Non-Multiplexed Address Signals

To save signals 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 signals 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 bits. For devices smaller than 32 bits, transactions must be broken down. For this reason, LA[30:31] are driven non-multiplexed. 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 five least significant addresses. Those should not be used from LAD[27:31].

All other addresses, A[0:26], must be reconstructed through the latch.

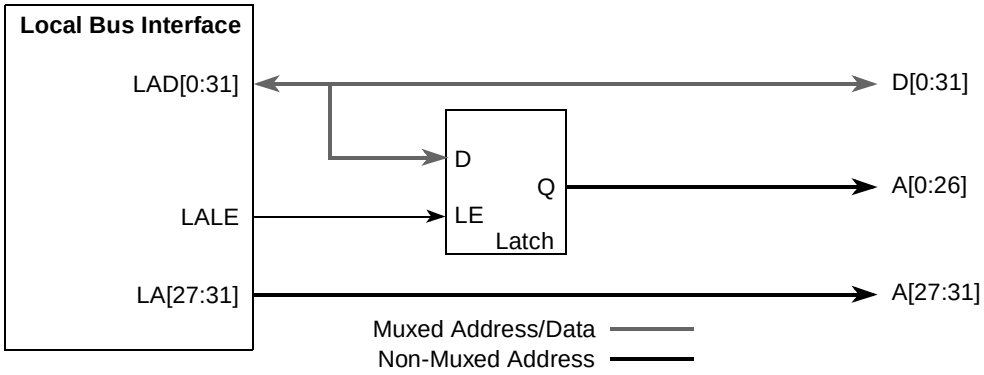


Figure 12-68. Multiplexed Address/Data Bus

12.5.1.2 Peripheral Hierarchy on the Local Bus

To achieve high bus speed interfaces for synchronous SRAMs or SDRAMs, a hierarchy of the memories/peripherals connected to the local bus is suggested, as shown in Figure 12-69.

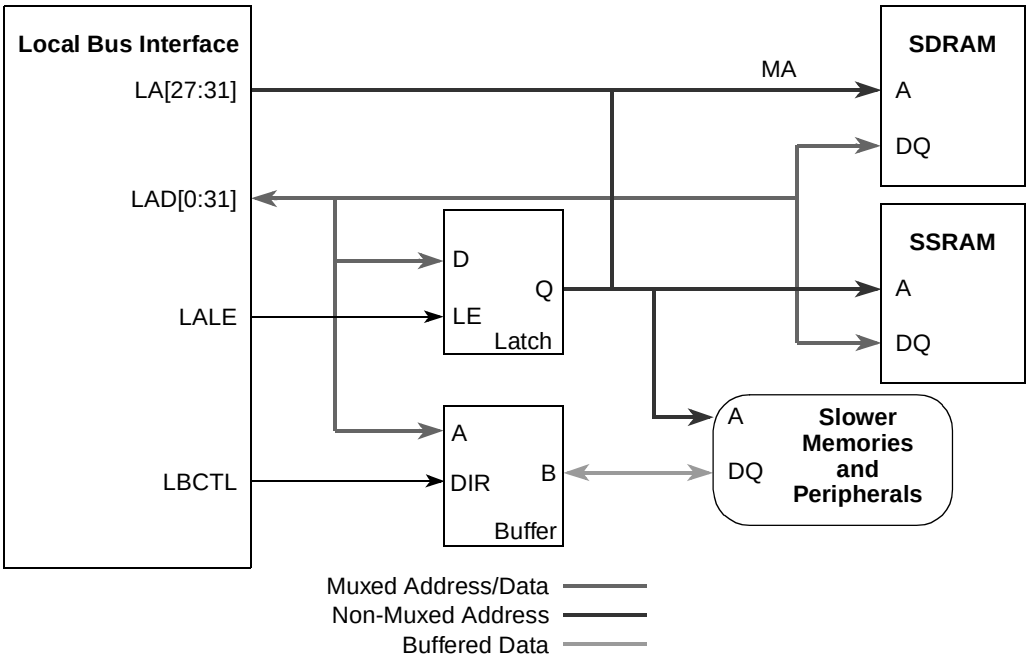


Figure 12-69. Local Bus Peripheral Hierarchy

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

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

To achieve the highest possible bus speeds on the local bus, it is recommended to reduce the number of devices connected directly to the local bus even further. For those cases probably only one bank of synchronous SRAMs or SDRAMs should be used and instead of using a separate latch and a separate bus transceiver, a bus demultiplexer combining those two functions into one device should be used.

Figure 12-70 shows an example of such a hierarchy. This section is only a guideline and the board designer must simulate the electric characteristics of his scenario to determine the maximum operating frequency.

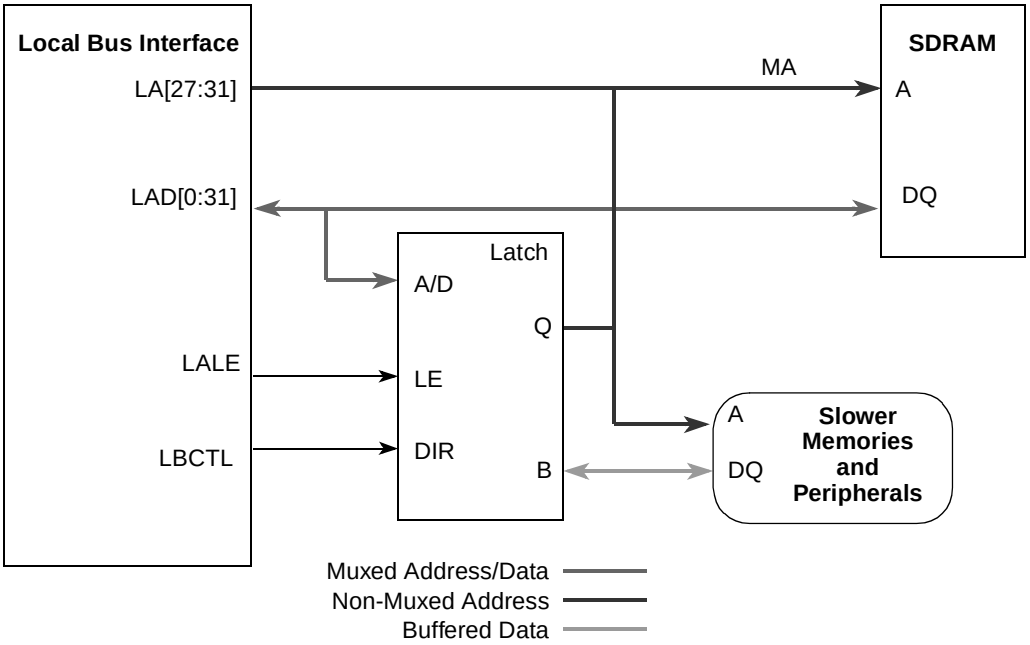


Figure 12-70. Local Bus Peripheral Hierarchy for Very High Bus Speeds

12.5.1.4 GPCM Timings

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

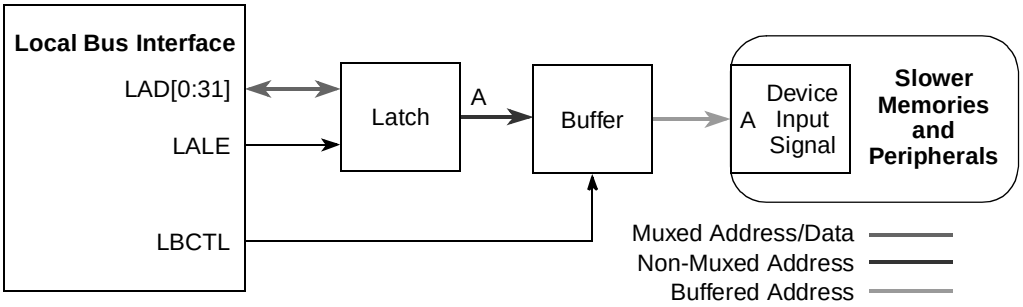


Figure 12-71. GPCM Address Timings

To calculate address setup timing for a slower peripheral/memory device, several parameters have to be added: propagation delay for the address latch, propagation delay for the buffer and the address setup for the actual peripheral. Typical values for the 2 propagation delays are in the order of 3–6 ns, so for a 133-MHz bus frequency, for example, $\overline{\text{LCS}}$ should arrive on the order of 3 bus clocks later.

For data timings, only the propagation delay of one buffer plus the actual data setup time has to be considered.

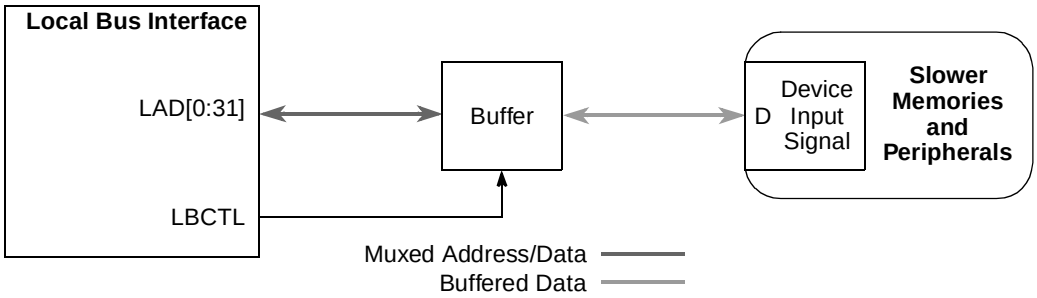


Figure 12-72. GPCM Data Timings

12.5.2 Bus Turnaround

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

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

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

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

12.5.2.1 Address Phase After Previous Read

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

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

12.5.2.2 Read Data Phase After Address Phase

During the address phase, LAD actively drives the address and LBCTL is high, driving the bus transceivers in the same direction as during a write. After the end of the address phase, LBCTL goes low and changes the direction of the bus transceiver. The LBC places the LAD signals in high impedance after its t_{dis}(LB). The LBCTL have its new state after t_{en}(LB) and, because this is an asynchronous input, the transceiver starts to drive those signals after its t_{en}(transceiver) time. The system designer has to ensure, that [t_{en}(LB) + t_{en}(transceiver)] is larger than t_{dis} (LB) to avoid bus contention.

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

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

12.5.2.4 UPM Cycles with Additional Address Phases

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

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

12.5.3 Interface to Different Port-Size Devices

The LBC supports 8-, 16-, and 32-bit data port sizes. However, the bus requires that the portion of the data bus used for a transfer to or from a particular port size be fixed. A 32-bit port must reside on D[0:31], a 16-bit port must reside on D[0:15], and an 8-bit port must reside on D[0:7]. The local bus always tries to transfer the maximum amount of data on all bus cycles.

Figure 12-73 shows the device connections on the data bus.

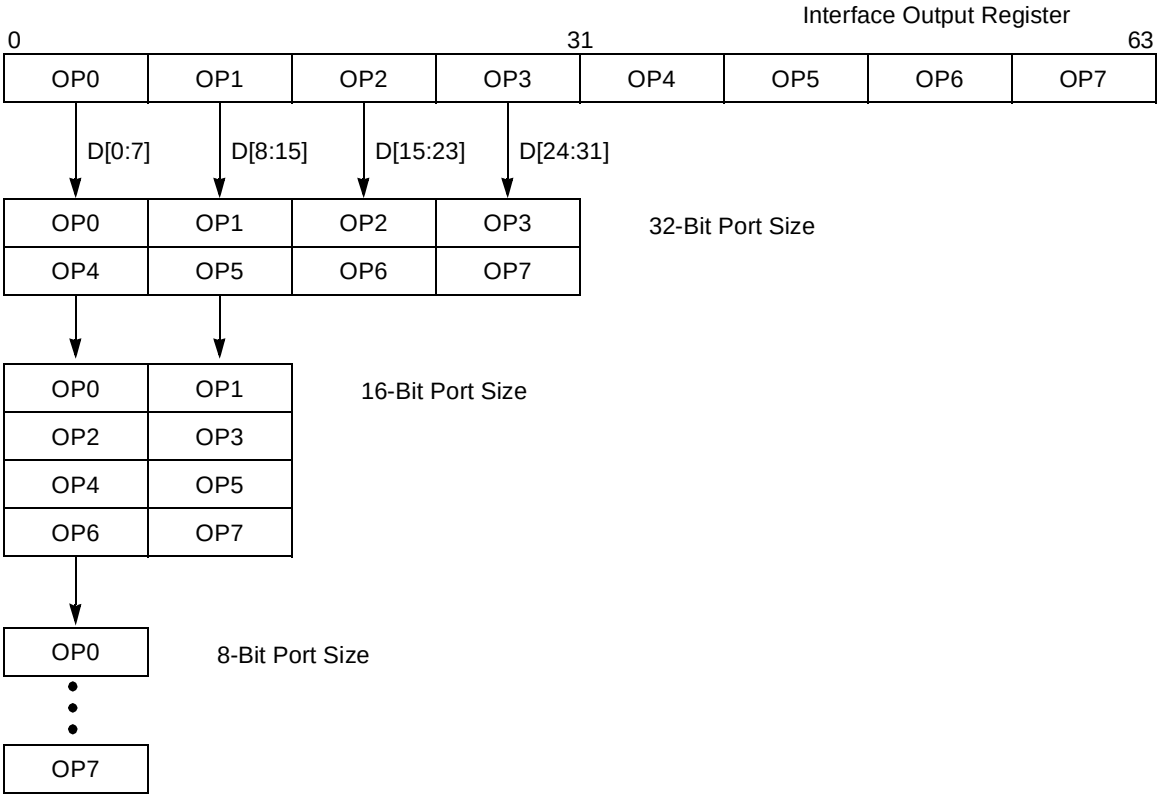


Figure 12-73. Interface to Different Port-Size Devices

Table 12-31 lists the bytes required on the data bus for read cycles.

Table 12-31. Data Bus Requirements For Read Cycle

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

Table 12-31. Data Bus Requirements For Read Cycle (continued)

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

1 Address state is the calculated address for port size.

2 OP*n*: These lanes are read or written during that bus transaction. OP0 is the most-significant byte of a word operand and OP3 is the least-significant byte.

3 — Denotes a byte not driven during that write cycle.

12.5.4 Interfacing to SDRAM

The following sections provide application information on interfacing to SDRAM.

12.5.4.1 Basic SDRAM Capabilities of the Local Bus

The LBC provides one SDRAM machine for the local bus. Although there is only one machine, multiple chip selects ($\overline{\text{LCS}}n$) can be programmed to support multiple SDRAM devices. Note that no limitation exists on the number of chip selects that can be programmed for SDRAM. This means that $\overline{\text{LCS}}$[1:7] can be programmed to support SDRAM, assuming $\overline{\text{LCS0}}$ is reserved for the GPCM to connect to Flash memory.

If multiple chip selects are configured to support SDRAM on the local bus, each SDRAM device should have the same port size and timing parameters. This means that all option registers (OR*n*) for the SDRAM chip selects should be programmed exactly the same.

NOTE

Although in principle it is possible to mix different port sizes and timing parameters, combinations are limited and this operation is not recommended.

All the chip selects share the same local bus SDRAM mode register (LSDMR) for initialization along with the local bus-assigned SDRAM refresh timer register (LSRT) and the memory refresh timer prescaler register (MPTPR) for refresh.

For refresh, the memory controller supplies auto refresh to SDRAM according to the time interval specified in LSRT and MPTPR as follows:

$$\text{Refresh Period} = \frac{\text{LSRTx (MPTPR[PTP])}}{\text{System Frequency}}$$

This represents the time period required between refreshes. When the refresh timer expires, the memory controller issues a CBR to each chip select. Each CBR is separated by one clock. A refresh timing diagram for multiple chip selects is shown in Figure 12-52 in Section 12.4.3.11.1, "SDRAM Refresh Timing."

During a memory transaction dispatched to the local bus, the memory controller compares the memory address with the address information of each chip select (programmed with BR*n* and OR*n*). If the comparison matches a chip select that is controlled by SDRAM, the memory controller requests service to the local bus SDRAM machine, depending on the information in BR*n*. Although multiple chip selects may be programmed for SDRAM, only one chip select is active at any given time; thus, multiple chip selects can share the same SDRAM machine.

12.5.4.2 Maximum Amount of SDRAM Supported

Table 12-32 summarizes information based on typical SDRAM data sheets.

Table 12-32. Typical 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	—
Column	10	9	8	8	11	10	9	8	11	10	9	8	12	11	10	—

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

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

12.5.4.3 SDRAM Machine Limitations

This section describes limitations of the local bus SDRAM machine.

12.5.4.3.1 Analysis of Maximum Row Number Due to Bank Select Multiplexing

LSDMR[BSMA] is used to multiplex the bank select address. The BSMA field and corresponding multiplexed address are shown below:

```
000 LA12–LA13
001 LA13–LA14
…
111 LA19–LA20
```

Note that LA12 is the latched value of LAD12.

The highest address signals that the bank selects can be multiplexed with are LA[12:13], which limits the signals for the row address to LA[14:31]. For a 32-bit port, the maximum width of the local bus, LA[30:31] are not connected, and the maximum row is LA[14:29]. The local bus SDRAM machine supports 15 rows, which is sufficient for all devices.

12.5.4.3.2 Bank Select Signals

Page-based interleaving allows bank signals to be multiplexed to the higher-order address signals 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 signal for a larger memory size.

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

12.5.4.3.3 128-Mbyte SDRAM

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

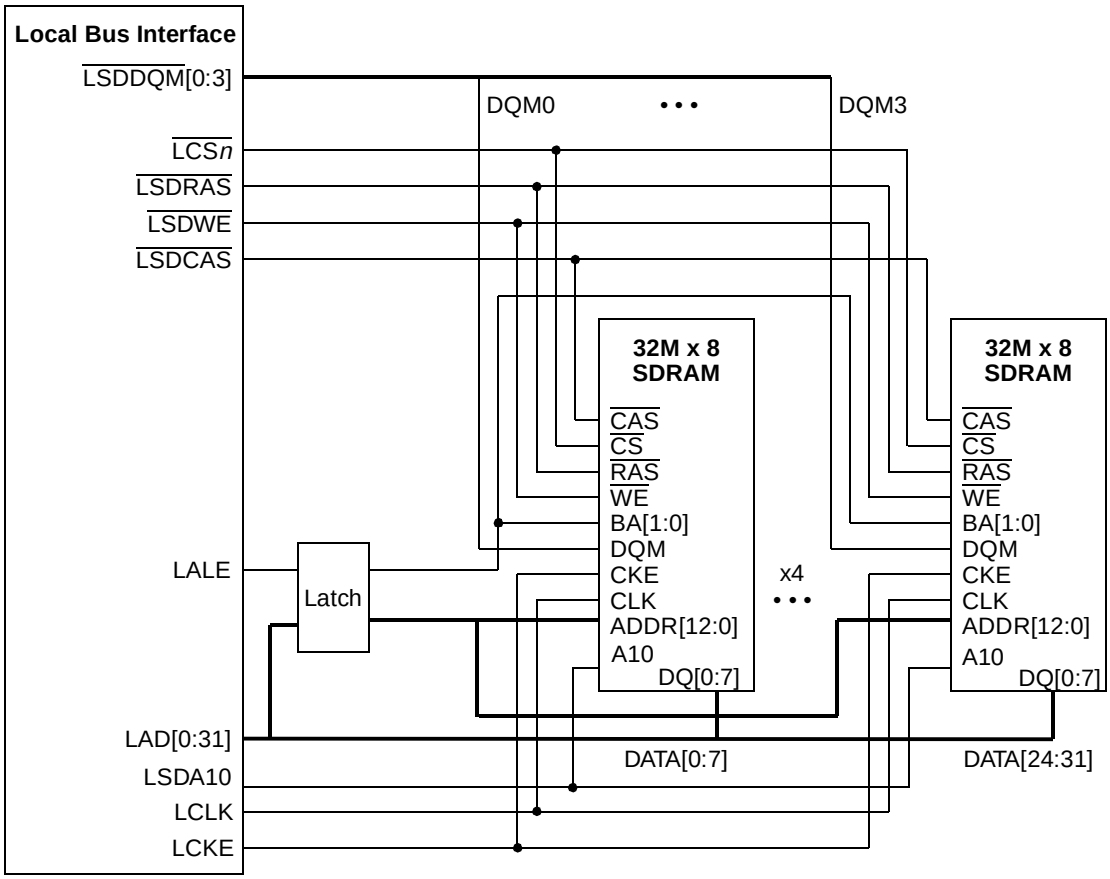


Figure 12-74. 128-Mbyte SDRAM Diagram

Table 12-33 shows details about LAD*n* signal connections for the example in Figure 12-74.

Table 12-33. LAD*n* Signal Connections to 128-Mbyte SDRAM

LAD (Latch Address)	SDRAM Address Signal
LAD29	A0
LAD28	A1
LAD27	A2
LAD26	A3
LAD25	A4
LAD24	A5
LAD23	A6
LAD22	A7

Table 12-33. LAD*n* Signal Connections to 128-Mbyte SDRAM (continued)

LAD (Latch Address)	SDRAM Address Signal
LAD21	A8
LAD20	A9
LAD19 (no connect)	A10 is connected to LSDA10
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 $8 \times 8 \times 32$ Mbits.
- Each device has 4 internal banks, 13 row address lines, and 10 column address lines.

The logical address is partitioned as shown in Table 12-34.

Table 12-34. Logical Address Bus Partitioning

A[0:4]	A[5:17]	A[18:19]	A[20:29]	A[30:31]
msb of start address	Row	Bank select	Column	lsb

The following parameters are extracted:

- COLS = 011, 10 column lines
- ROWS = 100, 13 row lines

During the address phase, the SDRAM address port is set as shown in Table 12-35.

Table 12-35. SDRAM Device Address Port During Address Phase

LA[0:14]	LA[15:16]	LA[17:29]	LA[30:31]
—	Internal bank select A[18:19]	Row A[5:17]	No connect

Because the internal bank selects are multiplexed over LA[15:16], LSDMR[BSMA] must be set to 011.

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

Table 12-36. SDRAM Device Address Port During READ/WRITE Command

LA[0:14]	LA[15:16]	LA[17:18]	LA[19]	LA[20:29]	LA[30:31]
msb of start address	Internal bank select	Don’t care	AP	Column	No connect

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

Table 12-37. Register Settings for 128-Mbyte SDRAMs

Register	Field	Value
BR*n*	BA XBA PS MS V	Base address Ext. Base Address 11 = 32-bit port size 011 = SDRAM-local bus 1
OR*n*	AM XAM COLS ROWS	11_1111_1000_0000_0000_0 11 011 100
LSDMR	RFEN OP BSMA RFRC PRETOACT ACTTOROW BL WRC BUFCMD CL	1 000 011 From device data sheet From device data sheet From device data sheet 0 From device data sheet 0 From device data sheet

12.5.4.3.4 256-Mbyte SDRAM

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

12.5.4.3.5 512-Mbyte SDRAM

This example uses the MT48LC64M4A2FB from Micron to implement 512 Mbytes.

In this SDRAM organization:

- The 32-bit port size is $8 \times 4 \times 64$ Mbit $\times$ 2 chip select lines.
- Each device has 4 internal banks, 13 row address lines, and 11 column address lines.

The logical address is partitioned as shown in Table 12-38.

Table 12-38. Logical Address Partitioning

A[0:3]	A[4:16]	A[17:18]	A[19:29]	A[30:31]
msb of start address	Row	Bank select	Column	lsb

The following parameters can be extracted:

- COLS = 100, 11 column lines
- ROWS = 100, 13 row lines

During the address phase, the SDRAM address port is set as in Table 12-39.

Table 12-39. SDRAM Device Address Port During Address Phase

LA[0:13]	LA[15:16]	LA[17:29]	LA[30:31]
—	Internal bank select (A[17:18])	Row (A[4:16])	No-connect

Because the internal bank selects are multiplexed over LA[15:16], LSDMR[BSMA] must be set to 011.

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

Table 12-40. SDRAM Device Address Port During READ/WRITE Command

LA[0:14]	LA[15:16]	LA[17]	LA[18]	LA[19:29]	LA[30:31]
msb of start address	Internal bank select	Don't care	AP	Column	No-connect

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

Table 12-41. Register Settings for 512-Mbyte SDRAMs

Register	Field	Value
BR*n*	BA XBA PS MS V	Base address ext. Base address 11 = 32-bit port size 011 = SDRAM-local bus 1
OR*n*	AM XAM BPD COLS ROWS	11_1110_0000_0000_0000_0 11 01 100 100
PSDMR	RFEN OP BSMA RFRC PRETOACT ACTTOROW BL WRC BUFCMD CL	1 000 011 From device data sheet From device data sheet From device data sheet 0 From device data sheet 0 From device data sheet

12.5.4.3.6 Power-Down Mode

SDRAMs offer a power-down mode during which the device is not refreshed, and therefore, data is not maintained. This mode is invoked by driving CKE low, while all internal banks are idle; note that they must be precharged first. Figure 12-75 shows the timing. Note that the figure does not show the precharge-all command that is issued by the LBC automatically prior to the self-refresh command.

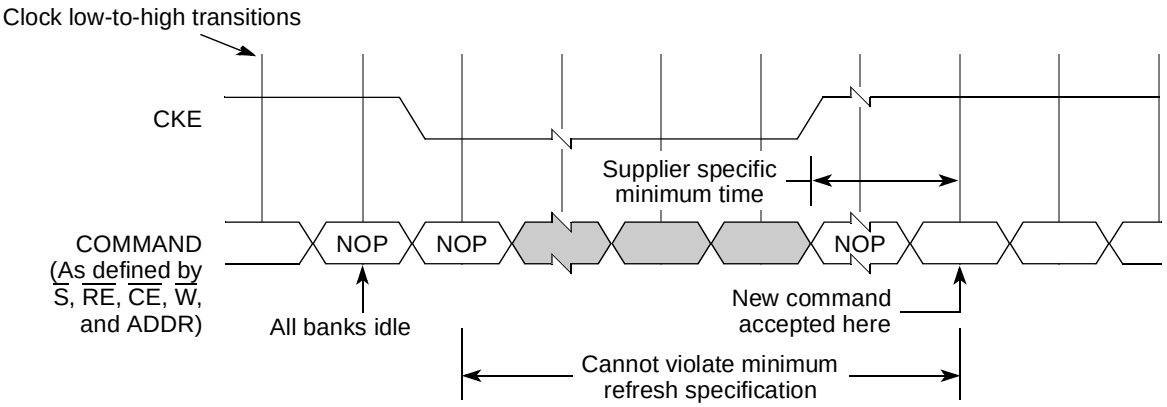


Figure 12-75. SDRAM Power-Down Timing

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

12.5.4.3.7 Self-Refresh

In order to be able to stop activity on the local bus (for power save or debug), while the content of the SDRAM is maintained, the self-refresh mode is supported. This mode is invoked by issuing a self-refresh command to the SDRAM. The LBC applies the same timing as for the auto refresh, but also pulls the SDRAM CKE (LCKE) signal low in the same cycle. This can only be done with all banks being idle; the SDRAM machine must precharge them ahead of this. As long as CKE stays low, the device refreshes itself and does not need to see any refreshes from the local bus. To exit self refresh, CKE simply has to be pulled high. Note that after returning from self-refresh mode the SDRAM needs a supplier-specific time before it can accept new commands and the auto-refresh mechanism has to be started again. Figure 12-76 shows this timing. The SDRAM controller always uses 200 local bus clocks, which should satisfy any SDRAM requirements. As in the case of the power-down mode, the figure does not show the precharge-all command that is issued by the LBC automatically prior to the self-refresh command. Refer to Section 12.4.3.3, “Intel PC133 and JEDEC-Standard SDRAM Interface Commands,” for SDRAM interface commands and information on the self-refresh command.

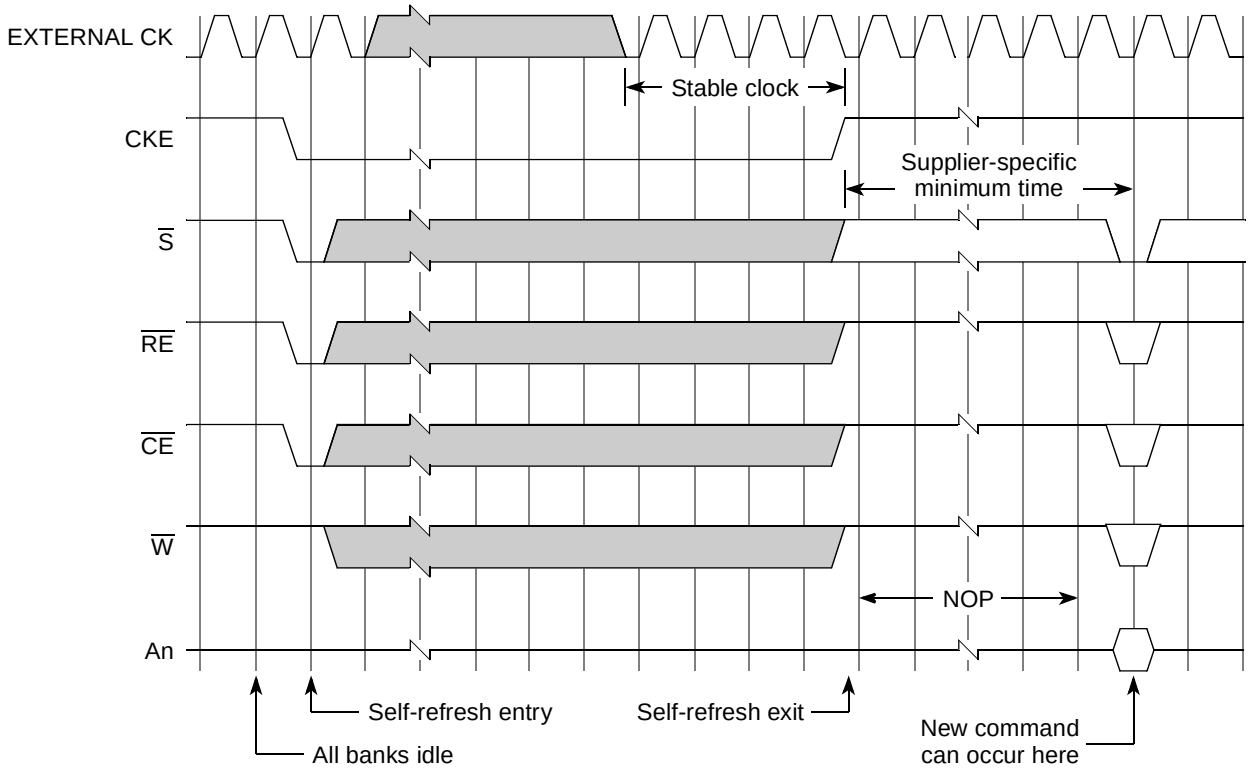


Figure 12-76. SDRAM Self-Refresh Mode Timing

12.5.4.3.8 SDRAM Timing

To allow for very high speeds on the memory bus, the capacitive loading on the local bus must be taken into consideration as shown in Table 12-42.

Table 12-42. SDRAM Capacitance

Signal	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 2 synchronous memory banks, 1 address latch, and 1 buffer loading the multiplexed address/data bus sees a loading of 4 loads of about 6.5 pF maximum. For a nominal load, 30 pF can be used.

Table 12-43. SDRAM AC Characteristics

Parameter		Device Speed 133 MHz		Unit
		Min	Max	
CLK cycle time	CAS latency = 3	7.5	1000	ns
	CAS latency = 2	7.5		
CLK to valid output delay	CAS latency = 3	—	5.4	ns
	CAS latency = 2	—	5.4	
Output data hold time	CAS latency = 3	3	—	ns
	CAS latency = 2	3	—	
Input setup time		2	—	ns
Input hold time		1	—	ns
CLK to output in Hi-Z	CAS latency = 3	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 PLL 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 PLL is used to generate external clocks, which minimize the skew between the local bus and the memory clock. Figure 12-77 shows relative timings for the local bus clock PLL.

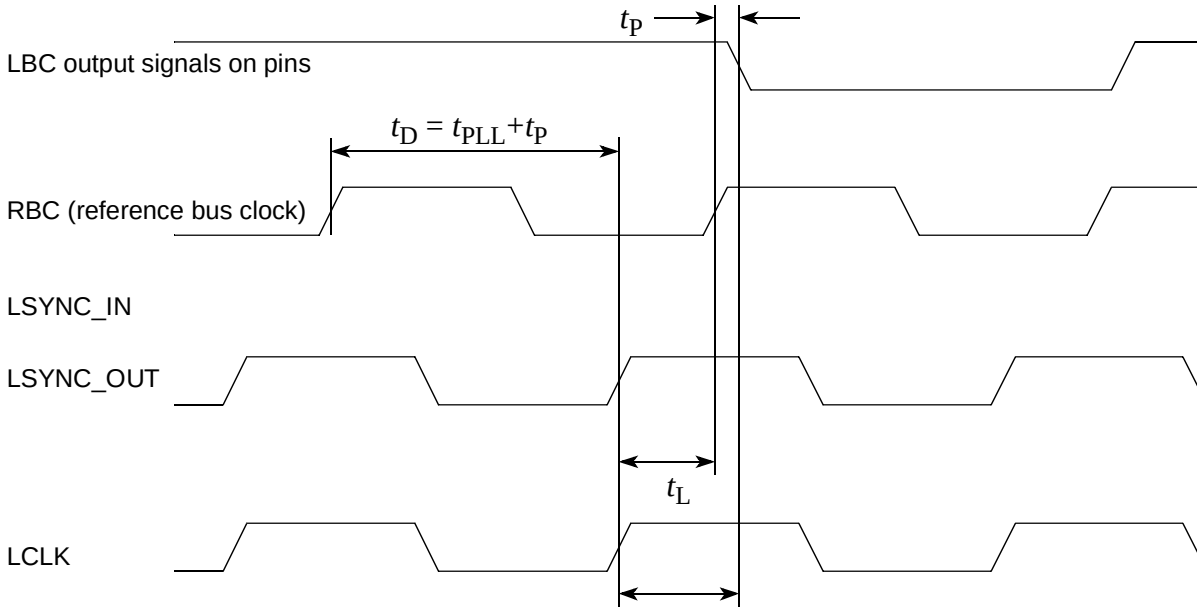


Figure 12-77. Local Bus PLL Operation

12.5.4.4 Parity Support for SDRAM

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

Figure 12-78 shows a connection diagram.

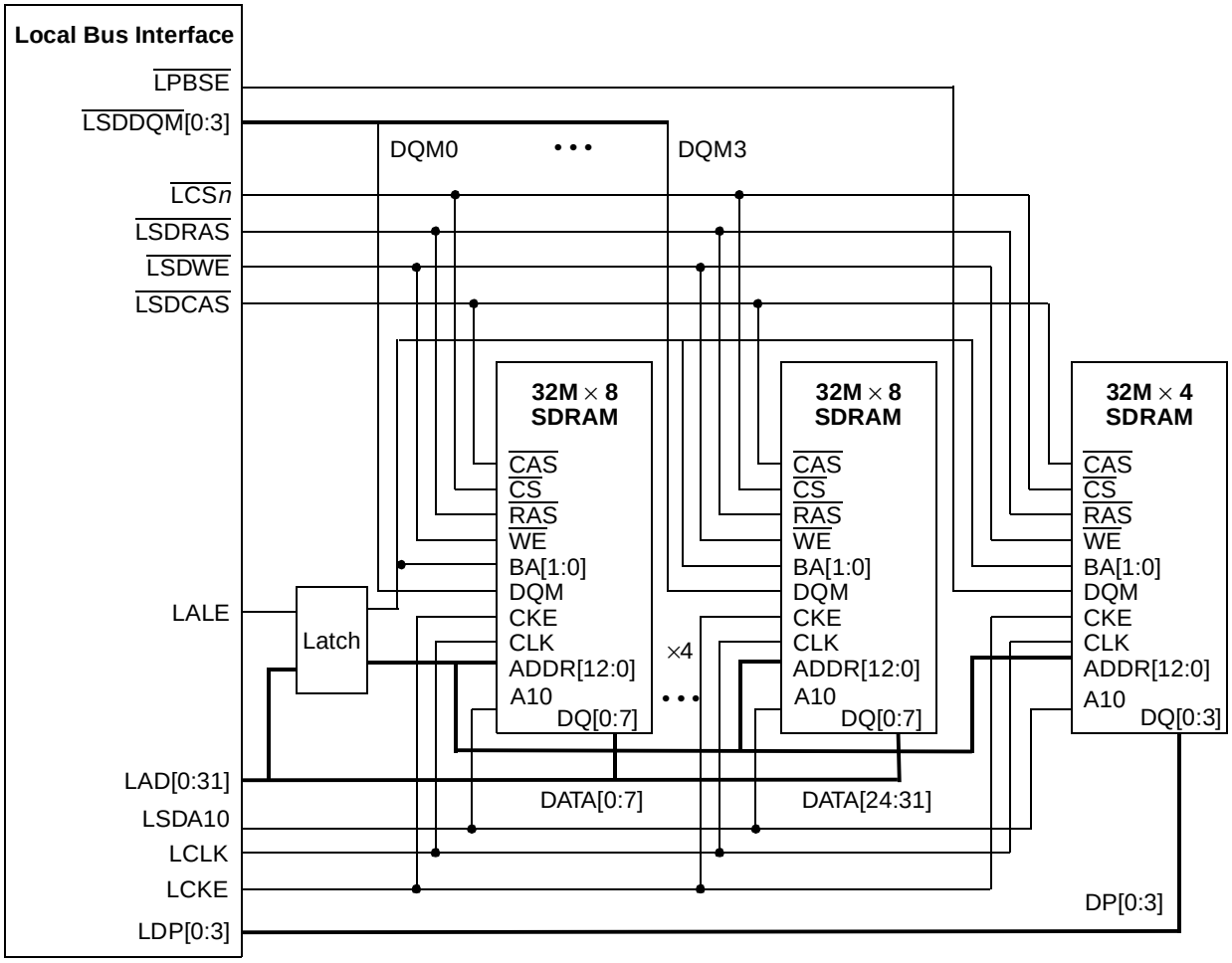


Figure 12-78. Parity Support for SDRAM

12.5.5 Interfacing to ZBT SRAM

In many applications, SDRAM provides sufficient performance for the local bus. However, especially in networking applications, memory access patterns are often random and SDRAM is not optimized for that case. ZBT SRAMs have been designed to optimize the performance in networking applications. This section describes how to interface to ZBT SRAMs. Figure 12-79 shows the connections. The UPM is used to generate control signals. The same interfacing is used for pipelined and flow-through versions of ZBT SRAMs. However different UPM patterns must be generated for those cases. Because ZBT SRAMs are mostly used by performance-critical applications, an assumption is made that, typically, the maximum width of the local bus of 32 bits is used.

ZBT SRAMs allow different configurations. For the local bus the burst order should be set to linear burst order by tying the mode signal to GND; $\overline{\text{CKE}}$ should also be tied to ground.

ZBT SRAMs perform four-beat bursts. Because the LBC generates eight-beat transactions (for 32-bit ports) the UPM breaks down each burst into two consecutive four-beat bursts. The internal address generator of the LBC generates the new A27 for the second burst.

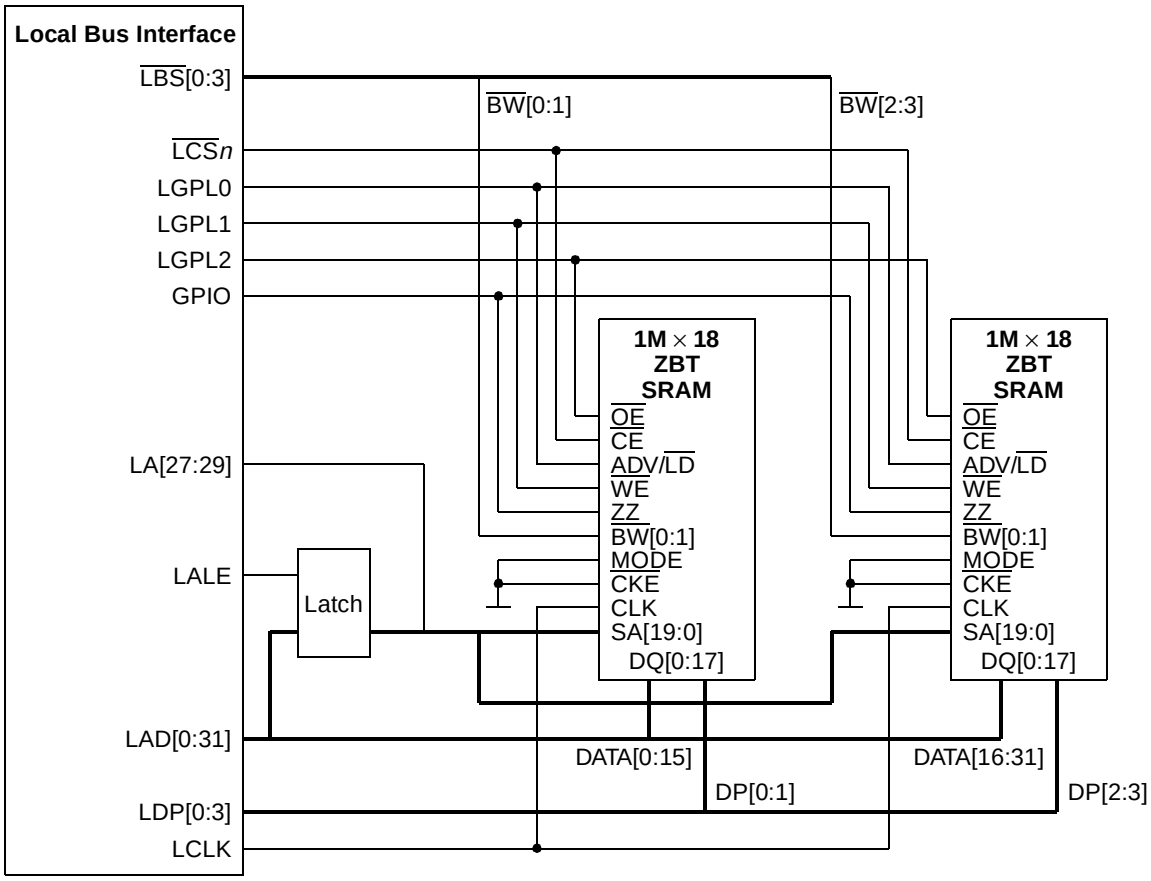


Figure 12-79. Interface to ZBT SRAM

Because linear burst is used on the SRAM, the device itself bursts with the burst addresses of [0:1:2:3]. The local bus always generates linear bursts and expects [0:1:2:3:4:5:6:7]. Therefore, two consecutive linear bursts of the ZBT SRAM with A27 = 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 $\overline{\text{WE}}$). 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 signal. Connecting a GPIO signal 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, a ×18 SRAM is used, which holds two data bytes and two parity bits. While for the support of parity on SDRAM banks the local

bus has to use read-modify-write cycles and compromise performance, SRAM banks can be used with natural parity and do not compromise performance for parity support.

12.5.6 Interfacing to DSP Host Ports

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

12.5.6.1 Interfacing to MSC8101 HDI16

This section describes how to interface to the HDI16 peripheral interface of the MSC8101. After initial set-up of the interface, the host and HDI16 device can communicate either by a read and write transaction from the core or, if the setup on the DSP and the host are implemented appropriately, by DMA transfers of the host DMA controller, which can be triggered automatically by signals generated by the HDI16 peripheral.

12.5.6.1.1 HDI16 Peripherals

The host interface (HDI16) is a 16-bit-wide, full-duplex, double-buffered parallel port that can directly connect to the data bus of a host processor. It supports a variety of buses and gluelessly connects with a number of industry-standard microcomputers, microprocessors, and DSPs. The HDI16 also supports the 8-bit host data bus, which makes it fully compatible with the DSP56300 HI08 (as viewed by the host side, not from the DSP side).

The host bus can operate asynchronously to the SC140 core clock, and the HDI16 registers are divided into two banks. The host register bank is accessible to the external host, and the core register bank is accessible to the SC140 core.

The MSC8101HDI16 host port peripheral has two sets of 16-bit-wide registers—one set is only visible internally to the DSP, while the other set is visible only to the external host processor. Figure 12-81 illustrates the relationship between the two sides.

All of the HDI16 peripheral's registers are mapped directly onto the MSC8101 QBus, as defined by the *MSC8101 16-Bit Digital Signal Processor Reference Manual* (MSC8101RM); the transmit and receive FIFOs are mapped onto the DMA data bus such that the DMA controller can access them directly without core intervention. The addressing for each of these registers is defined in Section 12.5.6.1.2, "Physical Interconnections."

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

The most important HDI16 host port facet is that it is specified as an asynchronous interface and so reduces concerns over clock skew between the HDI16 host port and the host device's buses. Furthermore, with all the host port registers being accessed with a single chip select and four address lines, as far as the local bus is concerned, the DSP host port is akin to an asynchronous memory mapped region. So, for the HDI16 port in single strobe mode, the host device asserts a chip select, a single data strobe and a read/write line to select HDI16 read or write bus operations.

The read and write strobes are also used as the data latch control to complete the bus transactions, obviating the need for any handshake termination signal from the DSP. The UPM programmer is responsible for satisfying the AC timings of the HDI16 transactions.

Through appropriate mode selection, the HDI16 peripheral's feature set can be fully supported by the local bus's UPM controlled signals. The UPM defined interface can be used with any of the local bus's eight chip selects to give the 16-bit port size and strobe generation that matches that of the HDI16 host port.

Figure 12-80 shows the internal register diagram of the HDI16.

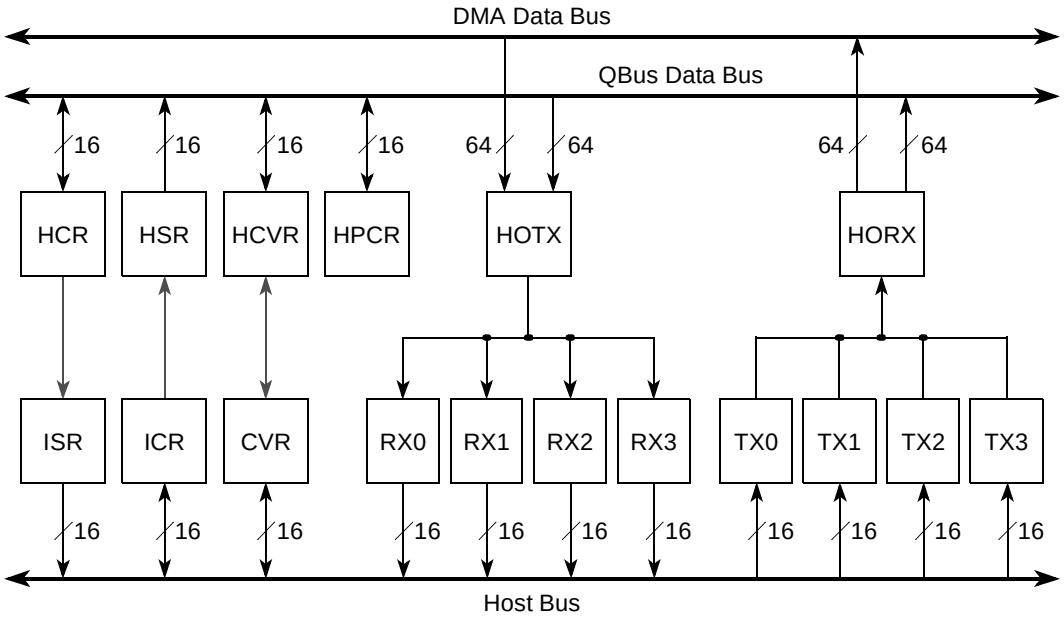


Figure 12-80. MSC8101 HDI16 Peripheral Registers

12.5.6.1.2 Physical Interconnections

The physical interconnections between the UPM controlled local bus and the HDI16 of the MSC8101 HDI16 peripheral are given in Table 12-44 and Figure 12-81.

Table 12-44. 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

Table 12-44. Local Bus to MSC8101 HDI16 Connections (continued)

MSC8101 Signal(s)	Type	Description	Connect with Local Bus Signal
$\overline{\text{HDS}}$	I	Host data strobe signal	LGPLy
HRRQ/HACK	O	Receive host request OP	As required in application
HTRQ/HREQ	O	Transmit host request OP	As required in application

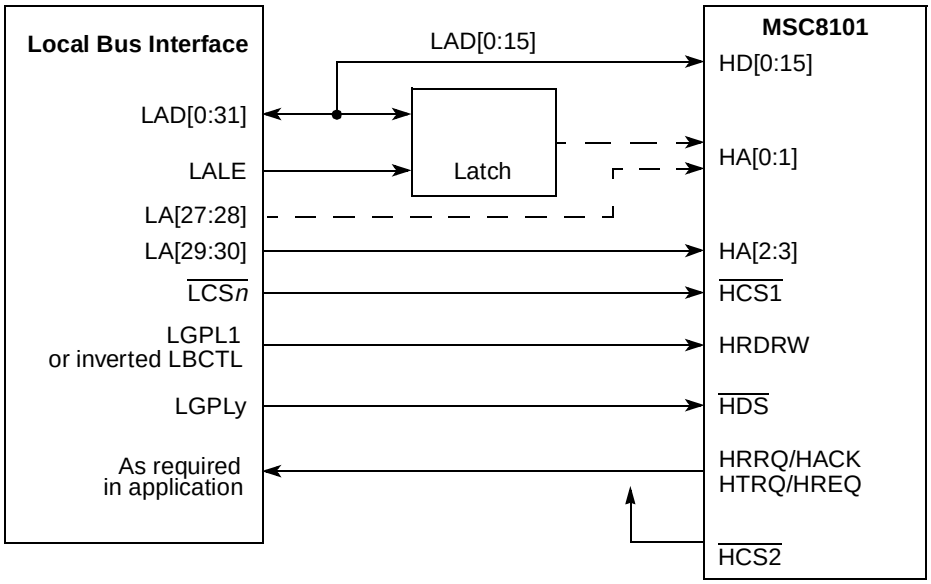


Figure 12-81. Interface to MSC8101 HDI16

The connections are specified as follows:

- One chip select, $\overline{\text{LCS}n}$ (whatever is available in the system), is used to memory map accesses from the host local bus to the HDI16 MSC8101 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/$\overline{\text{W}}$ signal. This can be generated 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 ($\overline{\text{HDS}}$), which must be asserted every 16-bit read or write transaction.
- Data lines—The bus data lines ($\overline{\text{LAD}n}$) are directly connected to the HDI16 data lines ($\overline{\text{HD}n}$).
- DMA request/service request signals (HRRQ and HTRQ)—as appropriate in the application

- Address lines—The address lines between the LBC and the HDI16 MSC8101 can either be connected in a straightforward or a specific manner to enable burst transfers across the HDI16. The connections are defined as follows and are described in more detail in the following sub-section:
  - Either LA[27] or latched value of A25 -> HA0
  - Either LA[28] or latched value of A26 -> HA1
  - LA[29] -> HA2
  - LA[30] -> HA3
- Ground lines—In order to provide the best ground plane, it is highly advised that all grounds are common and connected together.

12.5.6.1.3 Supporting Burst Transfers

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

For example, host transmit register 0 on the HDI16 peripheral (address 0x04) can be obtained by the host by accessing any of the following memory mapped addresses: 0x20, 0x28, 0x30, or 0x38. This is critical for burst accesses as the source or destination addresses increment after each 16-bit access to the interface for all 16 transactions within that burst. Using the addressing as defined, if the first access is at 0x20, the last is at 0x3e but, more importantly, the four host Tx/Rx registers are looped around four times.

12.5.6.1.4 Host 60x Bus: HDI16 Peripheral Interface Hardware Timings

The host UPM-controlled local bus and the HDI16 MSC8101 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 ($\overline{\text{HDS}}$ assert).

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

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

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

12.5.6.2 Interfacing to MSC8102 DSI

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

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

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

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

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

12.5.6.2.1 DSI in Asynchronous SRAM-Like Mode

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

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

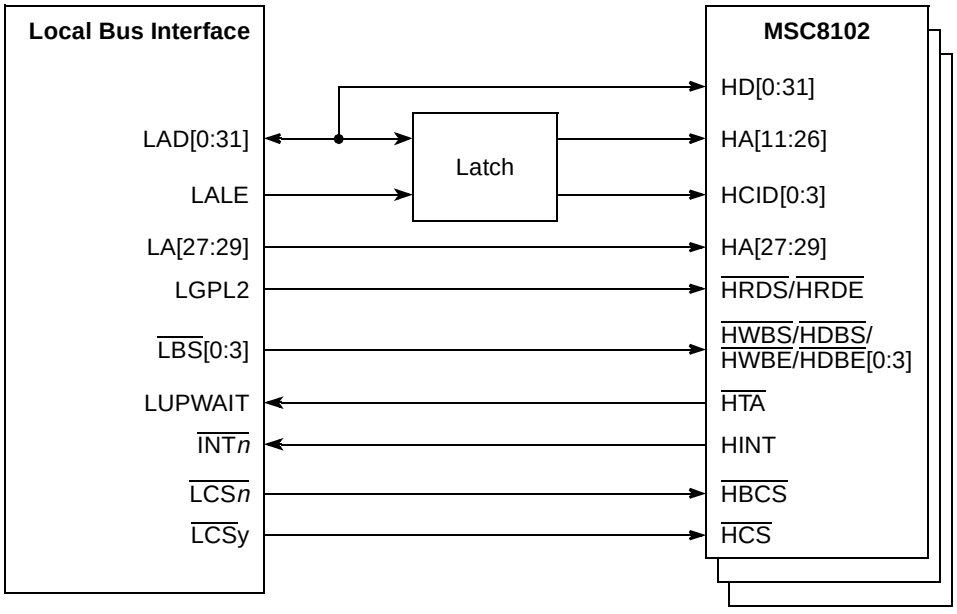


Figure 12-82. Interface to MSC8102 DSI in Asynchronous Mode

The asynchronous SRAM-like mode of the DSI is inherently slower than the synchronous mode and should be used, if only relatively small amounts of data are transferred between the communications controller and the MSC8102. To allow for maximum timing flexibility, the UPM machine of the LBC should be used.

The UPM programmer is responsible for ensuring correct setup and hold timings for all signals. The UPM allows sufficient control to satisfy any requirements here.

Figure 12-83 shows an asynchronous write access. The DSI samples the host chip ID signals (HCID[0:3]) on the first falling edge of the host write byte strobe signals ($\overline{\text{HWBS}}$) on which the host chip select signal ($\overline{\text{HCS}}$) is asserted. If the HCID[0:3] signals match the CHIPID value, the DSI is accessed. The DSI signals with the assertion of the host transfer acknowledge signal ($\overline{\text{HTA}}$), whether it is ready to sample the host data bus (HD), and the host can terminate the access by immediately negating $\overline{\text{HWBS}}$. The WAEN feature of the UPM must be used to insert wait states while the DSI is busy. The UWPL bit in the M*x*MR must be cleared to interpret the correct polarity of $\overline{\text{HTA}}$. The DSI samples the host address bus (HA) and the host data bus (HD) on the rising edge of $\overline{\text{HWBS}}$. In addition the assertion of $\overline{\text{HWBS}}$[0:3] are sampled at the end and are part of the access attributes.

Because the UPM is used for this mode, the DCR[4]:HTAAD should be set to 1 and DCR[9–10]:HTADT should be defined to a value different than 00. This mode is to be used in implementations with a pull-up resistor on $\overline{\text{HTA}}$. The host can start its next access (back-to-back accesses) without negating $\overline{\text{HCS}}$ between accesses. If the next access is not to the same MSC8102, then to prevent contention on the $\overline{\text{HTA}}$ signal, the host must wait until the previous DSI stops driving $\overline{\text{HTA}}$ before it accesses the next device. If the next access is to the same MSC8102, the host must not start consecutive accesses before $\overline{\text{HTA}}$ is actively driven to a value of 1 by the previous access. The easiest way to achieve this is to insert idle cycles at the end of the UPM pattern to guarantee that $\overline{\text{HTA}}$ is inactive.

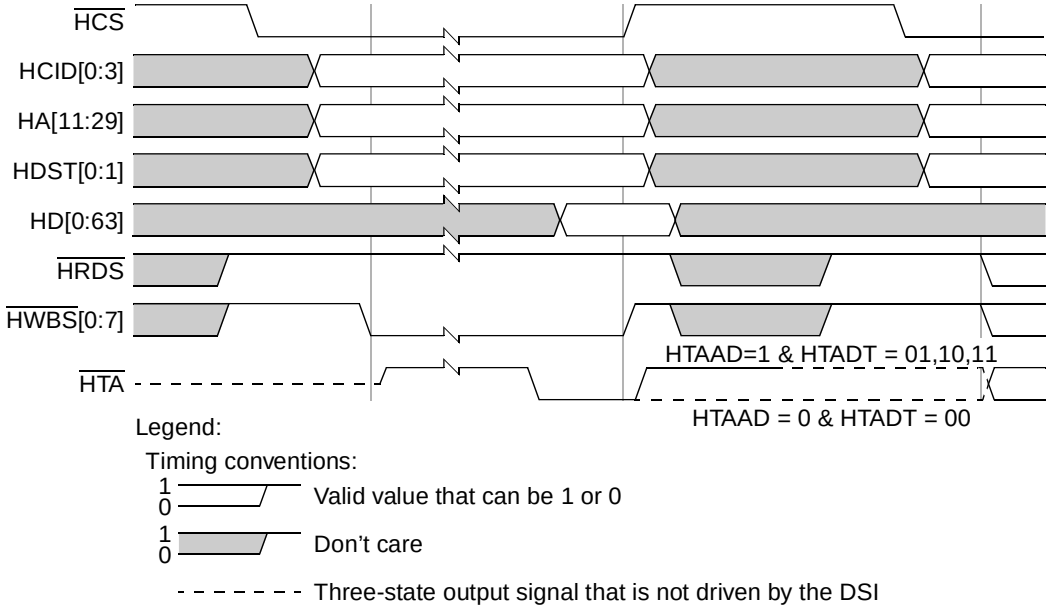


Figure 12-83. Asynchronous Write to MSC8102 DSI

Figure 12-84 shows an asynchronous read access. The DSI samples the host address bus (HA) and the HCID on the first falling edge of the host read strobe signal ($\overline{\text{HRDS}}$) on which the $\overline{\text{HCS}}$ is asserted. If HCID[0:3] match the CHIPID value, the DSI is accessed. When the DCR[8]:RPE bit is set, read access to the memory space (not to the register space) initiates data prefetching from consecutive addresses in the internal memory space. The DSI signals (with the assertion of the host transfer acknowledge signal ($\overline{\text{HTA}}$)) that data is valid, and the host can sample the host data bus (HD) and terminate the access by negating $\overline{\text{HRDS}}$. If the data for this access is already in the read buffer due to the prefetch mechanism, assertion time of $\overline{\text{HTA}}$ is improved. The WAEN feature of the UPM must be used to insert wait states while the DSI is busy. M*x*MR[UWPL] has to be cleared to interpret the correct polarity of the $\overline{\text{HTA}}$ signal.

Because the UPM is used in the mode, the DCR[4]:HTAAD should be set to 1 and the drive time control field, DCR[9–10]:HTADT, should be defined to a value different than 00. This mode is specially designed to be used for implementations with a pull-up resistor on $\overline{\text{HTA}}$.

The host can start its next access (back-to-back accesses) without negating the $\overline{\text{HCS}}$ signal between accesses. If the next access is not to the same MSC8102, then to prevent contention on $\overline{\text{HTA}}$, 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 access before $\overline{\text{HTA}}$ 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 $\overline{\text{HTA}}$ is inactive.

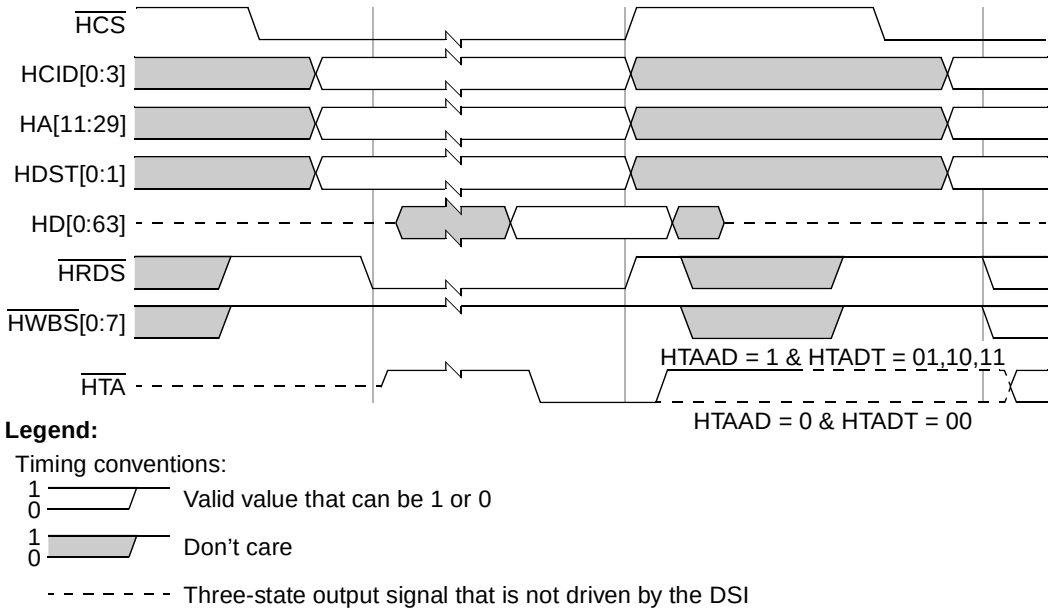


Figure 12-84. Asynchronous Read from MSC8102 DSI

12.5.6.2.2 DSI in Synchronous Mode

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

Figure 12-85 shows the interface for synchronous mode. Because the DSI asserts and negates $\overline{\text{HTA}}$ in synchronous mode even within a burst transfer on a clock-by-clock basis and because the DSI expects the host to react within one clock cycle, some tricks can be implemented to support the synchronous mode.

$\overline{\text{HTA}}$ drives LUPWAIT of the UPM. M*x*MR[UWPL] must be cleared to interpret the correct polarity of $\overline{\text{HTA}}$. Because this signal influences the internal state machine of the local bus clock, the local bus cannot react to $\overline{\text{HTA}}$ changes correctly within one local bus clock. Refer to Section 12.4.4.4.10, "Wait Mechanism (WAEN)," for more detailed information.

The solution to this lies in that the local bus operates at a higher frequency than the DSI interface of the DSP. The local bus clock can be divided by an integer divider (1:2, 1:3, or 1:4) to generate the DSI clock. This should not be a problem because the local bus is designed for much higher frequencies than the DSI. Because all timings are given in DSP DSI clock cycles, the UPM patterns must be adjusted appropriately and need to assert a signal for 2, 3, or 4 clocks (as many as the divider ratio) instead of one. Fortunately, the UPM has the REDO feature, which allows every UPM RAM entry to be executed 1×, 2×, 3×, or 4×, which should be sufficient for any divider ratio that would be used in this case.

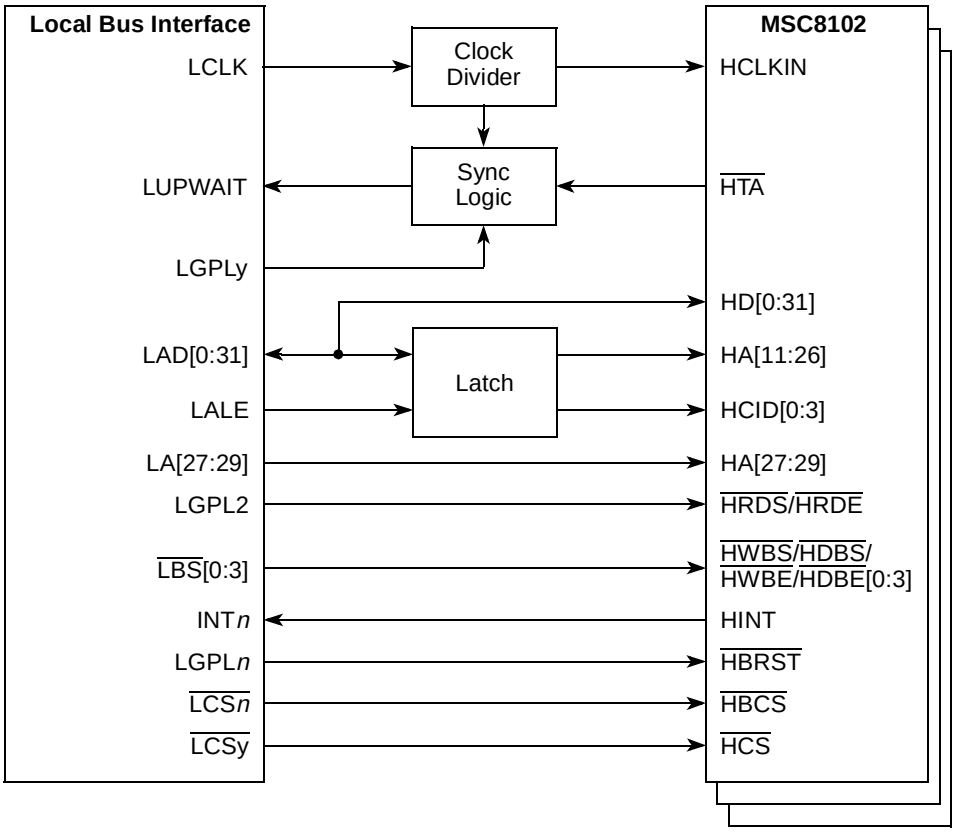


Figure 12-85. Interface to MSC8102 DSI in Synchronous Mode

This solution allows the local bus to react within multiple local bus clocks to the $\overline{\text{HTA}}$ signal and be still within one DSI clock. Typically, LUPWAIT is synchronized internally, and only 2 clocks after LUPWAIT changes, new data can be sampled or presented. For example, if the local bus clock ratio is 3× the DSI clock, data can be sampled in the third local bus sub-clock, which is the last third of the DSI clock. If the local bus clock ratio is only 2× the DSI clock, there is a special mode, where the LUPWAIT is NOT

synchronized. Refer to Section 12.4.4.5, “Synchronous Sampling of LUPWAIT for Early Transfer Acknowledge,” for more detailed information. In this mode, data is sampled in the second sub-clock, which is the second half of the DSI clock. AC timing of LUPWAIT must be met in this mode; otherwise indeterministic behavior may occur.

The remaining issue is the synchronization of the UPM cycles to the beginning of the DSI clock cycle. Because the UPM executes n cycles for every cycle of the DSI, a mechanism must be used to ensure that the UPM changes transitions in a way that is synchronized to the DSI clock. The solution is to use a special synchronize cycle at the beginning of the pattern. A GPL signal is used to control a multiplexer and to activate external synchronization logic, which uses the DSI clock to stall the UPM by asserting LUPWAIT until the beginning of the next DSI cycle. After that, this GPL signal must be negated and the multiplexer connects LUPWAIT to $\overline{\text{HTA}}$ instead, for the rest of the bus cycle. Note that the GPL signals should be used in the inverted state of their inactive state (GPL[0:4] are 1 when inactive, GPL5 is 0 when inactive) to start the synchronization process.

Figure 12-86 shows an example for a synchronization mechanism for a clock divider of 3. Note that the length of the synchronization cycle depends on the relative start of the synchronization process and varies with every access. It can vary in length from one to n (clock ratio) local bus clocks.

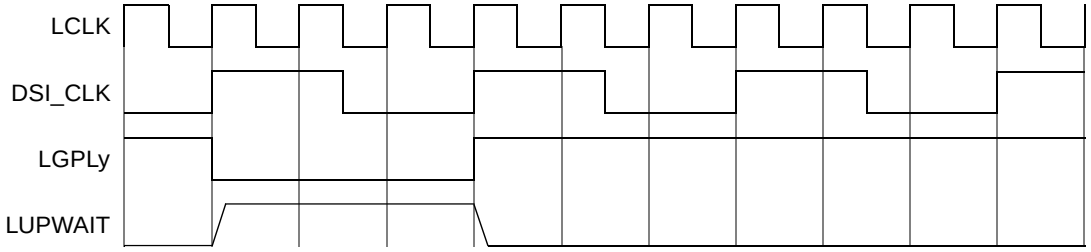


Figure 12-86. UPM Synchronization Cycle

The second column (compensation cycle) of Table 12-45 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.

Table 12-45. UPM Synchronization Cycles

	Sync Cycle	Compensation Cycle	DSI Cycle 1	Bits
cst1–cst4	—	—	—	0–3
bst1–bst4	—	—	—	4–7
g0xx	—	—	—	8–11
g1tx	—	—	—	12–13
g2tx	—	—	—	14–15
g3tx	—	—	—	16–17
g4t1	—	—	—	18
g4t3	**1**	**0**	—	19
g5tx	—	—	—	20–21

Table 12-45. UPM Synchronization Cycles (continued)

	Sync Cycle	Compensation Cycle	DSI Cycle 1	Bits
redo[0]	0	0	**1**	22
redo[1]	0	0	**0**	23
loop	0	0	—	24
exen	0	—	—	25
amx0	0	0	—	26
amx1	0	0	—	27
na	0	—	—	28
uta	0	—	—	29
todt	0	—	—	30
last	0	—	—	31

This section describes synchronous single write and read, and synchronous burst write and read operations.

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

Synchronous Single Write

Figure 12-87 shows a synchronous single write access.

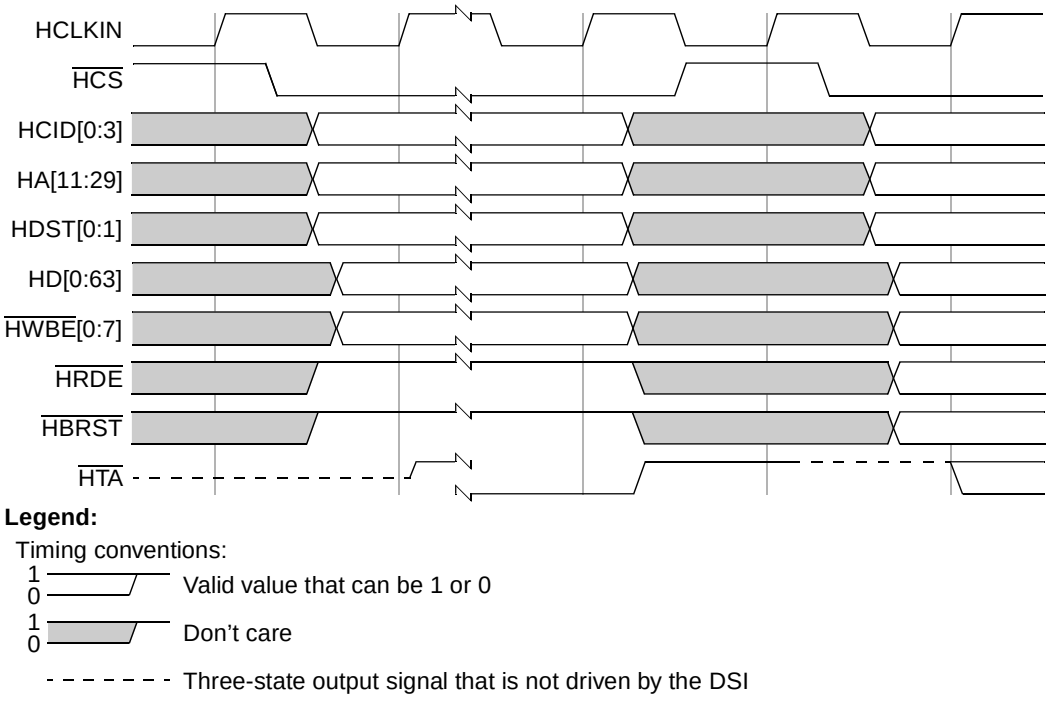


Figure 12-87. Synchronous Single Write to MSC8102 DSI

The DSI samples HA, HDST, HCID, HD, $\overline{\text{HWBE}}$, $\overline{\text{HRDE}}$, and $\overline{\text{HBRST}}$ on the first HCLKIN rising edge on which $\overline{\text{HCS}}$ is asserted. If HCID[0:3] match the CHIPID value, the DSI is accessed. At least one $\overline{\text{HWBE}}$ signal is asserted, and $\overline{\text{HRDE}}$ and $\overline{\text{HBRST}}$ are negated. Assertion of $\overline{\text{HTA}}$ indicates that the DSI is ready to complete the current access and the host must terminate this access. Because $\overline{\text{HTA}}$ is connected to the 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. Typically, $\overline{\text{HTA}}$ is asserted immediately. If the write buffer is full, $\overline{\text{HTA}}$ assertion is delayed. $\overline{\text{HTA}}$ is asserted for one HCLKIN cycle, driven to logic 1 in the next cycle, and stops being driven on the next rising edge of HCLKIN. The host can start its next access to the same MSC8102 immediately on the next HCLKIN rising edge without negating $\overline{\text{HCS}}$ between accesses. If the next access is not to the same MSC8102, then, to prevent contention on $\overline{\text{HTA}}$, the host must wait to access the next device until the previous DSI stops driving $\overline{\text{HTA}}$. The easiest way to achieve this is to insert idle cycles at the end of the UPM pattern to guarantee that $\overline{\text{HTA}}$ is inactive.

Synchronous Single Read

Figure 12-88 shows a synchronous single read access.

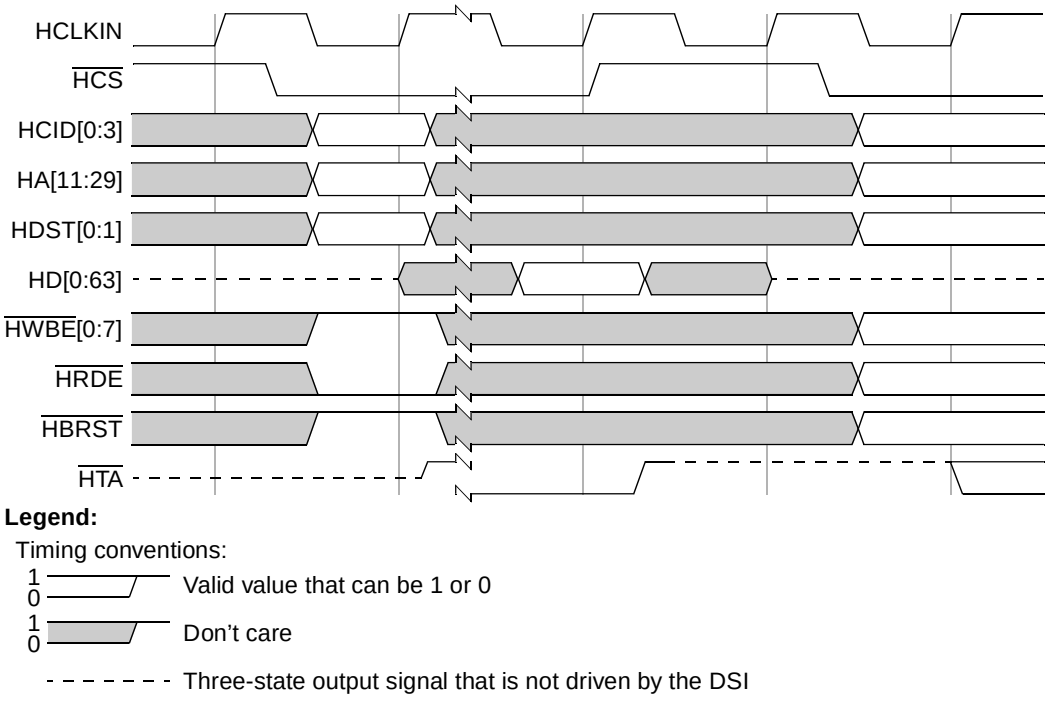


Figure 12-88. Synchronous Single Read from MSC8102 DSI

The DSI samples HA, HDST, HCID, $\overline{\text{HWBE}}$, $\overline{\text{HRDE}}$, and $\overline{\text{HBRST}}$ on the first HCLKIN rising edge on which $\overline{\text{HCS}}$ is asserted. If the HCID[0:3] signals match the CHIPID value, the DSI is accessed. $\overline{\text{HRDE}}$ is asserted, and $\overline{\text{HWBE}}$ and $\overline{\text{HBRST}}$ are negated. 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 $\overline{\text{HTA}}$ indicates that data is valid and the host must sample the HD and terminate the access. Because $\overline{\text{HTA}}$ is connected to the UPM LUPWAIT signal, all local bus signals are frozen until $\overline{\text{HTA}}$ goes to 0; then the UPM continues in its pattern. $\overline{\text{HTA}}$ is asserted earlier when the data for this access is already prefetched to the read buffer. It asserted for one HCLKIN cycle and

driven to logic 1 in the next cycle. It stops being driven on the next rising edge of HCLKIN. The host can start its next access to the same MSC8102 immediately in the next HCLKIN rising edge without negating HCS between accesses. If the next access is not to the same MSC8102, then, to prevent contention on HTA, the host must wait to access the next device until the previous DSI stops driving HTA. The easiest way to achieve this is to insert idle cycles at the end of the UPM pattern to guarantee that HTA is inactive.

Synchronous Burst Write

Figure 12-89 shows a synchronous burst write access.

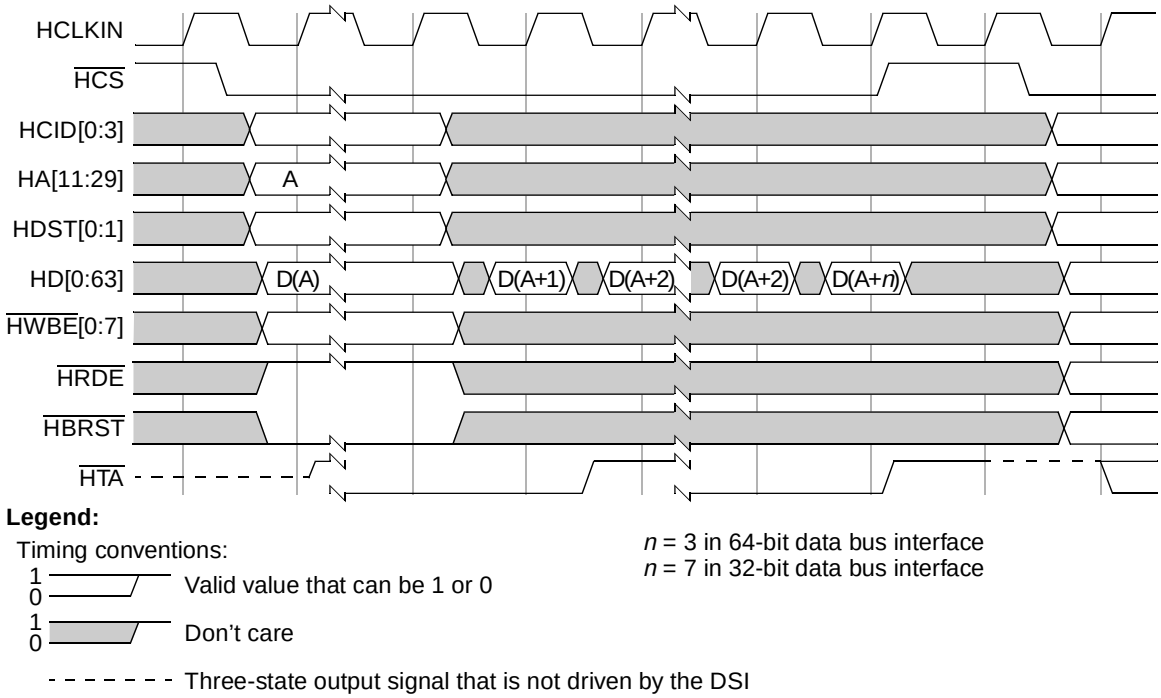


Figure 12-89. Synchronous Burst Write to MSC8102 DSI

The DSI samples HA, HDST, HCID, HD, 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 12-90 shows a synchronous burst read access. The DSI samples HA, HDST, HCID, $\overline{\text{HWBE}}$, $\overline{\text{HRDE}}$, and $\overline{\text{HBRST}}$ on the first HCLKIN rising edge on which $\overline{\text{HCS}}$ is asserted. If HCID[0:3] match the CHIPID value, the DSI is accessed.

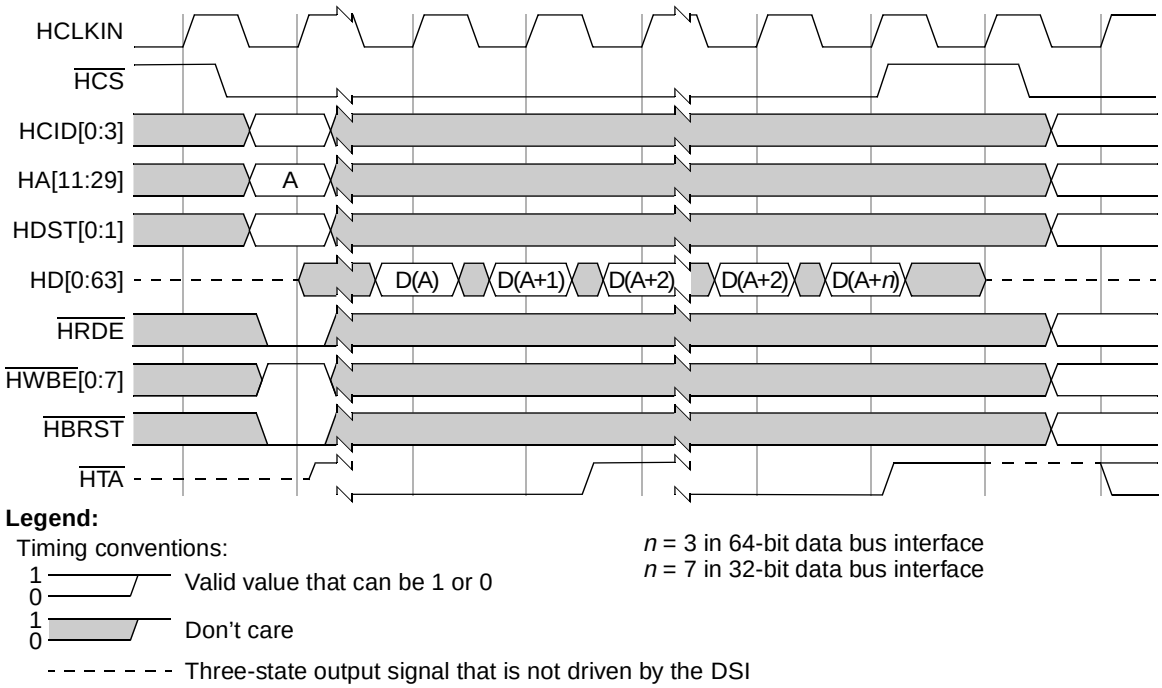


Figure 12-90. Synchronous Burst Read from MSC8102 DSI

$\overline{\text{HRDE}}$ and $\overline{\text{HBRST}}$ are asserted and $\overline{\text{HWBE}}$ are negated. When the DCR[8]:RPE bit (see the MSC8102 documentation) is set, a burst read access initiates data prefetching from consecutive addresses in the internal memory space. Assertion of $\overline{\text{HTA}}$ indicates that data is valid for the current beat of the access and the host must proceed to the next beat of this access. Because $\overline{\text{HTA}}$ is connected to the LUPWAIT signal of the UPM, all local bus signals are frozen until $\overline{\text{HTA}}$ goes to 0 and then the UPM continues in its pattern. When the host reaches the last beat of the access, it must terminate the burst access. The $\overline{\text{HTA}}$ is asserted earlier when the data for this access is already prefetched to the read buffer. Typically, after the first beat of the burst access, $\overline{\text{HTA}}$ remains asserted until the end of the access. After the last beat of the access, $\overline{\text{HTA}}$ is driven to 1 and stops being driven in the next rising edge of HCLKIN. The host can start its next access to the same MSC8102 immediately in the next HCLKIN rising edge without negating $\overline{\text{HCS}}$ between accesses. If the next access is not to the same MSC8102, to prevent contention on $\overline{\text{HTA}}$, the host must wait to access the next device until the previous DSI stops driving the $\overline{\text{HTA}}$ signal. The easiest way to achieve this is insert idle cycles at the end of the UPM pattern to guarantee that $\overline{\text{HTA}}$ is inactive.

12.5.6.2.3 Broadcast Accesses

Using $\overline{\text{HBCS}}$, a host can share one chip-select signal between multiple MSC8102 devices for broadcasting write accesses. In broadcast mode, the DSI does not drive its $\overline{\text{HTA}}$ signal to prevent contention between multiple devices driving different values to the same signal. Also, the DSI does not decode HCID[0:3].

Note that broadcasting is allowed only for write accesses.

The DSI sets the DSI error register (DER) OVF bit if there is an overflow during broadcast accesses. This bit can be cleared by writing a value of 1 to it.

NOTE

To avoid overflow when accessing DSI registers during broadcast accesses, wait at least 10 host clock cycles in synchronous mode or 8 internal clock cycles in asynchronous mode between each DSI register access.

To avoid data corruption, if DER[0]:OVF is set, no broadcast access is written until the bit is reset. Therefore, after the last broadcast access, and before any regular write access, DER[0]:OVF must first be read and reset if it is set.

NOTE

In asynchronous mode, write data from a previous access (even a normal write access) may be lost due to overflow during broadcast accesses. To prevent such a loss, ensure that previous access data has propagated to the FIFO or DSI registers, depending on the type of previous access. This can be achieved by performing a read access prior to the first broadcast access.

In broadcast accesses, the host must comply with the following rules:

- In asynchronous mode, $\overline{\text{HWBS}}$[0:3]/$\overline{\text{HDBS}}$[0:3] assertion time should be at least the minimum, which is defined in the AC characteristics section of the *MSC8102 Technical Data* sheet.
- In synchronous mode single access, the host must wait 1 cycle before terminating the access. Access signals must be in the same valid state during two positive edges of the host clock cycles. Access duration is two clock cycles (the DSI may translate accesses lasting longer than two clock cycles as two or more back-to-back accesses).
- In synchronous mode burst accesses, broadcast accesses are not allowed.

Chapter 13
Enhanced Three-Speed Ethernet Controllers

13.1 Overview

The enhanced three-speed Ethernet controllers (eTSECs) of the device interface to 10 Mbps, 100 Mbps, and 1 Gbps Ethernet/IEEE 802.3 networks and devices featuring generic 8-16-bit FIFO ports. For Ethernet, an external PHY or SerDes device is required to complete the interface to the media. Each eTSEC supports multiple standard media-independent interfaces, of which the FIFO interface bypasses the Ethernet MAC. Multiple eTSECs are available, providing flexible options for connectivity and control access at different speeds.

The eTSEC provides the flexibility to accelerate the identification and retrieval of standard and non-standard protocols carried over Ethernet, including both IP versions 4 and 6 and TCP/UDP. CPU-intensive parsing and checksum operations can be optionally off-loaded to an eTSEC to accelerate existing TCP/IP stacks. On transmission, varying fractions of link bandwidth can be allocated to each of multiple transmit queues through a modified weighted round-robin scheduler. On receive, an arbitrary set of queue selection rules can be programmed into each eTSEC to implement flexible quality of service or firewall strategies based on high-level protocol identification. Without enabling these advanced features, each eTSEC emulates a PowerQUICC III TSEC, allowing existing driver software to be re-used with minimal change. Each eTSEC is organized as shown in Figure 13-1.

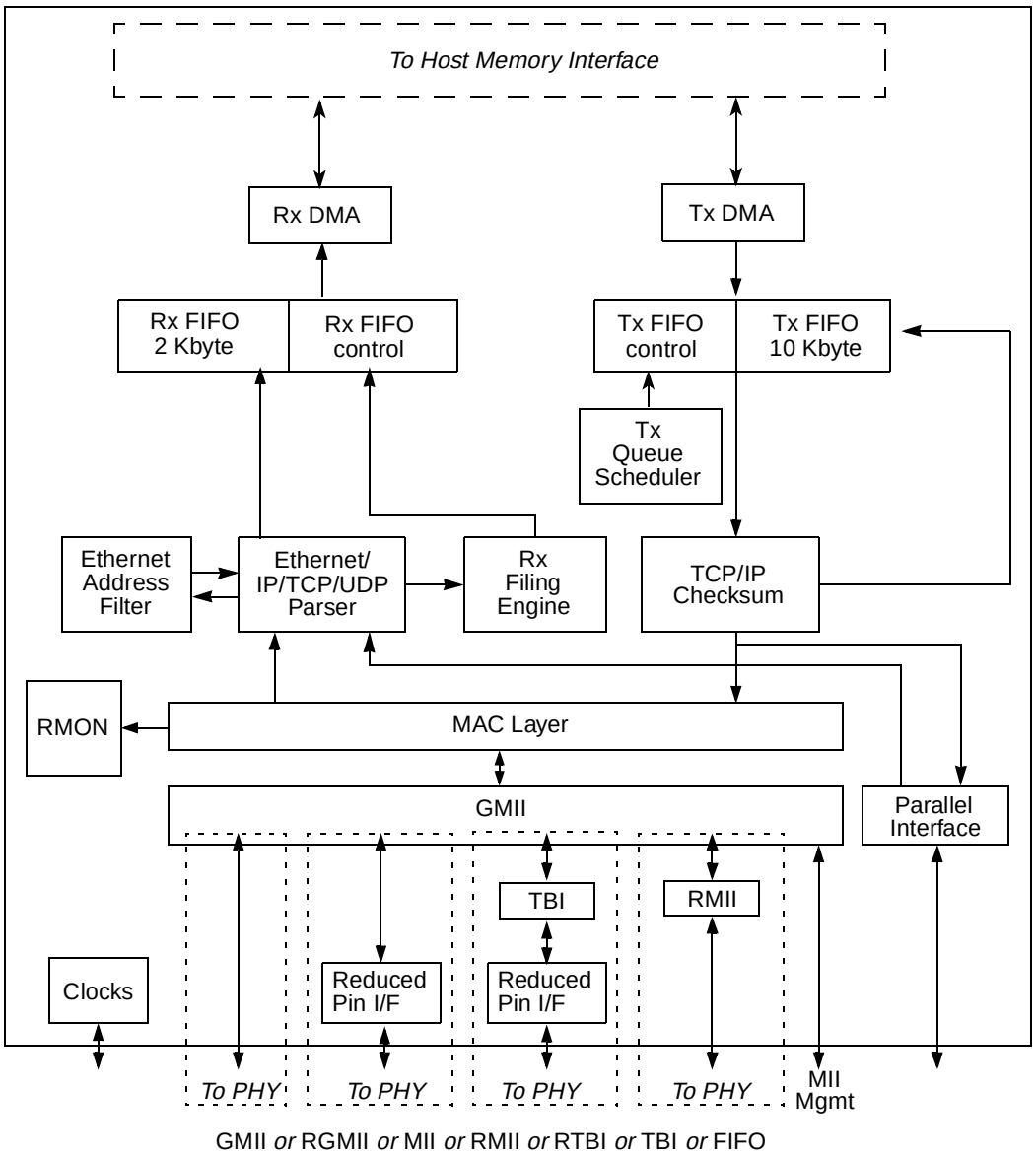


Figure 13-1. eTSEC Block Diagram

13.2 Features

The eTSECs of the device include these distinctive features:

- IEEE 802.3, 802.3u, 820.3x, 802.3z, 802.3ac, 802.3ab compatible
- Support for different Ethernet physical interfaces:
  - 10/100 Mbps IEEE 802.3 GMII
  - 1000 Mbps full-duplex IEEE 802.3 GMII
  - 10/100 Mbps IEEE 802.3 MII and RMII
  - 10/100 Mbps RGMII

  - 1000 Mbps full-duplex RGMII and RTBI
  - 1000 Mbps IEEE 802.3z TBI
  - Single-clock TBI
- Support for two full-duplex FIFO interface modes
  - 8-bit mode—GMII style and encoded packet
  - 16-bit mode—GMII style and encoded packet
  - Inter-packet and intra-packet flow control
  - Optional CRC-32 generation and checking
  - Minimal glue logic required to support POS PHY Level 3 conversion
  - TCP/IP off-load and QoS features available in all FIFO modes, at up to OC-48 rates
- TCP/IP off-load
  - IP v4 and IP v6 header recognition on receive
  - IP v4 header checksum verification and generation
  - TCP and UDP checksum verification and generation
  - Per-packet configurable off-load
  - Recognition of VLAN, stacked-VLAN, 802.2, PPPoE session, MPLS stacks, and ESP/AH IP-Security headers
- Quality of service (QoS) support
  - Transmission from up to eight queues
    - Priority-based queue selection
    - Modified weighted round-robin queue selection with fair bandwidth allocation
  - Reception to up to eight physical queues
    - 64 virtual receive queues overlaid on 8 physical buffer descriptor rings
    - Table-oriented queue filing strategy based on 16 header fields or flags
    - Frame rejection support for filtering applications
    - Filing based on Ethernet, IP, and TCP/UDP properties, including VLAN fields, Ether-type, IP protocol type, IP TOS or differentiated services, IP source and destination addresses, TCP/UDP port numbers
- Interrupt coalescing
  - Packet-count-based thresholds for both receive and transmit
  - Timer-based thresholds
- Full- and half-duplex Ethernet support (1000 Mbps supports only full duplex):
  - IEEE 802.3 full-duplex flow control (automatic PAUSE frame generation or software programmed PAUSE frame generation and recognition)
  - Programmable maximum frame length supports jumbo frames (up to 9.6 Kbytes) and IEEE 802.1 virtual local area network (VLAN) tags and priority
  - VLAN insertion and deletion
    - Per-frame VLAN control word or default VLAN for each eTSEC

  - Extracted VLAN control word passed to software separately
  - Programmable VLAN tag to support metropolitan bridging
- Retransmission following a collision
- Support for CRC generation and verification of inbound/outbound packets
- Programmable Ethernet preamble insertion and extraction of up to 7 bytes

- MAC address recognition:
  - Exact match on primary and virtual 48-bit unicast addresses
    - VRRP and HSRP support for seamless router fail-over
    - In addition to primary station address, up to fifteen additional exact-match MAC addresses supported
  - Broadcast address (accept/reject)
  - Hash table match on up to 256 unicast/multicast or 512 multicast-only addresses
  - Promiscuous mode
- Remote network monitoring (RMON) statistics support
  - 32-bit byte counters
  - Carry/Overflow of counter interrupts
- Backward compatibility with MPC8540E/MPC8560E (PowerQUICC III) TSEC
  - PowerQUICC III buffer descriptor (BD) format and rings supported
  - Common register memory map, with specific exceptions:
    - Out-of-sequence transmit BD not supported
    - Internal DMA BD pointers and data counts not visible
    - MINFLR register not supported
  - Reset state of eTSEC defaults to common PowerQUICC III TSEC subset
  - TSEC_ID register permits TSEC versus enhanced TSEC differentiation

13.3 Modes of Operation

The eTSEC's primary operational modes are the following:

- Ethernet and FIFO operation

  The ECNTRL register's FIFO mode enable bit (ECNTRL[FIFM]) allows bypass of the Ethernet MAC and enables I/O through the FIFO interface sharing the normal GMII signals. Each eTSEC supports an 8-bit FIFO interface independently. Pairs of GMII ports can be combined to create a 16-bit, full-duplex interface. If configured in FIFO mode, the FIFOCFG register determines operation. In FIFO mode data is transferred synchronously with respect to the external data clock. See the device hardware specifications document for maximum supported frequencies.
- Full- and half-duplex operation

  This is determined by the MACCFG2 register's full-duplex bit (MACCFG2[Full Duplex]). Full-duplex mode is intended for use on point-to-point links between switches or end node to switch. Half-duplex mode is used in connections between an end node and a repeater or between repeaters.

If configured in half-duplex mode (10- and 100-Mbps operation; MACCFG2[Full Duplex] is cleared), the MAC complies with the IEEE CSMA/CD access method.

If configured in full-duplex mode (10/100/1000 Mbps operation; MACCFG2[Full Duplex] is set), the MAC supports flow control. If flow control is enabled, it allows the MAC to receive or send PAUSE frames.

- 10- and 100-Mbps MII interface operation

  The MAC–PHY interface operates in MII mode by setting MACCFG2[I/F Mode] = 01. The MII is the media-independent interface defined by the 802.3 standard for 10/100 Mbps operation. The speed of operation is determined by the TSEC*n*_TX_CLK and TSEC*n*_RX_CLK signals, which are driven by the transceiver. The transceiver either auto-negotiates the speed, or it may be controlled by software using the serial management interface (MDC/MDIO signals) to the transceiver.

  Clause 22.2.4 of the IEEE 802.3 specification describes the MII management interface.

- 10- and 100-Mbps RMII interface operation

  The RMII is the reduced media-independent interface defined by the RMII Consortium (March 1998) for 10/100 Mbps operation. The speed of operation is determined by the TSEC*n*_TX_CLK signal, which is driven by the transceiver.

- 1000 Mbps GMII and TBI interface operation

  The MAC–PHY interface operates in GMII mode by setting MACCFG2[I/F Mode] = 10. The GMII is the gigabit media-independent interface defined by the 802.3 standard for 1000-Mbps operation.

  Independently, the MAC-PHY interface can also operate in TBI mode. Note that either the TBI or GMII interface is chosen, not both at the same time. TBI is the 10-bit interface which contains PCS functions (10-bit encoding/decoding) as defined by the 802.3 standard.

  In reduced-pin count mode (RGMII or RTBI), the MAC remains configured in GMII or TBI but the eTSEC muxes and decodes the input signals and provides the MAC with the expected interface.

  eTSEC provides the TSEC*n*_GTX_CLK to the PHY in either GMII or TBI mode of operation.

- MAC address recognition options

  The options supported are promiscuous, broadcast, exact unicast address match, exact unicast virtual address match to support router redundancy, and multicast hash match. For detailed descriptions refer to Section 13.6.3.7, “Frame Recognition.”

  eTSEC supports automatic LAN-initiated wake-up during power management through the AMD Magic Packet™ protocol, as described in Section 13.6.3.8, “Magic Packet Mode.”

- Receive frame parsing options

  Frame parsing options are to disable parsing (no TCP/IP off-load), IP header parsing, and TCP or UDP parsing. Parsing must be enabled to make use of receive queue filing algorithms. The options are detailed in Section 13.6.4, “TCP/IP Off-Load.”

- Receive queue selection options

  Received frames are by default sent to a single buffer descriptor ring. If multiple receive queues are enabled, a receive queue filer can be programmed with selection criteria to differentiate

received frames and file them to different buffer descriptor rings. See Section 13.6.5, “Quality of Service (QoS) Provision,” for detailed descriptions.

- TCP/IP transmit options

  Frames for transmission may be sent as-is, with IP header processing, or TCP header processing. The transmit buffer descriptors, described in Section 13.6.7.2, “Transmit Data Buffer Descriptors (TxBD),” enable these options and operate with parameters prepended to frame buffers, as described in Section 13.6.4, “TCP/IP Off-Load.”
- Transmit queue selection options

  The options supported are single transmit queue, priority-based queue selection, and modified weighted round-robin queueing. These options are described further in Section 13.5.3.2.1, “Transmit Control Register (TCTRL).”
- RMON support

  Standard Ethernet interface management information base (MIBs) can be generated through the RMON MIB counters.
- Internal loop back supported for all interfaces except when configured for half-duplex operation

  Internal loop back mode is selected through the loop back bit in the MACCFG1 register. See Section 13.7.1, “Interface Mode Configuration,” for details.

13.4 External Signals Description

This section defines the eTSEC interface signals. The buses are described using the bus convention used in IEEE 802.3 because the PHY follows this same convention. (That is, TxD[7:0] means 0 is the lsb.) Note that except for external physical interfaces the buses and registers follow a big-endian format, where 0 denotes the msb.

Each eTSEC network interface supports multiple options:

- The MII option requires 18 I/O signals (including the MDIO and MDC MII management interface) and supports both a data and a management interface to the PHY (transceiver) device. The MII option supports both 10- and 100-Mbps Ethernet rates.
- The GMII option is a superset of the MII signals and supports a 1000-Mbps Ethernet rate.
- The TBI interface shares signals with the GMII interface signals.
- The RGMII, RTBI, and RMII options are reduced-pin implementations of the GMII, TBI, and MII interfaces, respectively.
- Finally, the FIFO interfaces share the GMII signals—8 bits of data plus 3 bits of control signals.

Table 13-1 lists the network interface signals.

Table 13-1. eTSEC*n* Network Interface Signal Properties

Signal Name	Function	Reset State
TSEC*n*_COL	MII—collision, input FIFO—transmit flow control, input	—
TSEC*n*_CRS	MII—carrier sense, input TBI—signal detect, input FIFO—receive flow control, output	—
TSEC*n*_GTX_CLK	RTBI, RGMII—inverted transmit clock feedback, output TBI, FIFO—continuous transmit clock feedback, output GMII, MII, RMII—transmit clock feedback when transmission is enabled, zero otherwise, output	0
EC*n*_GTX_CLK125	Oscillator sources for GMII, TBI, RGMII, RTBI transmit clock, input, shared by all eTSECs Note: EC1_GTX_CLK125 is shared by eTSEC1 and eTSEC2; EC2_GTX_CLK125 is shared by eTSEC3 and eTSEC4.	—
EC_MDC	Management clock, output.	0
EC_MDIO	Management data, bidirectional.	Hi-Z (input)
TSEC*n*_RX_CLK	GMII, MII, RGMII—receive clock, input TBI—PMA receive clock 0, input FIFO—receive clock, input	—
TSEC*n*_RX_DV	GMII, MII—receive data valid, input TBI—receive code group (RCG) bit 8, input RGMII (RX_CLK rising)—receive data valid, input RGMII (RX_CLK falling)—receive error, input RTBI (RX_CLK rising)—receive code group (RCG) bit 4, input RTBI (RX_CLK falling)—receive code group (RCG) bit 9, input RMII—CRS_DV carrier sense/data valid, input FIFO—receive data valid or receive control bit, input	—
TSEC*n*_RXD[7:4]	GMII—receive data bits 7:4 input TBI—RCG bits 7:4, input FIFO—receive data bits 7:4 input MII, RGMII, RTBI, RMII—unused	—
TSEC*n*_RXD[3:0]	GMII, MII—Receive data bits 3:0, input TBI—RGC bits 3:0, input RGMII (RX_CLK rising) —Receive data bits 3:0, input RGMII (RX_CLK falling)—Receive data bits 7:4, input RTBI (RX_CLK rising)—RCG bits 3:0, input RTBI (RX_CLK falling)—RCG bits 8:5, input RMII—RXD[1:0] receive data bits, input RMII—RXD[3:2] are unused FIFO—Receive data bits 3:0, input	—
TSEC*n*_RX_ER	GMII, MII, RMII—Receive error, input TBI—RGC bit 9, input FIFO—Receive error or receive frame control bit, input RGMII, RTBI—Unused	—

Table 13-1. eTSEC*n* Network Interface Signal Properties (continued)

Signal Name	Function	Reset State
TSEC*n*_TX_CLK	MII—transmit clock, input TBI—PMA receive clock 1, input RMII—reference transmit and receive clock, input FIFO—transmit clock, input RGMII, RTBI—unused	—
TSEC*n*_TXD[7:4]	GMII—transmit data bit 7:4, output TBI—transmit code group (TCG) bit 7:4, output FIFO—transmit data bit 7:4, output MII, RGMII, RTBI, RMII—unused, output driven zero	0000
TSEC*n*_TXD[3:0]	GMII, MII—Transmit data bits 3:0, output TBI—TCG bits 3:0, output RGMII (TX_CLK rising)—Transmit data bits 3:0, output RGMII (TX_CLK falling)—Transmit data bits 7:4, output RTBI (TX_CLK rising)—TCG bits 3:0, output RTBI (TX_CLK falling)—TCG bits 8:5, output RMII—TXD[1:0] transmit data bits, output RMII—TXD[3:2] unused, output driven zero FIFO—Transmit data bits 3:0, output	0000
TSEC*n*_TX_ER	GMII, MII—transmit error, output RGMII, RTBI, RMII—unused, output driven zero TBI—TCG bit 9, output FIFO—transmit error or transmit frame control bit, output	0
TSEC*n*_TX_EN	GMII, MII, RMII—Transmit data valid, output TBI—TCG bit 8, output RGMII (TX_CLK rising)—Transmit data enabled, output RGMII (TX_CLK falling)—Transmit error, output RTBI (TX_CLK rising)—TCG bit 4, output RTBI (TX_CLK falling)—TCG bit 9, output FIFO—Transmit data valid or transmit control bit, output	0

13.4.1 Detailed Signal Descriptions

Below is a description of the eTSEC interface signals. For RGMII mode details please refer to the Hewlett-Packard reduced gigabit media-independent interface (RGMII) specification version 1.2a, dated 9/22/2000. RMII mode details follow the RMII Consortium Specification, dated 3/20/1998. All other modes follow the IEEE 802.3 standard, 2000 Edition. Input signals not used are internally disabled. Except for TSEC*n*_GTX_CLK, output signals not used are driven low.

Table 13-2. eTSEC Signals—Detailed Signal Descriptions

Signal	I/O	Description	
TSEC*n*_COL	I	Collision input. The behavior of this signal is not specified while in full-duplex mode.	
		State Meaning	Asserted/Negated—In MII mode, this signal is asserted upon detection of a collision, and must remain asserted while the collision persists. In FIFO mode this signal is used to effect flow control on the transmitter. This signal is not used in the following modes: • RMII • GMII • TBI • RTBI • RGMII
		Timing	Asserted/Negated—This signal is not required to transition synchronously with TSEC*n*_TX_CLK or TSEC*n*_RX_CLK.
TSEC*n*_CRS	I	Carrier sense input. In TBI and RTBI modes, this signal is used as SDET (signal detect).In TBI mode SDET must be tied high externally on the board. In RTBI mode SDET is tied high internally. This signal is not used in the following modes: • RMII • GMII • RGMII	
		State Meaning	Asserted/Negated—In MII mode, TSEC*n*_CRS is asserted while the transmit or receive medium is not idle. In the event of a collision, TSEC*n*_CRS must remain asserted for the duration of the collision.
		Timing	Asserted/Negated—This signal is not required to transition synchronously with TSEC*n*_TX_CLK or TSEC*n*_RX_CLK.
	O	Receiver flow control signal in FIFO mode. This signal is not used in the eTSEC Ethernet modes.	
		State Meaning	Asserted/Negated—TSEC*n*_CRS is asserted while the FIFO receiver is unprepared to accept additional receive data.
		Timing	Asserted/Negated—This signal transitions synchronously with TSEC*n*_RX_CLK.
TSEC*n*_GTX_CLK	O	Gigabit transmit clock. This signal is an output from the eTSEC into the PHY. TSEC*n*_GTX_CLK is a 125-MHz clock that provides a timing reference for TX_EN, TXD, and TX_ER in the following modes: • GMII • TBI • RTBI In RGMII mode, TSEC*n*_GTX_CLK becomes the transmit clock and provides timing reference during 1000Base-T (125 MHz), 100Base-T (25 MHz) and 10Base-T (2.5 MHz) transmissions. This signal feeds back the uninverted transmit clock in MII or FIFO modes, but feeds back an inverted transmit clock in RTBI or RGMII modes. This signal is driven low unless transmission is enabled, or the eTSEC is in TBI or FIFO mode.	

Table 13-2. eTSEC Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description
EC*m*_GTX_CLK125	I	Gigabit transmit 125-MHz source. This signal must be generated externally with a crystal or oscillator, or is sometimes provided by the PHY. EC*m*_GTX_CLK125 is a 125-MHz input into the eTSEC and is used to generate all 125-MHz related signals and clocks in the following modes: • GMII • TBI • RTBI • RGMII EC1_GTX_CLK125 sources eTSECs 1 and 2, while EC2_GTX_CLK125 sources eTSECs 3 and 4. This input is not used in these modes: • FIFO • RMII • MII
EC_MDC	O	Management data clock. This signal is a clock (typically 2.5 MHz) supplied by the MAC (IEEE set minimum period of 400 ns or a frequency of 2.5 MHz, but the device may be configured up to 12.5 MHz if supported by the PHY at that speed.) The frequency can be modified by writing to MIIMCFG[28:31] of the eTSEC1 controller.
EC_MDIO	I/O	Management data input/output. **State Meaning**: Asserted/Negated—EC_MDIO is a bidirectional signal to input PHY-supplied status during management read cycles and output control during MII management write cycles. Addressed using eTSEC1 memory-mapped registers. **Timing**: Asserted/Negated—This signal is required to be synchronous with the 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. In RMII mode this clock is not used for the receive clock, as RMII uses a shared reference clock. In FIFO mode the receive clock is a continuous clock. See the device hardware specifications document for maximum supported frequencies.
TSEC*n*_RX_DV	I	Receive data valid. In GMII or MII mode, if TSEC*n*_RX_DV is asserted, the PHY is indicating that valid data is present on the GMII and MII interfaces. In RGMII mode, TSEC*n*_RX_DV becomes RX_CTL. The RX_DV and RX_ERR are received on this signal on the rising and falling edges of TSEC*n*_RX_CLK. In TBI mode, TSEC*n*_RX_DV represents receive code group (RCG) bit 8. Together, with RCG[9] and RCG[7:0], they represents the 10-bit encoded symbol of GMII receive signals. In RTBI mode, TSEC*n*_RX_DV represents receive code group (RCG) bit 4 and 9. On the positive edge of the TSEC*n*_RX_CLK, RCG[4] and RCG[3:0] represent the first half of the 10-bit encoded symbol. On the negative edge of the TSEC*n*_RX_CLK, RCG[9] and RCG[8:5] represent the second half of the 10-bit encoded symbol. In RMII mode the PHY asserts TSEC*n*_RX_DV (CRS_DV) when the receive medium is non-idle. This signal asserts asynchronously with respect to the RMII reference clock, but negates synchronously to indicate loss of carrier. In FIFO mode TSEC*n*_RX_DV is used to indicate valid data (GMII-style protocols) or forms part of the receive control flags (encoded packet protocols).

Table 13-2. eTSEC Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description
TSEC*n*_RXD[7:0]	I	Receive data in. In GMII mode, TSEC*n*_RXD[7:4] with TSEC*n*_RXD[3:0], represent one complete octet of data to be transferred from the PHY to the MAC when TSEC*n*_RX_DV is asserted. In TBI mode, TSEC*n*_RXD[7:4] represents RCG[7:4]. Together, with RCG[9:8] and RCG[3:0], they represent the 10-bit encoded symbol of GMII receive signals. In GMII or MII mode, TSEC*n*_RXD[3:0] represents a nibble of data to be transferred from the PHY to the MAC when TSEC*n*_RX_DV is asserted. A completely-formed SFD must be passed across the MII. While TSEC*n*_RX_DV is not asserted, TSEC*n*_RXD has no meaning. In RGMII mode, data bits 3:0 are received on the rising edge of TSEC*n*_RX_CLK. In RTBI mode, TSEC*n*_RXD[3:0] represents RCG[3:0] on the rising edge of TSEC*n*_RX_CLK and RCG[8:5] are received on the falling edge of TSEC*n*_RX_CLK. In TBI mode, TSEC*n*_RXD[3:0] represents RCG[3:0]. Together, with RCG[9:4], they represent the 10-bit encoded symbol of GMII receive signals. In RMII mode TSEC*n*_RXD[1:0] represents RXD[1:0], which is considered valid when TSEC*n*_RX_DV (CRS_DV) is asserted, or invalid otherwise. In FIFO mode TSEC*n*_RXD[7:4] with TSEC*n*_RXD[3:0] represent one complete octet of data to be received from the external FIFO device. For 16-bit FIFO operation the TSEC*n*_RXD[7:0] signals of a pair of eTSECs are combined to represent two octets of data.
TSEC*n*_RX_ER	I	Receive error **State Meaning**: Asserted/Negated—In GMII, MII, or RMII mode, if TSEC*n*_RX_ER and TSEC*n*_RX_DV are asserted, the PHY has detected an error in the current frame. In TBI mode, this signal represents RCG[9]. Together, with RCG[8:0], they represent the 10-bit encoded symbol of GMII receive signals. In FIFO mode, this signal represents either receive data error (GMII-style protocols) or forms part of the receive control flags (encoded packet protocols). This signal is not used in the RTBI or RGMII modes.
TSEC*n*_TX_CLK	I	Transmit clock in. In MII mode, TSEC*n*_TX_CLK is a continuous clock (2.5 or 25 MHz) that provides a timing reference for the TSEC*n*_TX_EN, TSEC*n*_TXD, and TSEC*n*_TX_ER signals. In GMII mode, this signal provides the 2.5 or 25 MHz timing reference during 10Base-T and 100Base-T and comes from the PHY. In 1000Base-T this clock is not used and TSEC*n*_GTX_CLK (125 MHz) becomes the timing reference. The TSEC*n*_GTX_CLK is generated in the eTSEC and provided to the PHY and the MAC. The TSEC*n*_TX_CLK is generated in the PHY and provided to the MAC. In TBI mode, this signal is PMA receive clock 1 at 62.5 MHz, split phase with PMA_RX_CLK0, and is supplied by the SerDes. In RMII mode this signal is the reference clock shared between transmit and receive, and is supplied by the PHY. In FIFO mode the transmit clock is a continuous clock. See the device hardware specifications document for maximum supported frequencies. This signal is not used in the eTSEC RTBI or RGMII modes.

Table 13-2. eTSEC Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description
TSEC*n*_TXD[7:4]	O	Transmit data out. In GMII mode, TSEC*n*_TXD[7:0] represents one complete octet of data to be sent from the MAC to the PHY when TSEC_TX_DV is asserted and has no meaning while TSEC*n*_TX_EN is negated. In TBI mode, TSEC*n*_TXD[7:4] represents transmit code group (TCG) bits 7:4. Together, with TCG[9:8] and TCG[3:0], they represent the 10-bit encoded symbol. In GMII or MII mode, TSEC*n*_TXD[3:0] represent a nibble of data to be sent from the MAC to the PHY when TSEC*n*_TX_EN is asserted and have no meaning while TSEC*n*_TX_EN is negated. In RGMII or RTBI mode, data bits 3:0 are transmitted on the rising edge of TSEC*n*_TX_CLK, and data bits 7:4 are transmitted on the falling edge of TSEC*n*_TX_CLK. In TBI mode, TSEC*n*_TXD[3:0] represents TCG[3:0]. Together, with TCG[9:4], they represent the 10-bit encoded symbol. In RMII mode, TSEC*n*_TXD[1:0] represents TXD[1:0], which is valid data sent to the PHY when TSEC*n*_TX_EN is asserted, or undefined otherwise. In FIFO mode, TSEC*n*_TXD[7:4] with TSEC*n*_TXD[3:0] represent one complete octet of data to be received from the external FIFO device. For 16-bit FIFO operation the TSEC*n*_TXD[7:0] signals of a pair of eTSECs are combined to represent two octets of data. Note that some of these signals are also used during reset to configure the eTSEC interface mode.
TSEC*n*_TX_EN	O	Transmit data valid. In GMII, MII, or RMII mode, if TSEC*n*_TX_EN is asserted, the MAC is indicating that valid data is present on the GMII's or the MII's TSEC*n*_TXD signals. In RGMII mode, TSEC*n*_TX_EN becomes TX_CTL. TX_EN and TX_ERR are asserted on this signal on rising and falling edges of the TSEC*n*_GTX_CLK, respectively. In TBI mode, TSEC*n*_TX_EN represents TCG[8]. Together, with TCG[9] and TCG[7:0], they represent the 10-bit encoded symbol. In RTBI mode, TSEC*n*_TX_EN represents TCG[4] on the rising edge and TCG[9] on the falling edge of TSEC*n*_GTX_CLK, respectively. Together with TCG[3:0] and TCG[8:5], they represent the 10-bit encoded symbol. In FIFO mode TSEC*n*_TX_EN is used to indicate valid data (GMII-style protocols) or forms part of the transmit control flags (encoded packet protocols).
TSEC*n*_TX_ER	O	Transmit error. In GMII or MII mode, assertion of TSEC*n*_TX_ER for one or more clock cycles while TSEC*n*_TX_EN is asserted causes the PHY to transmit one or more illegal symbols. Asserting TSEC*n*_TX_ER has no effect while operating at 10 Mbps or while TSEC*n*_TX_EN is negated. This signal transitions synchronously with respect to TSEC*n*_TX_CLK. In TBI mode, TSEC*n*_TX_ER represents TCG[9]. Together, with TCG[8:0], they represents the 10-bit encoded symbol. In FIFO mode TSEC*n*_TX_ER represents either transmit data error (GMII-style protocols) or forms part of the transmit control flags (encoded packet protocols). This signal is not used in the eTSEC RMII, RTBI, or RGMII modes and is driven low.

13.5 Memory Map/Register Definition

The eTSECs use a software model that is a superset of the PowerQUICC III TSEC functionality and is similar to that employed by the Fast Ethernet function supported on the Freescale MPC8260 CPM FCC and in the FEC of the MPC860T.

The eTSEC device is programmed by a combination of control/status registers (CSRs) and buffer descriptors. The CSRs are used for mode control, interrupts, and to extract status information. The descriptors are used to pass data buffers and related buffer status or frame information between the hardware and software.

All accesses to and from the registers must be made as 32-bit accesses. There is no support for accesses of sizes other than 32 bits. Writes to reserved register bits must always store 0, as writing 1 to reserved bits may have unintended side-effects. Reads from unmapped register addresses return zero. Unless otherwise specified, the read value of reserved bits in mapped registers is not defined, and must not be assumed to be 0.

This section of the document defines the memory map and describes the registers in detail. The buffer descriptor is described in Section 13.6.7, “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 through the MII management interface in the same way the PHYs are accessed. For detailed descriptions of the TBI registers (the MII register set for the ten-bit interface) refer to Section 13.5.4, “Ten-Bit Interface (TBI).”

13.5.1 Top-Level Module Memory Map

Each of the eTSECs is allocated 4 Kbytes of memory-mapped space. The space for each eTSEC is divided as indicated in Table 13-3.

Table 13-3. Module Memory Map Summary

Address Offset	Function
000–0FF	eTSEC general control/status registers
100–2FF	eTSEC transmit control/status registers
300–4FF	eTSEC receive control/status registers
500–5FF	MAC registers
600–7FF	RMON MIB registers
800–8FF	Hash table registers
900–9FF	—
A00–AFF	FIFO control/status registers
B00–BFF	DMA system registers
C00–C3F	Lossless Flow Control registers
C40–FFF	—

13.5.2 Detailed Memory Map

The eTSEC memory mapped registers are accessed by reading and writing to an address comprised of the base address (specified in CCSRBAR as defined in Chapter 2, “Memory Map.”) plus the block base address, plus the offset of the specific register to be accessed. Note that all memory-mapped registers must only be accessed as 32-bit quantities.

Table 13-4 lists the offset, name, and a cross-reference to the complete description of each register. The offsets to the memory map table are applicable to each eTSEC. Block base addresses are as follows:

- eTSEC1 starts at 0x2_4000 address offset
- eTSEC2 starts at 0x2_5000 address offset

- eTSEC3 starts at 0x2_6000 address offset
- eTSEC4 starts at 0x2_7000 address offset

Table 13-4. eTSEC Memory Map

eTSEC1—Block Base Address 0x2_4000 eTSEC2—Block Base Address 0x2_5000 eTSEC3—Block Base Address 0x2_6000 eTSEC4—Block Base Address 0x2_7000				
Offset	**Register[1]**	**Access [2]**	**Reset**	**Section/Page**
eTSEC1				
eTSEC General Control and Status Registers				
0x000	TSEC_ID*—Controller ID register	R	0x0124_0000	13.5.3.1.1/13-23
0x004	TSEC_ID2*—Controller ID register	R	0x0030_00F0	13.5.3.1.2/13-24
0x008–0x00C	Reserved	—	—	—
0x010	IEVENT—Interrupt event register	w1c	0x0000_0000	13.5.3.1.3/13-25
0x014	IMASK—Interrupt mask register	R/W	0x0000_0000	13.5.3.1.4/13-28
0x018	EDIS—Error disabled register	R/W	0x0000_0000	13.5.3.1.5/13-30
0x01C	Reserved	—	—	—
0x020	ECNTRL—Ethernet control register	R/W	0x0000_0000	13.5.3.1.6/13-32
0x024	Reserved	—	—	—
0x028	PTV—Pause time value register	R/W	0x0000_0000	13.5.3.1.7/13-34
0x02C	DMACTRL—DMA control register	R/W	0x0000_0000	13.5.3.1.8/13-35
0x030	TBIPA—TBI PHY address register	R/W	0x0000_0000	13.5.3.1.9/13-36
0x034–0x0FC	Reserved	—	—	—
eTSEC Transmit Control and Status Registers				
0x100	TCTRL—Transmit control register	R/W	0x0000_0000	13.5.3.2.1/13-37
0x104	TSTAT—Transmit status register	w1c	0x0000_0000	13.5.3.2.2/13-39
0x108	DFVLAN*—Default VLAN control word	R/W	0x8100_0000	13.5.3.2.3/13-43
0x10C	Reserved	—	—	—
0x110	TXIC—Transmit interrupt coalescing register	R/W	0x0000_0000	13.5.3.2.4/13-44
0x114	TQUEUE*—Transmit queue control register	R/W	0x0000_8000	13.5.3.2.5/13-45
0x118–0x13C	Reserved	—	—	—
0x140	TR03WT*—TxBD Rings 0–3 round-robin weightings	R/W	0x0000_0000	13.5.3.2.6/13-46
0x144	TR47WT*—TxBD Rings 4–7 round-robin weightings	R/W	0x0000_0000	13.5.3.2.7/13-46
0x148–0x17C	Reserved	—	—	—

Table 13-4. eTSEC Memory Map (continued)

eTSEC1—Block Base Address 0x2_4000 eTSEC2—Block Base Address 0x2_5000 eTSEC3—Block Base Address 0x2_6000 eTSEC4—Block Base Address 0x2_7000				
Offset	**Register[1]**	**Access [2]**	**Reset**	**Section/Page**
0x180	TBDBPH*—Tx data buffer pointer high bits	R/W	0x0000_0000	13.5.3.2.8/13-47
0x184	TBPTR0—TxBD pointer for ring 0	R/W	0x0000_0000	13.5.3.2.9/13-47
0x188	Reserved	—	—	—
0x18C	TBPTR1*—TxBD pointer for ring 1	R/W	0x0000_0000	13.5.3.2.9/13-47
0x190	Reserved	—	—	—
0x194	TBPTR2*—TxBD pointer for ring 2	R/W	0x0000_0000	13.5.3.2.9/13-47
0x198	Reserved	—	—	—
0x19C	TBPTR3*—TxBD pointer for ring 3	R/W	0x0000_0000	13.5.3.2.9/13-47
0x1A0	Reserved	—	—	—
0x1A4	TBPTR4*—TxBD pointer for ring 4	R/W	0x0000_0000	13.5.3.2.9/13-47
0x1A8	Reserved	—	—	—
0x1AC	TBPTR5*—TxBD pointer for ring 5	R/W	0x0000_0000	13.5.3.2.9/13-47
0x1B0	Reserved	—	—	—
0x1B4	TBPTR6*—TxBD pointer for ring 6	R/W	0x0000_0000	13.5.3.2.9/13-47
0x1B8	Reserved	—	—	—
0x1BC	TBPTR7*—TxBD pointer for ring 7	R/W	0x0000_0000	13.5.3.2.9/13-47
0x1C0– 0x1FC	Reserved	—	—	—
0x200	TBASEH*—TxBD base address high bits	R/W	0x0000_0000	13.5.3.2.10/13-48
0x204	TBASE0—TxBD base address of ring 0	R/W	0x0000_0000	13.5.3.2.11/13-49
0x208	Reserved	—	—	—
0x20C	TBASE1*—TxBD base address of ring 1	R/W	0x0000_0000	13.5.3.2.11/13-49
0x210	Reserved	—	—	—
0x214	TBASE2*—TxBD base address of ring 2	R/W	0x0000_0000	13.5.3.2.11/13-49
0x218	Reserved	—	—	—
0x21C	TBASE3*—TxBD base address of ring 3	R/W	0x0000_0000	13.5.3.2.11/13-49
0x220	Reserved	—	—	—
0x224	TBASE4*—TxBD base address of ring 4	R/W	0x0000_0000	13.5.3.2.11/13-49
0x228	Reserved	—	—	—
0x22C	TBASE5*—TxBD base address of ring 5	R/W	0x0000_0000	13.5.3.2.11/13-49
0x230	Reserved	—	—	—
0x234	TBASE6*—TxBD base address of ring 6	R/W	0x0000_0000	13.5.3.2.11/13-49

Table 13-4. eTSEC Memory Map (continued)

eTSEC1—Block Base Address 0x2_4000 eTSEC2—Block Base Address 0x2_5000 eTSEC3—Block Base Address 0x2_6000 eTSEC4—Block Base Address 0x2_7000				
Offset	**Register[1]**	**Access [2]**	**Reset**	**Section/Page**
0x238	Reserved	—	—	—
0x23C	TBASE7*—TxBD base address of ring 7	R/W	0x0000_0000	13.5.3.2.11/13-49
0x240–0x2FC	Reserved	—	—	—
eTSEC Receive Control and Status Registers				
0x300	RCTRL—Receive control register	R/W	0x0000_0000	13.5.3.3.1/13-49
0x304	RSTAT—Receive status register	w1c	0x0000_0000	13.5.3.3.2/13-51
0x308–0x30C	Reserved	—	—	—
0x310	RXIC—Receive interrupt coalescing register	R/W	0x0000_0000	13.5.3.3.3/13-54
0x314	RQUEUE*—Receive queue control register.	R/W	0x0080_0080	13.5.3.3.4/13-55
0x318–0x32C	Reserved	—	—	—
0x330	RBIFX*—Receive bit field extract control register	R/W	0x0000_0000	13.5.3.3.5/13-55
0x334	RQFAR*—Receive queue filing table address register	R/W	0x0000_0000	13.5.3.3.6/13-57
0x338	RQFCR*—Receive queue filing table control register	R/W	0x0000_0000	13.5.3.3.7/13-58
0x33C	RQFPR*—Receive queue filing table property register	R/W	0x0000_0000	13.5.3.3.8/13-59
0x340	MRBLR—Maximum receive buffer length register	R/W	0x0000_0000	13.5.3.3.9/13-62
0x344–0x37C	Reserved	—	—	—
0x380	RBDBPH*—Rx data buffer pointer high bits	R/W	0x0000_0000	13.5.3.3.10/13-63
0x384	RBPTR0—RxBD pointer for ring 0	R/W	0x0000_0000	13.5.3.3.11/13-63
0x388	Reserved	—	—	—
0x38C	RBPTR1*—RxBD pointer for ring 1	R/W	0x0000_0000	13.5.3.3.11/13-63
0x390	Reserved	—	—	—
0x394	RBPTR2*—RxBD pointer for ring 2	R/W	0x0000_0000	13.5.3.3.11/13-63
0x398	Reserved	—	—	—
0x39C	RBPTR3*—RxBD pointer for ring 3	R/W	0x0000_0000	13.5.3.3.11/13-63
0x3A0	Reserved	—	—	—
0x3A4	RBPTR4*—RxBD pointer for ring 4	R/W	0x0000_0000	13.5.3.3.11/13-63
0x3A8	Reserved	—	—	—
0x3AC	RBPTR5*—RxBD pointer for ring 5	R/W	0x0000_0000	13.5.3.3.11/13-63
0x3B0	Reserved	—	—	—

Table 13-4. eTSEC Memory Map (continued)

eTSEC1—Block Base Address 0x2_4000 eTSEC2—Block Base Address 0x2_5000 eTSEC3—Block Base Address 0x2_6000 eTSEC4—Block Base Address 0x2_7000				
Offset	**Register[1]**	**Access [2]**	**Reset**	**Section/Page**
0x3B4	RBPTR6*—RxBD pointer for ring 6	R/W	0x0000_0000	13.5.3.3.11/13-63
0x3B8	Reserved	—	—	—
0x3BC	RBPTR7*—RxBD pointer for ring 7	R/W	0x0000_0000	13.5.3.3.11/13-63
0x3C0– 0x3FC	Reserved	—	—	—
0x400	RBASEH*—RxBD base address high bits	R/W	0x0000_0000	13.5.3.3.12/13-64
0x404	RBASE0—RxBD base address of ring 0	R/W	0x0000_0000	13.5.3.3.13/13-65
0x408	Reserved	—	—	—
0x40C	RBASE1*—RxBD base address of ring 1	R/W	0x0000_0000	13.5.3.3.13/13-65
0x410	Reserved	—	—	—
0x414	RBASE2*—RxBD base address of ring 2	R/W	0x0000_0000	13.5.3.3.13/13-65
0x418	Reserved	—	—	—
0x41C	RBASE3*—RxBD base address of ring 3	R/W	0x0000_0000	13.5.3.3.13/13-65
0x420	Reserved	—	—	—
0x424	RBASE4*—RxBD base address of ring 4	R/W	0x0000_0000	13.5.3.3.13/13-65
0x428	Reserved	—	—	—
0x42C	RBASE5*—RxBD base address of ring 5	R/W	0x0000_0000	13.5.3.3.13/13-65
0x430	Reserved	—	—	—
0x434	RBASE6*—RxBD base address of ring 6	R/W	0x0000_0000	13.5.3.3.13/13-65
0x438	Reserved	—	—	—
0x43C	RBASE7*—RxBD base address of ring 7	R/W	0x0000_0000	13.5.3.3.13/13-65
0x440– 0x4FC	Reserved	—	—	—
eTSEC MAC Registers				
0x500	MACCFG1—MAC configuration register 1	R/W	0x0000_0000	13.5.3.5.1/13-68
0x504	MACCFG2—MAC configuration register 2	R/W	0x0000_7000	13.5.3.5.2/13-69
0x508	IPGIFG—Inter-packet/inter-frame gap register	R/W	0x4060_5060	13.5.3.5.3/13-71
0x50C	HAFDUP—Half-duplex control	R/W	0x00A1_F037	13.5.3.5.4/13-72
0x510	MAXFRM—Maximum frame length	R/W	0x0000_0600	13.5.3.5.5/13-73
0x514– 0x51C	Reserved	—	—	—
0x520	MIIMCFG—MII management configuration	R/W	0x0000_0007	13.5.3.5.6/13-73

Table 13-4. eTSEC Memory Map (continued)

eTSEC1—Block Base Address 0x2_4000 eTSEC2—Block Base Address 0x2_5000 eTSEC3—Block Base Address 0x2_6000 eTSEC4—Block Base Address 0x2_7000				
Offset	**Register[1]**	**Access [2]**	**Reset**	**Section/Page**
0x524	MIIMCOM—MII management command	R/W	0x0000_0000	13.5.3.5.7/13-74
0x528	MIIMADD—MII management address	R/W	0x0000_0000	13.5.3.5.8/13-75
0x52C	MIIMCON—MII management control	WO	0x0000_0000	13.5.3.5.9/13-76
0x530	MIIMSTAT—MII management status	R	0x0000_0000	13.5.3.5.10/13-76
0x534	MIIMIND—MII management indicator	R	0x0000_0000	13.5.3.5.11/13-77
0x538	Reserved	—	—	—
0x53C	IFSTAT—Interface status	R	0x0000_0000	13.5.3.5.12/13-77
0x540	MACSTNADDR1—MAC station address register 1	R/W	0x0000_0000	13.5.3.5.13/13-78
0x544	MACSTNADDR2—MAC station address register 2	R/W	0x0000_0000	13.5.3.5.14/13-79
0x548	MAC01ADDR1*—MAC exact match address 1, part 1	R/W	0x0000_0000	13.5.3.5.15/13-79 13.5.3.5.16/13-80
0x54C	MAC01ADDR2*—MAC exact match address 1, part 2	R/W	0x0000_0000	
0x550	MAC02ADDR1*—MAC exact match address 2, part 1	R/W	0x0000_0000	
0x554	MAC02ADDR2*—MAC exact match address 2, part 2	R/W	0x0000_0000	
0x558	MAC03ADDR1*—MAC exact match address 3, part 1	R/W	0x0000_0000	
0x55C	MAC03ADDR2*—MAC exact match address 3, part 2	R/W	0x0000_0000	
0x560	MAC04ADDR1*—MAC exact match address 4, part 1	R/W	0x0000_0000	
0x564	MAC04ADDR2*—MAC exact match address 4, part 2	R/W	0x0000_0000	
0x568	MAC05ADDR1*—MAC exact match address 5, part 1	R/W	0x0000_0000	
0x56C	MAC05ADDR2*—MAC exact match address 5, part 2	R/W	0x0000_0000	
0x570	MAC06ADDR1*—MAC exact match address 6, part 1	R/W	0x0000_0000	
0x574	MAC06ADDR2*—MAC exact match address 6, part 2	R/W	0x0000_0000	
0x578	MAC07ADDR1*—MAC exact match address 7, part 1	R/W	0x0000_0000	
0x57C	MAC07ADDR2*—MAC exact match address 7, part 2	R/W	0x0000_0000	
0x580	MAC08ADDR1*—MAC exact match address 8, part 1	R/W	0x0000_0000	
0x584	MAC08ADDR2*—MAC exact match address 8, part 2	R/W	0x0000_0000	
0x588	MAC09ADDR1*—MAC exact match address 9, part 1	R/W	0x0000_0000	
0x58C	MAC09ADDR2*—MAC exact match address 9, part 2	R/W	0x0000_0000	
0x590	MAC10ADDR1*—MAC exact match address 10, part 1	R/W	0x0000_0000	

Table 13-4. eTSEC Memory Map (continued)

eTSEC1—Block Base Address 0x2_4000 eTSEC2—Block Base Address 0x2_5000 eTSEC3—Block Base Address 0x2_6000 eTSEC4—Block Base Address 0x2_7000				
Offset	**Register[1]**	**Access [2]**	**Reset**	**Section/Page**
0x594	MAC10ADDR2*—MAC exact match address 10, part 2	R/W	0x0000_0000	13.5.3.5.15/13-79 13.5.3.5.16/13-80
0x598	MAC11ADDR1*—MAC exact match address 11, part 1	R/W	0x0000_0000	
0x59C	MAC11ADDR2*—MAC exact match address 11, part 2	R/W	0x0000_0000	
0x5A0	MAC12ADDR1*—MAC exact match address 12, part 1	R/W	0x0000_0000	
0x5A4	MAC12ADDR2*—MAC exact match address 12, part 2	R/W	0x0000_0000	
0x5A8	MAC13ADDR1*—MAC exact match address 13, part 1	R/W	0x0000_0000	
0x5AC	MAC13ADDR2*—MAC exact match address 13, part 2	R/W	0x0000_0000	
0x5B0	MAC14ADDR1*—MAC exact match address 14, part 1	R/W	0x0000_0000	
0x5B4	MAC14ADDR2*—MAC exact match address 14, part 2	R/W	0x0000_0000	
0x5B8	MAC15ADDR1*—MAC exact match address 15, part 1	R/W	0x0000_0000	
0x5BC	MAC15ADDR2*—MAC exact match address 15, part 2	R/W	0x0000_0000	
0x5C0– 0x67C	Reserved	—	—	—
eTSEC Transmit and Receive Counters				
0x680	TR64—Transmit and receive 64-byte frame counter	R/W	0x0000_0000	13.5.3.6.1/13-81
0x684	TR127—Transmit and receive 65- to 127-byte frame counter	R/W	0x0000_0000	13.5.3.6.2/13-81
0x688	TR255—Transmit and receive 128- to 255-byte frame counter	R/W	0x0000_0000	13.5.3.6.3/13-82
0x68C	TR511—Transmit and receive 256- to 511-byte frame counter	R/W	0x0000_0000	13.5.3.6.4/13-82
0x690	TR1K—Transmit and receive 512- to 1023-byte frame counter	R/W	0x0000_0000	13.5.3.6.5/13-83
0x694	TRMAX—Transmit and receive 1024- to 1518-byte frame counter	R/W	0x0000_0000	13.5.3.6.6/13-83
0x698	TRMGV—Transmit and receive 1519- to 1522-byte good VLAN frame count	R/W	0x0000_0000	13.5.3.6.7/13-84
eTSEC Receive Counters				
0x69C	RBYT—Receive byte counter	R/W	0x0000_0000	13.5.3.6.8/13-84
0x6A0	RPKT—Receive packet counter	R/W	0x0000_0000	13.5.3.6.9/13-85
0x6A4	RFCS—Receive FCS error counter	R/W	0x0000_0000	13.5.3.6.10/13-85
0x6A8	RMCA—Receive multicast packet counter	R/W	0x0000_0000	13.5.3.6.11/13-86
0x6AC	RBCA—Receive broadcast packet counter	R/W	0x0000_0000	13.5.3.6.12/13-86
0x6B0	RXCF—Receive control frame packet counter	R/W	0x0000_0000	13.5.3.6.13/13-87
0x6B4	RXPF—Receive PAUSE frame packet counter	R/W	0x0000_0000	13.5.3.6.14/13-87
0x6B8	RXUO—Receive unknown OP code counter	R/W	0x0000_0000	13.5.3.6.15/13-88
0x6BC	RALN—Receive alignment error counter	R/W	0x0000_0000	13.5.3.6.16/13-88

Table 13-4. eTSEC Memory Map (continued)

eTSEC1—Block Base Address 0x2_4000 eTSEC2—Block Base Address 0x2_5000 eTSEC3—Block Base Address 0x2_6000 eTSEC4—Block Base Address 0x2_7000				
Offset	**Register[1]**	**Access [2]**	**Reset**	**Section/Page**
0x6C0	RFLR—Receive frame length error counter	R/W	0x0000_0000	13.5.3.6.17/13-89
0x6C4	RCDE—Receive code error counter	R/W	0x0000_0000	13.5.3.6.18/13-89
0x6C8	RCSE—Receive carrier sense error counter	R/W	0x0000_0000	13.5.3.6.19/13-90
0x6CC	RUND—Receive undersize packet counter	R/W	0x0000_0000	13.5.3.6.20/13-90
0x6D0	ROVR—Receive oversize packet counter	R/W	0x0000_0000	13.5.3.6.21/13-91
0x6D4	RFRG—Receive fragments counter	R/W	0x0000_0000	13.5.3.6.22/13-91
0x6D8	RJBR—Receive jabber counter	R/W	0x0000_0000	13.5.3.6.23/13-92
0x6DC	RDRP—Receive drop counter	R/W	0x0000_0000	13.5.3.6.24/13-92
eTSEC Transmit Counters				
0x6E0	TBYT—Transmit byte counter	R/W	0x0000_0000	13.5.3.6.25/13-93
0x6E4	TPKT—Transmit packet counter	R/W	0x0000_0000	13.5.3.6.26/13-93
0x6E8	TMCA—Transmit multicast packet counter	R/W	0x0000_0000	13.5.3.6.27/13-94
0x6EC	TBCA—Transmit broadcast packet counter	R/W	0x0000_0000	13.5.3.6.28/13-94
0x6F0	TXPF—Transmit PAUSE control frame counter	R/W	0x0000_0000	13.5.3.6.29/13-95
0x6F4	TDFR—Transmit deferral packet counter	R/W	0x0000_0000	13.5.3.6.30/13-95
0x6F8	TEDF—Transmit excessive deferral packet counter	R/W	0x0000_0000	13.5.3.6.31/13-96
0x6FC	TSCL—Transmit single collision packet counter	R/W	0x0000_0000	13.5.3.6.32/13-96
0x700	TMCL—Transmit multiple collision packet counter	R/W	0x0000_0000	13.5.3.6.33/13-97
0x704	TLCL—Transmit late collision packet counter	R/W	0x0000_0000	13.5.3.6.34/13-97
0x708	TXCL—Transmit excessive collision packet counter	R/W	0x0000_0000	13.5.3.6.35/13-98
0x70C	TNCL—Transmit total collision counter	R/W	0x0000_0000	13.5.3.6.36/13-98
0x710	Reserved	—	—	—
0x714	TDRP—Transmit drop frame counter	R/W	0x0000_0000	13.5.3.6.37/13-99
0x718	TJBR—Transmit jabber frame counter	R/W	0x0000_0000	13.5.3.6.38/13-99
0x71C	TFCS—Transmit FCS error counter	R/W	0x0000_0000	13.5.3.6.39/13-100
0x720	TXCF—Transmit control frame counter	R/W	0x0000_0000	13.5.3.6.40/13-100
0x724	TOVR—Transmit oversize frame counter	R/W	0x0000_0000	13.5.3.6.41/13-101
0x728	TUND—Transmit undersize frame counter	R/W	0x0000_0000	13.5.3.6.42/13-101
0x72C	TFRG—Transmit fragments frame counter	R/W	0x0000_0000	13.5.3.6.43/13-102
eTSEC Counter Control and TOE Statistics Registers				
0x730	CAR1—Carry register one register[3]	R	0x0000_0000	13.5.3.6.44/13-102

Table 13-4. eTSEC Memory Map (continued)

eTSEC1—Block Base Address 0x2_4000 eTSEC2—Block Base Address 0x2_5000 eTSEC3—Block Base Address 0x2_6000 eTSEC4—Block Base Address 0x2_7000				
Offset	**Register[1]**	**Access [2]**	**Reset**	**Section/Page**
0x734	CAR2—Carry register two register [3]	R	0x0000_0000	13.5.3.6.45/13-104
0x738	CAM1—Carry register one mask register	R/W	0xFE03_FFFF	13.5.3.6.46/13-105
0x73C	CAM2—Carry register two mask register	R/W	0x000F_FFFD	13.5.3.6.47/13-106
0x740	RREJ*—Receive filer rejected packet counter	R/W	0x0000_0000	13.5.3.6.48/13-107
0x744– 0x7FC	Reserved	—	—	—
Hash Function Registers				
0x800	IGADDR0—Individual/group address register 0	R/W	0x0000_0000	13.5.3.7.1/13-108
0x804	IGADDR1—Individual/group address register 1	R/W	0x0000_0000	
0x808	IGADDR2—Individual/group address register 2	R/W	0x0000_0000	
0x80C	IGADDR3—Individual/group address register 3	R/W	0x0000_0000	
0x810	IGADDR4—Individual/group address register 4	R/W	0x0000_0000	
0x814	IGADDR5—Individual/group address register 5	R/W	0x0000_0000	
0x818	IGADDR6—Individual/group address register 6	R/W	0x0000_0000	
0x81C	IGADDR7—Individual/group address register 7	R/W	0x0000_0000	
0x820– 0x87C	Reserved	—	—	—
0x880	GADDR0—Group address register 0	R/W	0x0000_0000	13.5.3.7.2/13-109
0x884	GADDR1—Group address register 1	R/W	0x0000_0000	
0x888	GADDR2—Group address register 2	R/W	0x0000_0000	
0x88C	GADDR3—Group address register 3	R/W	0x0000_0000	
0x890	GADDR4—Group address register 4	R/W	0x0000_0000	
0x894	GADDR5—Group address register 5	R/W	0x0000_0000	
0x898	GADDR6—Group address register 6	R/W	0x0000_0000	
0x89C	GADDR7—Group address register 7	R/W	0x0000_0000	
0x8A0– 0x9FC	Reserved	—	—	—
eTSEC FIFO Control Registers				
0xA00	FIFOCFG*—FIFO interface configuration register	R/W	0x0000_00C0	13.5.3.8.1/13-109
0xA04– 0xAFC	Reserved	—	—	—
eTSEC DMA Attribute Registers				
0xB00– 0xBF4	Reserved	—	—	—

Table 13-4. eTSEC Memory Map (continued)

eTSEC1—Block Base Address 0x2_4000 eTSEC2—Block Base Address 0x2_5000 eTSEC3—Block Base Address 0x2_6000 eTSEC4—Block Base Address 0x2_7000				
Offset	**Register[1]**	**Access [2]**	**Reset**	**Section/Page**
0xBF8	ATTR—Attribute register	R/W	0x0000_0000	13.5.3.9.1/13-111
eTSEC Lossless Flow Control Registers				
0xC00	RQPRM0*—Receive Queue Parameters register 0	R/W	0x0000_0000	13.5.3.10.1/13-112
0xC04	RQPRM1*—Receive Queue Parameters register 1	R/W	0x0000_0000	
0xC08	RQPRM2*—Receive Queue Parameters register 2	R/W	0x0000_0000	
0xC0C	RQPRM3*—Receive Queue Parameters register 3	R/W	0x0000_0000	
0xC10	RQPRM4*—Receive Queue Parameters register 4	R/W	0x0000_0000	
0xC14	RQPRM5*—Receive Queue Parameters register 5	R/W	0x0000_0000	
0xC18	RQPRM6*—Receive Queue Parameters register 6	R/W	0x0000_0000	
0xC1C	RQPRM7*—Receive Queue Parameters register 7	R/W	0x0000_0000	
0xC20–0xC40	Reserved	—	—	—
0xC44	RFBPTR0*—Last Free RxBD pointer for ring 0	R/W	0x0000_0000	13.5.3.10.2/13-113
0xC48	Reserved	—	—	—
0xC4C	RFBPTR1*—Last Free RxBD pointer for ring 1	R/W	0x0000_0000	13.5.3.10.2/13-113
0xC50	Reserved	—	—	—
0xC54	RFBPTR2*—Last Free RxBD pointer for ring 2	R/W	0x0000_0000	13.5.3.10.2/13-113
0xC58	Reserved	—	—	—
0xC5C	RFBPTR3*—Last Free RxBD pointer for ring 3	R/W	0x0000_0000	13.5.3.10.2/13-113
0xC60	Reserved	—	—	—
0xC64	RFBPTR4*—Last Free RxBD pointer for ring 4	R/W	0x0000_0000	13.5.3.10.2/13-113
0xC68	Reserved	—	—	—
0xC6C	RFBPTR5*—Last Free RxBD pointer for ring 5	R/W	0x0000_0000	13.5.3.10.2/13-113
0xC70	Reserved	—	—	—
0xC74	RFBPTR6*—Last Free RxBD pointer for ring 6	R/W	0x0000_0000	13.5.3.10.2/13-113
0xC78	Reserved	—	—	—
0xC7C	RFBPTR7*—Last Free RxBD pointer for ring 7	R/W	0x0000_0000	13.5.3.10.2/13-113
eTSEC Future Expansion Space				
0xC00–0xD94	Reserved	—	—	—

Table 13-4. eTSEC Memory Map (continued)

eTSEC1—Block Base Address 0x2_4000 eTSEC2—Block Base Address 0x2_5000 eTSEC3—Block Base Address 0x2_6000 eTSEC4—Block Base Address 0x2_7000				
Offset	**Register[1]**	**Access [2]**	**Reset**	**Section/Page**
eTSEC2				
0x000– 0xFF	eTSEC2 Registers **Note:** eTSEC2 has the same memory-mapped registers that are described for eTSEC1 but with a different block base address (0x2_5000).			
eTSEC3				
0x000– 0xFFF	eTSEC3 Registers **Note:** eTSEC3 has the same memory-mapped registers that are described for eTSEC1 but with a different block base address (0x2_6000).			
eTSEC4				
0x000– 0xFFF	eTSEC4 Registers **Note:** eTSEC4 has the same memory-mapped registers that are described for eTSEC1 but with a different block base address (0x2_7000).			

1 Registers denoted * are new to the enhanced TSEC and not supported by PowerQUICC III TSECs.

2 Key: R = read only, WO = write only, R/W = read and write, LH = latches high, SC = self-clearing.

3 Cleared on read.

13.5.3 Memory-Mapped Register Descriptions

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

13.5.3.1 eTSEC General Control and Status Registers

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

13.5.3.1.1 Controller ID Register (TSEC_ID)

The controller ID register (TSEC_ID) is a read-only register. The TSEC_ID register is used to identify the eTSEC block and revision.

Offset eTSEC1:0x2_4000; eTSEC2:0x2_5000; eTSEC3:0x2_6000; eTSEC4:0x2_7000 — Access: Read only

	0							15	16							23	24							31
R	TSEC_ID																TSEC_REV_MJ							
W																								
Reset	0	0	0	0	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0

Figure 13-2. TSEC_ID Register

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

Table 13-5. TSEC_ID Field Descriptions

Bits	Name	Description
0–15	TSEC_ID	Value identifying the eTSEC (10/100/1000 Ethernet MAC). 0124 Unique identifier for eTSEC with 8 Rx and 8 Tx BD rings.
16–23	TSEC_REV_MJ	Value identifies the major revision of the eTSEC. 00 Initial revision
24–31	TSEC_REV_MN	Value identifies the minor revision of the eTSEC. 02 Initial revision02Initial revision

13.5.3.1.2 Controller ID Register (TSEC_ID2)

The controller ID register (TSEC_ID2) is a read-only register. The TSEC_ID2 register is used to identify the eTSEC block configuration.

Offset eTSEC1:0x2_4004; eTSEC2:0x2_5004; eTSEC3:0x2_6004; eTSEC4:0x2_7004 — Access: Read only

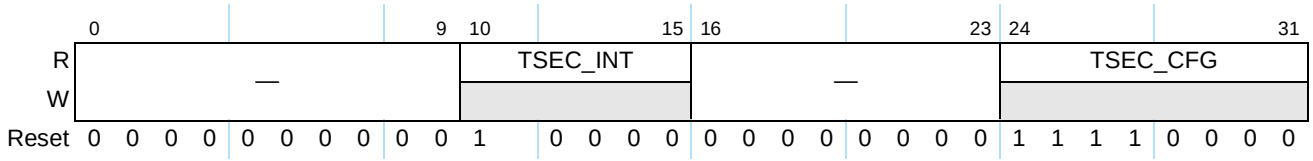


Figure 13-3. TSEC_ID2 Register

Table 13-6 describes the fields of the TSEC_ID2 register.

Table 13-6. TSEC_ID2 Field Descriptions

Bits	Name	Description
0–9	—	Reserved
10–15	TSEC_INT	Interface mode support. See Table 13-7 for settings. For eTSEC1: 0x30 For eTSEC2: 0x38 For eTSEC3: 0x30 For eTSEC4: 0x38
16–23	—	Reserved
24–31	TSEC_CFG	Value identifies configuration options of the eTSEC. 00 eTSEC multiple ring, Rx TOE, Filer and Tx TOE supports are off F0 eTSEC multiple ring, Rx TOE, Filer and Tx TOE supports are on 30 eTSEC multiple ring support is OFF and Rx TOE, Filer and Tx TOE supports are on 50 eTSEC multiple ring and filer supports are OFF and Rx TOE and Tx TOE supports are on

Table 13-7 describes the field settings for TSEC_ID2[TSEC_INT].

Table 13-7. TSEC_ID2[TSEC_INT] Field Settings

Bit	Mode
10	0 Ethernet mode not supported 1 Ethernet mode supported
11	0 FIFO mode not supported 1 FIFO mode supported
12	0 Can be configured to run in FIFO 16-bit mode 1 FIFO 16-bit mode off
12	Reserved
13	0 Can be configured to run in FIFO 8-bit mode 1 FIFO 8-bit mode off
14	0 Can be configured to run in Ethernet normal/full mode 1 Ethernet normal/full mode off
15	0 Can be configured to run in Ethernet reduced mode 1 Ethernet reduced mode off

13.5.3.1.3 Interrupt Event Register (IEVENT)

Interrupt events cause bits in the IEVENT register to be set. Software may poll this register at any time to check for pending interrupts. If an event occurs and its corresponding enable bit is set in the interrupt mask register (IMASK), the event also causes a hardware interrupt at the PIC. A bit in the interrupt event register is cleared by writing a 1 to that bit position. A write of 0 has no effect.

Each eTSEC can issue three kinds of hardware interrupt to the PIC:

1. Transmit interrupts—Issued whenever bits TXB or TXF of IEVENT are set to 1 and either transmit interrupt coalescing is disabled or the interrupt coalescing thresholds have been met for TXF. To negate this hardware interrupt, software must clear both TXB and TXF bits.
2. Receive interrupts—Issued whenever bits RXB or RXF of IEVENT are set to 1 and either receive interrupt coalescing is disabled or the interrupt coalescing thresholds have been met for RXF. To negate this hardware interrupt, software must clear both RXB and RXF bits.
3. Error and diagnostic interrupts—Issued whenever bits MAG, GTSC, GRSC, TXC, RXC, BABR, BABT, LC, CRL, FIR, FIQ, DPE, PERR, EBERR, TXE, XFUN or BSY of IEVENT are set to 1. Software must clear all of these bits to negate an error/diagnostic hardware interrupt.
   - Magic Packet reception event is: MAG
   - Operational diagnostics are events on: GTSC, GRSC, TXC and RXC
   - Interrupts resulting from errors/problems detected in the network or transceiver are: BABR, BABT, LC and CRL
   - Interrupts resulting from internal errors are: FIR, FIQ, DPE, PERR, EBERR, TXE, XFUN and BSY

Some of the error interrupts are independently counted in the MIB block counters. Software may choose to mask off these interrupts because these errors are visible to network management through the MIB counters.

Figure 13-4 describes the definition for the IEVENT register.

Offset eTSEC1:0x2_4010; eTSEC2: 0x2_5010; eTSEC3:0x2_6010; eTSEC4:0x2_7010 — Access: w1c

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BABR	RXC	BSY	EBERR	—	MSRO	GTSC	BABT	TXC	TXE	TXB	TXF	—	LC	CRL	XFUN
W	w1c	w1c	w1c	w1c		w1c	w1c	w1c	w1c	w1c	w1c	w1c		w1c	w1c	w1c
Reset	All zeros															

	16	17	19	20	21	22	23	24	25	27	28	29	30	31
R	RXB	—		MAG	MMRD	MMWR	GRSC	RXF	—	—	FIR	FIQ	DPE	PERR
W	w1c			w1c	w1c	w1c	w1c	w1c		—	w1c	w1c	w1c	w1c
Reset	All zeros													

Figure 13-4. IEVENT Register Definition

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

Table 13-8. IEVENT Field Descriptions

Bits	Name	Description
0	BABR	Babbling receive error. This bit indicates that a frame was received with length in excess of the MAC's maximum frame length register while MACCFG2[Huge Frame] is set. 0 Excessive frame not received. 1 Excessive frame received.
1	RXC	Receive control interrupt. A control frame was received while MACCFG1[Rx_Flow] is set. As soon as the transmitter finishes sending the current frame, a pause operation is performed. 0 Control frame not received. 1 Control frame received.
2	BSY	Busy condition interrupt. Indicates that a frame was received and discarded due to a lack of buffers. 0 No frame received and discarded. 1 Frame received and discarded.
3	EBERR	Internal bus error. This bit indicates that a system bus error occurred while a DMA transaction was underway. As a result, transferred data is expected to be partially or completely invalid. 0 No system bus error occurred. 1 System bus error occurred.
4	—	Reserved
5	MSRO	MIB counter overflow. This interrupt is asserted if the count for one of the MIB counters has exceeded the size of its register. 0 MIB count not exceeding its register size. 1 MIB count exceeds its register size.
6	GTSC	Graceful transmit stop complete. This interrupt is asserted for one of two reasons. Graceful stop means that the transmitter is put into a pause state after completion of the frame currently being transmitted. • A graceful stop, which was initiated by setting DMACTRL[GTS], is now complete. • A transmission of a flow control PAUSE frame, which was initiated by setting TCTRL[TFC_PAUSE], is now complete. 0 No graceful stop interrupt. 1 Graceful stop requested.

Table 13-8. IEVENT Field Descriptions (continued)

Bits	Name	Description
7	BABT	Babbling transmit error. This bit indicates that the transmitted frame length has exceeded the value in the MAC's maximum frame length register and MACCFG2[Huge Frame] is cleared. Frame truncation occurs when this condition occurs. 0 Transmitted frame length not exceeding maximum frame length. 1 Transmitted frame length exceeding maximum frame length when MACCFG2[Huge Frame] = 0.
8	TXC	Transmit control interrupt. This bit indicates that a control frame was transmitted. 0 Control frame not transmitted. 1 Control frame transmitted.
9	TXE	Transmit error. This bit indicates that an error occurred on the transmitted channel that has caused TSTAT[THLT] to be set by the eTSEC. This bit is set whenever any transmit error occurs that causes the transmitter to halt (EBERR, LC, CRL, XFUN). 0 No transmit channel error occurred. 1 Transmit channel error occurred.
10	TXB	Transmit buffer. This bit indicates that a transmit buffer descriptor was updated whose I (interrupt) bit was set in its status word and was not the last buffer descriptor of the frame. 0 No transmit buffer descriptor updated. 1 Transmit buffer descriptor updated.
11	TXF	Transmit frame interrupt. This bit indicates that a frame was transmitted and that the last corresponding transmit buffer descriptor (TxBD) was updated. This only occurs if the I (interrupt) bit in the status word of the buffer descriptor is set. The specific transmit queue that was updated has its TXF bit set in TSTAT. 0 No frame transmitted/TxBD not updated. 1 Frame transmitted/TxBD updated.
12	—	Reserved
13	LC	Late collision. This bit indicates that a collision occurred beyond the collision window (slot time) in half-duplex mode. The frame is truncated with a bad CRC and the remainder of the frame is discarded. 0 No late collision occurred. 1 Late collision occurred.
14	CRL	Collision retry limit. This bit indicates that the number of successive transmission collisions has exceeded the MAC's half-duplex register's retransmission maximum count (HAFDUP[Retransmission Maximum]). The frame is discarded without being transmitted and transmission of the next frame commences. This only occurs while in half-duplex mode. 0 Successive transmission collisions do not exceed maximum. 1 Successive transmission collisions exceed maximum.
15	XFUN	Transmit FIFO underrun. This bit indicates that the transmit FIFO became empty before the complete frame was transmitted. 0 Transmit FIFO not underrun. 1 Transmit FIFO underrun.
16	RXB	Receive buffer. This bit indicates that a receive buffer descriptor was updated which had the I (Interrupt) bit set in its status word and was not the last buffer descriptor of the frame. 0 Receive buffer descriptor not updated. 1 Receiver buffer descriptor updated.
17–19	—	Reserved
20	MAG	Magic Packet detected when the eTSEC is in Magic Packet detection mode (MACCFG2[MPEN] = 1). 0 No Magic Packet received, or Magic Packet mode was not enabled. 1 A Magic Packet was received while in Magic Packet mode. MACCFG2[MPEN] is also cleared upon receiving the Magic Packet.

Table 13-8. IEVENT Field Descriptions (continued)

Bits	Name	Description
21	MMRD	MII management read completion 0 MII management read not issued or in process. 1 MII management read completed that was initiated by a user through the MII Scan or Read cycle command.
22	MMWR	MII management write completion 0 MII management write not issued or in process. 1 MII management write completed that was initiated by a user write to the MIIMCON register.
23	GRSC	Graceful receive stop complete. This interrupt is asserted if a graceful receive stop is completed. It allows the user to know if the system has completed the stop and it is safe to write to receive registers (status, control or configuration registers) that are used by the system during normal operation. 0 Graceful stop not completed. 1 Graceful stop completed.
24	RXF	Receive frame interrupt. This bit indicates that a frame was received and the last receive buffer descriptor (RxBD) in that frame was updated. This occurs either if the I (interrupt) bit in the buffer descriptor status word is set, or an overrun error occurs. The specific receive queue that was updated has its RXF bit set in RSTAT. 0 Frame not received. 1 Frame received.
25–27	—	Reserved
28	FIR	The receive queue filer result is invalid, either because not enough time between frames was available to find a matching rule, or no entry in the filer table could be matched. 0 Receive queue filer reached a definite result; however, bit FIQ may still be set if a frame was filed to a disabled RxBD ring. 1 Receive queue filer was unable to reach a definite result. In this case, bit FIQ is also set if no entry in the filer table could provide a rule match.
29	FIQ	Filed frame to invalid receive queue. This bit indicates that either the receive queue filer chose to DMA a received frame to a disabled RxBD ring, or that no rule in the filer table could be matched. 0 Received frames filed to valid queues or rejected. Note that a frame may be rejected if the filer has insufficient time to reach a conclusive result between frames, in which case bit FIR is set. 1 Received frames filed to RxBD rings that are not enabled. The frame is discarded. If bit FIR is also set this indicates that the filer exhausted all of its table entries without a rule match.
30	DPE	Internal data parity error. This bit indicates that the eTSEC has detected a parity error on its stored data, which is likely to compromise the validity of recently transferred frames. 0 No parity errors detected. 1 Data held in the FIFO or filer arrays is expected to be corrupted due to a parity error.
31	PERR	Receive frame parse error for TCP/IP off-load. This bit indicates that a received frame could not be parsed unambiguously, due to encapsulated header type fields contradicting each other. 0 Received frame parsed successfully. 1 Received frame parse revealed header inconsistencies.

13.5.3.1.4 Interrupt Mask Register (IMASK)

The interrupt mask register provides control over which possible interrupt events in the IEVENT register are permitted to participate in generating hardware interrupts to the PIC. All implemented bits in this register are R/W and cleared upon a hardware reset. If the corresponding bits in both the IEVENT and IMASK registers are set, the PIC receives an interrupt (for each eTSEC these are grouped into transmit, receive, and error/diagnostic interrupts). The interrupt signal remains asserted until either the IEVENT bit is cleared, by writing a 1 to it, or by writing a 0 to the corresponding IMASK bit.

Figure 13-5 describes the IMASK register.

Offset eTSEC1:0x2_4014; eTSEC2:0x2_5014; eTSEC3:0x2_6014; eTSEC4:0x2_7014 — Access: Read/Write

	0	1	2	3	4	5	6	7
R W	BREN	RXCEN	BSYEN	EBERREN	—	MSROEN	GTSCEN	BTEN
Reset	All zeros							

	8	9	10	11	12	13	14	15
R W	TXCEN	TXEEN	TXBEN	TXFEN	—	LCEN	CRLEN	XFUNEN
Reset	All zeros							

	16	17	19	20	21	22	23
R W	RXBEN	—		MAGEN	MMRDEN	MMWREN	GRSCEN
Reset	All zeros						

	24	25	27	28	29	30	31
R W	RXFEN	—		FIREN	FIQEN	DPEEN	PERREN
Reset	All zeros						

Figure 13-5. IMASK Register Definition

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

Table 13-9. IMASK Field Descriptions

Bits	Name	Description
0	BREN	Babbling receiver interrupt enable
1	RXCEN	Receive control interrupt enable
2	BSYEN	Busy interrupt enable
3	EBERREN	Ethernet controller bus error enable
4	—	Reserved
5	MSROEN	MIB counter overflow interrupt enable
6	GTSCEN	Graceful transmit stop complete interrupt enable
7	BTEN	Babbling transmitter interrupt enable
8	TXCEN	Transmit control interrupt enable
9	TXEEN	Transmit error interrupt enable
10	TXBEN	Transmit buffer interrupt enable
11	TXFEN	Transmit frame interrupt enable
12	—	Reserved

Table 13-9. IMASK Field Descriptions (continued)

Bits	Name	Description
13	LCEN	Late collision enable
14	CRLEN	Collision retry limit enable
15	XFUNEN	Transmit FIFO underrun enable
16	RXBEN	Receive buffer interrupt enable
17–19	—	Reserved
20	MAGEN	Magic packet received interrupt enable
21	MMRDEN	MII management read completion interrupt enable
22	MMWREN	MII management write completion interrupt enable
23	GRSCEN	Graceful receive stop complete interrupt enable
24	RXFEN	Receive frame interrupt enable
25–27	—	Reserved
28	FIREN	Filer invalid result interrupt enable
29	FIQEN	Filed frame to invalid queue interrupt enable
30	DPEEN	Data parity error interrupt enable
31	PERREN	Receive frame parse error enable

13.5.3.1.5 Error Disabled Register (EDIS)

Figure 13-6 describes the definition for the EDIS register. The error disabled register allows the user to disable an error interruption, possibly to avoid spurious error indications external to the eTSECs.

Offset eTSEC1:0x2_4018; eTSEC2:0x2_5018; eTSEC3:0x2_6018; eTSEC4:0x2_7018 — Access: Read/Write

	0	1	2	3	4	6	7	8	9	10	12	13	14	15
R W	—		BSYDIS	EBERRDIS	—		BABTDIS	—	TXEDIS	—		LCDIS	CRLDIS	XFUNDIS
Reset	All zeros													

	16	27	28	29	30	31
R W	—		FIRDIS	FIQDIS	DPEDIS	PERRDIS
Reset	All zeros					

Figure 13-6. EDIS Register Definition

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

Table 13-10. EDIS Field Descriptions

Bits	Name	Description
0–1	—	Reserved
2	BSYDIS	Busy disable. 0 Allow eTSEC to report IEVENT[BSY] status and halt buffer descriptor queue if BSY condition occurs. 1 Do not set IEVENT[BSY] and do not halt buffer descriptor queue if BSY condition occurs.
3	EBERRDIS	Ethernet controller bus error disable. 0 Allow eTSEC to report IEVENT[EBERR] status and halt buffer descriptor queue if EBERR condition occurs. 1 Do not set IEVENT[EBERR] and do not halt buffer descriptor queue if EBERR condition occurs.
4–6	—	Reserved
7	BABTDIS	Babbling transmit error disable. 0 Allow eTSEC to report IEVENT[BABT] status and set the buffer descriptor TR field. 1 Do not set IEVENT[BABT] nor the buffer descriptor TR field.
8	—	Reserved
9	TXEDIS	Transmit error disable. 0 Allow eTSEC to report IEVENT[TXE] status. 1 Do not set IEVENT[TXE] if TXE condition occurs.
10–12	—	Reserved
13	LCDIS	Late collision disable. 0 Allow eTSEC to report IEVENT[LC] status, set the buffer descriptor LC field, and halt buffer descriptor queue if LC condition occurs. 1 Do not set IEVENT[LC] nor the buffer descriptor LC field, and do not halt buffer descriptor queue if LC condition occurs.
14	CRLDIS	Collision retry limit disable. 0 Allow eTSEC to report IEVENT[CRL] status, set the buffer descriptor RL field, and halt buffer descriptor queue if CRL condition occurs. 1 Do not set IEVENT[CRL] nor the buffer descriptor RL field, and do not halt buffer descriptor queue if CRL condition occurs.
15	XFUNDIS	Transmit FIFO underrun disable. 0 Allow eTSEC to report IEVENT[XFUN] status, set the buffer descriptor UN field, and halt buffer descriptor queue if XFUN condition occurs. 1 Do not set IEVENT[XFUN] nor the buffer descriptor UN field, and do not halt buffer descriptor queue if XFUN condition occurs.
16–27	—	Reserved
28	FIRDIS	Filer invalid result error disable. 0 Allow eTSEC to report IEVENT[FIR] status. 1 Do not set IEVENT[FIR] if eTSEC fails to reach a definite filer result when attempting to file a received frame, but discard the frame silently.
29	FIQDIS	Filed frame to invalid queue error disable. 0 Allow eTSEC to report IEVENT[FIQ] status. 1 Do not set IEVENT[FIQ] if eTSEC attempts to file a received frame to an invalid (disabled) RxBD ring, but discard the frame silently.

Table 13-10. EDIS Field Descriptions (continued)

Bits	Name	Description
30	DPEDIS	Data parity error disable. 0 Allow eTSEC to report IEVENT[DPE] status. 1 Do not set IEVENT[DPE] if a parity error occurs in eTSEC's FIFO or filer arrays.
31	PERRDIS	Receive frame parse error disable. 0 Allow eTSEC to report IEVENT[PERR] status. 1 Do not set IEVENT[PERR] if a parse error occurs on a received frame.

13.5.3.1.6 Ethernet Control Register (ECNTRL)

ECNTRL is a register writable by the user to reset, configure, and initialize the eTSEC. Note that the FIFM, GMIIM, TBIM, RPM, and RMM fields are read-only, having been set after sampling signals at power-on-reset.

Figure 13-7 describes the definition for the ECNTRL register.

Offset eTSEC1:0x2_4020; eTSEC2:0x2_5020; eTSEC3:0x2_6020; eTSEC4:0x2_7020 Access: Mixed

	0															15
R	—															
W																
Reset	All zeros															

	16	17	18	19	20	24	25	26	27	28	29	30	31
R	FIFM	CLRCNT	AUTOZ	STEN	—		GMIIM	TBIM	RPM	R100M	RMM	—	—
W													
Reset	All zeros												

Figure 13-7. ECNTRL Register Definition

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

Table 13-11. ECNTRL Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16	FIFM	FIFO mode enable. If this bit is set, 8- or 16-bit FIFO interface mode is enabled. This bit can be pin configured at reset to set or clear. See Section 4.4.3.13, "eTSECn Width." The FIFO width is configured according to RPM. 0 Interface to external signals through the Ethernet MAC. 1 Interface to external signals through the 8/16-bit FIFO interface, bypassing the Ethernet MAC. Frame parsing in this mode automatically assumes that IP packets are being received and transmitted. See FIFOCFG register for configuration of the FIFO interface.
17	CLRCNT	Clear all statistics counters 0 Allow MIB counters to continue to increment. 1 Reset all MIB counters. This bit is self-resetting.

Table 13-11. ECNTRL Field Descriptions (continued)

Bits	Name	Description
18	AUTOZ	Automatically zero MIB counter values. 0 The user must write the addressed counter zero after a host read. 1 The addressed counter value is automatically cleared to zero after a host read. This is a steady state signal and must be set prior to enabling the Ethernet controller and must not be changed without proper care.
19	STEN	MIB counter statistics enabled. 0 Statistics not enabled 1 Enables internal counters to update This is a steady state signal and must be set prior to enabling the Ethernet controller and must not be changed without proper care.
20–24	—	Reserved
25	GMIIM	GMII interface mode. If this bit is set, a PHY with a GMII or RGMII interface is expected to be connected. If cleared, a PHY with an MII or RMII interface is expected. The user should then set MACCFG2[I/F Mode] accordingly. The state of this status bit is defined during power-on reset. See Section 4.4.3.14, “eTSECn Protocol.” 0 MII or RMII mode interface expected 1 GMII or RGMII mode interface expected
26	TBIM	Ten-bit interface mode. If this bit is set, ten-bit interface mode is enabled. This bit can be pin-configured at reset to set or clear. See Section 4.4.3.14, “eTSECn Protocol.” 0 GMII or MII or RMII mode interface 1 TBI mode interface
27	RPM	Reduced-pin mode for Gigabit interfaces. If this bit is set, a reduced-pin interface is expected on either Ethernet and FIFO interfaces. RPM and RMM are never set together. This register can be pin-configured at reset to 0 or 1. See Section 4.4.3.13, “eTSECn Width.” 0 GMII or MII or TBI in non-reduced-pin mode configuration 1 RGMII or RTBI reduced-pin mode FIFO configured for 8-bit operation
27	RPM	Reduced-pin mode for Gigabit interfaces. If this bit is set, a reduced-pin interface is expected on either Ethernet and FIFO interfaces. RPM and RMM are never set together. This register can be pin-configured at reset to 0 or 1. See Section 4.4.3.13, “eTSECn Width.” 0 GMII or MII or TBI in non-reduced-pin mode configuration FIFO configured for 16-bit operation 1 RGMII or RTBI reduced-pin mode FIFO configured for 8-bit operation
28	R100M	RGMII/RMII 100 mode. This bit is ignored unless RPM or RMM are set and MACCFG2[I/F Mode] is assigned to 10/100 (01). If this bit is set, the eTSEC interface is in 100 Mbps speed. 0 RGMII is in 10 Mbps mode; RMII is in 10 Mbps mode, and every 10th RMII Reference clock is used to transfer data 1 RGMII is in 100 Mbps mode; RMII is in 100 Mbps mode, and data is transferred on every Reference clock
29	RMM	Reduced-pin mode for 10/100 interfaces. If this bit is set, an RMII pin interface is expected. RMM must be 0 if RPM = 1. This register can be pin-configured at reset to 0 or 1. See Section 4.4.3.13, “eTSECn Width.” 0 Non-RMII interface mode 1 RMII interface mode
30–31	—	Reserved

The different interface configurations indicated by registers ECNTRL and MACCFG2 are summarized in Table 13-12.

Table 13-12. eTSEC Interface Configurations

Interface Mode	ECNTRL Field						MACCFG2 Field
	FIFM	GMIIM	TBIM	RPM	R100M	RMM	I/F Mode
FIFO 8-bits	1	0	0	1	—	0	—
FIFO 16-bits	1	0	0	0	—	0	—
TBI 1 Gbps	0	0	1	0	—	0	10
RTBI 1 Gbps	0	0	1	1	—	0	10
GMII 1 Gbps[1]	0	1	0	0	—	0	10
RGMII 1 Gbps	0	1	0	1	—	—	10
RGMII 100 Mbps	0	1	0	1	1	—	01
RGMII 10 Mbps	0	1	0	1	0	0	01
MII 10/100 Mbps	0	0	0	0	—	0	01
RMII 100 Mbps	0	0	0	0	1	1	01
RMII 10 Mbps	0	0	0	0	0	1	01

1 See MII 10/100 Mbps mode for GMII 10/100 Mbps ‘fall-back’ mode.

13.5.3.1.7 Pause Time Value Register (PTV)

PTV is a 32-bit register written by the user to store the pause duration used when the eTSEC initiates an IEEE 802.3 PAUSE control frame 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 13.6.3.9, “Flow Control,” for additional details. Figure 13-8 describes the definition for the PTV register.

Offset eTSEC1:0x2_4028; eTSEC2:0x2_5028; eTSEC3:0x2_6028; eTSEC4:0x2_7028 Access: Read/Write

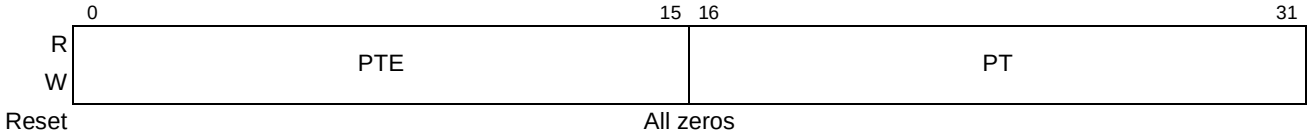


Figure 13-8. PTV Register Definition

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

Table 13-13. PTV Field Descriptions

Bits	Name	Description
0–15	PTE	Extended pause control. This field allows software to add a 16-bit additional control parameter into the PAUSE frame to be sent when TCTRL[TFC_PAUSE] is set. Note that current IEEE 802.3 PAUSE frame format requires this parameter to be cleared.
16–31	PT	Pause time value. Represents the 16-bit pause quanta (that is, 512 bit times). This pause value is used as part of the PAUSE frame to be sent when TCTRL[TFC_PAUSE] is set. See Section 13.6.3.9, “Flow Control,” on page 13-154 for more information.

13.5.3.1.8 DMA Control Register (DMACTRL)

DMACTRL is writable by the user to configure the DMA block. Figure 13-9 describes the definition for the DMACTRL register.

Offset eTSEC1:0x2_402C; eTSEC2:0x2_502C; eTSEC3:0x2_602C; eTSEC4:0x2_702C — Access: Read/Write

	0–15	16	17–23	24	25	26	27	28	29	30	31
R W	—	LE	—	TDSEN	TBDSEN	—	GRS	GTS	TOD	WWR	WOP
Reset	All zeros										

Figure 13-9. DMACTRL Register

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

Table 13-14. DMACTRL Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16	LE	Little-endian descriptor mode enable. This bit controls both the reading and writing of descriptors; data buffers are always transferred in network byte order. 0 RxBDs and TxBDs are interpreted with big-endian byte ordering, as shown in Section 13.6.7.1, “Data Buffer Descriptors.” 1 RxBDs and TxBDs are interpreted with little-endian byte ordering. That is, the 16 bits of flags are considered a complete half-word unit, the buffer length is considered another complete half-word unit, and the buffer pointer is considered a complete word unit.
17–23	—	Reserved
24	TDSEN	Tx Data snoop enable. 0 Disables snooping of all transmit frames from memory. 1 Enables snooping of all transmit frames from memory.
25	TBDSEN	TxBD snoop enable. 0 Disables snooping of all transmit BD memory accesses. 1 Enables snooping of all transmit BD memory accesses.
26	—	Reserved

Table 13-14. 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 contents of the Rx FIFO are then written to memory, and the IEVENT[GRSC] is set to indicate that all current receive buffers have been closed. Because the receive enable bit of the MAC may still be set, the MAC may continue to receive but the eTSEC ignores the receive data until GRS is cleared. If this bit is cleared, the eTSEC scans the input data stream for the start of a new frame (preamble sequence and start of frame delimiter) and the first valid frame received uses the next RxBD. If GRS is set, the user must monitor the graceful receive stop complete (GRSC) bit in the IEVENT register to insure that the graceful receive stop was completed. The user can then clear IEVENT[GRSC] and can write to receive registers that are accessible to both user and the eTSEC hardware without fear of conflict. 0 eTSEC scans input data stream for valid frame. 1 eTSEC stops receiving frames following completion of current frame.
28	GTS	Graceful transmit stop. If this bit is set, the Ethernet controller stops transmission after all frames that are currently in the Tx FIFO or scheduled have been transmitted, and the GTSC interrupt in the IEVENT register is asserted. A frame that has started reading buffer descriptors or data from memory is read to completion and transmitted before the GTSC interrupt occurs. However, if no frame has been scheduled for transmission and the Tx FIFO is empty, the GTSC interrupt is asserted immediately. Once transmission has completed, clearing GTS “restart” transmit. 0 Controller continues. 1 Controller stops transmission after completion of current frame.
29	TOD	Transmit on demand for TxBD ring 0. This bit is applicable only to the transmitter, and requires both TCTRL[TXSCHED] = 00 and DMACTRL[WOP] = 0. If 1 is written to this bit, the eTSEC immediately begins fetching the next TxBD from ring 0, avoiding waiting the normal polling time to check the TxBD’s R bit. This bit is always read as 0. 0 eTSEC continues waiting for the TxBD ring 0 poll timer to expire. 1 eTSEC immediately fetches a new TxBD from ring 0, and resets the poll timer.
30	WWR	Write with response. This bit gives the user the assurance that a BD was updated in memory before it receives an interrupt concerning a transmit or receive frame. 0 Do not wait for acknowledgement from system for BD writes before setting IEVENT bits. 1 Before setting IEVENT bits TXB, TXF, TXE, XFUN, LC, CRL, RXB, RXF, the eTSEC waits for acknowledgement from system that the transmit or receive BD being updated was stored in memory.
31	WOP	Wait or poll for TxBD ring 0. This bit, which is applicable only to the transmitter and when TCTRL[TXSCHED] = 00, provides the user the option for the eTSEC to periodically poll TxBDs or to wait for software to tell eTSEC to fetch a buffer descriptor. While operating in the “Wait” mode, the eTSEC allows two additional reads of a descriptor which is not ready before entering a halt state. No interrupt is driven. To resume transmission, software must clear TSTAT[THLT]. 0 Poll TxBD on ring 0 every 512 serial clocks. 1 Do not poll, but wait for TSTAT[THLT] to be cleared by the user.

13.5.3.1.9 TBI Physical Address Register (TBIPA)

The TBIPA, shown in Figure 13-10, is writable by the user to assign a physical address to the TBI for MII management configuration. The TBI registers are accessed at the offset of TBIPA. For detailed descriptions of the TBI registers (the MII register set for the ten-bit interface) please refer to Section 13.5.4, “Ten-Bit Interface (TBI).”

Offset eTSEC1:0x2_4030; eTSEC2:0x2_5030; eTSEC3:0x2_6030; eTSEC4:0x2_7030 Access: Read/Write

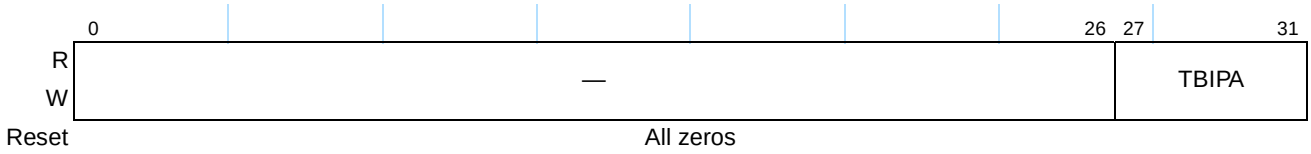


Figure 13-10. TBIPA Register Definition

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

Table 13-15. TBIPA Field Descriptions

Bits	Name	Description
0–26	—	Reserved
27–31	TBIPA	This field is used to program the PHY address of the ten-bit interface's MII management bus. To access the TBI register the user must write the TBIPA value to the MIIMADD [PHY Address] register located in the MAC register section. PHY Address 0 is reserved. Refer to Section 13.5.3.5.8, "MII Management Address Register (MIIMADD)."

13.5.3.2 eTSEC Transmit Control and Status Registers

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

13.5.3.2.1 Transmit Control Register (TCTRL)

This register is writable by the user to configure the transmit block. Figure 13-11 describes the TCTRL register.

Offset eTSEC1:0x2_4100; eTSEC2:0x2_5100; eTSEC3:0x2_6100; eTSEC4:0x2_7100 Access: Mixed

Figure 13-11. TCTRL Register Definition

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

Table 13-16. TCTRL Field Descriptions

Bits	Name	Description
0–16	—	Reserved
17	IPCSEN	IP header checksum generation enable. When set, the eTSEC offloads IPv4 header checksum generation. See Section 13.6.4.2, "Transmit Path Off-Load and Tx PTP Packet Parsing," on page 13-161. 0 IP header checksum generation is disabled even if enabled in a transmit frame control block. 1 IP header checksum generation is performed for IPv4 headers as determined by the settings in the current transmit frame control block.

Table 13-16. TCTRL Field Descriptions (continued)

Bits	Name	Description
18	TUCSEN	TCP/UDP header checksum generation enable. When set, the eTSEC offloads TCP or UDP header checksum generation. See Section 13.6.4.2, "Transmit Path Off-Load and Tx PTP Packet Parsing," on page 13-161. 0 TCP or UDP header checksum generation is disabled even if enabled in a transmit frame control block. 1 TCP or UDP header checksum generation is performed as determined by the settings in the current transmit frame control block.
19	VLINS	VLAN (IEEE Std. 802.1Q) tag insertion enable. Applicable only for transmission through the Ethernet MAC. 0 Do not insert a VLAN tag into the frame. 1 Insert a VLAN tag into the frame. If the frame FCB has a valid VLAN field, use the FCB to source the VLAN control word, otherwise take the default VLAN control word from register DFVLAN.
20	THDF	Transmit half-duplex flow control under software control for 10-/100-Mbps half-duplex media. This bit is not self-resetting. 0 Disable back pressure 1 Back pressure is applied to media by raising carrier
21–26	—	Reserved
27	RFC_PAUSE	Receive flow control pause frame (written by the eTSEC). This read-only status bit is set if a flow control pause frame was received and the transmitter is paused for the duration defined in the received pause frame. This bit automatically clears after the pause duration is complete. 0 Pause duration complete. 1 Flow control pause frame received.
28	TFC_PAUSE	Transmit flow control pause frame. Set this bit to transmit a PAUSE frame. If this bit is set, the MAC stops transmission of data frames after the currently transmitting frame completes. Next, the MAC transmits a pause control frame with the duration value obtained from the PTV register. The TXC event occurs after sending the pause control frame. Finally, the controller clears TFC_PAUSE and resumes transmitting data frames as before. Note that pause control frames can still be transmitted if the Tx controller is stopped due to user assertion of DMACTRL[GTS] or reception of a PAUSE frame. 0 No request for Tx PAUSE frame pending or transmission complete. 1 Software request for Tx PAUSE frame pending.

Table 13-16. TCTRL Field Descriptions (continued)

Bits	Name	Description
29–30	TXSCHED	Transmit ring scheduling algorithm. This field determines which scheme the transmit scheduler uses to arbitrate between the enabled TxBD rings. The scheme chosen also controls how the DMACTRL and TQUEUE bits are interpreted. Ring polling is supported only by mode 00; the other modes require software to restart rings with the TSTAT register. TCP/IP offload can be enabled with any scheduling mode. 00 Single polled ring mode. TxBD ring 0 is the only ring serviced, even if other rings are enabled and ready. In this scheduler mode, the DMACTRL[WOP] and DMACTRL[TOD] bits control polling and retry behavior. This mode supports ring polling, and allows fetching of a non-ready TxBD to be retried twice. 01 Priority scheduling mode. All enabled TxBD rings are serviced in ascending ring index order. Once a non-ready TxBD has been fetched from the lowest-numbered ring, the eTSEC attempts to fetch TxBDs from the next enabled ring having a higher index, until transmission stops for lack of data. TSTAT records whenever a TxBD ring is exhausted. 10 Modified weighted round-robin scheduling mode. Each TxBD ring is polled in sequence for frames that are ready for transmission. If a non-ready TxBD is fetched from a ring, that ring is removed from the scheduling pool until software re-enables it. Ready frames are repeatedly transmitted from a chosen ring until its transmission quota is exhausted. The transmission quota for TxBD ring *n* is set to WT*n* × 64 bytes, where WT*n* is a weight from the TR03WT/TR47WT registers. If a ring transmits more data than its quota allows, the excess is deducted from its quota on the next transmission opportunity, thereby preventing large frames from monopolizing the eTSEC bandwidth. 11 Reserved
31	—	Reserved

13.5.3.2.2 Transmit Status Register (TSTAT)

This register is read/write-one-to-clear and is written by the eTSEC to convey DMA status information for each TxBD ring. The halt bit only has meaning for enabled rings. After processing transmit-related interrupts, software should use TSTAT to restart transmission from rings that may have been affected by the interrupt condition. In particular, an error condition that prevents eTSEC from continuing transmission halts DMA from all rings, including the ring that gave rise to the error. Figure 13-12 describes the TSTAT register.

Offset eTSEC1:0x2_4104; eTSEC2:0x2_5104; eTSEC3:0x2_6104; eTSEC4:0x2_7104 — Access: w1c

	0	1	2	3	4	5	6	7	8–15
R	THLT0	THLT1	THLT2	THLT3	THLT4	THLT5	THLT6	THLT7	—
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	
Reset	All zeros								

	16	17	18	19	20	21	22	23	24–31
R	TXF0	TXF1	TXF2	TXF3	TXF4	TXF5	TXF6	TXF7	—
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	
Reset	All zeros								

Figure 13-12. TSTAT Register Definition

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

Table 13-17. TSTAT Field Descriptions

Bits	Name	Description
0	THLT0	Transmit halt of ring 0. Set by the eTSEC if is no longer processing transmit frames from this TxBD ring, and DMA from this ring is disabled. To re-start transmission from this TxBD ring, this bit must be cleared by writing 1 to it. This bit is set only on a general error condition (as in IEVENT[TXE]), regardless of TQUEUE[EN0], or if no ready TxBDs can be fetched. DMACTRL[GTS] being set by the user does not cause this bit to be set. Software should examine the halted queue's buffer descriptors for repeatable error conditions before taking it out of the halt state. Failure to do so may cause an effective livelock, in which the error condition recurs and halts all queues again. Repeatable error conditions which cause halt include: Bus error: • Invalid BD or data address • Uncorrectable error on BD or data read TxBD programming errors: • Ready=1 and length=0
1	THLT1	Transmit halt of ring 1. Set by the eTSEC if is no longer processing transmit frames from this TxBD ring, and DMA from this ring is disabled. To re-start transmission from this TxBD ring, this bit must be cleared by writing 1 to it. This bit is set only on a general error condition (as in IEVENT[TXE]), regardless of TQUEUE[EN1], or if no ready TxBDs can be fetched.DMACTRL[GTS] being set by the user does not cause this bit to be set. Software should examine the halted queue's buffer descriptors for repeatable error conditions before taking it out of the halt state. Failure to do so may cause an effective livelock, in which the error condition recurs and halts all queues again. Repeatable error conditions which cause halt include: Bus error: • Invalid BD or data address • Uncorrectable error on BD or data read TxBD programming errors: • Ready=1 and length=0
2	THLT2	Transmit halt of ring 2. Set by the eTSEC if is no longer processing transmit frames from this TxBD ring, and DMA from this ring is disabled. To re-start transmission from this TxBD ring, this bit must be cleared by writing 1 to it. This bit is set only on a general error condition (as in IEVENT[TXE]), regardless of TQUEUE[EN2], or if no ready TxBDs can be fetched. DMACTRL[GTS] being set by the user does not cause this bit to be set. Software should examine the halted queue's buffer descriptors for repeatable error conditions before taking it out of the halt state. Failure to do so may cause an effective livelock, in which the error condition recurs and halts all queues again. Repeatable error conditions which cause halt include: Bus error: • Invalid BD or data address • Uncorrectable error on BD or data read TxBD programming errors: • Ready=1 and length=0

Table 13-17. TSTAT Field Descriptions (continued)

Bits	Name	Description
3	THLT3	Transmit halt of ring 3. Set by the eTSEC if is no longer processing transmit frames from this TxBD ring, and DMA from this ring is disabled. To re-start transmission from this TxBD ring, this bit must be cleared by writing 1 to it. This bit is set only on a general error condition (as in IEVENT[TXE]), regardless of TQUEUE[EN3], or if no ready TxBDs can be fetched. DMACTRL[GTS] being set by the user does not cause this bit to be set. Software should examine the halted queue's buffer descriptors for repeatable error conditions before taking it out of the halt state. Failure to do so may cause an effective livelock, in which the error condition recurs and halts all queues again. Repeatable error conditions which cause halt include: Bus error: • Invalid BD or data address • Uncorrectable error on BD or data read TxBD programming errors: • Ready=1 and length=0
4	THLT4	Transmit halt of ring 4. Set by the eTSEC if is no longer processing transmit frames from this TxBD ring, and DMA from this ring is disabled. To re-start transmission from this TxBD ring, this bit must be cleared by writing 1 to it. This bit is set only on a general error condition (as in IEVENT[TXE]), regardless of TQUEUE[EN4], or if no ready TxBDs can be fetched. DMACTRL[GTS] being set by the user does not cause this bit to be set. Software should examine the halted queue's buffer descriptors for repeatable error conditions before taking it out of the halt state. Failure to do so may cause an effective livelock, in which the error condition recurs and halts all queues again. Repeatable error conditions which cause halt include: Bus error: • Invalid BD or data address • Uncorrectable error on BD or data read TxBD programming errors: • Ready=1 and length=0
5	THLT5	Transmit halt of ring 5. Set by the eTSEC if is no longer processing transmit frames from this TxBD ring, and DMA from this ring is disabled. To re-start transmission from this TxBD ring, this bit must be cleared by writing 1 to it. This bit is set only on a general error condition (as in IEVENT[TXE]), regardless of TQUEUE[EN5], or if no ready TxBDs can be fetched. DMACTRL[GTS] being set by the user does not cause this bit to be set. Software should examine the halted queue's buffer descriptors for repeatable error conditions before taking it out of the halt state. Failure to do so may cause an effective livelock, in which the error condition recurs and halts all queues again. Repeatable error conditions which cause halt include: Bus error: • Invalid BD or data address • Uncorrectable error on BD or data read TxBD programming errors: • Ready=1 and length=0

Table 13-17. TSTAT Field Descriptions (continued)

Bits	Name	Description
6	THLT6	Transmit halt of ring 6. Set by the eTSEC if is no longer processing transmit frames from this TxBD ring, and DMA from this ring is disabled. To re-start transmission from this TxBD ring, this bit must be cleared by writing 1 to it. This bit is set only on a general error condition (as in IEVENT[TXE]), regardless of TQUEUE[EN6], or if no ready TxBDs can be fetched. DMACTRL[GTS] being set by the user does not cause this bit to be set. Software should examine the halted queue's buffer descriptors for repeatable error conditions before taking it out of the halt state. Failure to do so may cause an effective livelock, in which the error condition recurs and halts all queues again. Repeatable error conditions which cause halt include: Bus error: • Invalid BD or data address • Uncorrectable error on BD or data read TxBD programming errors: • Ready=1 and length=0
7	THLT7	Transmit halt of ring 7. Set by the eTSEC if is no longer processing transmit frames from this TxBD ring, and DMA from this ring is disabled. To re-start transmission from this TxBD ring, this bit must be cleared by writing 1 to it. This bit is set only on a general error condition (as in IEVENT[TXE]), regardless of TQUEUE[EN7], or if no ready TxBDs can be fetched. DMACTRL[GTS] being set by the user does not cause this bit to be set. Software should examine the halted queue's buffer descriptors for repeatable error conditions before taking it out of the halt state. Failure to do so may cause an effective livelock, in which the error condition recurs and halts all queues again. Repeatable error conditions which cause halt include: Bus error: • Invalid BD or data address • Uncorrectable error on BD or data read TxBD programming errors: • Ready=1 and length=0
8–15	—	Reserved
16	TXF0	Transmit frame event occurred on ring 0. Set by the eTSEC if IEVENT[TXF] was set in relation to transmitting a frame from this ring.
17	TXF1	Transmit frame event occurred on ring 1. Set by the eTSEC if IEVENT[TXF] was set in relation to transmitting a frame from this ring.
18	TXF2	Transmit frame event occurred on ring 2. Set by the eTSEC if IEVENT[TXF] was set in relation to transmitting a frame from this ring.
19	TXF3	Transmit frame event occurred on ring 3. Set by the eTSEC if IEVENT[TXF] was set in relation to transmitting a frame from this ring.
20	TXF4	Transmit frame event occurred on ring 4. Set by the eTSEC if IEVENT[TXF] was set in relation to transmitting a frame from this ring.
21	TXF5	Transmit frame event occurred on ring 5. Set by the eTSEC if IEVENT[TXF] was set in relation to transmitting a frame from this ring.
22	TXF6	Transmit frame event occurred on ring 6. Set by the eTSEC if IEVENT[TXF] was set in relation to transmitting a frame from this ring.

Table 13-17. TSTAT Field Descriptions (continued)

Bits	Name	Description
23	TXF7	Transmit frame event occurred on ring 7. Set by the eTSEC if IEVENT[TXF] was set in relation to transmitting a frame from this ring.
24–31	—	Reserved

13.5.3.2.3 Default VLAN Control Word Register (DFVLAN)

This register defines the default value for the VLAN Ethertype and control word when VLAN tags are automatically inserted by the eTSEC, and no per-frame VLAN data is supplied by software. On receive, this register defines a customizable VLAN Ethertype for automatic deletion. Note that an Ethertype of 0x8808 (Control Word) is not permitted as a custom VLAN tag. Frames with an Ethertype of 0x8808 are dropped by the receiver. In the case of frames containing stacked VLAN tags, this register defines the tag associated with the outer or metropolitan area VLAN. Figure 13-13 describes the DFVLAN register.

Offset eTSEC1:0x2_4108; eTSEC2:0x2_5108; eTSEC3:0x2_6108; eTSEC4:0x2_7108

Access: Read/Write

Bits	0–15	16–18	19	20–31
R/W	TAG	PRI	CFI	VID
Reset	1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0	0 0 0	0	0 0 0 0 0 0 0 0 0 0 0 0

Figure 13-13. DFVLAN Register Definition

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

Table 13-18. DFVLAN Field Descriptions

Bits	Name	Description
0–15	TAG	This is the default Ethertype used to tag VLAN frames. On transmit, this tag is inserted ahead of the VLAN control word; TAG should be set to 0x8100 for IEEE 802.1Q VLAN. On receive, an Ethertype matching TAG or an Ethertype of 0x8100 marks a VLAN-tagged frame. Note that if using DFVLAN to set a custom ethertype (that is, using a value other than 0x8100), packets received with a custom tag are not counted by any of the RMON counters. Affected counters include TRMGV, RMCA, RBCA, RXCF, RXPF, RXUO, RALN, RFLR, ROVR, RJBR, TMCA, TBCA, TXPF, TXCF.
16–18	PRI	This is the default value used for the IEEE Std. 802.1p frame priority.
19	CFI	This is the default value used for the IEEE Std. 802.1Q canonical format indicator.
20–31	VID	This is the default value used for the virtual-LAN identifier in VLAN-tagged frames. A value of zero is defined as the null VLAN, however field PRI may be still set independently.

13.5.3.2.4 Transmit Interrupt Coalescing Register (TXIC)

The TXIC register enables and configures the operational parameters for interrupt coalescing associated with transmitted frames. Figure 13-14 describes the definition for the TXIC register.

Offset eTSEC1:0x2_4110; eTSEC2:0x2_5110; eTSEC3:0x2_6110; eTSEC4:0x2_7110 — Access: Read/Write

	0	1	2	3 – 10	11 – 15	16 – 31
R W	ICEN	ICCS	—	ICFT	—	ICTT
Reset	All zeros					

Figure 13-14. TXIC Register Definition

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

Table 13-19. TXIC Field Descriptions

Bits	Name	Description
0	ICEN	Interrupt coalescing enable 0 Interrupt coalescing is disabled. Interrupts are raised as they are received. 1 Interrupt coalescing is enabled. If the eTSEC transmit frame interrupt is enabled (IMASK[TXFEN] is set), an interrupt is raised when the threshold number of frames is reached (defined by TXIC[ICFT]) or when the threshold timer expires (determined by TXIC[ICTT]).
1	ICCS	Interrupt coalescing timer clock source. 0 The coalescing timer advances count every 64 eTSEC Tx interface clocks (TSEC*n*_GTX_CLK). 1 The coalescing timer advances count every 64 system clocks. This mode is recommended for FIFO operation.
2	—	Reserved
3–10	ICFT	Interrupt coalescing frame count threshold. While interrupt coalescing is enabled (TXIC[ICEN] is set), this value determines how many frames are transmitted before raising an interrupt. The eTSEC threshold counter is reset to ICFT following an interrupt. The value of ICFT must be greater than zero to avoid unpredictable behavior.
11–15	—	Reserved
16–31	ICTT	Interrupt coalescing timer threshold. While interrupt coalescing is enabled (TXIC[ICEN] is set), this value determines the maximum amount of time after transmitting a frame before raising an interrupt. If frames have been transmitted but the frame count threshold has not been met, an interrupt is raised when the threshold timer reaches zero. The threshold timer is reset to the value in this field and begins counting down upon transmission of the first frame having its TxBD[I] bit set. The threshold value is represented in units of 64 clock periods as specified by the timer clock source (TXIC[ICCS[). The value of ICTT must be greater than zero to avoid unpredictable behavior.

13.5.3.2.5 Transmit Queue Control Register (TQUEUE)

The TQUEUE register, shown in Figure 13-15, selectively enables each of the TxBD rings 0–7. By default, TxBD ring 0 is enabled.

Offset eTSEC1:0x2_4114; eTSEC2:0x2_5114; eTSEC3:0x2_6114; eTSEC4:0x2_7114 — Access: Read/Write

	0–15	16	17	18	19	20	21	22	23	24–31
R W	—	EN0	EN1	EN2	EN3	EN4	EN5	EN6	EN7	—
Reset	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	1	0	0	0	0	0	0	0	0 0 0 0 0 0 0 0

Figure 13-15. TQUEUE Register Definition

Table 13-20 describes the TQUEUE register.

Table 13-20. TQUEUE Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16	EN0	Transmit queue 0 enable. 0 TxBD ring is not queried for transmission. In effect the transmit queue is disabled. 1 TxBD ring is queried for transmission.
17	EN1	Transmit queue 1 enable. 0 TxBD ring is not queried for transmission. In effect the transmit queue is disabled. 1 TxBD ring is queried for transmission.
18	EN2	Transmit queue 2 enable. 0 TxBD ring is not queried for transmission. In effect the transmit queue is disabled. 1 TxBD ring is queried for transmission.
19	EN3	Transmit queue 3 enable. 0 TxBD ring is not queried for transmission. In effect the transmit queue is disabled. 1 TxBD ring is queried for transmission.
20	EN4	Transmit queue 4 enable. 0 TxBD ring is not queried for transmission. In effect the transmit queue is disabled. 1 TxBD ring is queried for transmission.
21	EN5	Transmit queue 5 enable. 0 TxBD ring is not queried for transmission. In effect the transmit queue is disabled. 1 TxBD ring is queried for transmission.
22	EN6	Transmit queue 6 enable. 0 TxBD ring is not queried for transmission. In effect the transmit queue is disabled. 1 TxBD ring is queried for transmission.
23	EN7	Transmit queue 7 enable. 0 TxBD ring is not queried for transmission. In effect the transmit queue is disabled. 1 TxBD ring is queried for transmission.
24–31	—	Reserved

13.5.3.2.6 TxBD Ring 0–3 Weighting Register (TR03WT)

When modified weighted round-robin Tx scheduling is enabled (TCTRL[TXSCHED] = 10), this register determines the weighting applied to each transmit queue for queues 0 to 3. For priority-based scheduling, TR03WT has no effect. A description of how queue weights affect eTSEC's round-robin algorithm appears in Section 13.6.5.3.2, "Modified Weighted Round-Robin Queuing (MWRR)." Figure 13-16 describes the TR03WT register.

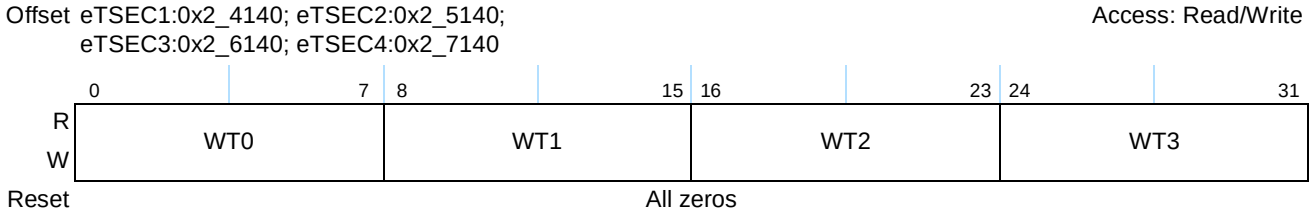


Figure 13-16. TR03WT Register Definition

Table 13-21 describes the fields of the TR03WT register.

Table 13-21. TR03WT Field Descriptions

Bits	Name	Description
0–7	WT0	Weighting value for TxBD ring 0 when TCTRL[TXSCHED] = 10. On each round of the Tx scheduler, a minimum of WT0 × 64 bytes of data are scheduled for transmission from TxBD ring 0. Clearing this field prevents transmission.
8–15	WT1	Weighting value for TxBD ring 1 when TCTRL[TXSCHED] = 10. On each round of the Tx scheduler, a minimum of WT1 × 64 bytes of data are scheduled for transmission from TxBD ring 1. Clearing this field prevents transmission.
16–23	WT2	Weighting value for TxBD ring 2 when TCTRL[TXSCHED] = 10. On each round of the Tx scheduler, a minimum of WT2 × 64 bytes of data are scheduled for transmission from TxBD ring 2. Clearing this field prevents transmission.
24–31	WT3	Weighting value for TxBD ring 3 when TCTRL[TXSCHED] = 10. On each round of the Tx scheduler, a minimum of WT3 × 64 bytes of data are scheduled for transmission from TxBD ring 3. Clearing this field prevents transmission.

13.5.3.2.7 TxBD Ring 4–7 Weighting Register (TR47WT)

When modified weighted round-robin Tx scheduling is enabled (TCTRL[TXSCHED] = 10), this register determines the weighting applied to each enabled transmit queue for queues 4 to 7. For priority-based scheduling, TR47WT has no effect. A description of how queue weights affect eTSEC's modified weighted round-robin algorithm appears in Section 13.6.5.3.2, "Modified Weighted Round-Robin Queuing (MWRR)." Figure 13-17 describes the definition for the TR47WT register.

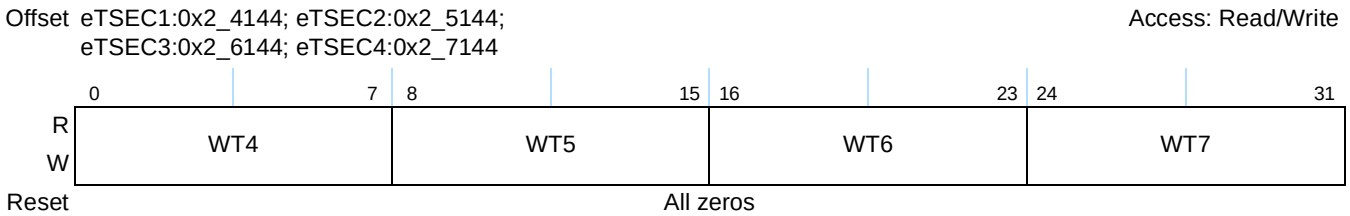


Figure 13-17. TR47WT Register Definition

Table 13-22 describes the fields of the TR47WT register.

Table 13-22. TR47WT Field Descriptions

Bits	Name	Description
0–7	WT4	Weighting value for TxBD ring 4 when TCTRL[TXSCHED] = 10. On each round of the Tx scheduler, a minimum of WT4 × 64 bytes of data are scheduled for transmission from TxBD ring 4. Clearing this field prevents transmission.
8–15	WT5	Weighting value for TxBD ring 5 when TCTRL[TXSCHED] = 10. On each round of the Tx scheduler, a minimum of WT5 × 64 bytes of data are scheduled for transmission from TxBD ring 5. Clearing this field prevents transmission.
16–23	WT6	Weighting value for TxBD ring 6 when TCTRL[TXSCHED] = 10. On each round of the Tx scheduler, a minimum of WT6 × 64 bytes of data are scheduled for transmission from TxBD ring 6. Clearing this field prevents transmission.
24–31	WT7	Weighting value for TxBD ring 7 when TCTRL[TXSCHED] = 10. On each round of the Tx scheduler, a minimum of WT7 × 64 bytes of data are scheduled for transmission from TxBD ring 7. Clearing this field prevents transmission.

13.5.3.2.8 Transmit Data Buffer Pointer High Register (TBDBPH)

The TBDBPH register is written by the user with the most significant address bits common to all TxBD buffer addresses, TxBD[Data Buffer Pointer]. As a consequence, all Tx buffers must be placed in a 4 gigabyte segment of memory whose base address is prefixed by the bits in TBDBPH. The TxBD ring itself can reside in a different memory region (based at TBASEH). Figure 13-18 describes the definition for the TBDBPH register.

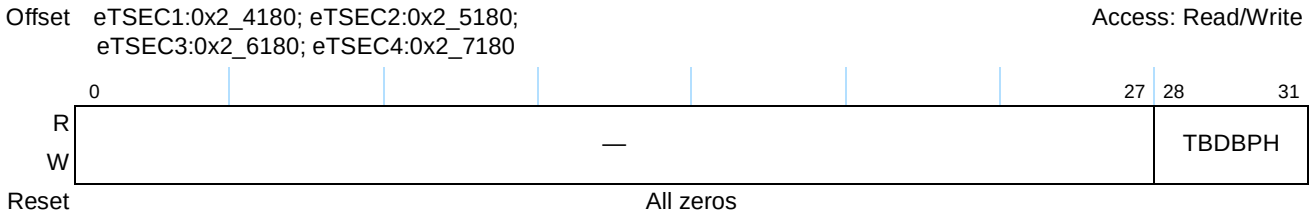


Figure 13-18. TBDBPH Register Definition

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

Table 13-23. TBDBPH Field Descriptions

Bits	Name	Description
0–27	—	Reserved
28–31	TBDBPH	Most significant bits common to all data buffer addresses contained in TxBDs. The user must initialize TBDBPH before enabling the eTSEC transmit function.

13.5.3.2.9 Transmit Buffer Descriptor Pointers 0–7 (TBPTR0–TBPTR7)

TBPTR0–TBPTR7 each contains the low-order 32 bits of the next transmit buffer descriptor address for their respective TxBD ring. Figure 13-19 describes the TBPTR registers. These registers takes on the value of their ring's associated TBASE when the TBASE register is written by software. Software must not write TBPTR0–TBPTR7 while eTSEC is actively transmitting frames. However, TBPTR0– TBPTR7 can be

modified when the transmitter is disabled or when no Tx buffer is in use (after a GRACEFUL STOP TRANSMIT command is issued and the frame completes its transmission) in order to change the next TxBD eTSEC transmits.

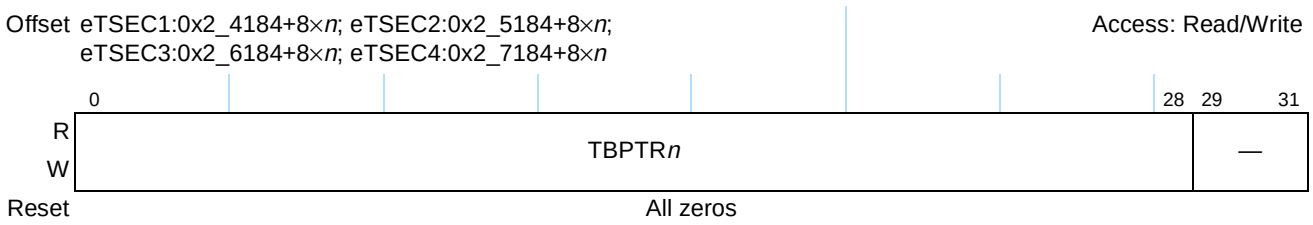


Figure 13-19. TBPTR0–TBPTR7 Register Definition

Table 13-24 describes the fields of the TBPTR*n* register.

Table 13-24. TBPTR*n* Field Descriptions

Bits	Name	Description
0–28	TBPTR*n*	Current TxBD pointer for TxBD ring *n*. Points to the current BD being processed or to the next BD the transmitter uses when it is idling. When the end of the TxBD ring is reached, eTSEC initializes TBPTR*n* to the value in the corresponding TBASE*n*. The TBPTR register is internally written by the eTSEC's DMA controller during transmission. The pointer increments by eight (bytes) each time a descriptor is closed successfully by the eTSEC. Note that the three least significant bits of this register are read-only and zero. After an error condition, the eTSEC returns TBPTR*n* to point to the first BD of the frame partially transmitted.
29–31	—	Reserved

13.5.3.2.10 Transmit Descriptor Base Address High Register (TBASEH)

The TBASEH register is written by the user with the most significant address bits common to all TxBD addresses, including TBASE0–TBASE7 and TBPTR0–TBPTR7. As a consequence, all TxBD rings must be placed in a 4 Gbyte segment of memory whose base address is prefixed by the bits in TBASEH. Data buffers are located in a potentially different region, based at TBDBPH. Figure 13-20 describes the TBASEH register.

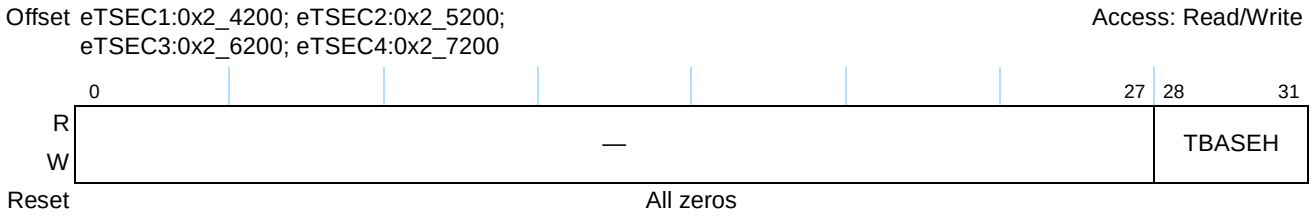


Figure 13-20. TBASEH Register Definition

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

Table 13-25. TBASEH Field Descriptions

Bits	Name	Description
0–27	—	Reserved
28–31	TBASEH	Most significant bits common to all TxBD addresses—except data buffer pointers. The user must initialize TBASEH before enabling the eTSEC transmit function.

13.5.3.2.11 Transmit Descriptor Base Address Registers (TBASE0–TBASE7)

The TBASE*n* registers are written by the user with the base address of each TxBD ring *n*. Each such value must be divisible by eight, since the three least significant bits always write as 000. Figure 13-21 describes the definition for the TBASE*n* registers.

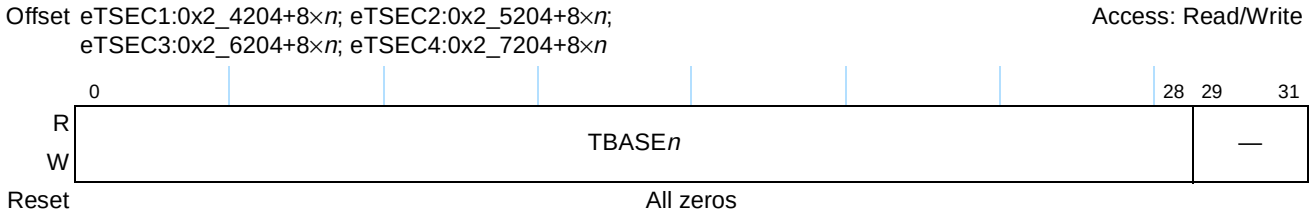


Figure 13-21. TBASE Register Definition

Table 13-26 describes the fields of the TBASE*n* registers.

Table 13-26. TBASE0–TBASE7 Field Descriptions

Bits	Name	Description
0–28	TBASE*n*	Transmit base for ring *n*. TBASE defines the starting location in the memory map for the eTSEC TxBDs. This field must be 8-byte aligned. Together with setting the W (wrap) bit in the last BD, the user can select how many BDs to allocate for the transmit packets. The user must initialize TBASE before enabling the eTSEC transmit function on the associated ring.
29–31	—	Reserved

13.5.3.3 eTSEC Receive Control and Status Registers

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

13.5.3.3.1 Receive Control Register (RCTRL)

The RCTRL register is programmed by the user and controls the operational mode of the receiver. It must be written only after a system reset (at initialization) or after a graceful receive stop has completed. Figure 13-22 describes the RCTRL register.

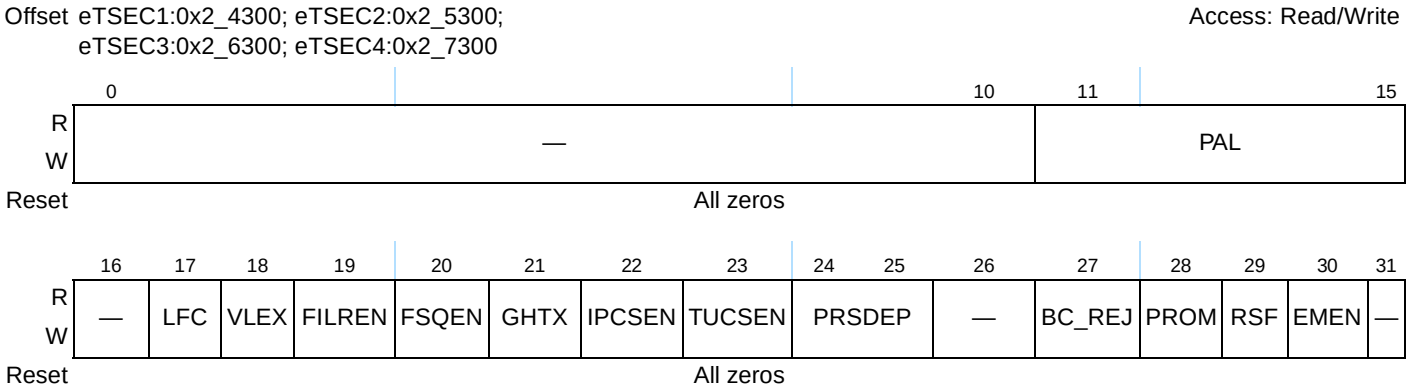


Figure 13-22. RCTRL Register Definition

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

Table 13-27. RCTRL Field Descriptions

Bits	Name	Description
0–10	—	Reserved
11–15	PAL	Packet alignment padding length. If not zero, PAL (1–31) bytes of zero padding are inserted before the start of each received frame, but following the RxFCB if TOE is enabled. For Ethernet where optional preamble extraction is enabled, the padding appears before the preamble, otherwise the padding precedes the layer 2 header. The value of PAL can be set so that the start of the IP header in the receive data buffer is aligned to a 32-bit boundary. Normally, setting PAL = 2 provides minimal padding to ensure such alignment of the IP header.
17	LFC	Lossless flow control. When set, the eTSEC determines the number of free BDs (through RQPARM*n*[LEN] and RBTPTR*n*) in each active ring. Should the free BD count in an active ring drop below its setting for RQPARM*n*[FBTHR], the eTSEC asserts link layer flow control. For full-duplex ethernet connections, the eTSEC emits a pause frame as if TCTRL[TFC_PAUSE] was set. For FIFO packet interface connections, the RFC signal is asserted. 0 Disabled. This is the default 1 Enabled, calculate the free BDs in each active ring and assert link layer flow control if required.
18	VLEX	Enable automatic VLAN tag extraction and deletion from Ethernet frames. Note that VLEX must be cleared if L2OFF is non-zero. 0 Do not delete VLAN tags from received Ethernet frames. 1 If a VLAN tag is seen after the Ethernet source address, and PRSDEP is non-zero, delete the VLAN tag and return the VLAN control word in the frame control block returned with this frame. Note that if PRSDEP is cleared, VLEX must be cleared as well. (VLAN tag extraction is only supported when the parser is enabled.)
19	FILREN	Filer enable. When set, the receive frame filer is enabled. This file accepted frames to a particular RxBD ring according to rules defined in the filer table. In this case, PRSDEP must not be cleared. 0 Do not search the receive queue filer table for received frames. All received frames are sent to RxBD ring 0 by default. 1 Search the receive queue filer table for received frames, and let the filer determine the index of the RxBD ring for each frame. Note that if PRSDEP is cleared, FILREN must be cleared as well.
20	FSQEN	Enable single-queue mode for the receive frame filer. This bit is ignored unless FILREN is also set. 0 The filer chooses the RxBD ring using the least significant bits of the virtual queue ID as a ring index. 1 The filer always attempts to file received frames to ring 0, regardless of virtual queue ID. This mode is intended for operating the filer as a packet classification engine.
21	GHTX	Group address hash table extend. By default, the group address hash table is 256 entries (as defined by registers GADDR0–GADDR7); registers IGADDR0–IGADDR7 are then used to define the individual address hash table. When this bit is set, the hash table is extended to a total of 512 entries (IGADDR0–IGADDR7 are then the first 256 entries of the extended 512-entry group address hash table). 0 Both the individual and group hash functions are the 8 MSBs of the CRC-32 of the Ethernet destination address. 1 The group hash function is the 9 MSBs of the CRC-32 of the Ethernet destination address. The individual address hash function is unavailable.
22	IPCSEN	IP Checksum verification enable. See Section 13.6.4.3, “Receive Path Off-Load.” 0 IPv4 header checksums are not verified by the eTSEC—even if layer 3 parsing is enabled. 1 Perform IPv4 header checksum verification if PRSDEP > 01.

Table 13-27. RCTRL Field Descriptions (continued)

Bits	Name	Description
23	TUCSEN	TCP or UDP Checksum verification enable. See Section 13.6.4.3, "Receive Path Off-Load." 0 TCP or UDP checksums are not verified by the eTSEC—even if layer 4 parsing is enabled. 1 Perform TCP or UDP checksum verification if PRSDEP = 11.
24–25	PRSDEP	Parser control. The level of parser layer recognition is determined as follows: 00 Parser disabled. Receive frame filer must also be disabled by clearing RCTRL[FILREN]. This should be the setting for raw (non-IP) packets received over a FIFO interface. 01 Only L2 (Ethernet) protocols are recognized. For packets received over a FIFO interface, this parse level is unavailable. 10 L2 and L3 (IP) protocols are recognized. This is the minimum parse level for IP packets received over a FIFO interface. 11 L2, L3, and L4 (TCP/UDP) protocols are recognized. If this field is non-zero, a TOE frame control block is prepended to the received frame, and the first RxBD points to the FCB. Note that if PRSDEP is cleared, VLEX must be cleared as well. (VLAN tag extraction is only supported when the parser is enabled.) Also, if PRSDEP is cleared, FILREN must also be cleared.
26	—	Reserved
27	BC_REJ	Broadcast frame reject. If this bit is set, frames with DA (destination address) = FFFF_FFFF_FFFF are rejected unless RCTRL[PROM] is set. If both BC_REJ and RCTRL[PROM] are set, then frames with broadcast DA are accepted and the M (MISS) bit is set in the receive BD.
28	PROM	Promiscuous mode. All Ethernet frames, regardless of destination address, are accepted.
29	RSF	Receive short frame mode. When set, enables the reception of frames shorter than 64 bytes. For packets received over the FIFO packet interface, this bit has no effect (packets shorter than 64 bytes are always accepted). 0 Ethernet frames less than 64B in length are silently dropped. 1 Frames more than 16B and less than 64B in length are accepted upon a DA match. Note that frames less than or equal to 16B in length are always silently dropped.
30	EMEN	Exact match MAC address enable. If this bit is set, the MAC01ADDR1–MAC15ADDR1 and MAC01ADDR2–MAC15ADDR2 registers are recognized as containing MAC addresses aliasing the MAC's station address. Setting this bit therefore allows eTSEC to receive Ethernet frames having a destination address matching one of these 15 addresses.
31	—	Reserved

13.5.3.3.2 Receive Status Register (RSTAT)

The eTSEC writes to this register under the following conditions:

- A frame interrupt event occurred on one or more RxBD rings
- The receiver runs out of descriptors due to a busy condition on a RxBD ring
- The receiver was halted because an error condition was encountered while receiving a frame

Writing 1 to any bit of this register clears it. Software should clear the QHLT bit to take eTSEC's receiver function out of halt state for the associated queue. Figure 13-23 describes the definition for the RSTAT register.

Offset eTSEC1:0x2_4304; eTSEC2:0x2_5304; eTSEC3:0x2_6304; eTSEC4:0x2_7304 Access: w1c

	0–7	8	9	10	11	12	13	14	15
R	—	QHLT0	QHLT1	QHLT2	QHLT3	QHLT4	QHLT5	QHLT6	QHLT7
W		w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset		All zeros							

	16–23	24	25	26	27	28	29	30	31
R	—	RXF0	RXF1	RXF2	RXF3	RXF4	RXF5	RXF6	RXF7
W		w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset		All zeros							

Figure 13-23. RSTAT Register Definition

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

Table 13-28. RSTAT Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8	QHLT0	RxBD queue 0 is halted. It is a hardware-initiated stop indication. (DMACTRL[GRS] being set by the user does not cause a QHLT0 to be set.). The current frame and all other frames directed to a halted queue are discarded. A write with a value of 1 re-enables the queue for receiving. 0 This queue is enabled for reception. (That is, it is not halted) 1 All controller receive activity to this queue is halted.
9	QHLT1	RxBD queue 1 is halted. It is a hardware-initiated stop indication. (DMACTRL[GRS] being set by the user does not cause a QHLT1 to be set.). The current frame and all other frames directed to a halted queue are discarded. A write with a value of 1 re-enables the queue for receiving. 0 This queue is enabled for reception. (That is, it is not halted) 1 All controller receive activity to this queue is halted.
10	QHLT2	RxBD queue 2 is halted. It is a hardware-initiated stop indication. (DMACTRL[GRS] being set by the user does not cause a QHLT2 to be set.). The current frame and all other frames directed to a halted queue are discarded. A write with a value of 1 re-enables the queue for receiving. 0 This queue is enabled for reception. (That is, it is not halted) 1 All controller receive activity to this queue is halted.
11	QHLT3	RxBD queue 3 is halted. It is a hardware-initiated stop indication. (DMACTRL[GRS] being set by the user does not cause a QHLT3 to be set.). The current frame and all other frames directed to a halted queue are discarded. A write with a value of 1 re-enables the queue for receiving. 0 This queue is enabled for reception. (That is, it is not halted) 1 All controller receive activity to this queue is halted.
12	QHLT4	RxBD queue 4 is halted. It is a hardware-initiated stop indication. (DMACTRL[GRS] being set by the user does not cause a QHLT4 to be set.). The current frame and all other frames directed to a halted queue are discarded. A write with a value of 1 re-enables the queue for receiving. 0 This queue is enabled for reception. (That is, it is not halted) 1 All controller receive activity to this queue is halted.

Table 13-28. RSTAT Field Descriptions (continued)

Bits	Name	Description
13	QHLT5	RxBD queue 5 is halted. It is a hardware-initiated stop indication. (DMACTRL[GRS] being set by the user does not cause a QHLT5 to be set.). The current frame and all other frames directed to a halted queue are discarded. A write with a value of 1 re-enables the queue for receiving. 0 This queue is enabled for reception. (That is, it is not halted) 1 All controller receive activity to this queue is halted.
14	QHLT6	RxBD queue 6 is halted. It is a hardware-initiated stop indication. (DMACTRL[GRS] being set by the user does not cause a QHLT6 to be set.). The current frame and all other frames directed to a halted queue are discarded. A write with a value of 1 re-enables the queue for receiving. 0 This queue is enabled for reception. (That is, it is not halted) 1 All controller receive activity to this queue is halted.
15	QHLT7	RxBD queue 7 is halted. It is a hardware-initiated stop indication. (DMACTRL[GRS] being set by the user does not cause a QHLT7 to be set.). The current frame and all other frames directed to a halted queue are discarded. A write with a value of 1 re-enables the queue for receiving. 0 This queue is enabled for reception. (That is, it is not halted) 1 All controller receive activity to this queue is halted.
16–23	—	Reserved
24	RXF0	Receive frame event occurred on ring 0. Set by the eTSEC if IEVENT[RXF] was set in relation to receiving a frame to this ring.
25	RXF1	Receive frame event occurred on ring 1. Set by the eTSEC if IEVENT[RXF] was set in relation to receiving a frame to this ring.
26	RXF2	Receive frame event occurred on ring 2. Set by the eTSEC if IEVENT[RXF] was set in relation to receiving a frame to this ring.
27	RXF3	Receive frame event occurred on ring 3. Set by the eTSEC if IEVENT[RXF] was set in relation to receiving a frame to this ring.
28	RXF4	Receive frame event occurred on ring 4. Set by the eTSEC if IEVENT[RXF] was set in relation to receiving a frame to this ring.
29	RXF5	Receive frame event occurred on ring 5. Set by the eTSEC if IEVENT[RXF] was set in relation to receiving a frame to this ring.
30	RXF6	Receive frame event occurred on ring 6. Set by the eTSEC if IEVENT[RXF] was set in relation to receiving a frame to this ring.
31	RXF7	Receive frame event occurred on ring 7. Set by the eTSEC if IEVENT[RXF] was set in relation to receiving a frame to this ring.

13.5.3.3.3 Receive Interrupt Coalescing Register (RXIC)

The RXIC register enables and configures the operational parameters for interrupt coalescing associated with received frames. Figure 13-24 describes the RXIC register.

Offset eTSEC1:0x2_4310; eTSEC2:0x2_5310; eTSEC3:0x2_6310; eTSEC4:0x2_7310 Access: Read/Write

	0	1	2	3–10	11–15	16–31
R W	ICEN	ICCS	—	ICFT	—	ICTT
Reset	All zeros					

Figure 13-24. RXIC Register Definition

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

Table 13-29. RXIC Field Descriptions

Bits	Name	Description
0	ICEN	Interrupt coalescing enable 0 Interrupt coalescing is disabled. Interrupts are raised as they are received. 1 Interrupt coalescing is enabled. If the eTSEC receive frame interrupt is enabled (IMASK[RXFEN] is set), an interrupt is raised when the threshold number of frames is reached (defined by RXIC[ICFT]) or when the threshold timer expires (determined by RXIC[ICTT]).
1	ICCS	Interrupt coalescing timer clock source. 0 The coalescing timer advances count every 64 eTSEC Rx interface clocks (TSECn_GTX_CLK). 1 The coalescing timer advances count every 64 system clocks. This mode is recommended for FIFO operation.
2	—	Reserved
3–10	ICFT	Interrupt coalescing frame count threshold. While interrupt coalescing is enabled (RXIC[ICE] is set), this value determines how many frames are received before raising an interrupt. The eTSEC threshold counter is reset to ICFT following an interrupt. The value of ICFT must be greater than zero avoid unpredictable behavior.
11–15	—	Reserved
16–31	ICTT	Interrupt coalescing timer threshold. While interrupt coalescing is enabled (RXIC[ICE] is set), this value determines the maximum amount of time after receiving a frame before raising an interrupt. If frames have been received but the frame count threshold has not been met, an interrupt is raised when the threshold timer reaches zero. The threshold timer is reset to the value in this field and begins counting down upon receiving the first frame having its RxBD[I] bit set. The threshold value is represented in units equal to 64 periods of the clock specified by RXIC[ICCS]. ICTT must be greater than zero to avoid unpredictable behavior.

13.5.3.3.4 Receive Queue Control Register (RQUEUE)

The RQUEUE register enables each of the RxBD rings 0–7. By default, RxBD ring 0 is enabled. Figure 13-25 describes the definition for the RQUEUE register.

Offset eTSEC1:0x2_4314; eTSEC2:0x2_5314; eTSEC3:0x2_6314; eTSEC4:0x2_7314 Access: Read/Write

	0–23	24	25	26	27	28	29	30	31
R W	—	EN0	EN1	EN2	EN3	EN4	EN5	EN6	EN7
Reset	0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	1	0	0	0	0	0	0	0

Figure 13-25. RQUEUE Register Definition

Table 13-30 describes the RQUEUE register.

Table 13-30. RQUEUE Field Descriptions

Bits	Name	Description
0–23	—	Reserved
24	EN0	Receive queue 0 enable. 0 RxBD ring is not queried for reception. In effect the receive queue is disabled. 1 RxBD ring is queried for reception.
25	EN1	Receive queue 1 enable. 0 RxBD ring is not queried for reception. In effect the receive queue is disabled. 1 RxBD ring is queried for reception.
26	EN2	Receive queue 2 enable. 0 RxBD ring is not queried for reception. In effect the receive queue is disabled. 1 RxBD ring is queried for reception.
27	EN3	Receive queue 3 enable. 0 RxBD ring is not queried for reception. In effect the receive queue is disabled. 1 RxBD ring is queried for reception.
28	EN4	Receive queue 4 enable. 0 RxBD ring is not queried for reception. In effect the receive queue is disabled. 1 RxBD ring is queried for reception.
29	EN5	Receive queue 5 enable. 0 RxBD ring is not queried for reception. In effect the receive queue is disabled. 1 RxBD ring is queried for reception.
30	EN6	Receive queue 6 enable. 0 RxBD ring is not queried for reception. In effect the receive queue is disabled. 1 RxBD ring is queried for reception.
31	EN7	Receive queue 7 enable. 0 RxBD ring is not queried for reception. In effect the receive queue is disabled. 1 RxBD ring is queried for reception.

13.5.3.3.5 Receive Bit Field Extract Control Register (RBIFX)

The RBIFX register provides a set of four 6-bit offsets for locating up to four octets in a received frame and passing them to the receive queue filer as the user-defined ARB property. Through RBIFX a custom

ARB filer property can be constructed from arbitrary bytes, which allows frame filing on the basis of bitfields not ordinarily provided to the filer, such as bits from the Ethernet preamble or TCP flags. The value of property ARB is the concatenation of {B0, B1, B2, B3} to 32-bits, where B0–B3 are the bytes as defined by RBIFX.

Figure 13-26 describes the definition for the RBIFX register.Note: when the eTSEC is configured to receive frame through the FIFO packet interface, a value of B*n*CTL = 01 is not supported unless RCTRL[PRSFM]=1 and RCTRL[PRSDEP] is configured to parse L2 packets over the FIFO interface. Below is a list of arbitrary extraction requirements:

- Byte extraction level cannot exceed the parser depth: a value of B*n*CTL=10 requires RCTRL[PRSDEP]=1x and a value of B*n*CTL=11 requires RCTRL[PRSDEP]=11.
- For B*n*CTL = 01, B*n*OFFSET = 7 is not supported.
- For values of B*n*CTL=10 or B*n*CTL=11, the controller extracts the defined bytes even if it does not recognize the L3 or L4 header, respectively.
- No L4 extraction is done if a packet is an IPV4 or IPV6 fragment frame.
- If no extraction occurs due to B*n*OFFSET longer than frame data or it is an unsupported B*n*OFFSET, the B*n* extraction values are filled with zeros.

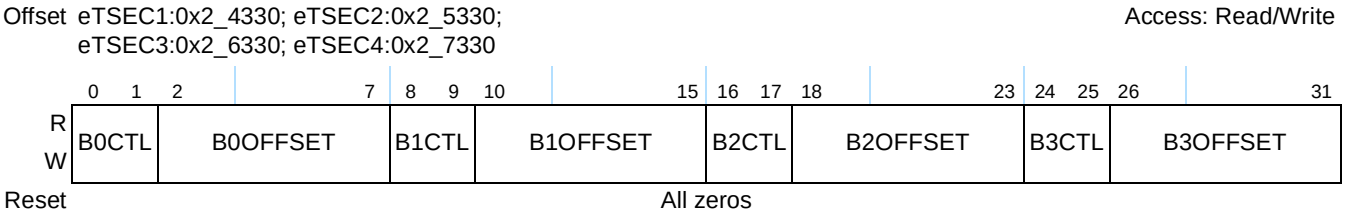


Figure 13-26. RBIFX Register Definition

Table 13-31 describes the RBIFX register.

Table 13-31. RBIFX Field Descriptions

Bits	Name	Description
0–1	B0CTL	Location of byte 0 of property ARB. 00 Byte 0 is not extracted, and appears as zero in property ARB. 01 Byte 0 is located in the received frame at offset (B0OFFSET – 8) bytes from the first byte of the Ethernet DA. In non-FIFO modes, a negative effective offset points to bytes of the standard Ethernet preamble. Values of B0OFFSET less than 8 are reserved in FIFO modes. 10 Byte 0 is located in the received frame at offset B0OFFSET bytes from the byte after the last byte of the layer 2 header. 11 Byte 0 is located in the received frame at offset B0OFFSET bytes from the byte after the last byte of the layer 3 header.
2–7	B0OFFSET	Offset relative to the header defined by B0CTL that locates byte 0 of property ARB. An effective offset of zero points to the first byte of the specified header.

Table 13-31. RBIFX Field Descriptions (continued)

Bits	Name	Description
8–9	B1CTL	Location of byte 1 of property ARB. 00 Byte 1 is not extracted, and appears as zero in property ARB. 01 Byte 1 is located in the received frame at offset (B1OFFSET – 8) bytes from the first byte of the Ethernet DA. In non-FIFO modes, a negative effective offset points to bytes of the standard Ethernet preamble. Values of B1OFFSET less than 8 are reserved in FIFO modes. 10 Byte 0 is located in the received frame at offset B1OFFSET bytes from the byte after the last byte of the layer 2 header. 11 Byte 0 is located in the received frame at offset B1OFFSET bytes from the byte after the last byte of the layer 3 header.
10–15	B1OFFSET	Offset relative to the header defined by B1CTL that locates byte 1 of property ARB. An effective offset of zero points to the first byte of the specified header.
16–17	B2CTL	Location of byte 2 of property ARB. 00 Byte 2 is not extracted, and appears as zero in property ARB. 01 Byte 2 is located in the received frame at offset (B2OFFSET – 8) bytes from the first byte of the Ethernet DA. In non-FIFO modes, a negative effective offset points to bytes of the standard Ethernet preamble. Values of B2OFFSET less than 8 are reserved in FIFO modes. 10 Byte 0 is located in the received frame at offset B2OFFSET bytes from the byte after the last byte of the layer 2 header. 11 Byte 0 is located in the received frame at offset B2OFFSET bytes from the byte after the last byte of the layer 3 header.
18–23	B2OFFSET	Offset relative to the header defined by B2CTL that locates byte 2 of property ARB. An effective offset of zero points to the first byte of the specified header.
24–25	B3CTL	Location of byte 3 of property ARB. 00 Byte 3 is not extracted, and appears as zero in property ARB. 01 Byte 3 is located in the received frame at offset (B3OFFSET – 8) bytes from the first byte of the Ethernet DA. In non-FIFO modes, a negative effective offset points to bytes of the standard Ethernet preamble. Values of B3OFFSET less than 8 are reserved in FIFO modes. 10 Byte 0 is located in the received frame at offset B3OFFSET bytes from the byte after the last byte of the layer 2 header. 11 Byte 0 is located in the received frame at offset B3OFFSET bytes from the byte after the last byte of the layer 3 header.
26–31	B3OFFSET	Offset relative to the header defined by B3CTL that locates byte 3 of property ARB. An effective offset of zero points to the first byte of the specified header.

13.5.3.3.6 Receive Queue Filer Table Address Register (RQFAR)

RQFAR, shown in Figure 13-27, contains the index of the current, indirectly accessible entry of the received queue filer table. Each table entry occupies a pair of 32-bit words, denoted RQCTRL and RQPROP. To access the RQCTRL and RQPROP words of entry *n*, write *n* to RQFAR. Then read or write the indexed RQCTRL and RQPROP words by reading or writing the RQFCR and RQFPR registers, respectively.

Offset eTSEC1:0x2_4334; eTSEC2:0x2_5334; eTSEC3:0x2_6334; eTSEC4:0x2_7334 — Access: Read/Write

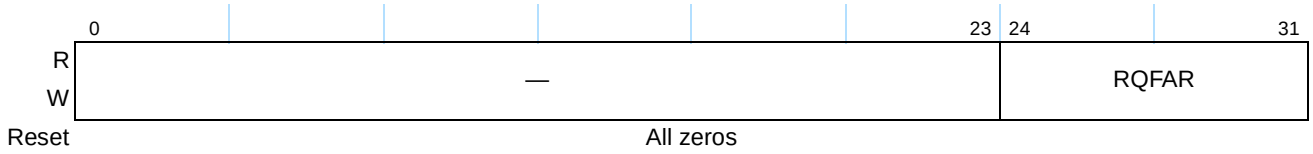


Figure 13-27. Receive Queue Filer Table Address Register Definition

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

Table 13-32. RQFAR Field Descriptions

Bits	Name	Description
0–23	—	Reserved
24–31	RQFAR	Current index of receive queue filer table, which spans a total of 256 entries.

13.5.3.3.7 Receive Queue Filer Table Control Register (RQFCR)

RQFCR is accessed to read or write the RQCTRL words in entries of the receive queue filer table. The table entries are described in greater detail in Section 13.6.5.2, "Receive Queue Filer." The word accessed through RQFCR is defined by the current value of RQFAR.

Figure 13-28 describes the definition for the RQFCR register.

Offset eTSEC1:0x2_4338; eTSEC2:0x2_5338; eTSEC3:0x2_6338; eTSEC4:0x2_7338 — Access: Read/Write

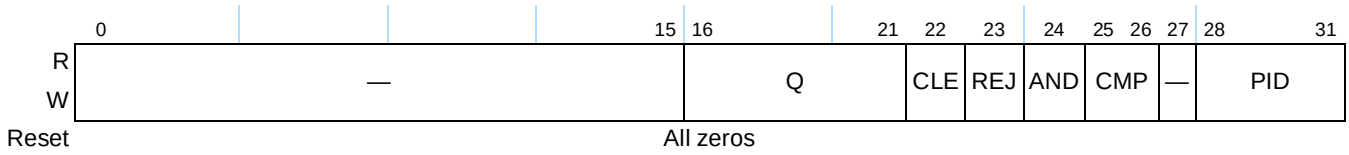


Figure 13-28. Receive Queue Filer Table Control Register Definition

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

Table 13-33. RQFCR Field Descriptions

Bit	Name	Description
0–15	—	Reserved, should be written with zero.
16–21	Q	Receive queue index, from 0 to 63, inclusive, written into the Rx frame control block associated with the received frame. When a property matches the value in the RQPROP entry at this index, and REJ = 0 and AND = 0, the frame is sent to either RxBD ring 0 (if RCTRL[FSQEN] = 1) or the RxBD ring with index (Q mod 8) and the filing table search is terminated. In the case where RCTRL[FSQEN] = 0, 8 virtual receive queues are overlaid on every RxBD ring, and software needs to consult the RQ field of the Rx frame control block to determine which virtual receive queue was chosen.
22	CLE	Cluster entry/exit (used in combination with AND bit). This bit brackets clusters, marking the start and end entries of a cluster. Clusters cannot be nested. 0 Regular RQCTRL entry. 1 If entry matches and AND = 1, treat subsequent entries as belonging to a nested cluster and enter the cluster; otherwise skip all entries up to and including the next cluster exit. If AND = 0, exit current cluster.

Table 13-33. RQFCR Field Descriptions (continued)

Bit	Name	Description
23	REJ	Reject frame. This bit and its specified action are ignored if AND = 1. 0 If entry matches, accept frame and file it to RxBD ring Q. 1 If entry matches, reject frame and discard it, ignoring Q.
24	AND	Match this entry and the next entry as a pair. 0 Match property[PID] against RQPROP, independent of the next entry. 1 Match property[PID] against RQPROP. If matched and CLE = 0, attempt to match next entry, otherwise, skip all entries up to and including the entry with AND = 0. If matched and CLE = 1, enter cluster of entries, otherwise, skip all entries up to and including the entry with CLE = 1 (cluster exit).
25–26	CMP	Comparison operation to perform on the RQPROP entry at this index when PID > 0. The property value extracted by the frame parser is masked by the 32-bit *mask_register* prior to comparison against RQPROP. However, the property value is not permanently altered by the value in *mask_register*. By default, *mask_register* is initialized to 0xFFFF_FFFF before each frame is processed. In the case where PID = 0, CMP is interpreted as follows: 00/01 Filer *mask_register* is set to all 32 bits of RQPROP, and this entry always *matches*. 10/11 Filer *mask_register* is set to all 32 bits of RQPROP, and this entry always *fails to match*. In the case where PID > 0, CMP is interpreted as follows (& is bit-wise AND operator): 00 *property*[PID] & *mask_register* = RQPROP 01 *property*[PID] & *mask_register* >= RQPROP 10 *property*[PID] & *mask_register* != RQPROP 11 *property*[PID] & *mask_register* < RQPROP
27	—	Reserved, should be written with zero.
28–31	PID	Property identifier. The value in the RQPROP entry at this index is interpreted according to PID (see Table 13-34).

13.5.3.3.8 Receive Queue Filer Table Property Register (RQFPR)

RQFPR (see Figure 13-29) is accessed to read or write the RQPROP words in entries of the receive queue filer table. The table entries are described in greater detail in Section 13.6.5.2, “Receive Queue Filer.” The word accessed through RQFPR is defined by the current value of RQFAR. Figure 13-29 and Figure 13-30 describe the fields of the RQFPR register according to property ID.

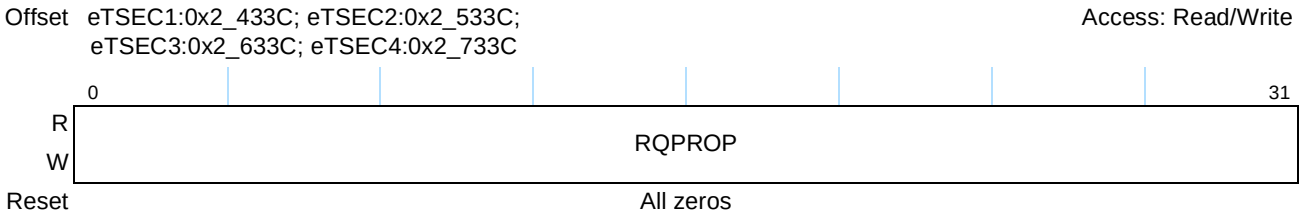


Figure 13-29. Receive Queue Filer Table Property IDs 0, 2–15 Register Definition

Offset eTSEC1:0x2_433C; eTSEC2:0x2_533C; eTSEC3:0x2_633C; eTSEC4:0x2_733C — Access: Read/Write

	0															15
R W	—															
Reset	All zeros															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	EBC	VLN	CFI	JUM	IPF	FIF	IP4	IP6	ICC	ICV	TCP	UDP	—		PER	EER
Reset	All zeros															

Figure 13-30. Receive Queue Filer Table Property ID1 Register Definition

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

Table 13-34. RQFPR Field Descriptions

PID[1]	Bit	Name	Description
0000	0–31	MASK	Mask bits to be written to Filer *mask_register* for masking of property values. The rule match/fail status for this PID is determined by RQCTRL[CMP]. Since *mask_register* is bit-wise ANDed with properties, every bit of MASK that is cleared also results in the corresponding property bit being cleared in comparisons. Therefore setting MASK to 0xFFFF_FFFF ensures that all property bits participate in rule matches.
0001	0–15	—	Reserved
	16	EBC	Set if the destination Ethernet address is to the broadcast address.
	17	VLN	Set if a VLAN tag (Ethertype DFVLAN[TAG] or 0x8100) was seen in the frame.
	18	CFI	Set to the value of the Canonical Format Indicator in the VLAN control tag if VLAN is set, zero otherwise.
	19	JUM	Set if a jumbo Ethernet frame was parsed.
	20	IPF	Set if a fragmented IPv4 or IPv6 header was encountered. See the descriptions of receive FCB fields IP and PRO in Section 13.6.4.3, “Receive Path Off-Load,” for more information on determining the status of received packets for which IPF is set.
	21	FIF	Set if the packet entered on eTSEC’s FIFO interface.
	22	IP4	Set if an IPv4 header was parsed.
	23	IP6	Set if an IPv6 header was parsed.
	24	ICC	Set if the IPv4 header checksum was checked.
	25	ICV	Set if the IPv4 header checksum was verified correct.
	26	TCP	Set if a TCP header was parsed.
	27	UDP	Set if a UDP header was parsed.
	28-29	—	Reserved.
	30	PER	Set on a parse error, such as header inconsistency.
	31	EER	Set on an Ethernet framing error that prevents parsing.

Table 13-34. RQFPR Field Descriptions (continued)

PID[1]	Bit	Name	Description
0010	0–7	ARB	User-defined arbitrary bit field property: byte 0 extracted. Defaults to 0x00.
	8–15		User-defined arbitrary bit field property: byte 1 extracted. Defaults to 0x00.
	16–23		User-defined arbitrary bit field property: byte 2 extracted. Defaults to 0x00.
	24–31		User-defined arbitrary bit field property: byte 3 extracted. Defaults to 0x00.
0011	0–7	—	Reserved, should be written with zero.
	8–31	DAH	Destination MAC address, most significant 24 bits. Defaults to 0x000000.
0100	0–7	—	Reserved, should be written with zero.
	8–31	DAL	Destination MAC address, least significant 24 bits. Defaults to 0x000000.
0101	0–7	—	Reserved, should be written with zero.
	8–31	SAH	Source MAC address, most significant 24 bits. Defaults to 0x000000.
0110	0–7	—	Reserved, should be written with zero.
	8–31	SAL	Source MAC address, least significant 24 bits. Defaults to 0x000000.
0111	0–15	—	Reserved, should be written with zero.
	16–31	ETY	Ethertype of next layer protocol, that is, last ethertype if layer 2 headers nest. Defaults to 0xFFFF. Using the filer to match ETY does not work in the case of PPPoE packets, because the PPPoE ethertype in the original packet, 0x8864, is always overwritten with the PPP protocol field. Thus, matches on ETY == 0x8864 always fail. Instead, software should use PID=1 fields IP4 (ETY = 0x0021) and IP6 (ETY = 0x0057) to distinguish PPPoE session packets carrying IPv4 and IPv6 datagrams. Other PPP protocols are encoded in the ETY field, but many of them overlap with real ethertype definitions. Consult IANA and IEEE for possible ambiguities. Packets with a value in the length/type field greater than 1500 and less than 1536 are treated as payload length. If the eTSEC is used in a network where there are packets carrying a type designation between 1500 and 1536 (note there are none currently publicly defined by IANA), then the S/W must confirm the parser and filer results by checking the type/length field after the packet has been written to memory to see if it falls in this range. Note that the eTSEC filer gets multiple packet attributes as a result of parsing the packet. The behavior of the eTSEC is that it pulls the innermost ethertype found in the packet; this means that in many supported protocols, it is impossible to create a filer rule that matches on the outer ethertype. There are four cases that need to be highlighted. 1. The jumbo ethertype (0x8870)—In this case, the eTSEC assumes that the following header is LLC/SNAP. LLC/SNAP has an associated Ethertype, and the ETY field is populated with that ethertype. This makes it impossible to file on jumbo frames. In this case, one can use arbitrary extracted bytes to pull the outermost Ethertype. 2. The PPPoE ethertype described above. 3. The VLAN tag ethertype (0x8100)—In this case, one can use the PID1 VLN bit to indicate that the packet had a VLAN tag. 4. The MPLS tagged packets. In this case, one can use arbitrary extraction bytes to compare to the actual ethertype if a filer rule is intending to file based on an MPLS label existence.
1000	0–19	—	Reserved, should be written with zero.
	20–31	VID	VLAN network identifier (as per IEEE Std 802.1Q). This value defaults to 0x000 if no VLAN tag was found, or the VLAN tag contained only priority information.

Table 13-34. RQFPR Field Descriptions (continued)

PID[1]	Bit	Name	Description
1001	0–28	—	Reserved, should be written with zero.
	29–31	PRI	VLAN user priority (as per IEEE Std 802.1p). This value defaults to 000 (best effort priority) if no VLAN tag was found.
1010	0–23	—	Reserved, should be written with zero.
	24–31	TOS	IPv4 header Type Of Service field or IPv6 Traffic Class field. This value defaults to 0x00 (default RFC 2474 best-effort behavior) if no IP header appeared. Note that for IPv6 the Traffic Class field is extracted using the IP header definition in RFC 2460. IPv6 headers formed using the earlier RFC 1883 have a different format and must be handled with software.
1011	0–23	—	Reserved, should be written with zero.
	24–31	L4P	Layer 4 protocol identifier as per published IANA specification. This is the last recognized protocol type recognized in the case of IPv6 extension headers. This value defaults to 0xFF to indicate that no layer 4 header was recognized (possibly due to absence of an IP header).
1100	0–31	DIA	Destination IP address. If an IPv4 header was found, this is the entire destination address. If an IPv6 header was found, this is the 32 most significant bits of the 128-bit destination address. This value defaults to 0x0000_0000 if no IP header appeared.
1101	0–31	SIA	Source IP address. If an IPv4 header was found, this is the entire source address. If an IPv6 header was found, this is the 32 most significant bits of the 128-bit source address. This value defaults to 0x0000_0000 if no IP header appeared.
1110	0–15	—	Reserved, should be written with zero.
	16–31	DPT	Destination port number for TCP or UDP headers. This value defaults to 0x0000 if no TCP or UDP headers were recognized.
1111	0–15	—	Reserved, should be written with zero.
	16–31	SPT	Source port number for TCP or UDP headers. This value defaults to 0x0000 if no TCP or UDP headers were recognized.

[1] PID is the property identifier field of the filer table control entry (see RQFCR[PID]) at the same index.

13.5.3.3.9 Maximum Receive Buffer Length Register (MRBLR)

The MRBLR register is written by the user. It informs the eTSEC how much space is in the receive buffer pointed to by the RxBD. Figure 13-31 describes the definition for the MRBLR.

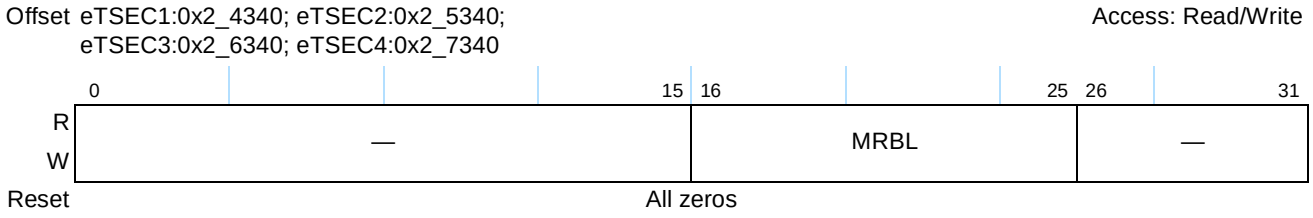


Figure 13-31. MRBLR Register Definition

Table 13-35. MRBLR Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–25	MRBL	Maximum receive buffer length. MRBL is the number of bytes that the eTSEC receiver writes to the receive buffer. The MRBL register is written by the user with a multiple of 64 for all modes. The eTSEC can write fewer bytes to the buffer than the value set in MRBL if a condition such as an error or end-of-frame occurs, but it never exceeds the MRBL value; therefore, user-supplied buffers must be at least as large as the MRBL. MRBL must be set, together with the number of buffer descriptors, to ensure adequate space for received frames. See Section 13.5.3.5.5, “Maximum Frame Length Register (MAXFRM),” for further discussion.
26–31	—	To ensure that MRBL is a multiple of 64, these bits are reserved and should be cleared.

13.5.3.3.10 Receive Data Buffer Pointer High Register (RBDBPH)

The RBDBPH register is written by the user with the most significant address bits common to all RxBD buffer addresses, RxBD[Data Buffer Pointer]. As a consequence, Rx buffers must be placed in a 4 Gbyte segment of memory whose base address is prefixed by the bits in RBDBPH. The RxBD ring itself can reside in a different memory region (based at RBASEH). Figure 13-32 describes the definition for the RBDBPH register.

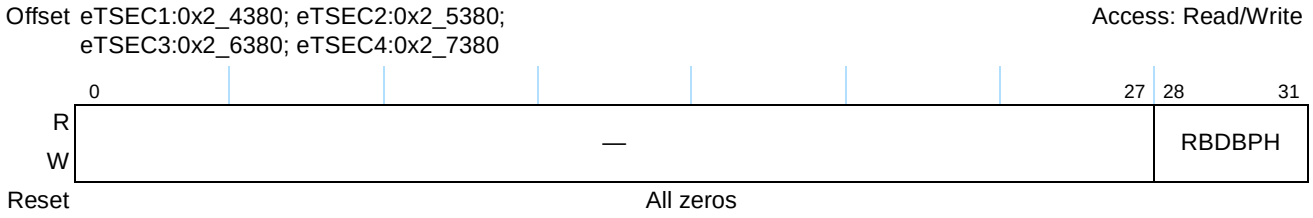


Figure 13-32. RBDBPH Register Definition

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

Table 13-36. RBDBPH Field Descriptions

Bits	Name	Description
0–27	—	Reserved
28–31	RBDBPH	Most significant bits common to all data buffer addresses contained in RxBDs. The user must initialize RBDBPH before enabling the eTSEC receive function.

13.5.3.3.11 Receive Buffer Descriptor Pointers 0–7 (RBPTR0–RBPTR7)

RBPTR0–RBPTR7 each contains the low-order 32 bits of the next receive buffer descriptor address for their respective RxBD ring. Figure 13-33 describes the RBPTR registers. These registers takes on the value of their ring’s associated RBASE when the RBASE register is written by software. Software must not write RBPTR*n* while eTSEC is actively receiving frames. However, RBPTR*n* can be modified when the receiver is disabled or when no Rx buffer is in use (after a GRACEFUL STOP RECEIVE command is issued and the frame completes its reception) in order to change the next RxBD eTSEC receives.

Offset eTSEC1:0x2_4384+8×*n*; eTSEC2:0x2_5384+8×*n*; eTSEC3:0x2_6384+8×*n*; eTSEC4:0x2_7384+8×*n* Access: Read/Write

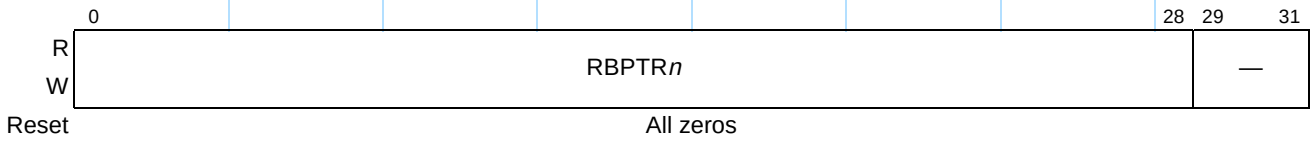


Figure 13-33. RBPTR0–RBPTR7 Register Definition

Table 13-24 describes the fields of the RBPTR*n* register.

Table 13-37. RBPTR*n* Field Descriptions

Bits	Name	Description
0–28	RBPTR*n*	Current RxBD pointer for RxBD ring *n*. Points to the current BD being processed or to the next BD the receiver uses when it is idling. After reset or when the end of the RxBD ring is reached, eTSEC initializes RBPTR*n* to the value in the corresponding RBASE*n*. The RBPTR register is internally written by the eTSEC's DMA controller during reception. The pointer increments by 8 (bytes) each time a descriptor is closed successfully by the eTSEC. Note that the 3 least-significant bits of this register are read only and zero.
29–31	—	Reserved

13.5.3.3.12 Receive Descriptor Base Address High Register (RBASEH)

The RBASEH register is written by the user with the most significant address bits common to all RxBD addresses, including RBASE0–RBASE7 and RBPTR0–RBPTR7. As a consequence, RxBD rings must be placed in a 4 Gbyte segment of memory whose base address is prefixed by the bits in RBASEH. However, Rx data buffers may potentially reside in a different memory region based at RBDBPH. Figure 13-34 describes the definition for the RBASEH register.

Offset eTSEC1:0x2_4400; eTSEC2:0x2_5400; eTSEC3:0x2_6400; eTSEC4:0x2_7400 Access: Read/Write

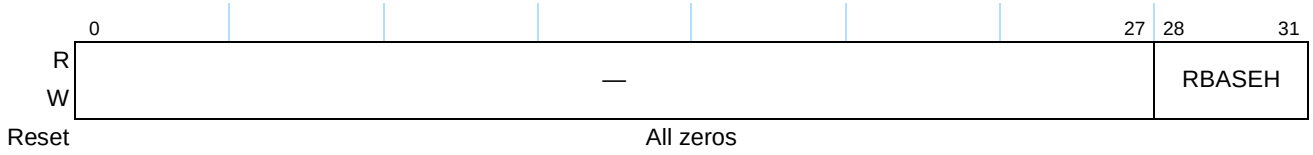


Figure 13-34. RBASEH Register Definition

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

Table 13-38. RBASEH Field Descriptions

Bits	Name	Description
0–27	—	Reserved
28–31	RBASEH	Most significant bits common to all RxBD addresses—except data buffer pointers. The user must initialize RBASEH before enabling the eTSEC receive function.

13.5.3.3.13 Receive Descriptor Base Address Registers (RBASE0–RBASE7)

The RBASE*n* registers are written by the user with the base address of each RxBD ring *n*. Each such value must be divisible by eight, since the 3 least-significant bits always write as 000. Figure 13-35 describes the RBASE*n* registers.

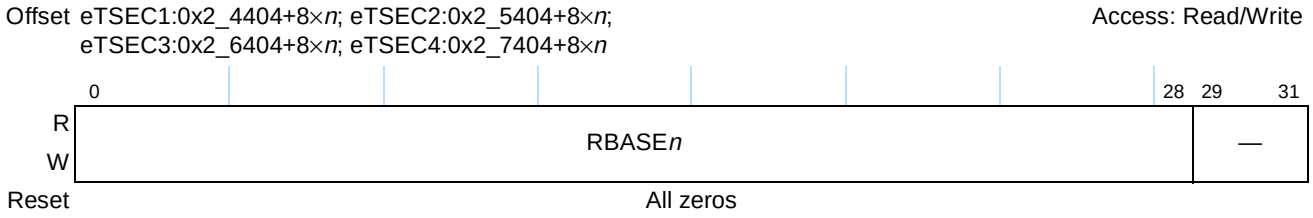


Figure 13-35. RBASE Register Definition

Table 13-26 describes the fields of the RBASE*n* registers.

Table 13-39. RBASE0–RBASE7 Field Descriptions

Bits	Name	Description
0–28	RBASE*n*	Receive base for ring *n*. RBASE defines the starting location in the memory map for the eTSEC RxBDs. This field must be 8-byte aligned. Together with setting the W (wrap) bit in the last BD, the user can select how many BDs to allocate for the receive packets. The user must initialize RBASE before enabling the eTSEC receive function on the associated ring.
29–31	—	Reserved

13.5.3.4 MAC Functionality

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

13.5.3.4.1 Configuring the MAC

The MAC configuration registers 1 and 2 provide for configuring the MAC in multiple ways:

- Adjusting the preamble length—The length of the preamble can be adjusted from the nominal seven bytes to some other (non-zero) value. Should custom preamble insertion/extraction be configured, then this register must by left at its default value.
- Varying pad/CRC combinations—Three different pad/CRC combinations are provided to handle a variety of system requirements. Simplest are frames that already have a valid frame check sequence (FCS) field. The other two options include appending a valid CRC or padding and then appending a valid CRC, resulting in a minimum frame of 64 octets. In addition to the programmable register set, the pad/CRC behavior can be dynamically adjusted on a per-packet basis.

13.5.3.4.2 Controlling CSMA/CD

The half-duplex register (HAFDUP) allows control over the carrier-sense multiple access/collision detection (CSMA/CD) logic of the eTSEC. Half-duplex mode is only supported for 10- and 100-Mbps

operation. Following the completion of the packet transmission the part begins timing the inter packet gap (IPG) as programmed in the back-to-back IPG configuration register. The system is now free to begin another frame transfer.

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

13.5.3.4.3 Handling Packet Collisions

While transmitting a packet in half-duplex mode, the eTSEC is sensitive to COL. If a collision occurs, it aborts the packet and outputs the 32-bit jam sequence. The jam sequence is comprised of several bits of the CRC, inverted to guarantee an invalid CRC upon reception. A signal is sent to the system indicating that a collision occurred and that the start of the frame is needed for retransmission. The eTSEC then backs off of the medium for a time determined by the truncated binary exponential back off (BEB) algorithm. Following this back-off time, the packet is retried. The back-off time can be skipped if configured through the half-duplex register. However, this is non-standard behavior and its use must be carefully applied. Should any one packet experience excessive collisions, the packet is aborted. The system should flush the frame and move to the next one in line. If the system requests to send a packet while the eTSEC is deferring to a carrier, the eTSEC simply waits until the end of the carrier event and the timing of IPG before it honors the request.

If packet transmission attempts experience collisions, the eTSEC outputs the jam sequence and waits some amount of time before retrying the packet. This amount of time is determined by a controlled randomization process called truncated binary exponential back-off. The amount of time is an integer number of slot times. The number of slot times to delay before the *n*th retransmission attempt is chosen as a uniformly-distributed random integer *r* in the range:

$$0 \le r \le 2^k, \text{ where } k = \min(n,10).$$

So after the first collision, the eTSEC backs off either 0 or 1 slot times. After the fifth collision, the eTSEC backs off between 0 and 32 slot times. After the tenth collision, the maximum number of slot times to back off is 1024. This can be adjusted through the half-duplex register. An alternate truncation point, such as 7 for instance, can be programmed. On average, the MAC is more aggressive after seven collisions than other stations on the network.

13.5.3.4.4 Controlling Packet Flow

Packet flow can be dealt with in a number of ways within eTSEC. A default retransmit attempt limit of 15 can be reduced using the half-duplex register. The slot time or collision window can be used to gate the retry window and possibly reduce the amount of transmit buffering within the system. The slot time for 10/100 Mbps is 512 bit times. Because the slot time begins at the beginning of the packet (including preamble), the end occurs around the 56th byte of the frame data. Slot time in 1000-Mbps mode is not supported.

Full-duplex flow control is provided for in IEEE 802.3x. Currently the standard does not address flow control in half-duplex environments. Common in the industry, however, is the concept of back pressure. The eTSEC implements the optional back pressure mechanism using the raise carrier method. If the system receive logic wishes to stop the reception of packets in a network-friendly way, transmit half-duplex flow control (THDF) is set (TCTRL[THDF]). If the medium is idle, the eTSEC raises carrier by transmitting preamble. Other stations on the half-duplex network then defer to the carrier.

In the event the preamble transmission happens to cause a collision, the eTSEC ensures the minimum 96-bit presence on the wire, then drops preamble and waits a back-off time depending on the value of the back-pressure-no-back-off configuration bit HAFDUP[BP No BackOff]. These transmitting-preamble-for-back pressure collisions are not counted. If HAFDUP[BP No BackOff] is set, the eTSEC waits an inter-packet gap before resuming the transmission of preamble following the collision and does not defer. If HAFDUP[BP No BackOff] is cleared, the eTSEC adheres to the truncated BEB algorithm that allows the possibility of packets being received. This also can be detrimental in that packets can now experience excessive collisions, causing them to be dropped in the stations from which they originate. To reduce the likelihood of lost packets and packets leaking through the back pressure mechanism, HAFDUP[BP No BackOff] must be set.

The eTSEC drops carrier (cease transmitting preamble) periodically to avoid excessive defer conditions in other stations on the shared network. If, while applying back pressure, the eTSEC is requested to send a packet, it stops sending preamble, and waits one IPG before sending the packet. HAFDUP[BP No BackOff] applies for any collision that occurs during the sending of this packet. Collisions for packets while half duplex back pressure is asserted are counted. The eTSEC does not defer while attempting to send packets while in back pressure. Again, back pressure is non-standard, yet it can be effective in reducing the flow of receive packets.

13.5.3.4.5 Controlling PHY Links

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

The eTSEC MII’s registers provide the ability to perform continuous read cycles (called a scan cycle); although, scan cycles are not explicitly defined in the standard. If requested (by setting MIIMCOM[Scan Cycle]), the part performs repetitive read cycles of the PHY status register, for example. In this way, link characteristics may be monitored more efficiently. The different fields in the MII management indicator register (scan, not valid and busy) are used to indicate availability of each read of the scan cycle to the host from MIIMSTAT[PHY scan].

Yet another parameter that can be modified through the MII registers is the length of the MII management interface preamble. After establishing that a PHY supports preamble suppression, the host may so configure the eTSEC. While enabled, the length of MII management frames are reduced from 64 clocks to 32 clocks. This effectively doubles the efficiency of the interface.

13.5.3.5 MAC Registers

This section describes the MAC registers.

13.5.3.5.1 MAC Configuration 1 Register (MACCFG1)

MACCFG1 is written by the user. Figure 13-36 describes the definition for the MACCFG1 register.

Offset eTSEC1:0x2_4500; eTSEC2:0x2_5500; eTSEC3:0x2_6500; eTSEC4:0x2_7500 Access: Mixed

	0	1	11	12	13	14	15
R W	Soft_Reset	—		Reset Rx MC	Reset Tx MC	Reset Rx Fun	Reset Tx Fun
Reset	All zeros						

	16	22	23	24	25	26	27	28	29	30	31
R W	—		Loop Back	—		Rx_Flow	Tx_Flow	Sync'd Rx EN	Rx_EN	Sync'd Tx EN	Tx_EN
Reset	All zeros										

Figure 13-36. MACCFG1 Register Definition

Table 13-40 describes the fields of the MACCFG1 register.

Table 13-40. MACCFG1 Field Descriptions

Bits	Name	Description
0	Soft_Reset	Soft reset. This bit is cleared by default. See Section 13.6.3.2, "Soft Reset and Reconfiguring Procedure," for more information on setting this bit. 0 Normal operation. 1 Place the entire MAC in reset except for the host interface.
1–11	—	Reserved
12	Reset Rx MC	Reset receive MAC control block. This bit is cleared by default. 0 Normal operation. 1 Place the receive part of the MAC in reset. This block detects control frames and contains the pause timers.
13	Reset Tx MC	Reset transmit MAC control block. This bit is cleared by default. 0 Normal operation. 1 Place the transmit part of the MAC in reset. This block multiplexes data and control frame transfers. It also responds to XOFF PAUSE control frames.
14	Reset Rx Fun	Reset receive function block. This bit is cleared by default. 0 Normal operation. 1 Place the receive function in reset. This block performs the receive frame protocol.
15	Reset Tx Fun	Reset transmit function block. This bit is cleared by default. 0 Normal operation. 1 Place the transmit function in reset. This block performs the frame transmission protocol.
16–22	—	Reserved
23	Loop Back	Loop back. This bit is cleared by default. 0 Normal operation. 1 Loop back the MAC transmit outputs to the MAC receive inputs.

Table 13-40. MACCFG1 Field Descriptions (continued)

Bits	Name	Description
24–25	—	Reserved
26	Rx_Flow	Receive flow. This bit is cleared by default. 0 The receive MAC control ignores PAUSE flow control frames. 1 The receive MAC control detects and acts on PAUSE flow control frames.
27	Tx_Flow	Transmit flow. This bit is cleared by default. 0 The transmit MAC control may not send PAUSE flow control frames if requested by the system. 1 The transmit MAC control may send PAUSE flow control frames if requested by the system.
28	Sync'd Rx EN	Receive enable synchronized to the receive stream. (Read-only) 0 Frame reception is not enabled. 1 Frame reception is enabled.
29	Rx_EN	Receive enable. This bit is cleared by default. If set, prior to clearing this bit, set DMACTRL[GRS] then confirm subsequent occurrence of the graceful receive stop interrupt (IEVENT[GRSC] is set). 0 The MAC may not receive frames from the PHY. 1 The MAC may receive frames from the PHY.
30	Sync'd Tx EN	Transmit enable synchronized to the transmit stream. (Read-only) 0 Frame transmission is not enabled. 1 Frame transmission is enabled.
31	Tx_EN	Transmit enable. This bit is cleared by default. If set, prior to clearing this bit, set DMACTRL[GTS] then confirm subsequent occurrence of the graceful receive stop interrupt (IEVENT[GTSC] is set). 0 The MAC may not transmit frames from the system. 1 The MAC may transmit frames from the system.

13.5.3.5.2 MAC Configuration 2 Register (MACCFG2)

The MACCFG2 register is written by the user. Figure 13-37 describes the definition for the MACCFG2 register.

Offset eTSEC1:0x2_4504; eTSEC2:0x2_5504; eTSEC3:0x2_6504; eTSEC4:0x2_7504 — Access: Read/Write

	0–15
R W	—
Reset	All zeros

	16–19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Preamble Length	—		I/F Mode		PreAmRxEN	PreAmTxEN	Huge Frame	Length check	MPEN	PAD/CRC	CRC EN	Full Duplex
Reset	0 1 1 1	0	0	0	0	0	0	0	0	0	0	0	0

Figure 13-37. MACCFG2 Register Definition

Table 13-41 describes the fields of the MACCFG2 register.

Table 13-41. MACCFG2 Field Descriptions

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>0–15</td><td>—</td><td>Reserved</td></tr>
<tr><td>16–19</td><td>Preamble Length</td><td>This field determines the length in bytes of the preamble field preceding each Ethernet start-of-frame delimiter byte. Values from 0x3 to 0xF are supported by the controller. The default value of 0x7 should not be altered in order to guarantee reliable operation with IEEE 802.3 compliant hardware.</td></tr>
<tr><td>20–21</td><td>—</td><td>Reserved</td></tr>
<tr><td>22–23</td><td>I/F Mode</td><td>This field determines the type of interface to which the MAC is connected. Its default is 00.
00 Reserved bit mode (not supported) (10 Mbps GENDEC/GPSI)
01 Nibble mode (MII) (10/100 Mbps MII/RMII)
10 Byte mode (GMII/TBI) (1000 MbpsGMII/TBI). Reserved if neither GMII or TBI are supported.
11 Reserved</td></tr>
<tr><td>24</td><td>PreAM RxEN</td><td>User defined preamble enable for received frames. This bit is cleared by default.
0 The MAC skips the Ethernet preamble without returning it.
1 The MAC recovers the received Ethernet preamble and passes it to the driver at the start of each received frame. If the preamble is less than 7 bytes, 0’s are prepended to pad it to 7 bytes. Not applicable to FIFO or RMII 10/100 modes.</td></tr>
<tr><td>25</td><td>PreAM TxEN</td><td>User defined preamble enable for transmitted frames. This bit is cleared by default.
0 The MAC generates a standard Ethernet preamble.
1 If a user-defined preamble has been passed to the MAC it is transmitted instead of the standard preamble. Otherwise the standard Ethernet preamble is generated. The Preamble Length field should be left at its default setting if a user-defined preamble is transmitted. Not applicable to FIFO or RMII 10/100 modes.</td></tr>
<tr><td>26</td><td>Huge Frame</td><td>Huge frame enable. This bit is cleared by default.
0 Limit the length of frames received to less than or equal to the maximum frame length value (MAXFRM[Maximum Frame]) and limit the length of frames transmitted to less than the maximum frame length.
See Section 13.6.7, “Buffer Descriptors,” for further details of buffer descriptor bit updating.
<table>
<tr><th>Frame type</th><th>Frame length</th><th>Packet truncation</th><th>Buffer descriptor updated</th></tr>
<tr><td>Receive or transmit</td><td>> maximum frame length</td><td>yes</td><td>yes</td></tr>
<tr><td>Receive</td><td>= maximum frame length</td><td>no</td><td>yes</td></tr>
<tr><td>Transmit</td><td>= maximum frame length</td><td>no</td><td>no</td></tr>
<tr><td>Receive or transmit</td><td>< maximum frame length</td><td>no</td><td>no</td></tr>
</table>
1 Frames are transmitted and received regardless of their relationship to the maximum frame length.
Note that if Huge Frame is cleared, the user must ensure that adequate buffer space is allocated for received frames. See Section 13.5.3.5.5, “Maximum Frame Length Register (MAXFRM),” for further information.</td></tr>
<tr><td>27</td><td>Length check</td><td>Length check. This bit is cleared by default.
0 No length field checking is performed.
1 The MAC checks the frame’s length field on receive to ensure it matches the actual data field length. Transmitted frames are not checked.</td></tr>
</table>

Table 13-41. MACCFG2 Field Descriptions (continued)

Bits	Name	Description
28	MPEN	Magic packet enable for Ethernet modes. This bit is cleared by default. MPEN should be enabled only after GRACEFUL RECEIVE STOP and GRACEFUL TRANSMIT STOP are completed successfully (in other words, transmission and reception have stopped). 0 Normal receive behavior on receive, or Magic Packet mode has exited with reception of a valid Magic Packet. 1 Commence Magic Packet detection by the MAC provided that frame reception is enabled in MACCFG1. In this mode the MAC ignores all received frames until the specific Magic Packet frame is received, at which point this bit is cleared by the eTSEC, and a maskable interrupt through IEVENT[MAG] occurs.
29	PAD/CRC	Pad and append CRC. This bit is cleared by default.This bit must be set when in half-duplex mode (MACCFG2[Full Duplex] is cleared). 0 Frames presented to the MAC have a valid length and contain a CRC. 1 The MAC pads all transmitted short frames and appends a CRC to every frame regardless of padding requirement.
30	CRC EN	CRC enable. If the configuration bit PAD/CRC ENABLE or the per-packet PAD/CRC ENABLE is set, CRC ENABLE is ignored. This bit is cleared by default. 0 Frames presented to the MAC have a valid length and contain a valid CRC. 1 The MAC appends a CRC on all frames. Clear this bit if frames presented to the MAC have a valid length and contain a valid CRC.
31	Full Duplex	Full duplex configure. This bit is cleared by default. 0 The MAC operates in half-duplex mode only. 1 The MAC operates in full-duplex mode.

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

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

Offset eTSEC1:0x2_4508; eTSEC2:0x2_5508; eTSEC3:0x2_6508; eTSEC4:0x2_7508 — Access: Read/Write

	0	1–7	8	9–15	16–23	24	25–31
R W	—	Non-Back-to-Back Inter-Packet-Gap, Part 1	—	Non-Back-to-Back Inter-Packet-Gap, Part 2	Minimum IFG Enforcement	—	Back-to-Back Inter-Packet-Gap
Reset	0	1 0 0 0 0 0 0	0	1 1 0 0 0 0 0	0 1 0 1 0 0 0 0	0	1 1 0 0 0 0 0

Figure 13-38. IPGIFG Register Definition

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

Table 13-42. IPGIFG Field Descriptions

Bits	Name	Description
0	—	Reserved
1–7	Non-Back-to-Back Inter-Packet-Gap, Part 1	This is a programmable field representing the optional carrier sense window referenced in IEEE 802.3/4.2.3.2.1 ‘carrier deference’. If carrier is detected during the timing of IPGR1, the MAC defers to carrier. If, however, carrier becomes active after IPGR1, the MAC continues timing IPGR2 and transmits, knowingly causing a collision, thus ensuring fair access to medium. Its range of values is 0x00 to IPGR2. Its default is 0x40 (64d) which follows the two-thirds/one-third guideline.

Table 13-42. IPGIFG Field Descriptions (continued)

Bits	Name	Description
8	—	Reserved
9–15	Non-Back-to-Back Inter-Packet-Gap, Part 2	This is a programmable field representing the non-back-to-back inter-packet-gap in bits. Its default is 0x60 (96d), which represents the minimum IPG of 96 bits.
16–23	Minimum IFG Enforcement	This is a programmable field representing the minimum number of bits of IFG to enforce between frames. A frame is dropped whose IFG is less than that programmed. The default setting of 0x50 (80d) represents half of the nominal minimum IFG which is 160 bits.
24	—	Reserved
25–31	Back-to-Back Inter-Packet-Gap	This is a programmable field representing the IPG between back-to-back packets. This is the IPG parameter used exclusively in full-duplex mode and in half-duplex mode if two transmit packets are sent back-to-back. Set this field to the number of bits of IPG desired. The default setting of 0x60 (96d) represents the minimum IPG of 96 bits.

13.5.3.5.4 Half-Duplex Register (HAFDUP)

The HAFDUP register is written by the user. Figure 13-39 describes the HAFDUP register.

Offset eTSEC1:0x2_450C; eTSEC2:0x2_550C; eTSEC3:0x2_650C; eTSEC4:0x2_750C — Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	—								Alternate BEB Truncation				Alt BEB	BP No BackOff	No BackOff	Excess Defer
Reset	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Retransmission Maximum				—				Collision Window							
Reset	1	1	1	1	0	0	0	0	0	0	1	1	0	1	1	1

Figure 13-39. Half-Duplex Register Definition

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

Table 13-43. HAFDUP Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–11	Alternate BEB Truncation	This field is used while ALTERNATE BINARY EXPONENTIAL BACKOFF ENABLE is set. The value programmed is substituted for the Ethernet standard value of ten. Its default is 0xA.
12	Alt BEB	Alternate binary exponential backoff. This bit is cleared by default. 0 The Tx MAC follows the standard binary exponential back off rule. 1 The Tx MAC uses the ALTERNATE BINARY EXPONENTIAL BACKOFF TRUNCATION setting instead of the 802.3 standard tenth collision. The standard specifies that any collision after the tenth uses one less than 210 as the maximum backoff time.
13	BP No BackOff	Back pressure no backoff. This bit is cleared by default. 0 The Tx MAC follows the binary exponential back off rule. 1 The Tx MAC immediately re-transmits, following a collision, during back pressure operation.

Table 13-43. HAFDUP Field Descriptions (continued)

Bits	Name	Description
14	No BackOff	No backoff. This bit is cleared by default. 0 The Tx MAC follows the binary exponential back off rule. 1 The Tx MAC immediately re-transmits following a collision.
15	Excess Defer	Excessively deferred. This bit is set by default. 0 The Tx MAC aborts the transmission of a packet that is excessively deferred. 1 The Tx MAC allows the transmission of a packet that is excessively deferred.
16–19	Retransmission Maximum	This is a programmable field specifying the number of retransmission attempts following a collision before aborting the packet due to excessive collisions. The standard specifies the attempt limit to be 0xF (15d). Its default value is 0xF.
20–21	—	Reserved
22–31	Collision Window	This is a programmable field representing the slot time or collision window during which collisions occur in properly configured networks. Because the collision window starts at the beginning of transmission, the preamble and SFD are included. Its default of 0x37 (55d) corresponds to the count of frame bytes at the end of the window.

13.5.3.5.5 Maximum Frame Length Register (MAXFRM)

The MAXFRM register is written by the user. Figure 13-40 shows the MAXFRM register.

Offset eTSEC1:0x2_4510; eTSEC2:0x2_5510; eTSEC3:0x2_6510; eTSEC4:0x2_7510 — Access: Read/Write

	0															15	16															31
R W	—																Maximum Frame															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0

Figure 13-40. Maximum Frame Length Register Definition

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

Table 13-44. MAXFRM Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	Maximum Frame	By default this field is set to 0x0600 (1536 bytes). It sets the maximum Ethernet frame size in both the transmit and receive directions. (Refer to MACCFG2[Huge Frame].) It does not affect the size of packets sent or received via the FIFO packet interface. Note that if MACCFG2[Huge Frame] = 0, the value of this field must be less than or equal to MRBLR[MRBL] × (minimum number of RxBDs per ring). See Section 13.5.3.5.2, “MAC Configuration 2 Register (MACCFG2),” Section 13.5.3.3.9, “Maximum Receive Buffer Length Register (MRBLR),” and Section 13.6.7.3, “Receive Buffer Descriptors (RxBD).”

13.5.3.5.6 MII Management Configuration Register (MIIMCFG)

The MIIMCFG register is written by the user to configure all MII management operations. Note that MII management hardware is shared by all eTSECs. Thus, only through the MIIM registers of eTSEC1 can external PHYs be accessed and configured. Note: when an eTSEC is configured to use TBI/RTBI,

configuration of the TBI/RTBI (described in Section 13.5.4, “Ten-Bit Interface (TBI)”) is done through the MIIM registers for that eTSEC. For example, if a TBI/RTBI interface is required on eTSEC2, then the MIIM registers starting at offset 0x2_5520 are used to configure it.

Figure 13-41 describes the definition for the MIIMCFG register.

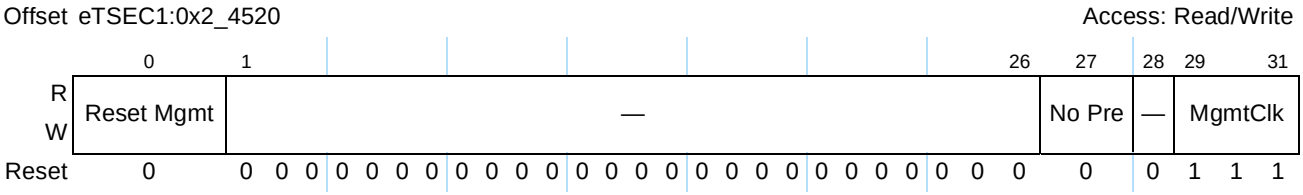


Figure 13-41. MII Management Configuration Register Definition

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

Table 13-45. MIIMCFG Field Descriptions

Bits	Name	Description
0	Reset Mgmt	Reset management. This bit is cleared by default. 0 Allow the MII MGMT to perform mgmt read/write cycles if requested through the host interface. 1 Reset the MII MGMT.
1–26	—	Reserved
27	No Pre	Preamble suppress. This bit is cleared by default. 0 The MII MGMT performs Mgmt read/write cycles with 32 clocks of preamble. 1 The MII MGMT suppresses preamble generation and reduces the Mgmt cycle from 64 clocks to 32 clocks. This is in accordance with IEEE 802.3/22.2.4.4.2.
28	—	Reserved
29–31	MgmtClk	This field determines the clock frequency of the MII management clock (EC_MDC). Its default value is 111. 000 1/4 of the eTSEC system clock divided by 8 001 1/4 of the eTSEC system clock divided by 8 010 1/6 of the eTSEC system clock divided by 8 011 1/8 of the eTSEC system clock divided by 8 100 1/10 of the eTSEC system clock divided by 8 101 1/14 of the eTSEC system clock divided by 8 110 1/20 of the eTSEC system clock divided by 8 111 1/28 of the eTSEC system clock divided by 8

13.5.3.5.7 MII Management Command Register (MIIMCOM)

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

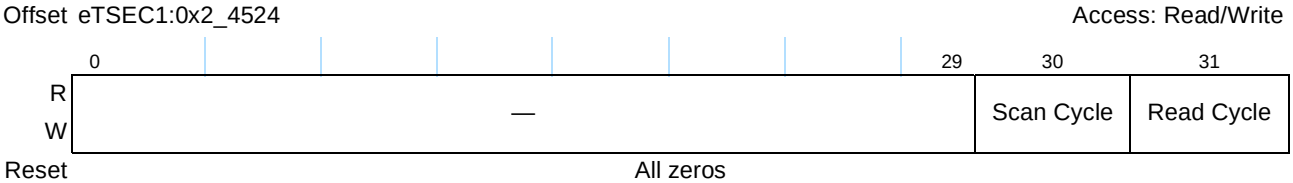


Figure 13-42. MIIMCOM Register Definition

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

Table 13-46. MIIMCOM Descriptions

Bits	Name	Description
0–29	—	Reserved
30	Scan Cycle	Scan cycle. This bit is cleared by default. 0 Normal operation. 1 The MII management continuously performs read cycles. This is useful for monitoring link fail, for example.
31	Read Cycle	Read cycle. This bit is cleared by default but is not self-clearing once set. 0 Normal operation. 1 The MII management performs a single read cycle upon the transition of this bit from 0 to 1 using the PHY address (at MIIMADD[PHY Address]) and the register address (at MIIMADD[Register Address]). The 0-to-1 transition of this bit also causes the MIIMIND[Busy] bit to be set. The read is complete when the MIIMIND[Busy] bit clears. Data is returned in register MIIMSTAT[PHY Status].

13.5.3.5.8 MII Management Address Register (MIIMADD)

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

Offset eTSEC1:0x2_4528 Access: Read/Write

	0–18	19–23	24–26	27–31
R W	—	PHY Address	—	Register Address
Reset	All zeros			

Figure 13-43. MIIMADD Register Definition

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

Table 13-47. MIIMADD Field Descriptions

Bits	Name	Description
0–18	—	Reserved
19–23	PHY Address	This field represents the 5-bit PHY address field of Mgmt cycles. Up to 31 PHYs can be addressed (0 is reserved). Its default value is 0x00.
24–26	—	Reserved
27–31	Register Address	This field represents the 5-bit register address field of Mgmt cycles. Up to 32 registers can be accessed. Its default value is 0x00.

13.5.3.5.9 MII Management Control Register (MIIMCON)

MIIMCON, shown in Figure 13-44, is written by the user.

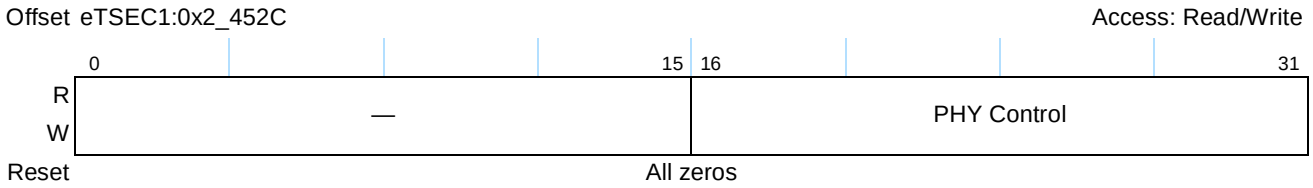


Figure 13-44. MII Mgmt Control Register Definition

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

Table 13-48. MIIMCON Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	PHY Control	If written, an MII Mgmt write cycle is performed using this 16-bit data, the pre-configured PHY address (at MIIMADD[PHY Address]) and the register address (at MIIMADD[Register Address]). Its default value is 0x0000.

13.5.3.5.10 MII Management Status Register (MIIMSTAT)

The MIIMSTAT register is read only by the user. Figure 13-45 describes the definition for the MIIMSTAT register.

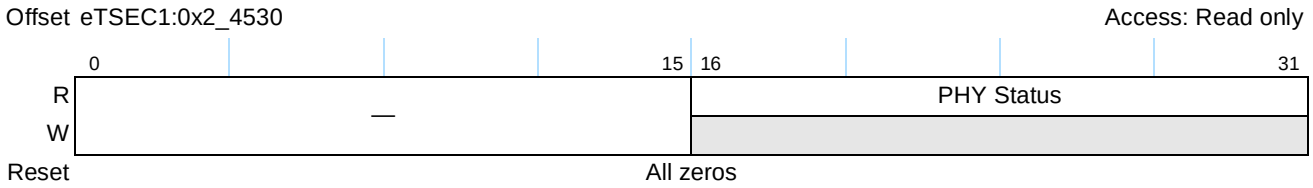


Figure 13-45. MIIMSTAT Register Definition

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

Table 13-49. MIIMSTAT Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	PHY Status	Following an MII Mgmt read cycle, the 16-bit data can be read from this location. Its default value is 0x0000.

13.5.3.5.11 MII Management Indicator Register (MIIMIND)

The MIIMIND register is read-only by the user. Figure 13-46 describes the definition for the MIIMIND register.

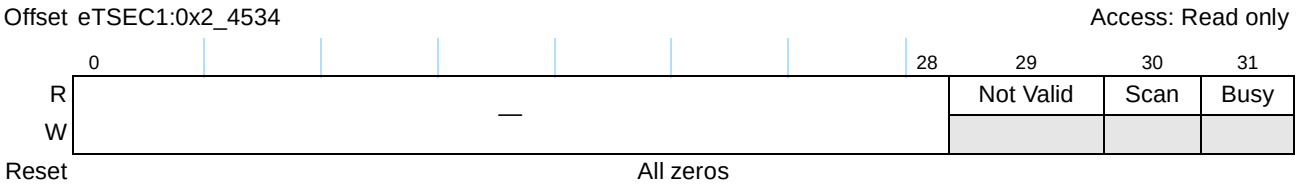


Figure 13-46. MII Mgmt Indicator Register Definition

Table 13-50. MIIMIND Field Descriptions

Bits	Name	Description
0–28	—	Reserved
29	Not Valid	Not valid. 0 MII Mgmt read cycle has completed and the read data is valid. 1 MII Mgmt read cycle has not completed and the read data is not yet valid.
30	Scan	Scan in progress. 0 A scan operation (continuous MII Mgmt read cycles) is not in progress. 1 A scan operation (continuous MII Mgmt read cycles) is in progress.
31	Busy	Busy. 0 MII Mgmt block is not currently performing an MII Mgmt read or write cycle. 1 MII Mgmt block is currently performing an MII Mgmt read or write cycle.

13.5.3.5.12 Interface Status Register (IFSTAT)

Figure 13-47 shows the IFSTAT register.

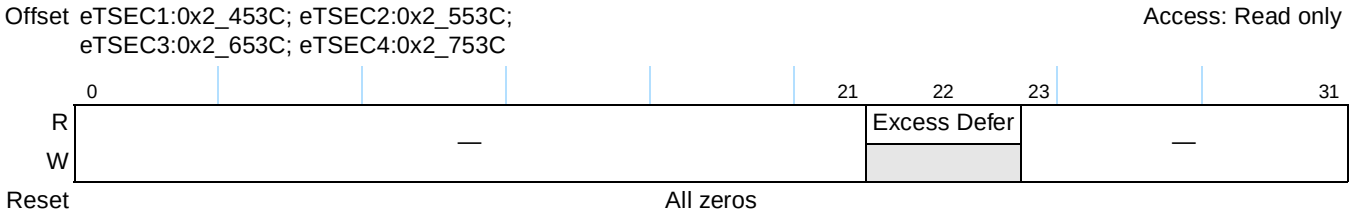


Figure 13-47. Interface Status Register Definition

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

Table 13-51. IFSTAT Field Descriptions

Bits	Name	Description
0–21	—	Reserved

Table 13-51. IFSTAT Field Descriptions (continued)

Bits	Name	Description
22	Excess Defer	Excessive transmission defer. This bit latches high and is cleared when read. This bit is cleared by default. 0 Normal operation. 1 The MAC excessively defers a transmission.
23–31	—	Reserved

13.5.3.5.13 MAC Station Address Part 1 Register (MACSTNADDR1)

The MACSTNADDR1 register is written by the user. The value of the station address written into MACSTNADDR1 and MACSTNADDR2 is byte reversed from how it would appear in the DA field of a frame in memory. For example, for a station address of 0x12345678ABCD, MACSTNADDR1 is set to 0xCDAB7856 and MACSTNADDR2 is set to 0x34120000.

Figure 13-48 shows the MACSTNADDR1 register.

Offset eTSEC1:0x2_4540; eTSEC2:0x2_5540; eTSEC3:0x2_6540; eTSEC4:0x2_7540 — Access: Read/Write

	0–7	8–15	16–23	24–31
R W	Station Address, 6th Octet	Station Address, 5th Octet	Station Address, 4th Octet	Station Address, 3rd Octet
Reset	All zeros			

Figure 13-48. MAC Station Address Part 1 Register Definition

Table 13-52 describes the fields of the MACSTNADDR1 register.

Table 13-52. MACSTNADDR1 Field Descriptions

Bit	Name	Description
0–7	Station Address, 6th Octet	This field holds the sixth octet of the station address. The sixth octet (station address bits 40–47) defaults to a value of 0x0.
8–15	Station Address, 5th Octet	This field holds the fifth octet of the station address. The fifth octet (station address bits 32–39) defaults to a value of 0x0.
16–23	Station Address, 4th Octet	This field holds the fourth octet of the station address. The fourth octet (station address bits 24–31) defaults to a value of 0x0.
24–31	Station Address, 3rd Octet	This field holds the third octet of the station address. The third octet (station address bits 16–23) defaults to a value of 0x0.

13.5.3.5.14 MAC Station Address Part 2 Register (MACSTNADDR2)

The MACSTNADDR2 register is written by the user. Figure 13-49 describes the definition for the MACSTNADDR2 register.

Offset eTSEC1:0x2_4544; eTSEC2:0x2_5544; eTSEC3:0x2_6544; eTSEC4:0x2_7544 — Access: Read/Write

	0–7	8–15	16–31
R W	Station Address, 2nd Octet	Station Address, 1st Octet	—
Reset	All zeros		

Figure 13-49. MAC Station Address Part 2 Register Definition

Table 13-53 describes the fields of the MACSTNADDR2 register.

Table 13-53. MACSTNADDR2 Field Descriptions

Bit	Name	Description
0–7	Station Address, 2nd Octet	This field holds the second octet of the station address. The second octet (station address bits 8–15) defaults to a value of 0x0.
8–15	Station Address, 1st Octet	This field holds the first octet of the station address. The first octet (station address bits 0–7) defaults to a value of 0x0.
16–31	—	Reserved

13.5.3.5.15 MAC Exact Match Address 1–15 Part 1 Registers (MAC01ADDR1–MAC15ADDR1)

The MAC01ADDR1–MAC15ADDR1 registers are written by the user with the unicast or multicast addresses aliasing the MAC. Figure 13-50 describes the definition for all of the fifteen MAC*n*ADDR1 registers. The value of the address written into MAC*x*ADDR1 and MAC*n*ADDR2 is byte reversed from how it would appear in the DA field of a frame in memory. For example, for a MAC address of 0x12345678ABCD, MACnADDR1 is set to 0xCDAB7856 and MACnADDR2 is set to 0x34120000. For any valid, non-zero MAC address received, exact match registers can be excluded individually by clearing them to all zero bytes.

Offset eTSEC1:0x2_4548+8×*n*; eTSEC2:0x2_5548+8×*n*; eTSEC3:0x2_6548+8×*n*; eTSEC4:0x2_7548+8×*n* — Access: Read/Write

	0–7	8–15	16–23	24–31
R W	Exact Match Address, 6th Octet	Exact Match Address, 5th Octet	Exact Match Address, 4th Octet	Exact Match Address, 3rd Octet
Reset	All zeros			

Figure 13-50. MAC Exact Match Address *n* Part 1 Register Definition

Table 13-52 describes the fields of a MAC*n*ADDR1 register.

Table 13-54. MAC*n*ADDR1 Field Descriptions

Bit	Name	Description
0–7	Exact Match Address, 6th Octet	Holds the sixth octet of the exact match address. The sixth octet (destination address bits 40–47) defaults to a value of 0x0.
8–15	Exact Match Address, 5th Octet	Holds the fifth octet of the exact match address. The fifth octet (destination address bits 32–39) defaults to a value of 0x0.
16–23	Exact Match Address, 4th Octet	Holds the fourth octet of the exact match address. The fourth octet (destination address bits 24–31) defaults to a value of 0x0.
24–31	Exact Match Address, 3rd Octet	Holds the third octet of the exact match address. The third octet (destination address bits 16–23) defaults to a value of 0x0.

13.5.3.5.16 MAC Exact Match Address 1–15 Part 2 Registers (MAC01ADDR2–MAC15ADDR2)

The MAC01ADDR2–MAC15ADDR2 registers are written by the user with the unicast or multicast addresses aliasing the MAC. Figure 13-51 describes the definition for all of the fifteen MACxADDR2 registers.

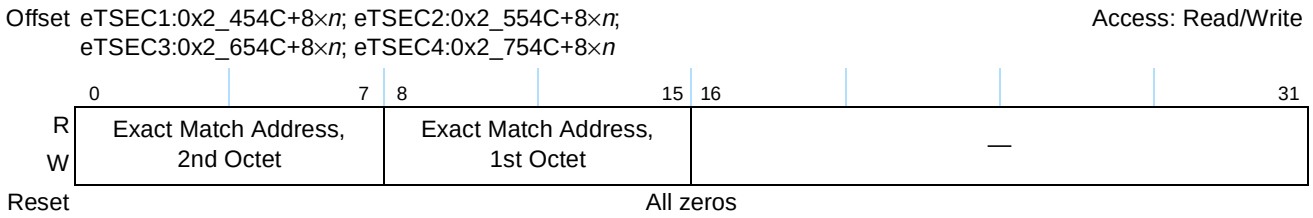


Figure 13-51. MAC Exact Match Address *x* Part 2 Register Definition

Table 13-53 describes the fields of a MAC*x*ADDR2 register.

Table 13-55. MAC01ADDR2–MAC15ADDR2 Field Descriptions

Bit	Name	Description
0–7	Exact Match Address, 2nd Octet	This field holds the second octet of the exact match address. The second octet (destination address bits 8–15) defaults to a value of 0x0.
8–15	Exact Match Address, 1st Octet	This field holds the first octet of the exact match address. The first octet (destination address bits 0–7) defaults to a value of 0x0.
16–31	—	Reserved

13.5.3.6 MIB Registers

This section describes the MIB registers. The eTSEC RMON module has 37 separate statistics counters, which simply count or accumulate statistical events that occur as packets transmitted and received. These counters support RMON MIB group 1, RMON MIB group 2 if table counters, RMON MIB group 3, RMON MIB group 9, RMON MIB 2, and the 802.3 Ethernet MIB.

An interrupt can be generated upon any one counter's rollover condition through a carry interrupt output from the RMON. Each counter's rollover condition can be discretely masked from causing an interrupt by internal masking registers. In addition, each individual counter value may be reset on read access, or all counters may be simultaneously reset by setting ECNTRL[CLRCNT].

The majority of MIB counters are Ethernet-specific.

In FIFO modes, only the following registers are updated:

- Transmit: TBYT, TPKT, TDRP
- Receive: RBYT, RPKT, RFCS

Note: RMON counters do not comprehend custom VLAN tagged frames. Affected counters include TRMGV, RMCA, RBCA, RXCF, RXPF, RXUO, RALN, RFLR, ROVR, RJBR, TMCA, TBCA, TXPF, TXCF. Specifically, custom VLAN tagged frames are not afforded the ability to be greater than 1518, as compared to the IEEE standard tagged frames.

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

Figure 13-52 describes the definition for the TR64 register.

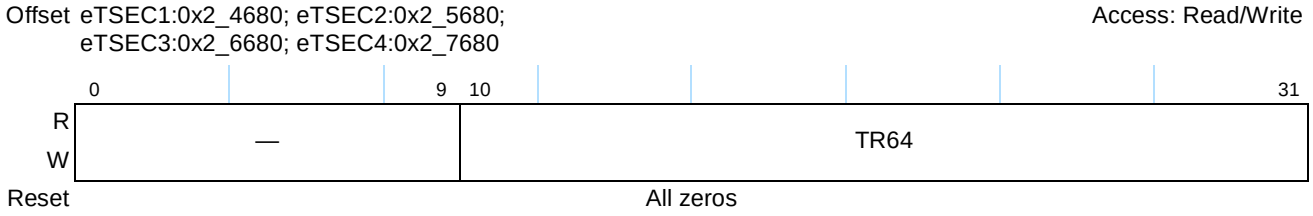


Figure 13-52. Transmit and Receive 64-Byte Frame Register Definition

Table 13-56 describes the fields of the TR64 register.

Table 13-56. TR64 Field Descriptions

Bits	Name	Description
0–9	—	Reserved
10–31	TR64	Transmit and receive 64-byte frame counter—Increment for each good or bad frame transmitted and received which is 64 bytes in length, inclusive (excluding preamble and SFD but including FCS bytes).

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

Figure 13-53 describes the definition for the TR127 register.

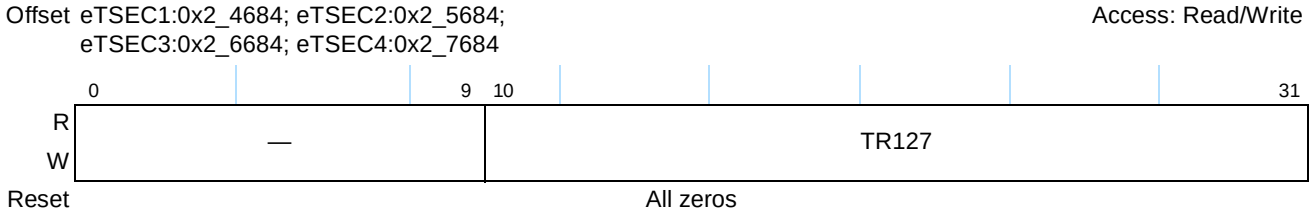


Figure 13-53. Transmit and Receive 65- to 127-Byte Frame Register Definition

Table 13-57 describes the fields of the TR127 register.

Table 13-57. TR127 Field Descriptions

Bits	Name	Description
0–9	—	Reserved
10–31	TR127	Transmit and receive 65- to 127-byte frame counter—Increments for each good or bad frame transmitted and received which is 65–127 bytes in length, inclusive (excluding preamble and SFD but including FCS bytes).

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

Figure 13-54 describes the definition for the TR255 register.

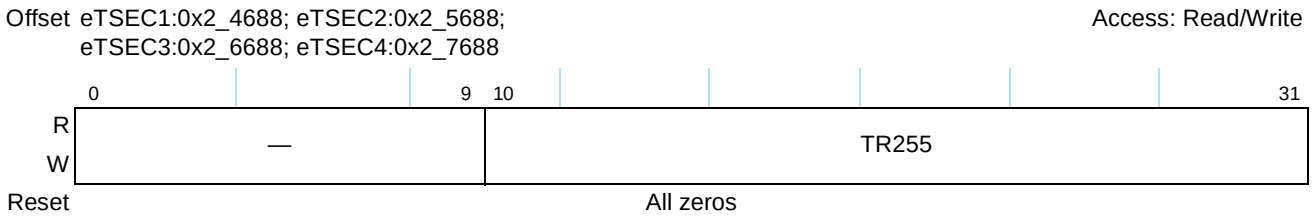


Figure 13-54. Transmit and Received 128- to 255-Byte Frame Register Definition

Table 13-58 describes the fields of the TR255 register.

Table 13-58. TR255 Field Descriptions

Bits	Name	Description
0–9	—	Reserved
10–31	TR255	Transmit and receive 128- to 255-byte frame counter—Increments for each good or bad frame transmitted and received which is 128–255 bytes in length, inclusive (excluding preamble and SFD but including FCS bytes).

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

Figure 13-55 describes the definition for the TR511 register.

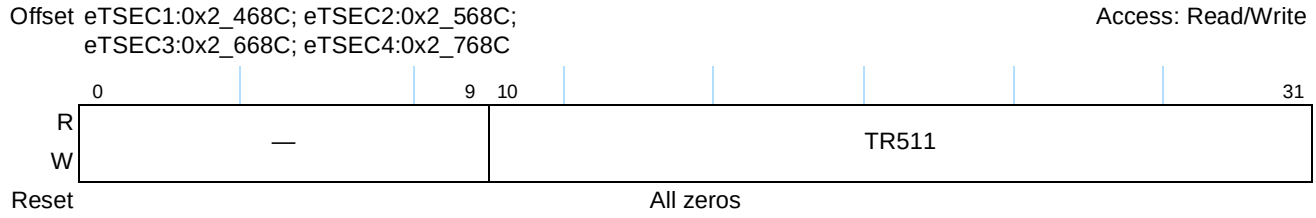


Figure 13-55. Transmit and Received 256- to 511-Byte Frame Register Definition

Table 13-59 describes the fields of the TR511 register.

Table 13-59. TR511 Field Descriptions

Bits	Name	Description
0–9	—	Reserved
10–31	TR511	Increments for each good or bad frame transmitted and received which is 256–511 bytes in length, inclusive (excluding preamble and SFD but including FCS bytes).

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

Figure 13-56 shows the TR1K register.

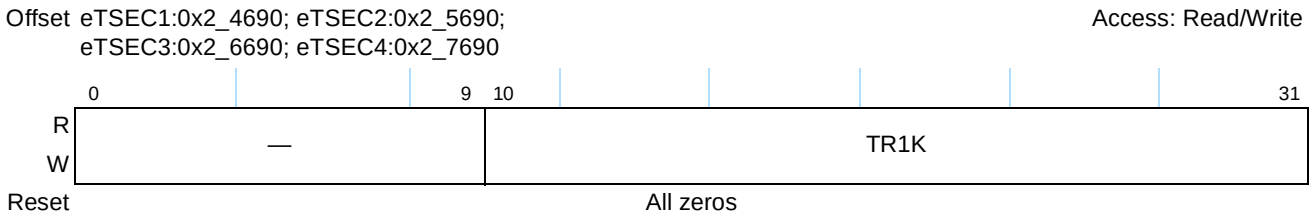


Figure 13-56. Transmit and Received 512- to 1023-Byte Frame Register Definition

Table 13-60 describes the fields of the TR1K register.

Table 13-60. TR1K Field Descriptions

Bits	Name	Description
0–9	—	Reserved
10–31	TR1K	Increments for each good or bad frame transmitted and received which is 512–1023 bytes in length, inclusive (excluding preamble and SFD but including FCS bytes).

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

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

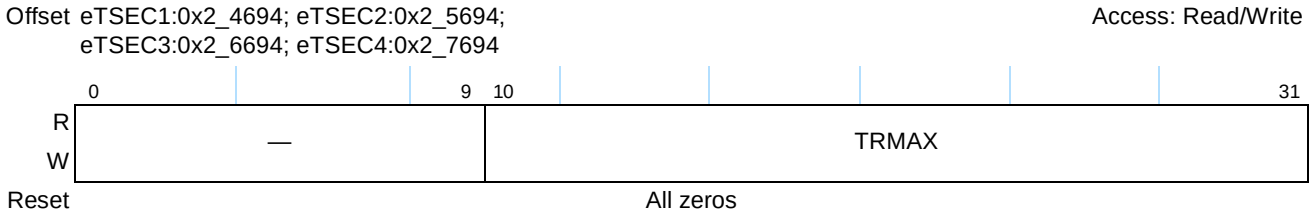


Figure 13-57. Transmit and Received 1024- to 1518-Byte Frame Register Definition

Table 13-61 describes the fields of the TRMAX register.

Table 13-61. TRMAX Field Descriptions

Bits	Name	Description
0–9	—	Reserved
10–31	TRMAX	Increments for each good or bad frame transmitted and received which is 1024–1518 bytes in length, inclusive (excluding preamble and SFD but including FCS bytes).

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

Figure 13-58 describes the definition for the TRMGV register.

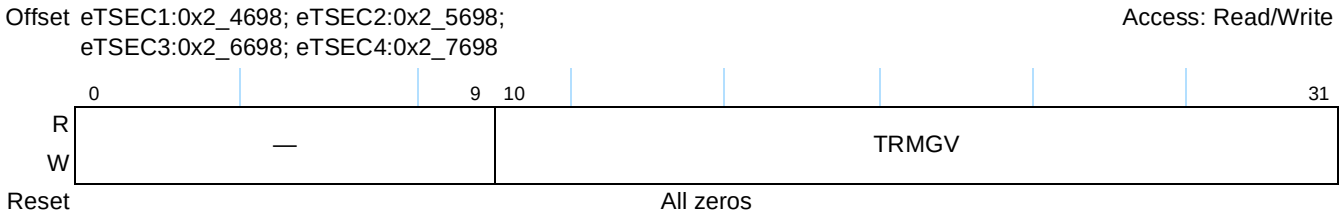


Figure 13-58. Transmit and Received 1519- to 1522-Byte VLAN Frame Register Definition

Table 13-62 describes the fields of the TRMGV register.

Table 13-62. TRMGV Field Descriptions

Bits	Name	Description
0–9	—	Reserved
10–31	TRMGV	Increments for each good or bad frame transmitted and received which is 1519–1522 bytes in length, inclusive (excluding preamble and SFD but including FCS bytes).

13.5.3.6.8 Receive Byte Counter (RBYT)

Figure 13-59 shows the RBYT register.

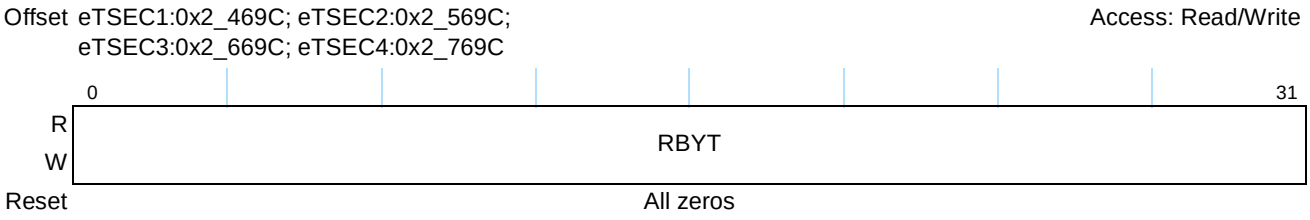


Figure 13-59. Receive Byte Counter Register Definition

Table 13-63 describes the fields of the RBYT register.

Table 13-63. RBYT Field Descriptions

Bits	Name	Description
0–31	RBYT	Receive byte counter. The statistic counter register increments by the byte count of frames received, including those in bad packets, excluding preamble and SFD but including FCS bytes. In FIFO mode, all bytes (including FCS bytes) are counted.

13.5.3.6.9 Receive Packet Counter (RPKT)

Figure 13-60 describes the definition for the RPKT register.

Offset eTSEC1:0x2_46A0; eTSEC2:0x2_56A0; eTSEC3:0x2_66A0; eTSEC4:0x2_76A0 Access: Read/Write

Reset All zeros

Figure 13-60. Receive Packet Counter Register Definition

Table 13-64 describes the fields of the RPKT register.

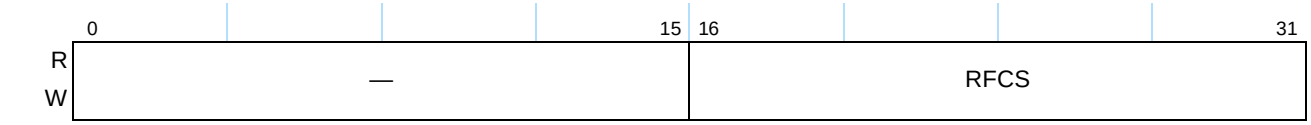
Table 13-64. RPKT Field Descriptions

Bits	Name	Description
0–9	—	Reserved
10-31	RPKT	Receive packet counter. Increments for each frame received packet (including bad packets, all unicast, broadcast, and multicast packets).

13.5.3.6.10 Receive FCS Error Counter (RFCS)

Figure 13-61 describes the definition for the RFCS register.

Offset eTSEC1:0x2_46A4; eTSEC2:0x2_56A4; eTSEC3:0x2_66A4; eTSEC4:0x2_76A4 Access: Read/Write



Reset All zeros

Figure 13-61. Receive FCS Error Counter Register Definition

Table 13-65 describes the fields of the RFCS register.

Table 13-65. RFCS Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	RFCS	Receive FCS error counter. In Ethernet mode, increments for each frame received that has an integral 64–1518 length and contains a frame check sequence error. In FIFO mode, increments for each frame received that contains a frame check sequence error (regardless of size).

13.5.3.6.11 Receive Multicast Packet Counter (RMCA)

Figure 13-62 describes the definition for the RMCA register.

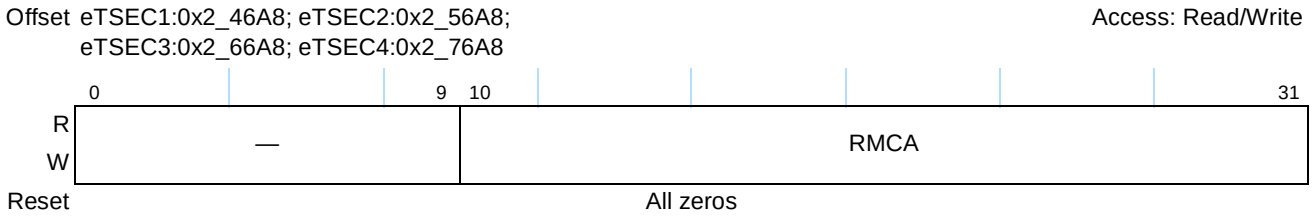


Figure 13-62. Receive Multicast Packet Counter Register Definition

Table 13-66 describes the fields of the RMCA register.

Table 13-66. RMCA Field Descriptions

Bits	Name	Description
0–9	—	Reserved
10–31	RMCA	Receive multicast packet counter. Increments for each multicast frame with valid CRC and of lengths 64 to 1518 (non VLAN) or 1522 (VLAN), excluding broadcast frames. This count does not include range/length errors.

13.5.3.6.12 Receive Broadcast Packet Counter (RBCA)

Figure 13-63 describes the definition for the RBCA register.

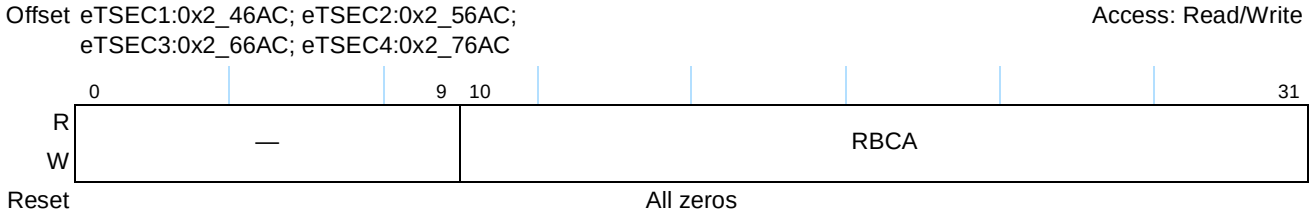


Figure 13-63. Receive Broadcast Packet Counter Register Definition

Table 13-67 describes the fields of the RBCA register.

Table 13-67. RBCA Field Descriptions

Bits	Name	Description
0–9	—	Reserved
10–31	RBCA	Receive broadcast packet counter. Increments for each broadcast frame with valid CRC and of lengths 64 to 1518 (non VLAN) or 1522 (VLAN), excluding multicast frames. Does not include range/length errors.

13.5.3.6.13 Receive Control Frame Packet Counter (RXCF)

Figure 13-64 describes the definition for the RXCF register.

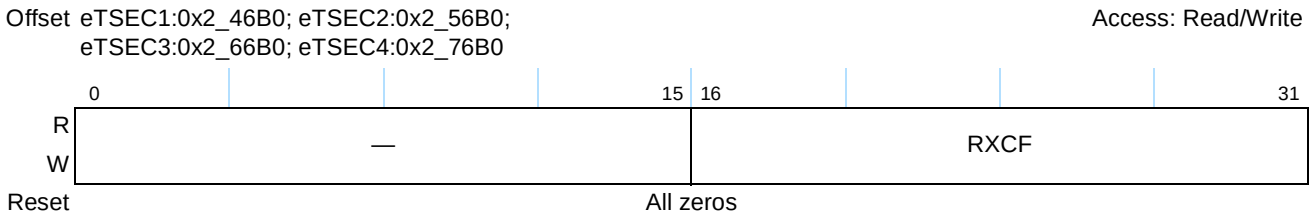


Figure 13-64. Receive Control Frame Packet Counter Register Definition

Table 13-68 describes the fields of the RXCF register.

Table 13-68. RXCF Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	RXCF	Receive control frame packet counter. Increments for each MAC control frame received (PAUSE and unsupported) with valid CRC and of lengths 64 to 1518 (non VLAN) or 1522 (VLAN).

13.5.3.6.14 Receive Pause Frame Packet Counter (RXPF)

Figure 13-65 describes the definition for the RXPF register.

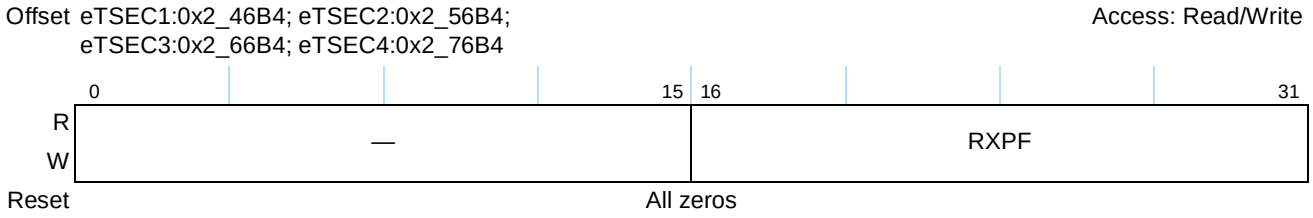


Figure 13-65. Receive Pause Frame Packet Counter Register Definition

Table 13-69 describes the fields of the RXPF register.

Table 13-69. RXPF Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	RXPF	Receive PAUSE frame packet counter. Increments each time a PAUSE MAC control frame is received with valid CRC and of lengths 64 to 1518 (non VLAN) or 1522 (VLAN).

13.5.3.6.15 Receive Unknown Opcode Packet Counter (RXUO)

Figure 13-66 describes the definition for the RXUO register.

Offset eTSEC1:0x2_46B8; eTSEC2:0x2_56B8; eTSEC3:0x2_66B8; eTSEC4:0x2_76B8 Access: Read/Write

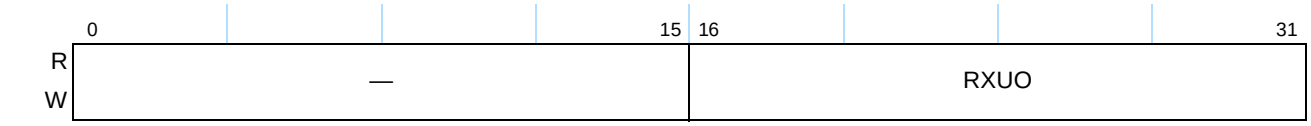


Figure 13-66. Receive Unknown OPCode Packet Counter Register Definition

Table 13-70 describes the fields of the RXUO register.

Table 13-70. RXUO Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	RXUO	Receive unknown opcode counter. Increments each time a MAC control frame is received which contains an opcode other than PAUSE, but the frame has valid CRC and length 64 to 1518 (non VLAN) or 1522 (VLAN).

13.5.3.6.16 Receive Alignment Error Counter (RALN)

Figure 13-67 describes the definition for the RALN register.

Offset eTSEC1:0x2_46BC; eTSEC2:0x2_56BC; eTSEC3:0x2_66BC; eTSEC4:0x2_76BC Access: Read/Write

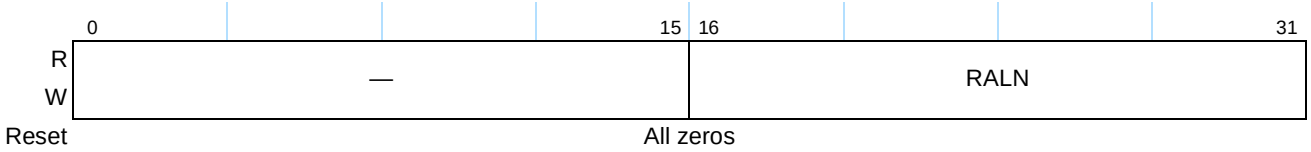


Figure 13-67. Receive Alignment Error Counter Register Definition

Table 13-71 describes the fields of the RALN register.

Table 13-71. RALN Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	RALN	Receive alignment error counter. Increments for each received frame from 64 to 1518 (non VLAN) or 1522 (VLAN) which contains an invalid FCS and is not an integral number of bytes.

13.5.3.6.17 Receive Frame Length Error Counter (RFLR)

Figure 13-68 describes the definition for the RFLR register.

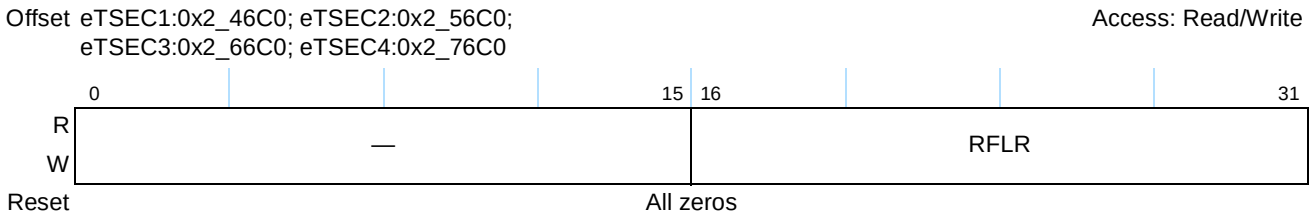


Figure 13-68. Receive Frame Length Error Counter Register Definition

Table 13-72 describes the fields of the RFLR register.

Table 13-72. RFLR Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	RFLR	Receive frame length error counter. Increments for each frame received in which the 802.3 length field did not match the number of data bytes actually received (46–1500 bytes). The counter does not increment if the length field is not a valid 802.3 length, such as an Ethertype value.

13.5.3.6.18 Receive Code Error Counter (RCDE)

Figure 13-69 describes the definition for the RCDE register.

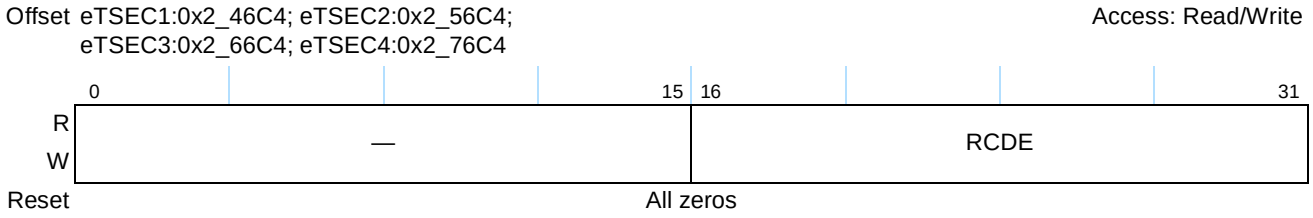


Figure 13-69. Receive Code Error Counter Register Definition

Table 13-73 describes the fields of the RCDE register.

Table 13-73. RCDE Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	RCDE	Receive code error counter. Increments each time a valid carrier is present and at least one invalid data symbol is detected.

13.5.3.6.19 Receive Carrier Sense Error Counter (RCSE)

Figure 13-70 describes the definition for the RCSE register.

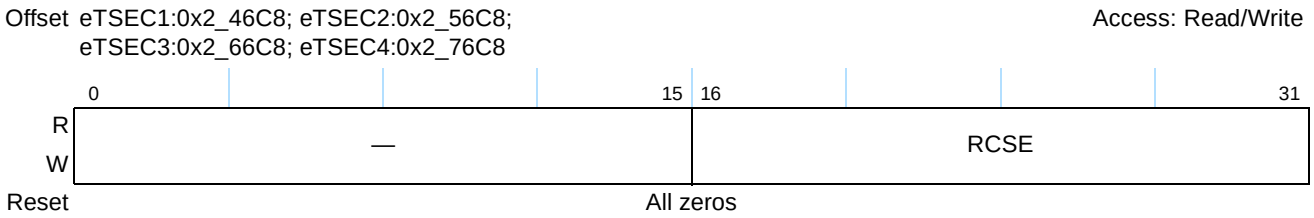


Figure 13-70. Receive Carrier Sense Error Counter Register Definition

Table 13-74 describes the fields of the RCSE register.

Table 13-74. RCSE Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	RCSE	Receive false carrier counter. Counts the number of times that the carrier sense condition was lost or never asserted when attempting to transmit a frame on a particular interface. The count represented by an instance of this object is incremented at most once per transmission attempt, even if the carrier sense condition fluctuates during a transmission attempt. The event is reported along with the statistics generated on the next received frame, as defined by a 1 on TSEC*n*_RX_ER and an 0xE on TSEC*n*_RXD. Only one false carrier condition can be detected and logged between frames.

13.5.3.6.20 Receive Undersize Packet Counter (RUND)

Figure 13-71 describes the definition for the RUND register.

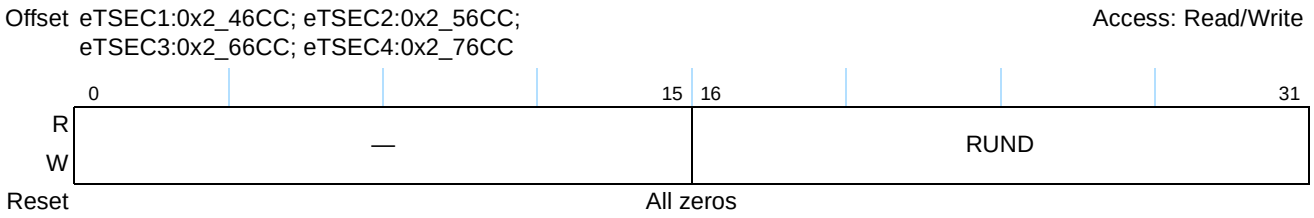


Figure 13-71. Receive Undersize Packet Counter Register Definition

Table 13-75 describes the fields of the RUND register.

Table 13-75. RUND Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	RUND	Receive undersize packet counter. Increments each time a frame is received which is less than 64 bytes in length and contains a valid FCS and were otherwise well formed. This count does not include range length errors.

13.5.3.6.21 Receive Oversize Packet Counter (ROVR)

Figure 13-72 describes the definition for the ROVR register.

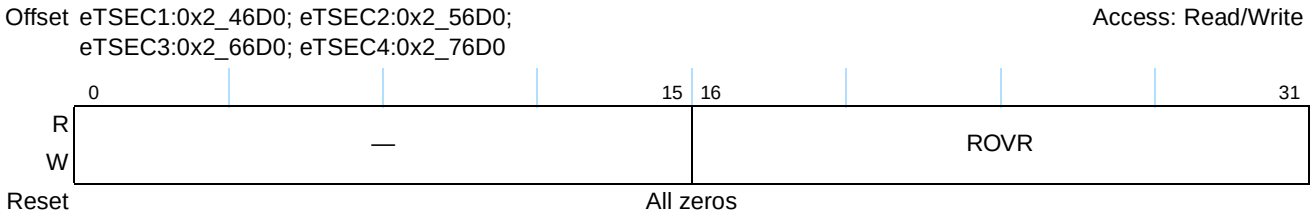


Figure 13-72. Receive Oversize Packet Counter Register Definition

Table 13-76 describes the fields of the ROVR register.

Table 13-76. 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.

13.5.3.6.22 Receive Fragments Counter (RFRG)

Figure 13-73 describes the definition for the RFRG register.

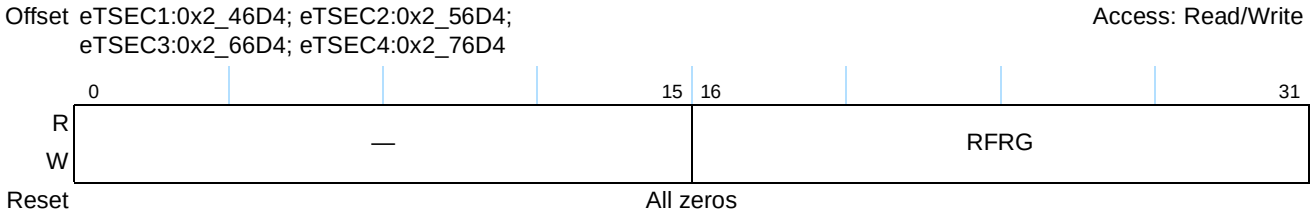


Figure 13-73. Receive Fragments Counter Register Definition

Table 13-77 describes the fields of the RFRG register.

Table 13-77. RFRG Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	RFRG	Receive fragments counter. Increments for each frame received which is less than 64 bytes in length and contains an invalid FCS. This includes integral and non-integral lengths.

13.5.3.6.23 Receive Jabber Counter (RJBR)

Figure 13-74 describes the definition for the RJBR register.

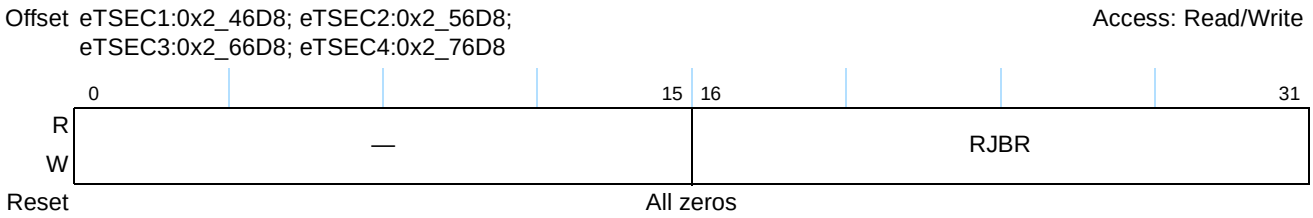


Figure 13-74. Receive Jabber Counter Register Definition

Table 13-78 describes the fields of the RJBR register.

Table 13-78. RJBR Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	RJBR	Receive jabber counter. Increments for frames received which exceed 1518 (non VLAN) or 1522 (VLAN) bytes and contain an invalid FCS. This includes alignment errors.

13.5.3.6.24 Receive Dropped Packet Counter (RDRP)

Figure 13-75 describes the definition for the RDRP register.

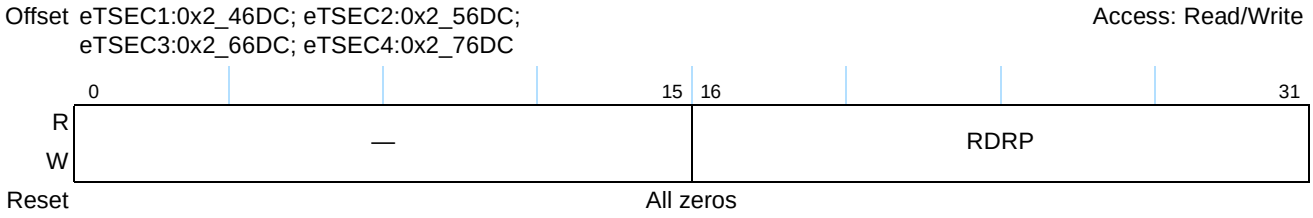


Figure 13-75. Receive Dropped Packet Counter Register Definition

Table 13-79 describes the fields of the RDRP register.

Table 13-79. RDRP Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	RDRP	Receive dropped packets counter. Increments for frames received which are streamed to system but are later dropped due to lack of system resources.

13.5.3.6.25 Transmit Byte Counter (TBYT)

Figure 13-76 describes the definition for the TBYT register.

Offset eTSEC1:0x2_46E0; eTSEC2:0x2_56E0; eTSEC3:0x2_66E0; eTSEC4:0x2_76E0 Access: Read/Write

Figure 13-76. Transmit Byte Counter Register Definition

Table 13-80 describes the fields of the TBYT register.

Table 13-80. TBYT Field Descriptions

Bits	Name	Description
0–31	TBYT	Transmit byte counter. Increments by the number of bytes that were put on the wire including fragments of frames that were involved with collisions. This count does not include preamble/SFD or jam bytes. Note that the value of TBYT may be greater than the actual number of bytes transmitted if the frame is truncated because it exceeds MAXFRM.

13.5.3.6.26 Transmit Packet Counter (TPKT)

Figure 13-77 describes the definition for the TPKT register.

Offset eTSEC1:0x2_46E4; eTSEC2:0x2_56E4; eTSEC3:0x2_66E4; eTSEC4:0x2_76E4 Access: Read/Write

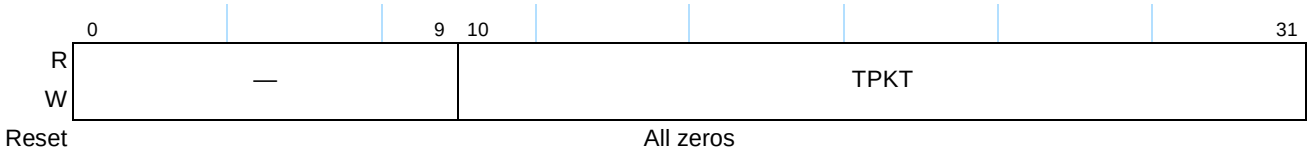


Figure 13-77. Transmit Packet Counter Register Definition

Table 13-81 describes the fields of the TPKT register.

Table 13-81. TPKT Field Descriptions

Bits	Name	Description
0–9	—	Reserved
10–31	TPKT	Transmit packet counter. Increments for each transmitted packet (including bad packets, excessive deferred packets, excessive collision packets, late collision packets, all unicast, broadcast, and multicast packets).

13.5.3.6.27 Transmit Multicast Packet Counter (TMCA)

Figure 13-78 describes the definition for the TMCA register.

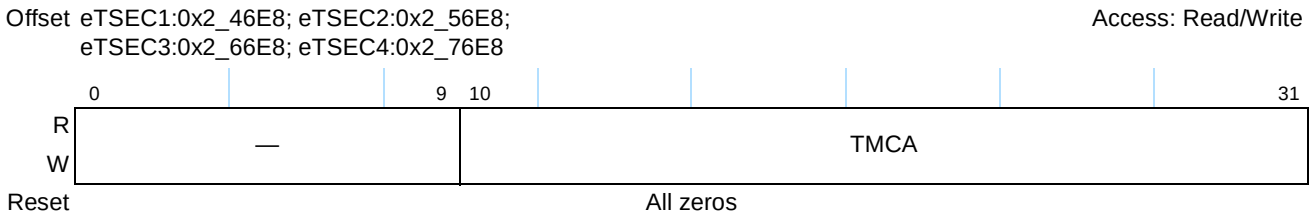


Figure 13-78. Transmit Multicast Packet Counter Register Definition

Table 13-82 describes the fields of the TMCA register.

Table 13-82. TMCA Field Descriptions

Bits	Name	Description
0–9	—	Reserved
10–31	TMCA	Transmit multicast packet counter. Increments for each multicast valid frame transmitted (excluding broadcast frames) with valid CRC and of lengths 64 to 1518 (non VLAN) or 1522 (VLAN).

13.5.3.6.28 Transmit Broadcast Packet Counter (TBCA)

Figure 13-79 describes the definition for the TBCA register.

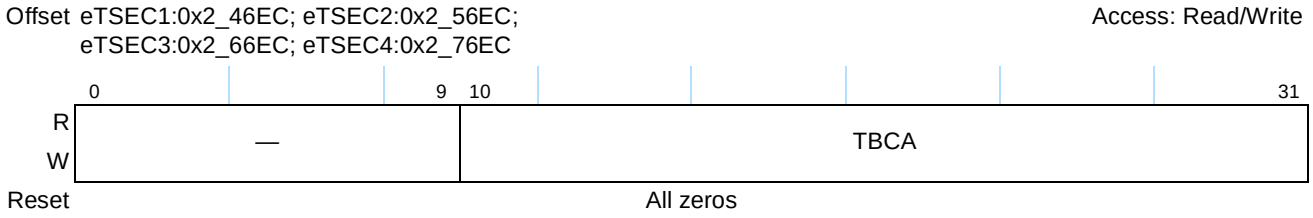


Figure 13-79. Transmit Broadcast Packet Counter Register Definition

Table 13-83 describes the fields of the TBCA register.

Table 13-83. TBCA Field Descriptions

Bits	Name	Description
0–9	—	Reserved
10–31	TBCA	Transmit broadcast packet counter. Increments for each broadcast frame transmitted (excluding multicast frames) with valid CRC and of lengths 64 to 1518 (non VLAN) or 1522 (VLAN).

13.5.3.6.29 Transmit Pause Control Frame Counter (TXPF)

Figure 13-80 describes the definition for the TXPF register.

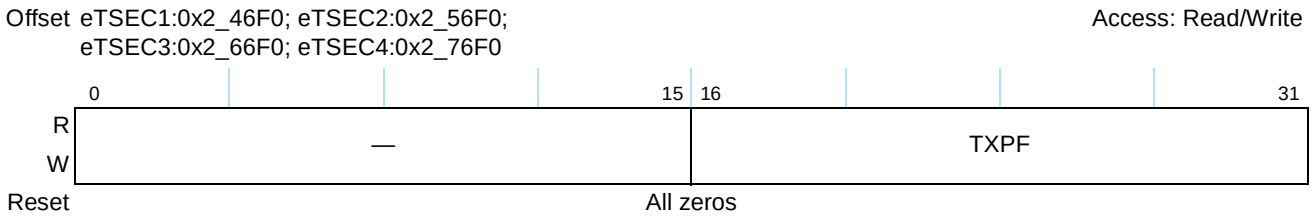


Figure 13-80. Transmit Pause Control Frame Counter Register Definition

Table 13-84 describes the fields of the TXPF register.

Table 13-84. TXPF Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	TXPF	Transmit PAUSE frame packet counter. Increments each time a valid PAUSE MAC control frame is transmitted with valid CRC and of lengths 64 to 1518 (non VLAN) or 1522 (VLAN).

13.5.3.6.30 Transmit Deferral Packet Counter (TDFR)

Figure 13-81 describes the definition for the TDFR register.

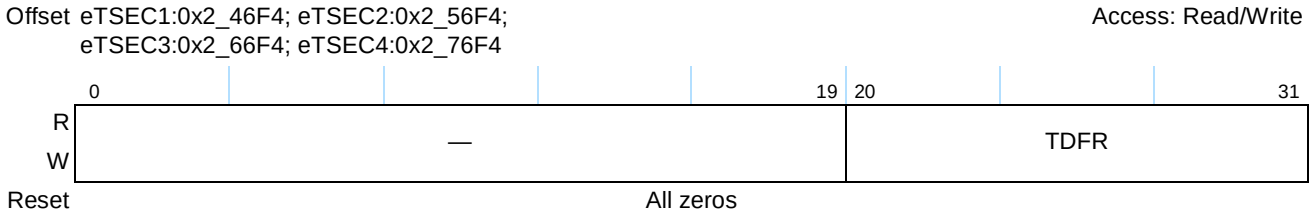


Figure 13-81. Transmit Deferral Packet Counter Register Definition

Table 13-85 describes the fields of the TDFR register.

Table 13-85. TDFR Field Descriptions

Bits	Name	Description
0–19	—	Reserved
20–31	TDFR	Transmit deferral packet counter. Increments for each frame, which was deferred on its first transmission attempt. This count does not include frames involved in collisions.

13.5.3.6.31 Transmit Excessive Deferral Packet Counter (TEDF)

Figure 13-82 describes the definition for the TEDF register.

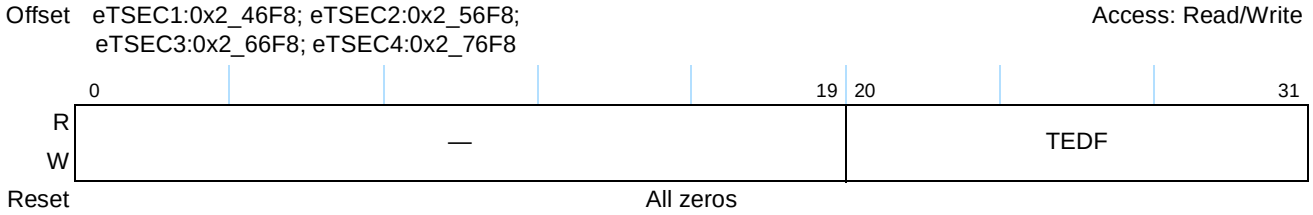


Figure 13-82. Transmit Excessive Deferral Packet Counter Register Definition

Table 13-86 describes the fields of the TEDF register.

Table 13-86. TEDF Field Descriptions

Bits	Name	Description
0–19	—	Reserved
20–31	TEDF	Transmit excessive deferral packet counter. Increments for frames aborted which were deferred for an excessive period of time (3036 byte times).

13.5.3.6.32 Transmit Single Collision Packet Counter (TSCL)

Figure 13-83 describes the definition for the TSCL register.

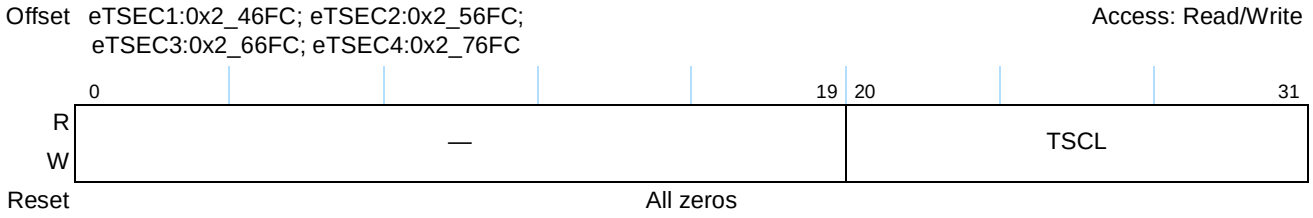


Figure 13-83. Transmit Single Collision Packet Counter Register Definition

Table 13-87 describes the fields of the TSCL register.

Table 13-87. TSCL Field Descriptions

Bits	Name	Description
0–19	—	Reserved
20–31	TSCL	Transmit single collision packet counter. Increments for each frame transmitted which experienced exactly one collision during transmission.

13.5.3.6.33 Transmit Multiple Collision Packet Counter (TMCL)

Figure 13-84 describes the definition for the TMCL register.

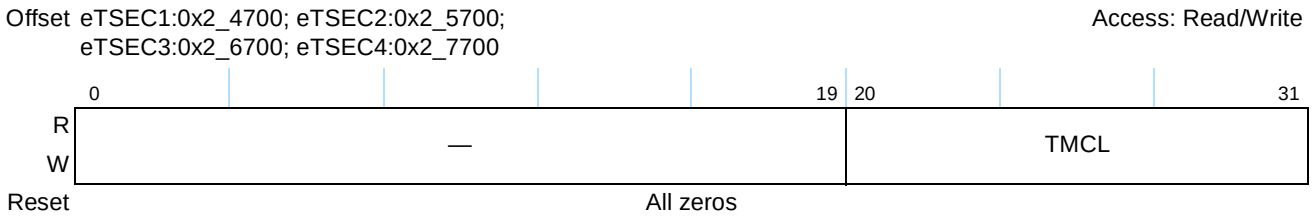


Figure 13-84. Transmit Multiple Collision Packet Counter Register Definition

Table 13-88 describes the fields of the TMCL register.

Table 13-88. TMCL Field Descriptions

Bits	Name	Description
0–19	—	Reserved
20–31	TMCL	Transmit multiple collision packet counter. Increments for each frame transmitted which experienced 2–15 collisions (including any late collisions) during transmission as defined using the Half_Duplex[RETRANSMISSION MAXIMUM] field.

13.5.3.6.34 Transmit Late Collision Packet Counter (TLCL)

Figure 13-85 describes the definition for the TLCL register.

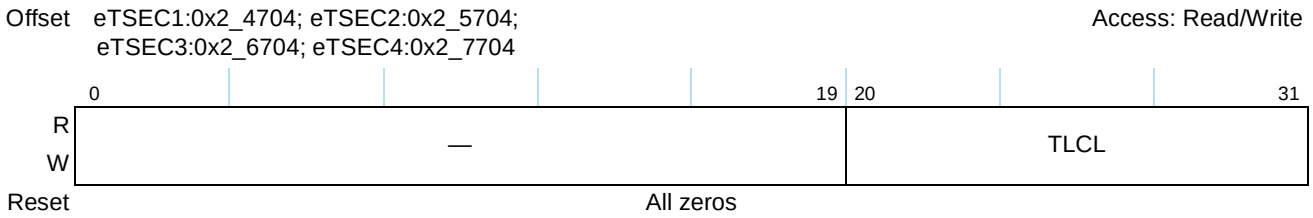


Figure 13-85. Transmit Late Collision Packet Counter Register Definition

Table 13-89 describes the fields of the TLCL register.

Table 13-89. TLCL Field Descriptions

Bits	Name	Description
0–19	—	Reserved
20–31	TLCL	Transmit late collision packet counter. Increments for each frame transmitted which experienced a late collision during a transmission attempt. Late collisions are defined using the collision window field of the half-duplex [26:31] register.

13.5.3.6.35 Transmit Excessive Collision Packet Counter (TXCL)

Figure 13-86 describes the definition for the TXCL register.

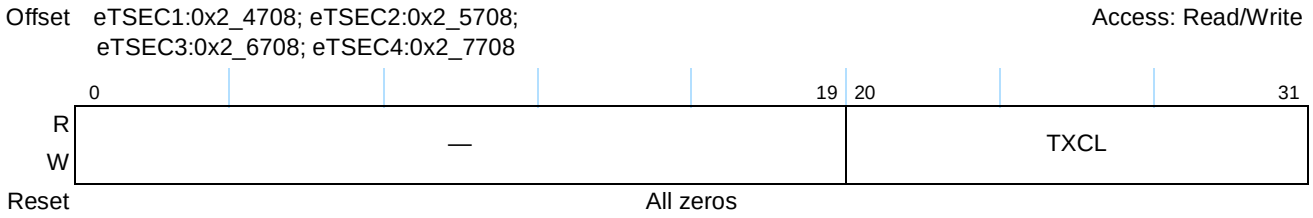


Figure 13-86. Transmit Excessive Collision Packet Counter Register Definition

Table 13-90 describes the fields of the TXCL register.

Table 13-90. TXCL Field Descriptions

Bits	Name	Description
0–19	—	Reserved
20–31	TXCL	Transmit excessive collision packet counter. Increments for each frame that experienced 16 collisions during transmission and was aborted.

13.5.3.6.36 Transmit Total Collision Counter (TNCL)

Figure 13-87 describes the definition for the TNCL register.

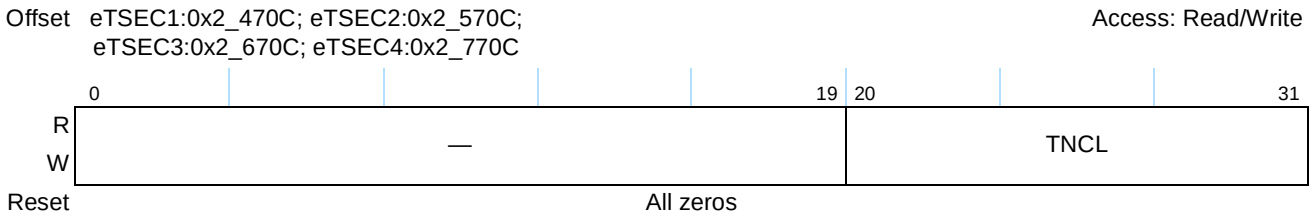


Figure 13-87. Transmit Total Collision Counter Register Definition

Table 13-91 describes the fields of the TNCL register.

Table 13-91. TNCL Field Descriptions

Bits	Name	Description
0–19	—	Reserved
20–31	TNCL	Transmit total collision counter. Increments by the number of collisions experienced during the transmission of a frame as defined as the simultaneous presence of signals on the DO and RD circuits (That is, transmitting and receiving at the same time). **Note:** This count does not include collisions that result in an excessive collision condition.

13.5.3.6.37 Transmit Drop Frame Counter (TDRP)

Figure 13-88 describes the definition for the TDRP register.

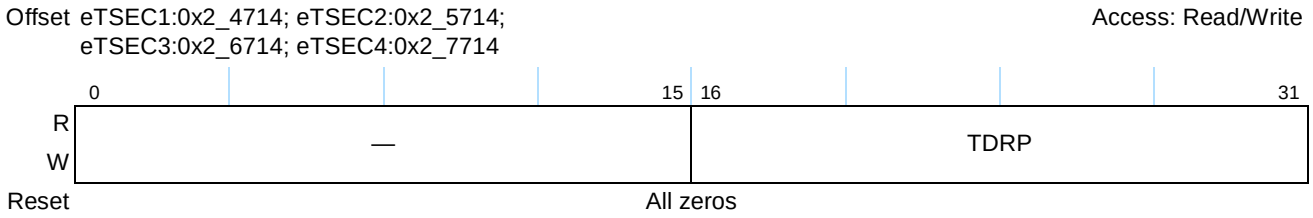


Figure 13-88. Transmit Drop Frame Counter Register Definition

Table 13-92 describes the fields of the TDRP register.

Table 13-92. TDRP Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	TDRP	Transmit drop frame counter. Increments each time a memory error or an underrun has occurred.

13.5.3.6.38 Transmit Jabber Frame Counter (TJBR)

Figure 13-89 describes the definition for the TJBR register.

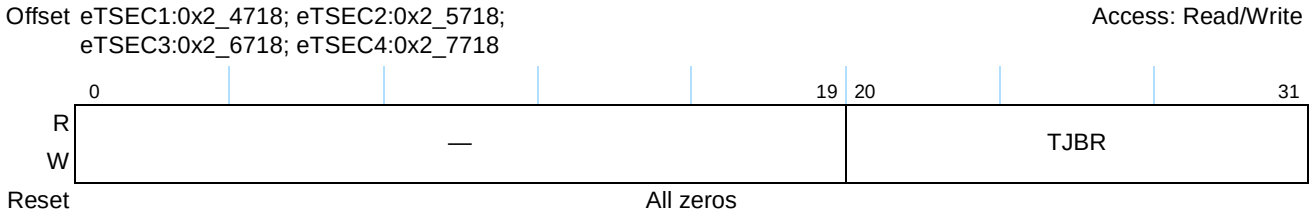


Figure 13-89. Transmit Jabber Frame Counter Register Definition

Table 13-93 describes the fields of the TJBR register.

Table 13-93. TJBR Field Descriptions

Bits	Name	Description
0–19	—	Reserved
20–31	TJBR	Transmit jabber frame counter. Increments for each oversized transmitted frame with an incorrect FCS value.

13.5.3.6.39 Transmit FCS Error Counter (TFCS)

Figure 13-90 describes the definition for the TFCS register.

Offset eTSEC1:0x2_471C; eTSEC2:0x2_571C; eTSEC3:0x2_671C; eTSEC4:0x2_771C — Access: Read/Write

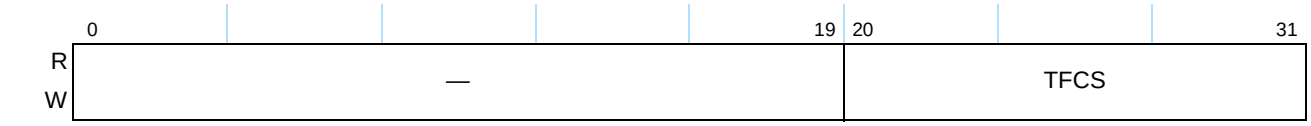


Figure 13-90. Transmit FCS Error Counter Register Definition

Table 13-94 describes the fields of the TFCS register.

Table 13-94. TFCS Field Descriptions

Bits	Name	Description
0–19	—	Reserved
20–31	TFCS	Transmit FCS error counter. Increments for every valid sized packet with an incorrect FCS value.

13.5.3.6.40 Transmit Control Frame Counter (TXCF)

Figure 13-91 describes the definition for the TXCF register.

Offset eTSEC1:0x2_4720; eTSEC2:0x2_5720; eTSEC3:0x2_6720; eTSEC4:0x2_7720 — Access: Read/Write

Figure 13-91. Transmit Control Frame Counter Register Definition

Table 13-95 describes the fields of the TXCF register.

Table 13-95. TXCF Field Descriptions

Bits	Name	Description
0–19	—	Reserved
20–31	TXCF	Transmit control frame counter. Increments for every control frame with valid CRC and of lengths 64 to 1518 (non VLAN) or 1522 (VLAN).

13.5.3.6.41 Transmit Oversize Frame Counter (TOVR)

Figure 13-92 describes the definition for the TOVR register.

Offset eTSEC1:0x2_4724; eTSEC2:0x2_5724; eTSEC3:0x2_6724; eTSEC4:0x2_7724 — Access: Read/Write

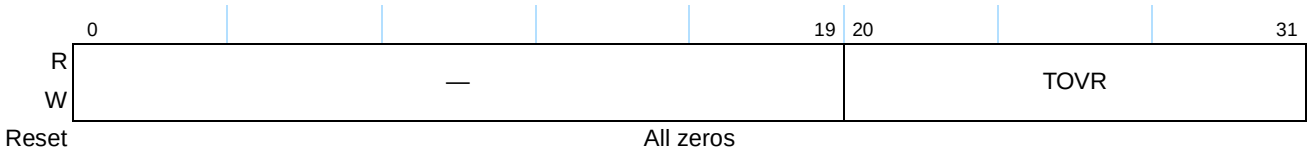


Figure 13-92. Transmit Oversized Frame Counter Register Definition

Table 13-96 describes the fields of the TOVR register.

Table 13-96. TOVR Field Descriptions

Bits	Name	Description
0–19	—	Reserved
20–31	TOVR	Transmit oversize frame counter. Increments for each oversized transmitted frame with a correct FCS value.

13.5.3.6.42 Transmit Undersize Frame Counter (TUND)

Figure 13-93 describes the definition for the TUND register.

Offset eTSEC1:0x2_4728; eTSEC2:0x2_5728; eTSEC3:0x2_6728; eTSEC4:0x2_7728 — Access: Read/Write

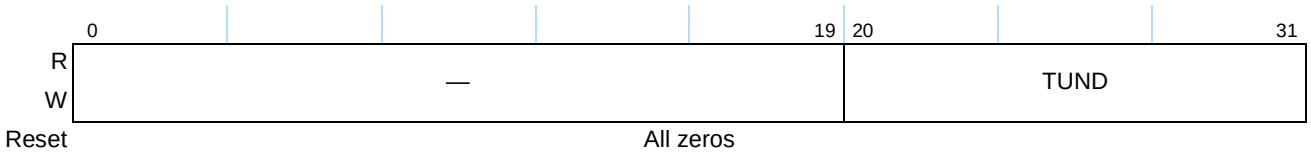


Figure 13-93. Transmit Undersize Frame Counter Register Definition

Table 13-97 describes the fields of the TUND register.

Table 13-97. TUND Field Descriptions

Bits	Name	Description
0–19	—	Reserved
20–31	TUND	Transmit undersize frame counter. Increments for every frame less then 64 bytes, with a correct FCS value.

13.5.3.6.43 Transmit Fragment Counter (TFRG)

Figure 13-94 describes the definition for the TFRG register.

Offset eTSEC1:0x2_472C; eTSEC2:0x2_572C; eTSEC3:0x2_672C; eTSEC4:0x2_772C — Access: Read/Write

	0	19	20	31
R W	—		TFRG	
Reset	All zeros			

Figure 13-94. Transmit Fragment Counter Register Definition

Table 13-98 describes the fields of the TFRG register.

Table 13-98. TFRG Field Descriptions

Bits	Name	Description
0–19	—	Reserved
20–31	TFRG	Transmit fragment counter. Increments for every frame less then 64 bytes, with an incorrect FCS value.

13.5.3.6.44 Carry Register 1 (CAR1)

Carry register bits are cleared on carry register writes when the respective bits are set. Figure 13-95 describes the definition for the CAR1 register.

Offset eTSEC1:0x2_4730; eTSEC2:0x2_5730; eTSEC3:0x2_6730; eTSEC4:0x2_7730 — Access: Read/Write

	0	1	2	3	4	5	6	7	13	14	15
R W	C1 64	C1 127	C1 255	C1 511	C1 1K	C1 MAX	C1 MGV	—		C1 REJ	C1 RBY
Reset	All zeros										

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	C1 RPK	C1 RFC	C1 RMC	C1 RBC	C1 RXC	C1 RXP	C1 RXU	C1 RAL	C1 RFL	C1 RCD	C1 RCS	C1 RUN	C1 ROV	C1 RFR	C1 RJB	C1 RDR
Reset	All zeros															

Figure 13-95. Carry Register 1 (CAR1) Register Definition

Table 13-99 describes the fields of the CAR1 register.

Table 13-99. 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

Table 13-99. CAR1 Field Descriptions (continued)

Bits	Name	Description
4	C11K	Carry register 1 TR1K counter carry bit
5	C1MAX	Carry register 1 TRMAX counter carry bit
6	C1MGV	Carry register 1 TRMGV counter carry bit
7–13	—	Reserved
14	C1REJ	Carry register 1 RREJ counter carry bit
15	C1RBY	Carry register 1 RBYT counter carry bit
16	C1RPK	Carry register 1 RPKT counter carry bit
17	C1RFC	Carry register 1 RFCS counter carry bit
18	C1RMC	Carry register 1 RMCA counter carry bit
19	C1RBC	Carry register 1 RBCA counter carry bit
20	C1RXC	Carry register 1 RXCF counter carry bit
21	C1RXP	Carry register 1 RXPF counter carry bit
22	C1RXU	Carry register 1 RXUO counter carry bit
23	C1RAL	Carry register 1 RALN counter carry bit
24	C1RFL	Carry register 1 RFLR counter carry bit
25	C1RCD	Carry register 1 RCDE counter carry bit
26	C1RCS	Carry register 1 RCSE counter carry bit
27	C1RUN	Carry register 1 RUND counter carry bit
28	C1ROV	Carry register 1 ROVR counter carry bit
29	C1RFR	Carry register 1 RFRG counter carry bit
30	C1RJB	Carry register 1 RJBR counter carry bit
31	C1RDR	Carry register 1 RDRP counter carry bit

13.5.3.6.45 Carry Register 2 (CAR2)

Figure 13-96 describes the definition for the CAR2 register.

Offset eTSEC1:0x2_4734; eTSEC2:0x2_5734; eTSEC3:0x2_6734; eTSEC4:0x2_7734 — Access: Read/Write

	0												12	13	14	15
R W	—												C2 TJB	C2 TFC	C2 TCF	C2 TOV
Reset	All zeros															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	C2 TUN	C2 TFG	C2 TBY	C2 TPK	C2 TMC	C2 TBC	C2 TPF	C2 TDF	C2 TED	C2 TSC	C2 TMA	C2 TLC	C2 TXC	C2 TNC	—	C2 TDP
Reset	All zeros															

Figure 13-96. Carry Register 2 (CAR2) Register Definition

Carry register bits are cleared on carry register write when the respective bits are set. Table 13-100 describes the fields of the CAR2 register.

Table 13-100. 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
29	C2TNC	Carry register 2 TNCL counter carry bit

Table 13-100. CAR2 Field Descriptions (continued)

Bits	Name	Description
30	—	Reserved, should be cleared
31	C2TDP	Carry register 2 TDRP counter carry bit

13.5.3.6.46 Carry Mask Register 1 (CAM1)

While one of the below mask bits are cleared, the corresponding carry bit in CAR1 is allowed to cause interrupt indications in register IEVENT[MSR0]. These bits all default to a set state. Figure 13-97 describes the definition for the CAM1 register.

Offset eTSEC1:0x2_4738; eTSEC2:0x2_5738; eTSEC3:0x2_6738; eTSEC4:0x2_7738 — Access: Read/Write

	0	1	2	3	4	5	6	7						13	14	15
R	M1	M1	M1	M1	M1	M1	M1								M1	M1
W	64	127	255	511	1K	MAX	MGV	—							REJ	RBY
Reset	1	1	1	1	1	1	1	0	0	0	0	0	0	0	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	M1	M1	M1	M1	M1	M1	M1	M1	M1	M1	M1	M1	M1	M1	M1	M1
W	RPK	RFC	RMC	RBC	RXC	RXP	RXU	RAL	RFL	RCD	RCS	RUN	ROV	RFR	RJB	RDR
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 13-97. Carry Mask Register 1 (CAM1) Register Definition

Table 13-101 describes the fields of the CAM1 register.

Table 13-101. 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
7–13	—	Reserved
14	M1REJ	Mask register 1 RREJ counter carry bit mask
15	M1RBY	Mask register 1 RBYT counter carry bit mask
16	M1RPK	Mask register 1 RPKT counter carry bit mask
17	M1RFC	Mask register 1 RFCS counter carry bit mask
18	M1RMC	Mask register 1 RMCA counter carry bit mask

Table 13-101. CAM1 Field Descriptions (continued)

Bits	Name	Description
19	M1RBC	Mask register 1 RBCA counter carry bit mask
20	M1RXC	Mask register 1 RXCF counter carry bit mask
21	M1RXP	Mask register 1 RXPF counter carry bit mask
22	M1RXU	Mask register 1 RXUO counter carry bit mask
23	M1RAL	Mask register 1 RALN counter carry bit mask
24	M1RFL	Mask register 1 RFLR counter carry bit mask
25	M1RCD	Mask register 1 RCDE counter carry bit mask
26	M1RCS	Mask register 1 RCSE counter carry bit mask
27	M1RUN	Mask register 1 RUND counter carry bit mask
28	M1ROV	Mask register 1 ROVR counter carry bit mask
29	M1RFR	Mask register 1 RFRG counter carry bit mask
30	M1RJB	Mask register 1 RJBR counter carry bit mask
31	M1RDR	Mask register 1 RDRP counter carry bit mask

13.5.3.6.47 Carry Mask Register 2 (CAM2)

While one of the below mask bits are cleared, the corresponding carry bit in CAR2 is allowed to cause interrupt indications in register IEVENT[MSR0]. These bits default to a set state. Figure 13-98 describes the definition for the CAM2 register.

Offset eTSEC1:0x2_473C; eTSEC2:0x2_573C; eTSEC3:0x2_673C; eTSEC4:0x2_773C — Access: Read/Write

	0											11	12	13	14	15
R	—												M2 TJB	M2 TFC	M2 TCF	M2 TOV
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	M2 TUN	M2 TFG	M2 TBY	M2 TPK	M2 TMC	M2 TBC	M2 TPF	M2 TDF	M2 TED	M2 TSC	M2 TMA	M2 TLC	M2 TXC	M2 TNC	—	M2 TDP
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1

Figure 13-98. Carry Mask Register 2 (CAM2) Register Definition

Table 13-102 describes the fields of the CAM2 register.

Table 13-102. CAM2 Field Descriptions

Bits	Name	Description
0–11	—	Reserved
12	M2TJB	Mask register 2 TJBR counter carry bit mask

Table 13-102. CAM2 Field Descriptions (continued)

Bits	Name	Description
13	M2TFC	Mask register 2 TFCS counter carry bit mask
14	M2TCF	Mask register 2 TXCF counter carry bit mask
15	M2TOV	Mask register 2 TOVR counter carry bit mask
16	M2TUN	Mask register 2 TUND counter carry bit mask
17	M2TFG	Mask register 2 TFRG counter carry bit mask
18	M2TBY	Mask register 2 TBYT counter carry bit mask
19	M2TPK	Mask register 2 TPKT counter carry bit mask
20	M2TMC	Mask register 2 TMCA counter carry bit mask
21	M2TBC	Mask register 2 TBCA counter carry bit mask
22	M2TPF	Mask register 2 TXPF counter carry bit mask
23	M2TDF	Mask register 2 TDFR counter carry bit mask
24	M2TED	Mask register 2 TEDF counter carry bit mask
25	M2TSC	Mask register 2 TSCL counter carry bit mask
26	M2TMA	Mask register 2 TMCL counter carry bit mask
27	M2TLC	Mask register 2 TLCL counter carry bit mask
28	M2TXC	Mask register 2 TXCL counter carry bit mask
29	M2TNC	Mask register 2 TNCL counter carry bit mask
30	—	Reserved
31	M2TDP	Mask register 2 TDRP counter carry bit mask

13.5.3.6.48 Receive Filer Rejected Packet Counter (RREJ)

Figure 13-99 describes the definition for the RREJ register.

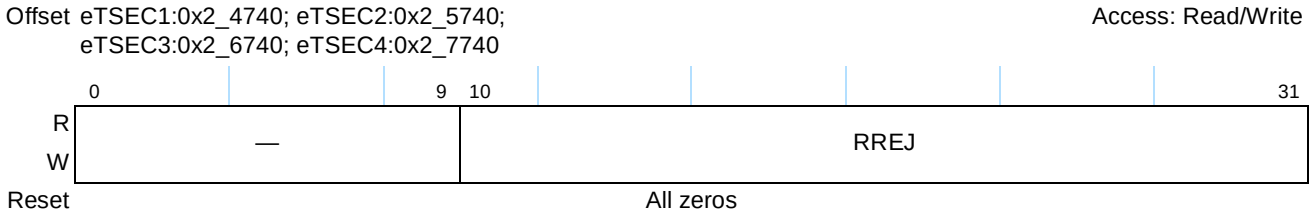


Figure 13-99. Receive Filer Rejected Packet Counter Register Definition

Table 13-67 describes the fields of the RREJ register.

Table 13-103. RREJ Field Descriptions

Bits	Name	Description
0–9	—	Reserved
10–31	RREJ	Receive filer rejected packet counter. Increments for each frame with valid CRC received, but rejected by the receive queue filer—either due to a matching rule that asserted the REJ flag or due to filing to a RxBD ring that was not enabled (see IEVENT[FIQ] error).

13.5.3.7 Hash Function Registers

This section provides detailed descriptions of the registers used for hash functions. All of the registers are 32 bits wide. The DA field of every received frame is processed through a 32-bit CRC generator (CRC-32 polynomial), and the 8 or 9 most significant bits of the CRC are mapped to a hash table entry. The user can enable a hash entry by setting its bit. A hash entry usually represents a set of addresses. A hash table hit occurs if the DA CRC result points to an enabled hash entry. Software may need to further filter the address in order to eliminate false-positive hits in the hash table.

If RCTRL[GHTX] = 0, the 8 most significant bits of the CRC are used as the hash table index. In this case, registers IGADDR0–IGADDR7 comprise a 256-entry hash table exclusively for individual (unicast) address matching, while registers GADDR0–GADDR7 comprise a 256-entry hash table for group (multicast) address matching. If RCTRL[GHTX] = 1, the group hash table is extended to all 512 entries, and the 9 most significant bits of the CRC are used as the hash table index. In this case, registers IGADDR0–IGADDR7 hold hash table entries 0–255 for group addresses, while registers GADDR0–GADDR7 hold entries 256–511 of the extended group hash table.

See Section 13.6.3.7.2, “Hash Table Algorithm” for more information on the hash algorithm.

13.5.3.7.1 Individual/Group Address Registers 0–7 (IGADDR*n*)

The IGADDR*n* registers are written by the user. Together these registers represent, depending on RCTRL[GHTX], either the 256 entries of the individual address hash table, or the first 256 entries of the extended group address hash table used in the address recognition process. The user can enable a hash entry by setting the appropriate bit. A hash table hit occurs if the DA CRC-32 result points to an enabled hash entry.

Figure 13-100 describes the definition for the IGADDR*n* register.

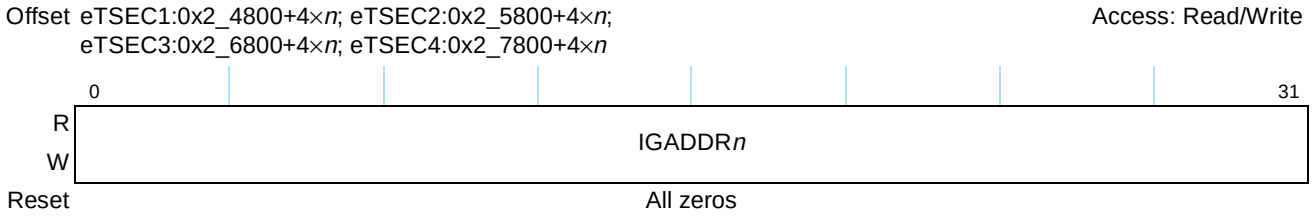


Figure 13-100. IGADDR*n* Register Definition

Table 13-105 describes the fields of the IGADDR*n* register.

Table 13-104. IGADDR*n* Field Descriptions

Bits	Name	Description
0–31	IGADDR*n*	Represents the 32-bit value associated with the corresponding register. When RCTRL[GHTX] = 0, IGADDR0 contains entries 0–31 of the 256-entry individual hash table and IGADDR7 represents entries 224–255. When RCTRL[GHTX] = 1, IGADDR0 contains entries 0–31 of the 512-entry extended group hash table and IGADDR7 represents entries 224–255.

13.5.3.7.2 Group Address Registers 0–7 (GADDR*n*)

The GADDR*n* registers are written by the user. Together these registers represent, depending on RCTRL[GHTX], either the 256 entries of the group address hash table, or the last 256 entries of the extended group address hash table used in the address recognition process. The user can enable a hash entry by setting the appropriate bit. A hash table hit occurs if the DA CRC result points to an enabled hash entry. Figure 13-101 describes the definition for the GADDR*n* register.

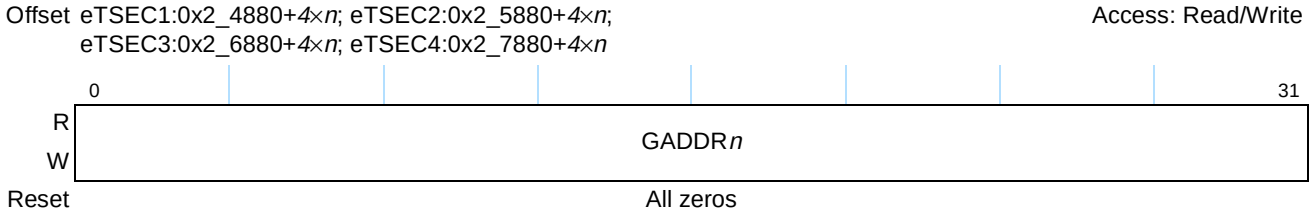


Figure 13-101. GADDR*n* Register Definition

Table 13-105 describes the fields of the GADDR*n* register.

Table 13-105. GADDR*n* Field Descriptions

Bits	Name	Description
0–31	GADDR*n*	Represents the 32-bit value associated with the corresponding register. When RCTRL[GHTX] = 0, GADDR0 contains entries 0–31 of the 256-entry group hash table and GADDR7 represents entries 224–255. When RCTRL[GHTX] = 1, GADDR0 contains entries 256–287 of the 512-entry extended group hash table and GADDR7 represents entries 480–511.

13.5.3.8 FIFO Registers

This section provides detailed descriptions of the registers used to configure the FIFO interface. All of the registers are 32 bits wide. The ECNTRL[FIFM] bit is set to indicate that data transfers take place over this interface. Please refer to Section 13.6.2, “Connecting to FIFO Interfaces,” for details of the signaling protocols available.

13.5.3.8.1 FIFO Configuration Register (FIFOCFG)

The FIFO Configuration Register configures and enables the 8/16-bit FIFO interface.

Figure 13-102 describes the definition for the FIFOCFG register.

Offset eTSEC1:0x2_4A00; eTSEC2:0x2_5A00; eTSEC3:0x2_6A00; eTSEC4:0x2_7A00 — Access: Read/Write

	0							7	8							15
R W	—								IPG							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20		22	23	24	25	26	27	28	29	30	31
R W	RRX	RTX	RXE	TXE	—			LPB	RFC	TFC	FFC	CRCAPP	CRCCHK	—	SIGM	
Reset	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0

Figure 13-102. FIFOCFG Register Definition

Table 13-106 describes the fields of the FIFOCFG register.

Table 13-106. FIFOCFG Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–15	IPG	Minimum inter packet gap. This sets the minimum number of cycles inserted between back-to-back frames transmitted over the FIFO interface. The minimum required is 3 cycles if CRCAPP=0, 5 cycles for 16-bit interfaces if CRCAPP=1 and 7 cycles for 8-bit interfaces if CRCAPP=1.
16	RRX	Enable reset of FIFO receive function. 0 Do not reset the FIFO receiver. 1 Reset the FIFO receiver for as long as this bit is set.
17	RTX	Enable reset of FIFO transmit function. 0 Do not reset the FIFO transmitter. 1 Reset the FIFO transmitter for as long as this bit is set.
18	RXE	Enable FIFO receive function. 0 Disable reception over the FIFO interface, ignoring data presented to the signals. 1 Enable normal reception over the FIFO interface.
19	TXE	Enable FIFO transmit function. 0 Disable transmission over the FIFO interface. 1 Enable normal transmission over the FIFO interface.
20–22	—	Reserved.
23	LPB	Loopback enable. 0 Do not loopback data in the FIFO interface. 1 Loopback transmitted data to the FIFO receiver rather than outputting transmitted data to signals.
24	RFC	Enable receive flow control. Setting FFC overrides this bit. 0 Do not allow the FIFO receiver to assert link-level flow control if eTSEC requires it. 1 Allow the FIFO receiver to assert link-level flow control if eTSEC requires it. This is the default setting.
25	TFC	Enable transmit flow control. 0 Do not allow the FIFO transmitter to assert link-level flow control if transmit data is unavailable, resulting in underruns. 1 Allow the FIFO transmitter to assert link-level flow control if transmit data is unavailable and SIGM = 01. This is the default setting.

Table 13-106. FIFOCFG Field Descriptions (continued)

Bits	Name	Description
26	FFC	Force flow control. This can be used by software to stop reception on the FIFO interface. 0 Do not assert link-level flow control through the RXFC signal unless eTSEC requires flow control. 1 Force flow control on the RXFC signal in encoded FIFO packet mode regardless of eTSEC pause requirements.
27	CRCAPP	Append a CRC (CRC-32 algorithm, as per IEEE Std. 802.3) to the end of every transmitted frame. 0 Do not automatically append a CRC to transmitted frames. Allow TxBD[TC], if set, to control when a CRC is appended. 1 Automatically append a CRC to transmitted frames. Ignore TxBD[TC].
28	CRCCHK	Check the CRC (CRC-32 algorithm, as per IEEE Std. 802.3) at the end of every frame. 0 Do not automatically check the last 4 bytes of received frames for a valid CRC. 1 Automatically check the last 4 bytes of received frames for a valid CRC. If a CRC error is detected, or insufficient data is received to recover the CRC, the RxBD[CR] bit is set.
29	—	Reserved
30–31	SIGM	FIFO signaling mode. Determines how the GMII signals are interpreted as framing signals. 00 GMII style mode. 01 Encoded packet mode. 10 Reserved 11 Reserved

13.5.3.9 DMA Attribute Registers

This section describes the two eTSEC DMA attribute registers.

13.5.3.9.1 Attribute Register (ATTR)

The attribute register defines memory access attributes and transaction types used to access buffer descriptors, to write receive data, and to read transmit data. Snoop enable attributes may be set for reading buffer descriptors and for reading transmit data.

Figure 13-103 describes the definition for the ATTR register.

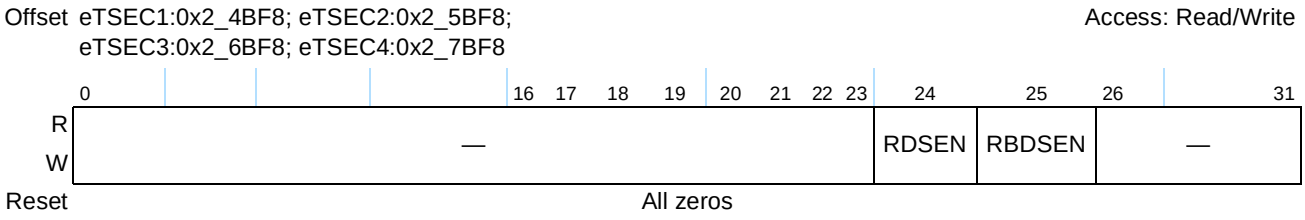


Figure 13-103. ATTR Register Definition

Table 13-107 describes the fields of the ATTR register.

Table 13-107. ATTR Field Descriptions

Bits	Name	Description
0–23	—	Reserved
24	RDSEN	Rx data snoop enable. 0 Disables snooping of all receive frames data to memory. 1 Enables snooping of all receive frames data to memory.
25	RBDSEN	RxBD snoop enable. 0 Disables snooping of all receive BD memory accesses. 1 Enables snooping of all receive BD memory accesses.
26–31	—	Reserved

13.5.3.10 Lossless Flow Control Configuration Registers

When enabled through RCTRL[LFC], the eTSEC tracks location of the last free BD in each Rx BD ring through the value of RFBPTR*n*. Using this pointer and the ring length stored in RQPRM*n*[LEN], the eTSEC continuously calculates the number of free BDs in the ring. Whenever the calculated number of free BDs in the ring drops below the pause threshold specified in RQPRM*n*[FBTHR], the eTSEC issues link layer flow control. It continues to assert flow control until the free BD count for each active ring reaches or exceeds RQPRM*n*[FBTHR]. See Section 13.6.6.1, “Back Pressure Determination through Free Buffers,” for the theory of operation of these registers.

13.5.3.10.1 Receive Queue Parameters 0–7 (RQPRM0–PQPRM7)

The RQPRM*n* registers specify the minimum number of BDs required to prevent flow control being asserted and the total number of Rx BDs in their respective ring. Whenever the free BD count calculated by the eTSEC for any active ring drops below the value of RQPRM*n*[FBTHR] for that ring, link level flow control is asserted. Software must not write to RQPRM*n* while LFC is enabled and the eTSEC is actively receiving frames. However, software may modify these registers after disabling LFC by clearing RCTRL[LFC]. Note that packets may be lost due to lack of RxBDs while RCTRL[LFC] is clear. Software can prevent packet loss by manually generating pause frames (through TCTRL[TFC_PAUSE]) to cover the time when RCTRL[LFC] is clear. Figure 13-104 describes the definition for the RQPRM*n* register.

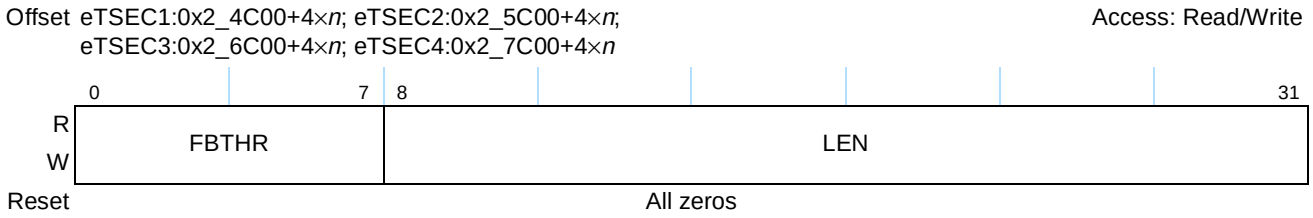


Figure 13-104. RQPRM Register Definition

Table 13-108 describes the fields of the RQPRM register.

Table 13-108. RQPRM Field Descriptions

Bits	Name	Description
0–7	FBTHR	Free BD threshold. Minimum number of BDs required for normal operation. If the eTSEC calculated number of free BDs drops below this threshold, link layer flow control is asserted.
8–31	LEN	Ring length. Total number of Rx BDs in this ring.

13.5.3.10.2 Receive Free Buffer Descriptor Pointer Registers 0–7 (RFBPTR0–RFBPTR7)

The RFBPTR*n* registers specify the location of the last free buffer descriptor in their respective ring. These registers live in the same 32b address space – and must share the same 4 most significant bits – as RBPTR*n*. That is, RFBPTR*n* and its associated RBPTR*n* must remain in the same 256MB page. Like RBPTR*n*, whenever RBASE*n* is updated, RFBPTR*n* is initialized to the value of RBASE*n*. This indicates that the ring is completely empty. As buffers are freed and their respective BDs are returned (by setting the EMPTY bit) to the ring, software is expected to update this register. The eTSEC then performs modulo arithmetic involving RBASE*n*, RBPTR*n* and RFBPTR*n* to determine the number of free BDs remaining in the ring. If, at any stage, the value written to RFBPTR*n* matches that of the respective RBPTR*n* the eTSEC free BD calculation assumes that the ring is now completely empty. For more information on the recommended use of these registers, see Section 13.6.6.1, “Back Pressure Determination through Free Buffers.”
Figure 13-105 describes the definition for the RFBPTR*n* register.

Offset eTSEC1:0x2_4C40+8×*n*; eTSEC2:0x2_5C40+8×*n*; eTSEC3:0x2_6C40+8×*n*; eTSEC4:0x2_7C40+8×*n* — Access: Read/Write

	0	28	29	31
R	RFBPTR*n*		—	
W				
Reset	All zeros			

Figure 13-105. RFBPTR0–RFBPTR7 Register Definition

Table 13-109 describes the fields of the RFBPTR*n* registers.

Table 13-109. RFBPTR0–RFBPTR7 Field Descriptions

Bits	Name	Description
0–28	RFBPTR	Pointer to the last free BD in RxBD Ring *n*. When RBASE*n* is updated, eTSEC initializes RFBPTR*n* to the value in the corresponding RBASE*n*. Software may update this register at any time to inform the eTSEC the location of the last free BD in the ring. Note that the 3 least-significant bits of this register are read only and zero.
29–31	—	Reserved.

13.5.4 Ten-Bit Interface (TBI)

This section describes the ten-bit interface (TBI) and the TBI MII set of registers.

13.5.4.1 TBI Transmit Process

The eTSEC's TBI implements the transmit portion of the physical coding sublayer as found in Clause 36 of IEEE 802.3z. In SerDes mode, packets conveyed across the GMII are encapsulated and encoded into 10-bit symbols and output to the SerDes. In GMII mode, the GMII signals are passed through to the attached GMII PHY.

13.5.4.1.1 Packet Encapsulation

If TX_EN is de-asserted the eTSEC outputs an idle stream. If TX_EN is asserted, a Start_of_Packet symbol is output. This symbol replaces the first byte of the preamble field. All other bytes of the packet pass through an 8B10B encoding module. After the last byte of the FCS field is signaled through the GMII, the MAC de-asserts TX_EN. The eTSEC then outputs an End_of_Packet symbol. Then, depending on the position of the End_of_Packet symbols (being in either an odd or even position) the eTSEC outputs one or two Carrier_Extend symbols. Following the last Carrier_Extend symbol, the eTSEC resumes sending idle codes. If, during a packet, the eTSEC wishes to mark a byte invalid, TX_ER is asserted. The eTSEC, upon detection of TX_ER, substitutes the data symbol for an Error_Propagation symbol.

13.5.4.1.2 8B10B Encoding

Every eight-bit data octet has two (not necessarily different) ten-bit symbols associated with it. Depending on the running disparity (the cumulative difference of ones and zeroes) the eTSEC module chooses the appropriate symbol.

Special encapsulation symbols are called ordered_sets. Ordered_sets are comprised of one to four ten-bit symbols. Ordered_sets can be found in Clause 36 of the IEEE 802.3z specification.

13.5.4.1.3 Preamble Shortening

Because the idle ordered_set comprises two symbols and begins on an even symbols boundary, packets can only begin on an even boundary. However, the GMII has no such restriction and may signal TX_EN on an odd boundary. If this happens, the eTSEC delays the Start_of_Packet symbol, effectively ignoring the first byte of preamble; thus, a seven octet preamble becomes six octets on the Ten-Bit Interface.

13.5.4.2 TBI Receive Process

The eTSEC's TBI Implements the receive portion of the physical coding sublayer as found in Clause 36 of IEEE 802.3z. The Receive portion includes the Synchronization state machine. In SerDes mode, the eTSEC first attempts to acquire synchronization on the link by examining received symbols. Once synchronization is acquired, received packets are decoded and sent across the Receive GMII interface. In GMII mode, the GMII signals are passed through to the MAC.

13.5.4.2.1 Synchronization

The eTSEC examines received symbols looking for the seven bit 'comma' string embedded in some special symbols. Both the idle ordered_set and the Configuration ordered_set contain a symbol which has the comma. Once a certain number of codes with comma are detected, the eTSEC is considered to have acquired synchronization.

13.5.4.2.2 Auto-Negotiation for 1000BASE-X

Once synchronization is acquired, ordered_sets are decoded. If Configuration ordered_sets are received, the eTSEC decodes the two octet data field and the sixteen-bit Configuration data is stored and used to Auto-Negotiate with the link partner. in the Receive Configuration Register (RXCR[15:0]) an internal register used to receive all the link partners informations and used to compare to local ability during negotiation. Not visible to user. If, during Auto-Negotiation an invalid symbol is detected, Auto-Negotiation re-starts. After Auto-Negotiation is completed the TBI MII Status Register SR[AN done] in set. In this mode, packets may be received from the link partner.

13.5.4.3 TBI MII Set Register Descriptions

This section describes the TBI MII registers. All of the TBI registers are 16 bits wide. The TBI registers are accessed at the offset of the TBI physical address. The eTSEC's TBI physical address is stored in the TBIPA register. Writing to the TBI registers is performed in a way similar to writing to an external PHY, using the MII management interface. Using TBIPA in place of the PHY address, in the MIIMADD[PHY Address] field, and setting the MIIMADD[Register Address] to the appropriate address offset that corresponds to the register that one wants to read or write (see Table 13-110), the user can read (set MIIMCOM[read cycle]) or write (writing to MIIMCON[PHY control]) to the TBI block. Refer to the TBI physical address register in Section 13.5.3.1, "eTSEC General Control and Status Registers," and the TBI MII register set in Table 13-110. Notice that jitter diagnostics and TBI control are not IEEE 802.3 required registers and are only used for test and control of the eTSEC TBI block. The TBI's TBI control register (TBI) is for configuring the eTSEC ten-bit interface block. However, because this TBI block has an MII management interface (just like any other PHY), it has an IEEE 802.3 register called the control register (CR).

Table 13-110. TBI MII Register Set

Offset Address	Name	Access	Size	Section/page
TEN-BIT INTERFACE (TBI) REGISTERS				13.5.4/13-113
0x00	Control (CR)	R/W[1]	16 bits	13.5.4.3.1/13-116
0x01	Status (SR)	R, LH, LL	16 bits	13.5.4.3.2/13-117
0x02–0x03	Reserved	R	2 bytes	—
0x04	AN advertisement (ANA)	RW, R	16 bits	13.5.4.3.2/13-117
0x05	AN link partner base page ability (ANLPBPA)	R	16 bits	13.5.4.3.4/13-120
0x06	AN expansion (ANEX)	R, LH	16 bits	13.5.4.3.5/13-121
0x07	AN next page transmit (ANNPT)	R/W, R	16 bits	13.5.4.3.6/13-121
0x08	AN link partner ability next page (ANLPANP)	R	16 bits	13.5.4.3.7/13-122
0x0F	Extended status (EXST)	R	16 bits	13.5.4.3.8/13-123
0x10	Jitter diagnostics (JD)	R/W	16 bits	13.5.4.3.9/13-124
0x11	TBI control (TBICON)	R/W	16 bits	13.5.4.3.10/13-125

[1] R = Read-only, WO = Write Only, R/W = Read and Write, LH = Latches High, LL = Latches Low, SC = Self-clearing,

13.5.4.3.1 Control Register (CR)

Figure 13-106 describes the definition for the CR register.

Offset 0x00 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	PHY Reset	—	Speed[0]	AN Enable	—		Reset AN	Full Duplex	—	Speed[1]	—					
Reset	0	0	0	1	0	0	0	1	0	1	0	0	0	0	0	0

Figure 13-106. Control Register Definition

Table 13-111 describes the fields of the CR register.

Table 13-111. CR Field Descriptions

Bits	Name	Description
0	PHY Reset	PHY reset. This bit is cleared by default. This bit is self-clearing. 0 Normal operation. 1 The internal state of the TBI is reset. This in turn may change the state of the TBI link partner.
1	—	Reserved
2	Speed[0]	Speed selection. This bit defaults to a cleared state and should always be cleared, which corresponds to 1000 Mbps speed.Setting this field controls the speed at which the TBI operates. The table for Speed[1] provides the appropriate encoding. Its default is bit[2] = ‘0’; bit[9] = ’1’.
3	AN Enable	Auto-negotiation enable. This bit is set by default. 0 The values programmed in bits 2, 7 and 9 determine the operating condition of the link. 1 Auto-negotiation process enabled.
4–5	—	Reserved
6	Reset AN	Reset auto-negotiation. This bit is cleared by default and is self-clearing. 0 Normal operation. 1 The auto-negotiation process restarts. This action is only available if auto-negotiation is enabled.
7	Full Duplex	Duplex mode. This bit is set by default. 0 Reserved. 1 Full-duplex operation.
8	—	Reserved, should be cleared.
9	Speed[1]	Speed selection. This bit defaults to a set state and should always be set, which corresponds to 1000 Mbps speed.Setting this field controls the speed at which the TBI operates. The following table provides the appropriate encoding. Its default is bit[2] = ‘0’; bit[9] = ’1’.
10–15	—	Reserved

Maximum Operating Speed	Bit 2	Bit 9
Reserved	0	0
Reserved	1	0
1000 Mbps	0	1
Reserved	1	1

13.5.4.3.2 Status Register (SR)

Figure 13-107 describes the definition for the SR register.

Offset 0x01 Access: Read only

	0	6	7	8	9	10	11	12	13	14	15
R	—		Extend Status	—	No Pre	AN Done	Remote Fault	AN Ability	Link Status	—	Extend Ability
W											
Reset	0 0 0 0	0 0 0	1	0	1	0	0	1	0	0	1

Figure 13-107. Status Register Definition

Table 13-112 describes the fields of the SR register.

Table 13-112. SR Descriptions

Bits	Name	Description
0–6	—	Reserved, should be cleared.
7	Extend Status	This bit indicates that PHY status information is also contained in the Register 15, Extended Status Register. Returns 1 on read. This bit is read-only.
8	—	Reserved, should be cleared.
9	No Pre	MF preamble suppression enable. This bit indicates whether or not the PHY is capable of handling MII management frames without the 32-bit preamble field. Returns 1, indicating support for suppressed preamble MII management frames. This bit is read-only.
10	AN Done	Auto-negotiation complete. This bit is read-only and is cleared by default. 0 Either the auto-negotiation process is underway or the auto-negotiation function is disabled. 1 The auto-negotiation process has completed.
11	Remote Fault	Remote fault. This bit is read-only and is cleared by default. Each read of the status register clears this bit. 0 Normal operation. 1 A remote fault condition was detected. This bit latches high in order for software to detect the condition.
12	AN Ability	Auto-negotiation ability. While read as set, this bit indicates that the PHY has the ability to perform auto-negotiation. While read as cleared, this bit indicates the PHY lacks the ability to perform auto-negotiation. Returns 1 on read. This bit is read-only.
13	Link Status	Link status. This bit is read-only and is cleared by default. 0 A valid link is not established. This bit latches low allowing for software polling to detect a failure condition. 1 A valid link is established.
14	—	Reserved, should be cleared.
15	Extend Ability	Extended capability. This bit indicates that the PHY contains the extended set of registers (those beyond control and status). Returns 1 on read. This bit is read-only.

13.5.4.3.3 AN Advertisement Register (ANA)

Figure 13-108 describes the definition for the ANA register.

Offset 0x04 Access: Read/Write

	0	1	2 3	4 6	7	8 9	10	11 15
R W	Next Page	—	Remote Fault	—	Pause	Half Duplex	Full Duplex	—
Reset	All zeros							

Figure 13-108. AN Advertisement Register Definition

Table 13-113 describes the fields of the ANA register.

Table 13-113. ANA Field Descriptions

Bits	Name	Description
0	Next Page	Next page configuration. The local device sets this bit to either request next page transmission or advertise next page exchange capability. 0 The local device wishes not to engage in next page exchange. 1 The local device has no next pages but wishes to allow reception of next pages. If the local device has no next pages and the link partner wishes to send next pages, the local device shall send null message codes and have the message page set to 0b000_0000_0001, as defined in annex 28C.
1	—	Reserved. (Ignore on read)
2–3	Remote Fault	The local device's remote fault condition is encoded in bits 2 and 3 of the base page. Values are shown in the following table. The default value is 00. Indicate a fault by setting a non-zero remote fault encoding and re-negotiating. **RF1 bit[3] / RF2 bit[2] / Description** 0 / 0 / No error, link OK 0 / 1 / Offline 1 / 0 / Link_Failure 1 / 1 / Auto-Negotiation_Error
4–6	—	Reserved, should be cleared.
7–8	Pause	The local device's PAUSE capability is encoded in bits 7 and 8, and the decodes are shown in the following table. For priority resolution information consult Table 13-114. **PAUSE bit[8] / ASM_DIR bit[7] / Capability** 0 / 0 / No PAUSE 0 / 1 / Asymmetric PAUSE toward link partner 1 / 0 / Symmetric PAUSE 1 / 1 / Both symmetric PAUSE and Asymmetric PAUSE toward local device

Table 13-113. ANA Field Descriptions (continued)

Bits	Name	Description
9	Half Duplex	Half-duplex capability. 0 Designates local device as not capable of half-duplex operation. 1 Designates local device as capable of half-duplex operation.
10	Full Duplex	Full-duplex capability. 0 Designates the local device as not capable of full-duplex operation. 1 Designates the local device as capable of full-duplex operation.
11–15	—	Reserved, should be cleared.

Table 13-114 describes the resolution of pause priority.

Table 13-114. PAUSE Priority Resolution

Local Device		Link Partner		Local Resolution	Link Partner Resolution
PAUSE	ASM_DIR	PAUSE	ASM_DIR		
0	0	x	x	Disable PAUSE transmit Disable PAUSE receive	Disable PAUSE transmit Disable PAUSE receive
0	1	0	x	Disable PAUSE transmit Disable PAUSE receive	Disable PAUSE transmit Disable PAUSE receive
0	1	1	0	Disable PAUSE transmit Disable PAUSE receive	Disable PAUSE transmit Disable PAUSE receive
0	1	1	1	Enable PAUSE transmit Disable PAUSE receive	Disable PAUSE transmit Enable PAUSE receive
1	0	0	x	Disable PAUSE transmit Disable PAUSE receive	Disable PAUSE transmit Disable PAUSE receive
1	0	1	x	Enable PAUSE transmit Enable PAUSE receive	Enable PAUSE transmit Enable PAUSE receive
1	1	0	0	Disable PAUSE transmit Disable PAUSE receive	Disable PAUSE transmit Disable PAUSE receive
1	1	0	1	Disable PAUSE transmit Enable PAUSE receive	Enable PAUSE transmit Disable PAUSE receive
1	1	1	x	Enable PAUSE transmit Enable PAUSE receive	Enable PAUSE transmit Enable PAUSE receive

13.5.4.3.4 AN Link Partner Base Page Ability Register (ANLPBPA)

Figure 13-109 describes the definition for the ANLPBPA register.

Offset 0x05 Access: Read only

	0	1	2 3	4 6	7	8	9	10	11 15
R	Next Page	—	Remote Fault	—		Pause	Half Duplex	Full Duplex	—
W									
Reset	All zeros								

Figure 13-109. AN Link Partner Base Page Ability Register Definition

Table 13-115 describes the fields of the ANLPBPA register.

Table 13-115. ANLPBPA Field Descriptions

Bits	Name	Description
0	Next Page	Next page. This bit is read-only. The link partner sets or clears this bit. 0 Link partner has no subsequent next pages or is not capable of receiving next pages. 1 Link partner either requesting next page transmission or indicating the capability to receive next pages.
1	—	Reserved. (Ignore on read)
2–3	Remote Fault	The link partner's remote fault condition is encoded in bits 2 and 3 of the base page. Values are shown in the remote fault encoding field table below. This bit is read-only.
4–6	—	Reserved, should be cleared.
7–8	Pause	Encoding of the link partner's PAUSE capability is shown in the PAUSE encoding table below. For priority resolution information consult. This bit is read-only
9	Half Duplex	Half-duplex capability. This bit is read-only. 0 Link partner is not capable of half-duplex mode. 1 Link partner is capable of half-duplex mode.

Remote Fault encoding:

RF1 bit[3]	RF2 bit[2]	Description
0	0	No error, link OK
0	1	Offline
1	0	Link_Failure
1	1	Auto-Negotiation_Error

PAUSE encoding:

PAUSE bit[8]	ASM_DIR bit[7]	Capability
0	0	No PAUSE
0	1	Asymmetric PAUSE toward link partner
1	0	Symmetric PAUSE
1	1	Both symmetric PAUSE and Asymmetric PAUSE toward local device

Table 13-115. ANLPBPA Field Descriptions (continued)

Bits	Name	Description
10	Full Duplex	Full-duplex capability. This bit is read-only. 0 Link partner is not capable of full-duplex mode. 1 Link partner is capable of full-duplex mode.
11–15	—	Reserved, should be cleared.

13.5.4.3.5 AN Expansion Register (ANEX)

Figure 13-110 describes the definition for the ANEX register.

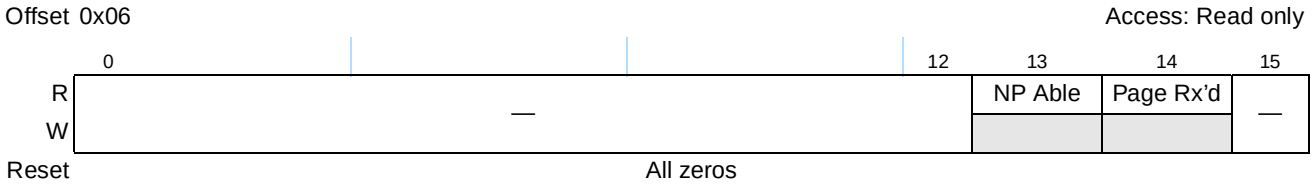


Figure 13-110. AN Expansion Register Definition

Table 13-116 describes the fields of the ANEX register.

Table 13-116. ANEX Field Descriptions

Bits	Name	Description
0–12	—	Reserved, should be cleared.
13	NP Able	Next page able. This bit is read-only and returns 1 on read. While read as set, indicates local device supports next page function.
14	Page Rx'd	Page received. This bit is read-only. The bit clears on a read to the register. 0 Normal operation. 1 A new page was received and stored in the applicable AN link partner ability or AN next page register. This bit latches high in order for software to detect while polling.
15	—	Reserved, should be cleared.

13.5.4.3.6 AN Next Page Transmit Register (ANNPT)

Figure 13-111 describes the definition for the ANNPT register.

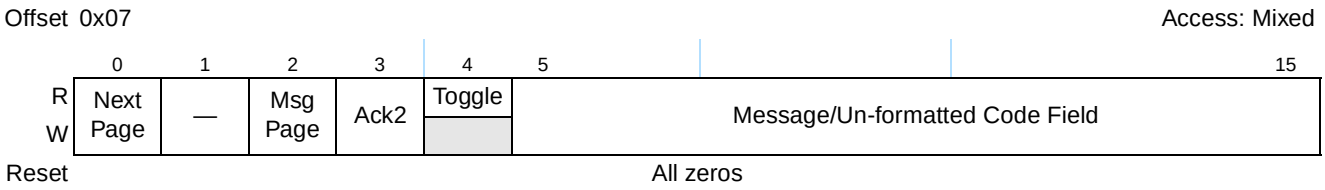


Figure 13-111. AN Next Page Transmit Register Definition

Table 13-117 describes the fields of the ANNPT register.

Table 13-117. ANNPT Field Descriptions

Bits	Name	Description
0	Next Page	Next page indication. [Reference MII bit 7.15 in IEEE 802.3, 2000 Edition Clause 28.2.4] 0 Last page. 1 Additional next pages to follow.
1	—	Reserved. (Ignore on read)
2	Msg Page	Message page. [Reference MII bit 7.13] 0 Unformatted page. 1 Message page.
3	Ack2	Acknowledge 2. Used by the next page function to indicate that the device has the ability to comply with the message. [Reference MII bit 7.12] 0 The local device cannot comply with message. 1 The local device complies with message.
4	Toggle	Toggle. Used to ensure synchronization with the link partner during next page exchange. This bit always takes the opposite value of the toggle bit of the previously-exchanged link code word. The initial value in the first next page transmitted is the inverse of bit 11 in the base link code word. [Reference MII bit 7.11] This bit is read-only. 0 Toggle bit of the previously-exchanged link code word was 1. 1 Toggle bit of the previously-exchanged link code word was 0.
5–15	Message/ Un-formatted Code Field	Message pages are formatted pages that carry a pre-defined message code, which is enumerated in IEEE 802.3u/Annex 28C. Unformatted code fields take on an arbitrary value. [Reference MII field 7.10:0]

13.5.4.3.7 AN Link Partner Ability Next Page Register (ANLPANP)

Figure 13-112 describes the definition for the ANLPANP register.

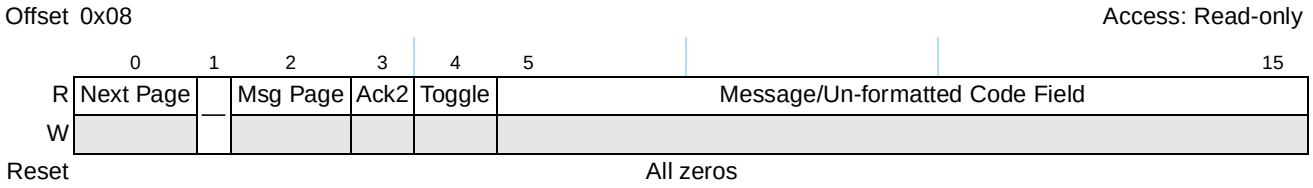


Figure 13-112. AN Link Partner Ability Next Page Register Definition

Table 13-118 describes the fields of the ANLPANP register.

Table 13-118. ANLPANP Field Descriptions

Bits	Name	Description
0	Next Page	Next page. The link partner sets and clears this bit. 0 Last page from link partner 1 Additional next pages to follow
1	—	Reserved. (Ignore on read)
2	Msg Page	Message page. 0 Unformatted page 1 Message page

Table 13-118. ANLPANP Field Descriptions (continued)

Bits	Name	Description
3	Ack2	Acknowledge 2. Indicates the link partner's ability to comply with the message. 0 Link partner cannot comply with message. 1 Link partner complies with message.
4	Toggle	Toggle. Used to ensure synchronization with the link partner during next page exchange. This bit always takes the opposite value of the toggle bit of the previously-exchanged link code word. The initial value in the first next page transmitted is the inverse of bit 11 in the base link code word. This bit is read-only. 0 Toggle bit of the previously-exchanged link code word was 1. 1 Toggle bit of the previously-exchanged link code word was 0.
5–15	Message/ Un-formatted Code Field	Message pages are formatted pages that carry a pre-defined message code, which is enumerated in IEEE 802.3u/Annex 28C. Unformatted code fields take on an arbitrary value.

13.5.4.3.8 Extended Status Register (EXST)

Figure 13-113 describes the definition for the EXST register.

Offset 0x0F Access: Read only

	0	1	2	3	4								15
R	1000X Full	1000X Half	1000T Full	1000T Half	—								
W													
Reset	1	0	1	0	0	0	0	0	0	0	0	0	0

Figure 13-113. Extended Status Register Definition

Table 13-119 describes the fields of the EXST register.

Table 13-119. EXST Field Descriptions

Bits	Name	Description
0	1000X Full	1000X full-duplex capability. Returns 1 on read. This bit is read-only. 0 PHY cannot operation in 1000BASE-X full-duplex mode. 1 PHY can operate in 1000BASE-X full-duplex mode.
1	1000X Half	1000X half-duplex capability. Returns 0 on read. This bit is read-only. 0 PHY cannot operation in 1000BASE-X half-duplex mode. 1 PHY can operate in 1000BASE-X half-duplex mode.
2	1000T Full	1000T full-duplex capability. Returns 1 on read. This bit is read-only. 0 PHY cannot operation in 1000BASE-T full-duplex mode. 1 PHY can operate in 1000BASE-T full-duplex mode.
3	1000T Half	1000T half-duplex capability. Returns 0 on read. This bit is read-only. 0 PHY cannot operation in 1000BASE-T half-duplex mode. 1 PHY can operate in 1000BASE-T half-duplex mode.
4–15	—	Reserved

13.5.4.3.9 Jitter Diagnostics Register (JD)

Annex 36A in IEEE 802.3z describes several jitter test patterns. These can be configured to be sent by writing the jitter diagnostics register. See the register description for more information. In may be wise to auto-negotiate and advertise a remote fault signaling of offline prior to beginning the test patterns. Figure 13-114 describes the definition for the JD register.

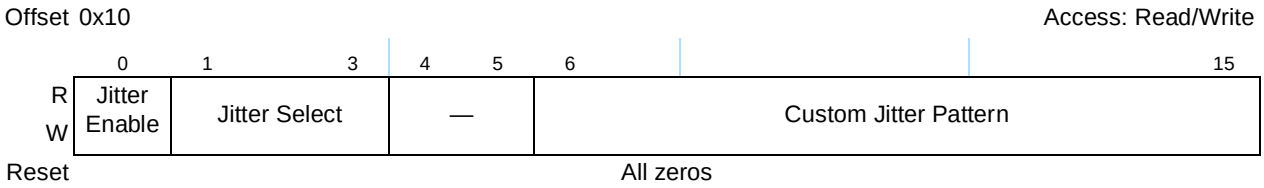


Figure 13-114. Jitter Diagnostics Register Definition

Table 13-120 describes the fields of the JD register.

Table 13-120. JD Field Descriptions

Bits	Name	Description
0	Jitter Enable	Jitter enable. This bit is cleared by default. 0 Normal transmit operation. 1 Enable the TBI to transmit the jitter test patterns defined in IEEE 802.3z 36A.
1–3	Jitter Select	Selects the jitter pattern to be transmitted in diagnostics mode. Encoding of this field is shown in the following table. Default is 00.

Jitter Pattern Select	bit[1]	bit[2]	bit[3]
User defined uses custom jitter pattern, bits 6–15	0	0	0
High frequency (+/- D21.5) 1010101010101010101010101010101010101010…	0	0	1
Mixed frequency (+/- K28.5) 1111101011000001010011111010110000010100…	0	1	0
Low frequency 1111100000111110000011111000001111100000…	0	1	1
Complex pattern (10’h17c,10’h0c9,10’h0e5,10’h2a3, 10’h17c,...)	1	0	0
Square Wave (- K28.7) 0011111000001111100000111110000011111000…	1	0	1
Reserved	1	1	0
Reserved	1	1	1

4–5	—	Reserved
6–15	Custom Jitter Pattern	Used in conjunction with jitter (pattern) select and jitter (diagnostic) enable; set this field to the desired custom pattern which is continuously transmitted. Its default is 0x000.

13.5.4.3.10 TBI Control Register (TBICON)

Figure 13-115 describes the definition for the TBICON register.

Offset 0x11 Access: Mixed

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Soft_Reset	—	Disable Rx Dis	Disable Tx Dis	—			AN Sense	—		Clock Select	MII Mode	—			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Figure 13-115. TBI Control Register Definition

Table 13-121 describes the fields of the TBICON register.

Table 13-121. TBICON Field Descriptions

Bits	Name	Description
0	Soft_Reset	Soft reset. This bit is cleared by default. 0 Normal operation. 1 Resets the functional modules in the TBI.
1	—	Reserved. (Ignore on read)
2	Disable Rx Dis	Disable receive disparity. This bit is cleared by default. 0 Normal operation. 1 Disables the running disparity calculation and checking in the receive direction.
3	Disable Tx Dis	Disable transmit disparity. This bit is cleared by default. 0 Normal operation. 1 Disables the running disparity calculation and checking in the transmit direction.
4–6	—	Reserved
7	AN Sense	Auto-negotiation sense enable. This bit is cleared by default. 0 IEEE 802.3z Clause 37 behavior is desired, which results in the link not completing. 1 Allow the auto-negotiation function to sense either a Gigabit MAC in auto-negotiation bypass mode or an older Gigabit MAC without auto-negotiation capability. If sensed, auto-negotiation complete becomes true; however, the page received is low, indicating no page was exchanged. Management can then act accordingly.
8–9	—	Reserved
10	Clock Select	Clock select. This bit is cleared by default. 0 Allow the TBI to accept dual split-phase 62.5 MHz receive clocks. 1 Configure the TBI to accept a 125 MHz receive clock from the SerDes/PHY. The 125 MHz clock must be physically connected to ‘PMA receive clock 0’ if using a parallel (non-SGMII) Ethernet protocol.
11	MI Mode	This bit describes the configuration mode of the TBI. The user reads a 1 while the TBI is configured in GMII/MII mode (connected to a GMII/MII PHY) and a 0 while configured in TBI mode (connected to a 1000BASE-X SerDes). Its value is the inverse of ECNTRL[TBIM]. 0 TBI mode. 1 GMII mode.
12–15	—	Reserved

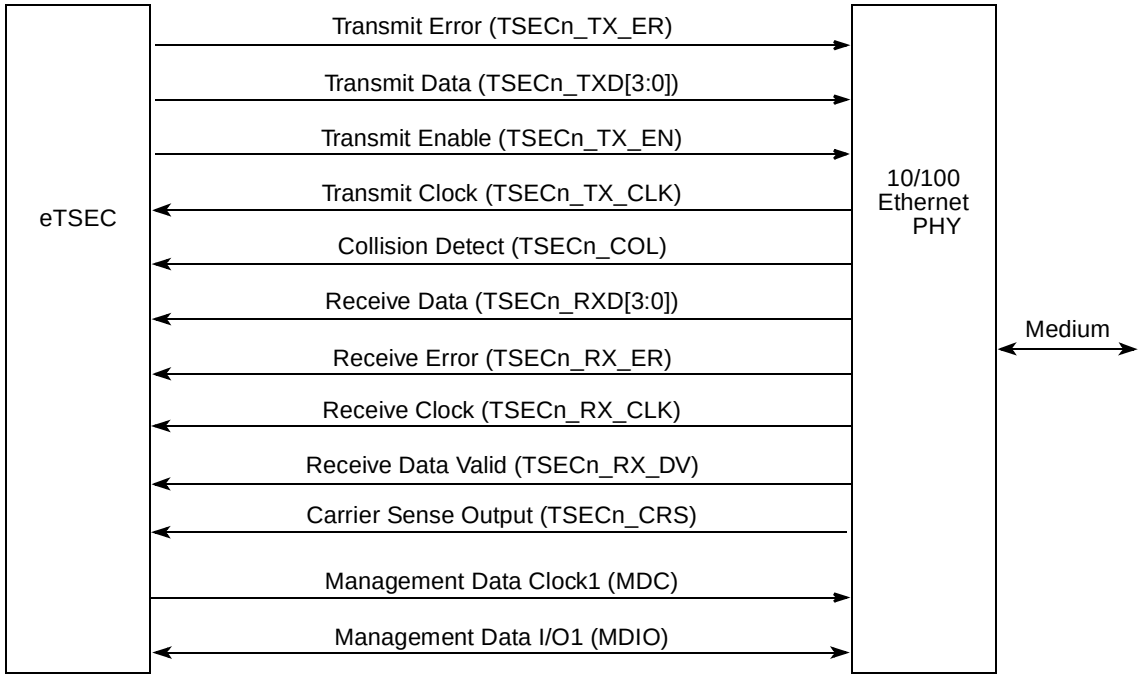
13.6 Functional Description

13.6.1 Connecting to Physical Interfaces on Ethernet

This section describes how to connect the eTSEC to various interfaces: MII, GMII, RMII, RGMII, TBI, and RTBI. To avoid confusion, all of the buses follow the bus conventions used in the IEEE 802.3 specification because the PHYs follow the same conventions. (For instance, in the bus TSEC*n*_TXD[7:0], bit 7 is the msb and bit 0 is the lsb). If a mode does not use all input signals available to a particular eTSEC, those inputs that are not used must be pulled low on the board.

13.6.1.1 Media-Independent Interface (MII)

This section describes the media-independent interface (MII) intended to be used between the PHYs and the eTSEC. Figure 13-116 depicts the basic components of the MII including the signals required to establish eTSEC module connection with a PHY.



1 The management signals (MDC and MDIO) are common to all of the Ethernet controllers' connections in the system, assuming that each PHY has a different management address.

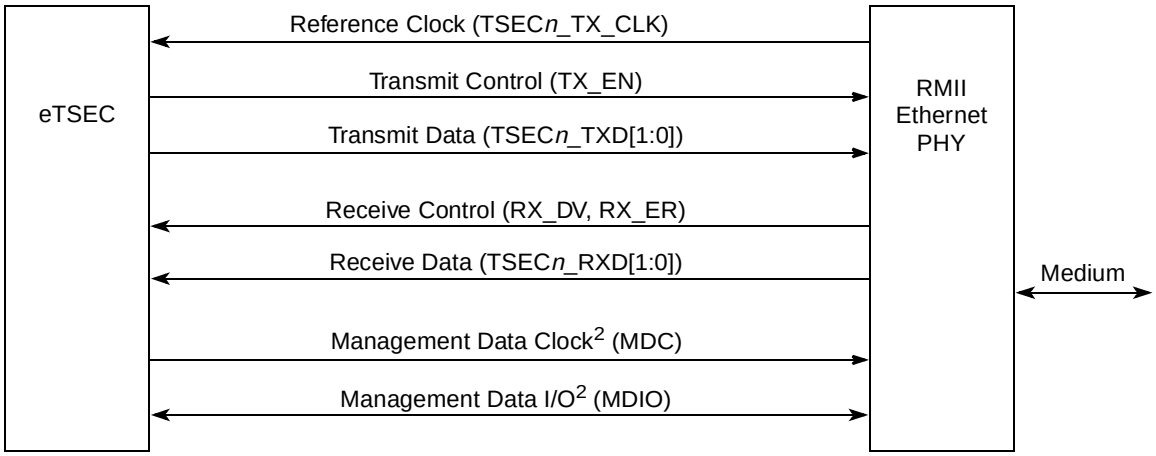
Figure 13-116. eTSEC-MII Connection

An MII interface has 18 signals (including the MDC and MDIO signals), as defined by the IEEE 802.3u standard, for connecting to an Ethernet PHY.

13.6.1.2 Reduced Media-Independent Interface (RMII)

This section describes the reduced media-independent interface (RMII) intended to be used between the PHYs and the GMII MAC. The RMII is a reduced-pin alternative to the IEEE802.3u MII. The RMII reduces the number of signals required to interconnect the MAC and the PHY from a maximum of 18

signals (MII) to 10 signals. To accomplish this objective, the data paths are halved in width and clocked at twice the MII clock frequency, while clocks, carrier sense and error signals have been partly combined. For 100 Mbps operation, the reference clock operates at 50 MHz, whereas for 10 Mbps operation, the clock remains at 50MHz, but only every 10th cycle is used. Figure 13-117 depicts the basic components of the reduced media-independent interface and the signals required to establish an eTSEC's connection with a PHY. The RMII is implemented as defined by the RMII Specification of the RMII Consortium, as of March 20, 1998.

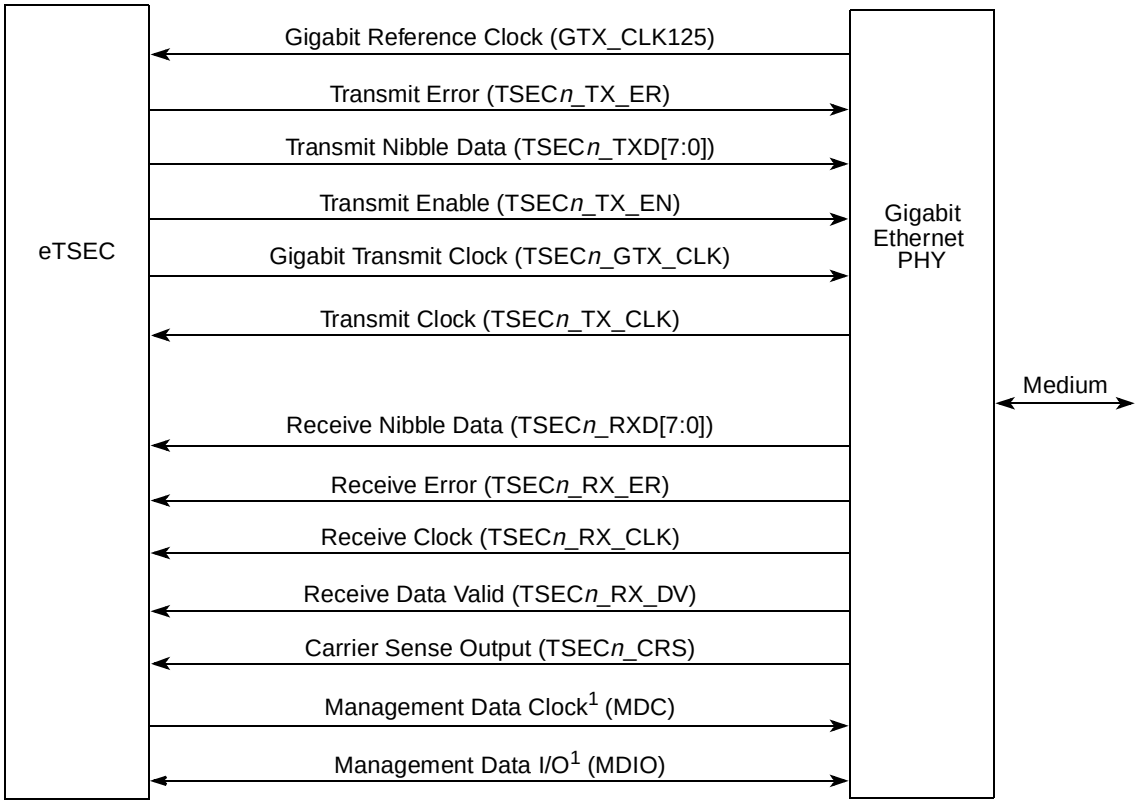


2 The management signals (MDC and MDIO) are common to all of the Ethernet controllers module connections in the system, assuming that each PHY has a different management address.

Figure 13-117. eTSEC-RMII Connection

13.6.1.3 Gigabit Media-Independent Interface (GMII)

This section describes the gigabit media-independent interface (GMII) intended to be used between the PHYs and the eTSEC. Figure 13-118 depicts the basic components of the GMII including the signals required to establish the eTSEC module connection with a PHY.



1 The management signals (MDC and MDIO) are common to all of the Ethernet controllers' connections in the system, assuming that each PHY has a different management address.

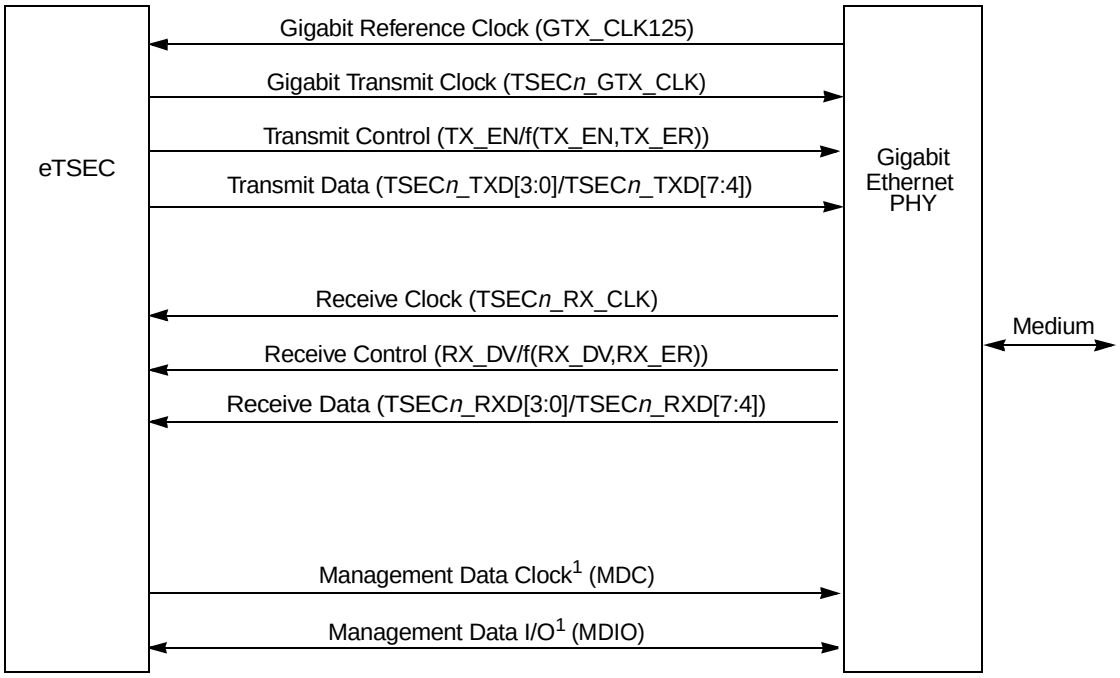
Figure 13-118. eTSEC-GMII Connection

A GMII interface has 28 signals (TSEC*n*_GTX_CLK + EC_GTX_CLK125 included), as defined by the IEEE 802.3u standard, for connecting to an Ethernet PHY.

13.6.1.4 Reduced Gigabit Media-Independent Interface (RGMII)

This section describes the reduced gigabit media-independent interface (RGMII) intended to be used between the PHYs and the GMII MAC. The RGMII is an alternative to the 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 (GTX_CLK125 included) in a cost effective and technology independent manner. To accomplish this objective, the data paths and all associated control signals are multiplexed using both edges of the clock. For gigabit operation, the clocks operate at 125MHz, and for 10/100 operation, the clocks operate at 2.5 MHz or 25 MHz, respectively. Note that the GTX_CLK125 input must be provided at 125 MHz for an RGMII interface, regardless of operation speed (1 Gbps, 100 Mbps, or 10 Mbps). Figure 13-119 depicts the basic components of the gigabit reduced media-independent interface and the signals required to establish the gigabit Ethernet controllers' module

connection with a PHY. The RGMII is implemented as defined by the RGMII specification Version 1.2a 9/22/00.

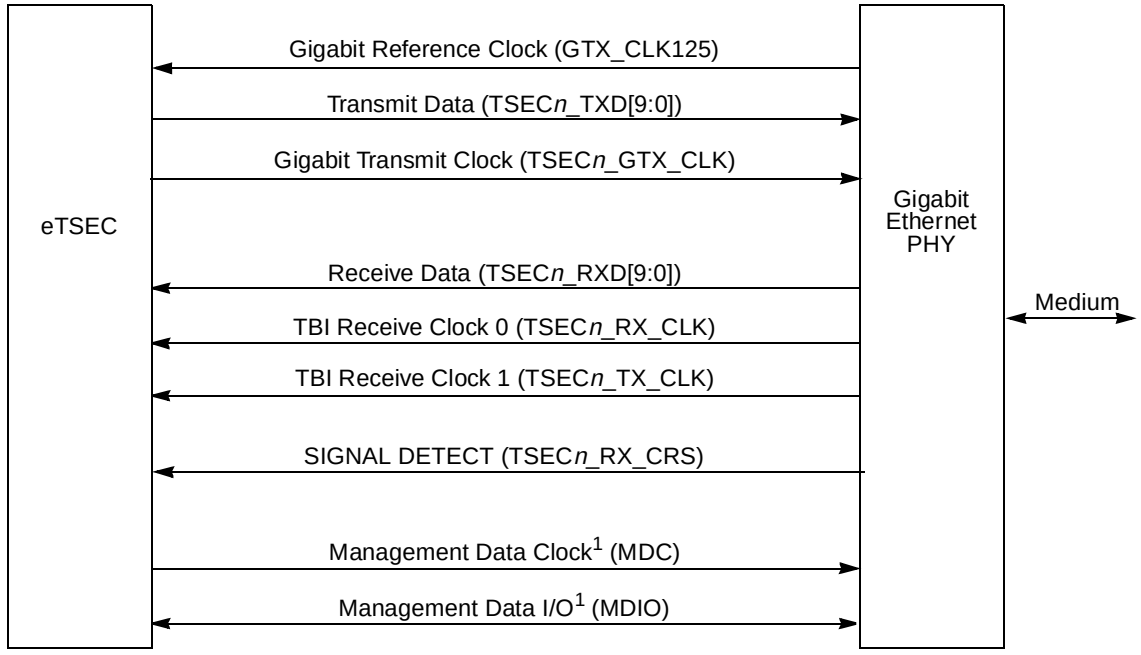


1 The management signals (MDC and MDIO) are common to all of the gigabit Ethernet controllers module connections in the system, assuming that each PHY has a different management address.

Figure 13-119. eTSEC-RGMII Connection

13.6.1.5 Ten-Bit Interface (TBI)

This section describes the ten-bit interface (TBI) intended to be used between the PHYs and the eTSEC to implement a standard SerDes interface for optical-fiber devices in 1000BASE-SX/LX applications. Figure 13-120 depicts the basic components of the TBI including the signals required to establish eTSEC module connection with a PHY. RBC0 and RBC1 are differential 62.5 MHz receive clocks. If not connected to the TBI PHY, the Signal Detect (SDET) input must be tied high. This causes the eTSEC to begin auto negotiation with the SERDES immediately upon the TBI module being enabled.



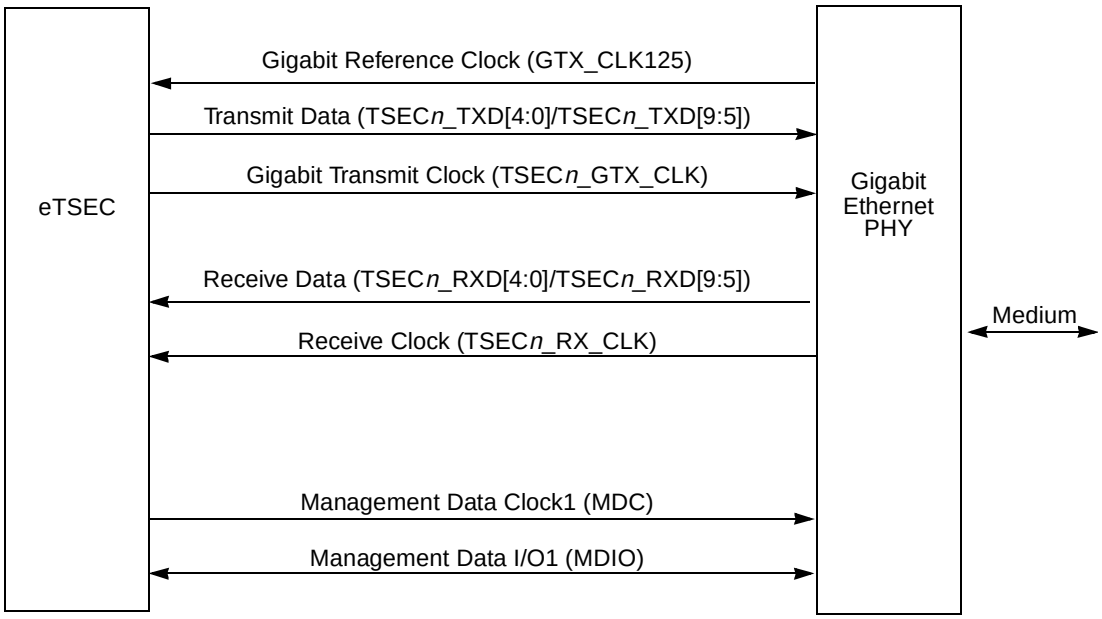
1 The management signals (MDC and MDIO) are common to all of the Ethernet controllers' connections in the system, assuming that each PHY has a different management address.

Figure 13-120. eTSEC-TBI Connection

A TBI interface has 26 signals (GE_GTX_CLK125 included) for connecting to an Ethernet PHY, as defined by IEEE 802.3z GMII and TBI standards.

13.6.1.6 Reduced Ten-Bit Interface (RTBI)

This section describes the reduced ten-bit interface (RTBI) intended to be used between the PHYs and the eTSEC to implement a reduced-pin count version of a SerDes interface for optical-fiber devices in 1000BASE-SX/LX applications. Figure 13-121 depicts the basic components of the RTBI including the signals required to establish eTSEC module connection with a PHY. Note that in RTBI the eTSEC immediately begins auto-negotiation with the SerDes.



1 The management signals (MDC and MDIO) are common to all of the Ethernet controllers' connections in the system, assuming that each PHY has a different management address.

Figure 13-121. eTSEC-RTBI Connection

A RTBI interface has 15 signals (GE_GTX_CLK125 included), as defined by the RGMII specification Version 1.2a 9/22/00, and is intended to be an alternative to the IEEE 802.3u MII, the IEEE 802.3z GMII and the TBI standard for connecting to an Ethernet PHY.

13.6.1.7 Ethernet Physical Interfaces Signal Summary

Table 13-122 describes the signal multiplexing for the following interfaces: GMII, MII, and RMII.

Table 13-122. GMII, MII, and RMII Signals Multiplexing

eTSEC Signals			GMII Interface			MII Interface			RMII Interface		
Frequency [MHz] 125			Frequency [MHz] 125			Frequency [MHz] 25			Frequency [MHz] 50		
Voltage[V] 3.3/2.5			Voltage[V] 3.3			Voltage[V] 3.3			Voltage[V] 3.3		
Signals (TSEC*n*_)	I/O	No. of Signals	Signals (TSEC*n*_)	I/O	No. of Signals	Signals (TSEC*n*_)	I/O	No. of Signals	Signals (TSEC*n*_)	I/O	No. of Signals
GTX_CLK	O	1	GTX_CLK	O	1	—	—	—	—	—	—
TX_CLK	I	1	TX_CLK	I	1	TX_CLK	I	1	REF_CLK	I	1
TxD[0]	O	1	TxD[0]	O	1	TxD[0]	O	1	TxD[0]	O	1
TxD[1]	O	1	TxD[1]	O	1	TxD[1]	O	1	TxD[1]	O	1
TxD[2]	O	1	TxD[2]	O	1	TxD[2]	O	1	—	—	—
TxD[3]	O	1	TxD[3]	O	1	TxD[3]	O	1	—	—	—
TxD[4]	O	1	TxD[4]	O	1	—	—	—	—	—	—
TxD[5]	O	1	TxD[5]	O	1	—	—	—	—	—	—
TxD[6]	O	1	TxD[6]	O	1	—	—	—	—	—	—
TxD[7]	O	1	TxD[7]	O	1	—	—	—	—	—	—
TX_EN	O	1	TX_EN	O	1	TX_EN	O	1	TX_EN	O	1
TX_ER	O	1	TX_ER	O	1	TX_ER	O	1	—	—	—
RX_CLK	I	1	RX_CLK	I	1	RX_CLK	I	1	—	—	—
RxD[0]	I	1	RxD[0]	I	1	RxD[0]	I	1	RxD[0]	I	1
RxD[1]	I	1	RxD[1]	I	1	RxD[1]	I	1	RxD[1]	I	1
RxD[2]	I	1	RxD[2]	I	1	RxD[2]	I	1	—	—	—
RxD[3]	I	1	RxD[3]	I	1	RxD[3]	I	1	—	—	—
RxD[4]	I	1	RxD[4]	I	1	—	—	—	—	—	—
RxD[5]	I	1	RxD[5]	I	1	—	—	—	—	—	—
RxD[6]	I	1	RxD[6]	I	1	—	—	—	—	—	—
RxD[7]	I	1	RxD[7]	I	1	—	—	—	—	—	—
RX_DV	I	1	RX_DV	I	1	RX_DV	I	1	CRS_DV	I	1
RX_ER	I	1	RX_ER	I	1	RX_ER	I	1	RX_ER	I	1
COL	I	1	—	—	—	COL	I	1	—	—	—
CRS	I	1	—	—	—	CRS	I	1	—	—	—
Sum		25	**Sum**		23	**Sum**		16	**Sum**		8

Table 13-123 describes the signal multiplexing for RGMII, TBI, and RTBI interfaces.

Table 13-123. RGMII, TBI, and RTBI Signals Multiplexing

eTSEC Signals			RGMII Interface			TBI Interface			RTBI Interface		
Frequency [MHz] 125			Frequency [MHz] 125			Frequency [MHz] 62.5			Frequency [MHz] 62.5		
Voltage[V] 3.3/2.5			Voltage[V] 2.5			Voltage[V] 3.3			Voltage[V] 2.5		
Signals (TSEC*n*_)	I/O	No. of Signals	Signals (TSEC*n*_)	I/O	No. of Signals	Signals (TSEC*n*_)	I/O	No. of Signals	Signals (TSEC*n*_)	I/O	No. of Signals
GTX_CLK	O	1	GTX_CLK	O	1	GTX_CLK	O	1	GTX_CLK	O	1
TX_CLK	I	1	—	—	—	RX_CLK1	I	1	—	—	—
TxD[0]	O	1	TxD[0]/TxD[4]	O	1	TCG[0]	O	1	TCG[0]/TCG[5]	O	1
TxD[1]	O	1	TxD[1]/TxD[5]	O	1	TCG[1]	O	1	TCG[1]/TCG[6]	O	1
TxD[2]	O	1	TxD[2]/TxD[6]	O	1	TCG[2]	O	1	TCG[2]/TCG[7]	O	1
TxD[3]	O	1	TxD[3]/TxD[7]	O	1	TCG[3]	O	1	TCG[3]/TCG[8]	O	1
TxD[4]	O	1	—	—	—	TCG[4]	O	1	—	—	—
TxD[5]	O	1	—	—	—	TCG[5]	O	1	—	—	—
TxD[6]	O	1	—	—	—	TCG[6]	O	1	—	—	—
TxD[7]	O	1	—	—	—	TCG[7]	O	1	—	—	—
TX_EN	O	1	TX_CTL (TX_EN/ TX_ERR)	O	1	TCG[8]	O	1	TCG[4]/TCG[9]	O	1
TX_ER	O	1	—	—	—	TCG[9]	O	1	—	—	—
RX_CLK	I	1	RX_CLK	I	1	RX_CLK0	I	1	RX_CLK	I	1
RxD[0]	I	1	RxD[0]/RxD[4]	I	1	RCG[0]	I	1	RCG[0]/RCG[5]	I	1
RxD[1]	I	1	RxD[1]/RxD[5]	I	1	RCG[1]	I	1	RCG[1]/RCG[6]	I	1
RxD[2]	I	1	RxD[2]/RxD[6]	I	1	RCG[2]	I	1	RCG[2]/RCG[7]	I	1
RxD[3]	I	1	RxD[3]/RxD[7]	I	1	RCG[3]	I	1	RCG[3]/RCG[8]	I	1
RxD[4]	I	1	—	—	—	RCG[4]	I	1	—	—	—
RxD[5]	I	1	—	—	—	RCG[5]	I	1	—	—	—
RxD[6]	I	1	—	—	—	RCG[6]	I	1	—	—	—
RxD[7]	I	1	—	—	—	RCG[7]	I	1	—	—	—
RX_DV	I	1	RX_CTL (RX_DV/ RX_ERR)	I	1	RCG[8]	I	1	RCG[4]/RCG[9]	I	1
RX_ER	I	1	—	—	—	RCG[9]	I	1	—	—	—
COL	I	1	—	—	—	—	—	—	—	—	—
CRS	I	1	—	—	—	SDET	I	1	—	I	—
Sum		25	**Sum**		12	**Sum**		24	**Sum**		12

Table 13-124. RGMII and RTBI Signals Multiplexing

eTSEC Signals			RGMII Interface			RTBI Interface		
Frequency [MHz] 125			Frequency [MHz] 125			Frequency [MHz] 62.5		
Voltage[V] 3.3/2.5			Voltage[V] 2.5			Voltage[V] 2.5		
Signals (TSEC*n*_)	I/O	No. of Signals	Signals (TSEC*n*_)	I/O	No. of Signals	Signals (TSEC*n*_)	I/O	No. of Signals
GTX_CLK	O	1	GTX_CLK	O	1	GTX_CLK	O	1
TX_CLK	I	1	—	—	—	—	—	—
TxD[0]	O	1	TxD[0]/TxD[4]	O	1	TCG[0]/TCG[5]	O	1
TxD[1]	O	1	TxD[1]/TxD[5]	O	1	TCG[1]/TCG[6]	O	1
TxD[2]	O	1	TxD[2]/TxD[6]	O	1	TCG[2]/TCG[7]	O	1
TxD[3]	O	1	TxD[3]/TxD[7]	O	1	TCG[3]/TCG[8]	O	1
TxD[4]	O	1	—	—	—	—	—	—
TxD[5]	O	1	—	—	—	—	—	—
TxD[6]	O	1	—	—	—	—	—	—
TxD[7]	O	1	—	—	—	—	—	—
TX_EN	O	1	TX_CTL (TX_EN/ TX_ERR)	O	1	TCG[4]/TCG[9]	O	1
TX_ER	O	1	—	—	—	—	—	—
RX_CLK	I	1	RX_CLK	I	1	RX_CLK	I	1
RxD[0]	I	1	RxD[0]/RxD[4]	I	1	RCG[0]/RCG[5]	I	1
RxD[1]	I	1	RxD[1]/RxD[5]	I	1	RCG[1]/RCG[6]	I	1
RxD[2]	I	1	RxD[2]/RxD[6]	I	1	RCG[2]/RCG[7]	I	1
RxD[3]	I	1	RxD[3]/RxD[7]	I	1	RCG[3]/RCG[8]	I	1
RxD[4]	I	1	—	—	—	—	—	—
RxD[5]	I	1	—	—	—	—	—	—
RxD[6]	I	1	—	—	—	—	—	—
RxD[7]	I	1	—	—	—	—	—	—
RX_DV	I	1	RX_CTL (RX_DV/ RX_ERR)	I	1	RCG[4]/RCG[9]	I	1
RX_ER	I	1	—	—	—	—	—	—
COL	I	1	—	—	—	—	—	—
CRS	I	1	—	—	—	—	I	—
Sum		25	**Sum**		12	**Sum**		12

Table 13-125. RGMII Signals Multiplexing

eTSEC Signals			RGMII Interface		
Frequency [MHz] 125			Frequency [MHz] 125		
Voltage[V] 3.3/2.5			Voltage[V] 2.5		
Signals (TSEC*n*_)	I/O	No. of Signals	Signals (TSEC*n*_)	I/O	No. of Signals
GTX_CLK	O	1	GTX_CLK	O	1
TX_CLK	I	1	—	—	—
TxD[0]	O	1	TxD[0]/TxD[4]	O	1
TxD[1]	O	1	TxD[1]/TxD[5]	O	1
TxD[2]	O	1	TxD[2]/TxD[6]	O	1
TxD[3]	O	1	TxD[3]/TxD[7]	O	1
TxD[4]	O	1	—	—	—
TxD[5]	O	1	—	—	—
TxD[6]	O	1	—	—	—
TxD[7]	O	1	—	—	—
TX_EN	O	1	TX_CTL (TX_EN/TX_ERR)	O	1
TX_ER	O	1	—	—	—
RX_CLK	I	1	RX_CLK	I	1
RxD[0]	I	1	RxD[0]/RxD[4]	I	1
RxD[1]	I	1	RxD[1]/RxD[5]	I	1
RxD[2]	I	1	RxD[2]/RxD[6]	I	1
RxD[3]	I	1	RxD[3]/RxD[7]	I	1
RxD[4]	I	1	—	—	—
RxD[5]	I	1	—	—	—
RxD[6]	I	1	—	—	—
RxD[7]	I	1	—	—	—
RX_DV	I	1	RX_CTL (RX_DV/RX_ERR)	I	1
RX_ER	I	1	—	—	—
COL	I	1	—	—	—
CRS	I	1	—	—	—
Sum		25	**Sum**		12

Table 13-126 describes the signals shared by all interfaces.

Table 13-126. Shared Signals

Signals	I/O	No. of Signals	Function
MDIO	I/O	1	Management interface I/O
MDC	O	1	Management interface clock
EC1_GTX_CLK125	I	1	Reference clock for eTSEC1/2
EC2_GTX_CLK125	I	1	Reference clock for eTSEC3/4
Sum		4	—

13.6.2 Connecting to FIFO Interfaces

This section describes how to connect an eTSEC to third-party communication devices, including users' ASICs and FPGAs, through the FIFO interface.

Each eTSEC provides an 8-bit or 16-bit full-duplex packet FIFO interface port that bypasses the Ethernet MAC, but re-uses the GMII signals. As a result, the FIFO interface normally does not impose the overheads of Ethernet framing. The FIFO interface operates synchronously, at a maximum frequency defined by a ratio of 4.2:1 (platform:TxClk) in GMII mode and 3.2:1 (platform:TxClk) in encoded mode providing OC-48 full-duplex transfer rates. For example, a FIFO frequency of 127 MHz in GMII mode requires a platform frequency of 533 MHz; a FIFO frequency of 200 MHz in encoded mode requires a platform frequency of 667 MHz; a FIFO frequency of 167 MHz in encoded mode requires a platform frequency of 533 MHz.

Bare IP packets—with an optional 32-bit CRC check sequence—can be transferred to the eTSEC directly. The eTSEC Tx and Rx FIFOs, TOE functions, and DMA continue to be used in packet FIFO mode.

The ECNTRL[FIFM] bit determines whether eTSEC is communicating with its Ethernet MAC or FIFO interface. There are two main modes of FIFO operation:

- 8-bit packet FIFO
  - The GMII signals of each eTSEC can be used to create a FIFO port, therefore eTSEC can support up to four simultaneous 8-bit FIFO interfaces Choosing between 8-bit FIFO and Ethernet affects each eTSEC independently, therefore a mix of FIFO and Ethernet interfaces can be configured.
  - The data signals of GMII and 8-bit FIFO remain the same. The data valid (RX_DV, TX_EN) and error (RX_ER, TX_ER) signals are used to signal framing information. If required, the collision (COL) and carrier sense (CRS) signals can be used in an encoded mode to provide link-level flow control.
- 16-bit packet FIFO
  - The GMII signals of eTSECs 1 & 2 (or 3 & 4) may be combined to create a 16-bit FIFO port Note, therefore, that with four eTSEC controllers, up to 4 8-bit FIFO ports or up to 2 16-bit

FIFO ports are available. The eTSEC that loses access to its GMII signals (eTSEC2 or 4) in this mode cannot be used for communications.

— Either a GMII-style or encoded framing protocol is available, the latter allowing relatively simple conversion to a POS PHY Level 3 interface with additional glue logic. When the FIFO interface is in 16-bit mode, RX_DV and TX_EN signals from both ports are used to determine data valid. Only one of the RX_ER and TX_ER signals are used to determine framing errors. The collision (COL) and carrier sense (CRS) signals can be used to provide link-level flow control.

The possible port mode combinations supported across all four eTSECs are shown in Table 13-127.

Table 13-127. Valid Combinations of eTSEC Signals and Interface Modes

Mode Option	eTSEC1 Signals	eTSEC2 Signals	eTSEC3 Signals	eTSEC4 Signals
Ethernet only	All Ethernets	All Ethernets	All Ethernets	All Ethernets
8-bit FIFO only	8-bit FIFO port 1	8-bit FIFO port 2	8-bit FIFO port 3	8-bit FIFO port 4
8-bit FIFO & Ethernet, mixed	8-bit FIFO port 1 or Reduced Ethernets	8-bit FIFO port 2 or Reduced Ethernets	8-bit FIFO port 3 or Reduced Ethernets	8-bit FIFO port 4 or Reduced Ethernets
16-bit FIFO only	16-bit FIFO port 1		16-bit FIFO port 3	
16-bit FIFO & 8-bit FIFO mixed	16-bit FIFO port 1		8-bit FIFO port 3	8-bit FIFO port 4
	8-bit FIFO port 1	8-bit FIFO port 2	16-bit FIFO port 3	
16-bit FIFO & Ethernet	16-bit FIFO port 1		All Ethernets	All Ethernets
	All Ethernets	All Ethernets	16-bit FIFO port 3	

The following restrictions apply in any of the FIFO modes:

- Transferred packets must be a minimum of 10 bytes, and no more than 9600 bytes in length.
- Although TCP/IP offload is supported, the receive queue filer table must be limited to as many entries as eTSEC can search every packet. See Section 13.6.5.2.1, “Filing Rules,” on page 13-167 for guidance on how to determine maximum table size for an application.
- eTSEC requires received packets to have a minimum inter-packet gap of three cycles.
- On transmission, the minimum inter-packet gap (set in FIFOCFG[IPG]) is three cycles if CRC is not automatically appended. Each CRC data beat adds to this requirement. For 16-bit FIFO interfaces the minimum Tx IPG is 5 cycles and for 8-bit FIFO interfaces the minimum is 7 cycles.

No Ethernet-specific features (such as MAC address matching) or layer 2 properties (such as Ethertype) are available in FIFO mode.

13.6.2.1 Flow Control

In the encoded (non GMII-style) FIFO modes, link-level flow control is provided to the eTSEC transmitter on the COL signal of the controlling eTSEC, while back pressure to the remote transmitter is sent on the CRS signal (which acts as an output signal only in FIFO mode). Owing to the synchronization delay of responding to flow control on signal COL, the eTSEC cannot stop transmission immediately, but may

require up to 8 clock cycles before transmission is paused. The eTSEC issues flow control either when software forces it (through the FIFOCFG[FFC] bit), or when the Rx FIFO reaches its high watermark.

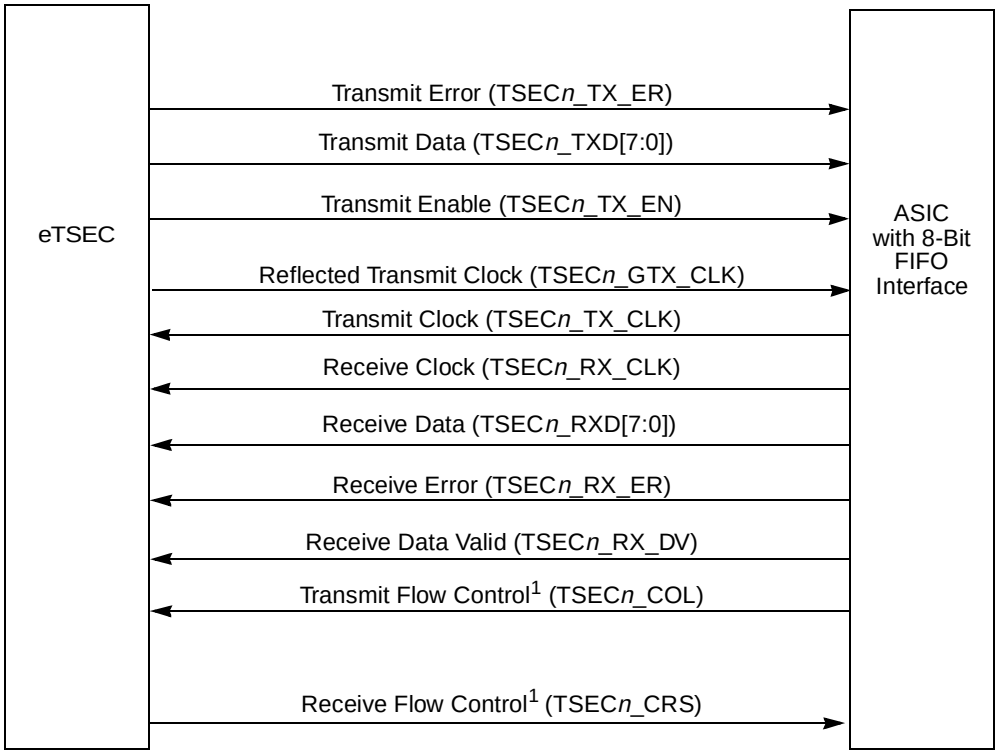
13.6.2.2 CRC Appending and Checking

If FIFOCFG[CRCAPP] is enabled, the FIFO interface automatically appends a 4-byte CRC to each transmitted packet. Alternatively, if FIFOCFG[CRCAPP] is cleared, TxBD[TC] provides a per-packet override to append CRC. The IEEE 802.3 standard CRC-32 algorithm is used, where the least significant bit of each byte (TXD[0]) is combined into the CRC ahead of the most significant bit (TXD[7]). Accordingly, the CRC result, CRC[31:0] is transmitted onto the interface in bit-reversed order, CRC[24:31], CRC[16:23], CRC[8:15], CRC[0:7].

Automatic checking of CRC-32 checksums received over the FIFO interface is enabled by setting FIFOCFG[CRCCHK]. CRC errors are recorded in the RxBD[CR] flag of every last buffer. Like transmit, the receiver combines data into the CRC in the order least significant data bit (RXD[0]) to most significant bit (RXD[7]). The last 4 bytes of the packet are assumed to be CRC whenever FIFOCFG[CRCCHK] is enabled, and these bytes are returned as part of the data buffer.

13.6.2.3 8-Bit GMII-Style Packet FIFO Mode

Figure 13-122 depicts the signals required to establish eTSEC module connection with an external device using the 8-bit FIFO interface.



[1]The flow control signals (TSEC*n*_CRS and TSEC*n*_COL) are common to all of the FIFO modes. TSEC*n*_CRS becomes an output signal in FIFO modes only.

Figure 13-122. eTSEC-FIFO (8-Bit) Connection

The 8-bit FIFO interface has 25 signals (including the flow control signals). Illustrative timing of the GMII-style FIFO mode is shown in Figure 13-123.

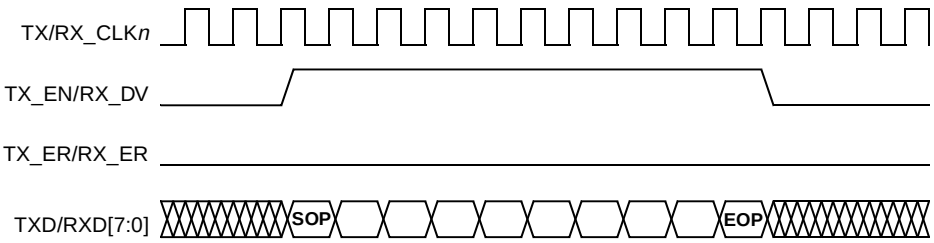


Figure 13-123. 8-Bit GMII-Style Packet FIFO Timing

The encoding of the eTSEC GMII signals in this FIFO mode is shown in Table 13-128.

Table 13-128. Signal Encoding for GMII-Style 8-Bit FIFO

Condition	TX_EN/RX_DV	TX_ER/RX_ER
Valid data, start of packet	0 to 1 transition at start of cycle	0
Valid data	1	0
Valid data, end of packet	1 to 0 transition at end of cycle	0
Error	1	1 until TX_EN/RX_DV falls

In this mode flow control can control only the decision to continue transmitting packets, as packet transfers cannot be suspended once started.

13.6.2.4 8-Bit Encoded Packet FIFO Mode

The encoded packet 8-bit FIFO mode uses the signals shown in Figure 13-124. The control lines encode four states that can be associated with each beat of data. This mode should be used where invalid bytes can appear between the start and end of packet. Illustrative timing of the encoded packet FIFO mode is shown in Figure 13-124.

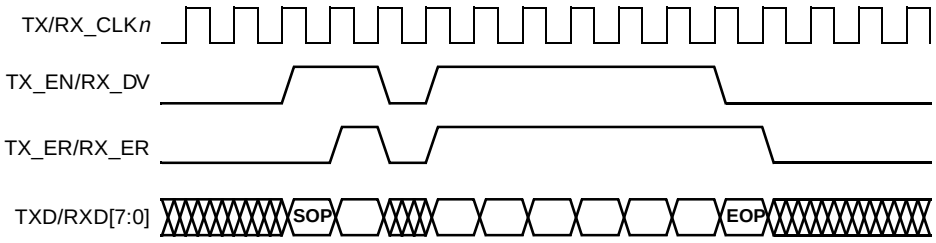


Figure 13-124. 8-Bit Encoded Packet FIFO Timing

The encoding of the eTSEC GMII signals in this FIFO mode is shown in Table 13-129.

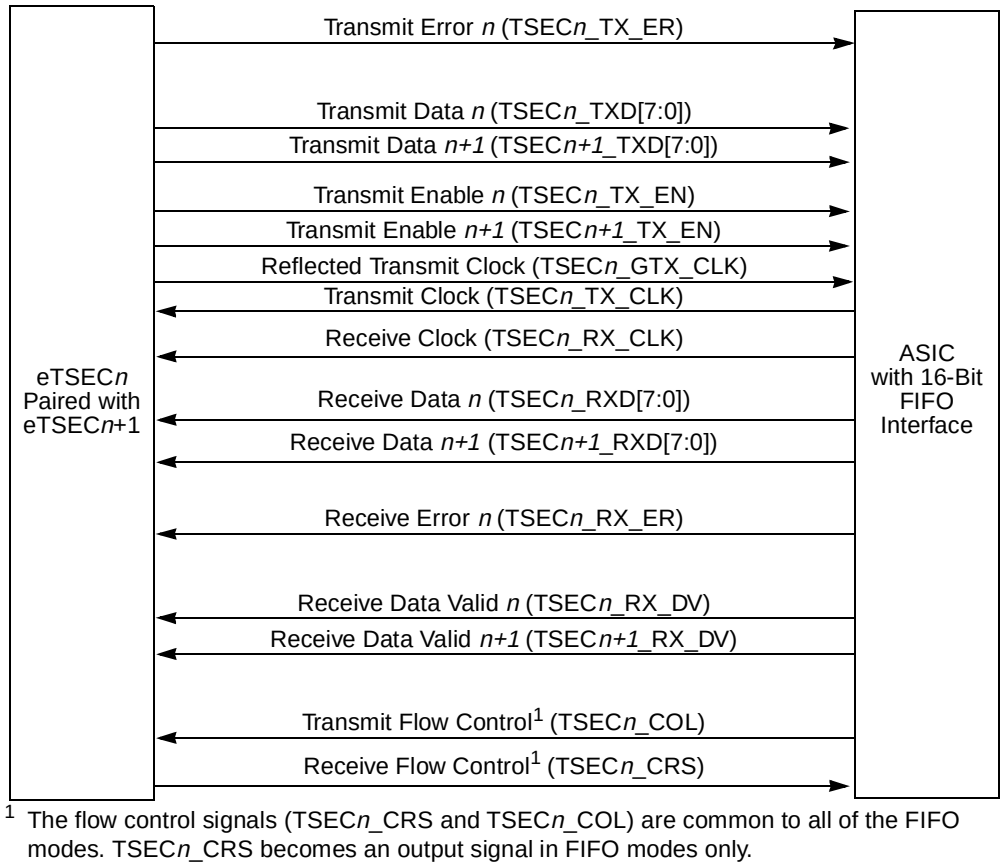
Table 13-129. Signal Encoding for Encoded 8-Bit FIFO

Condition	TX_EN/RX_DV	TX_ER/RX_ER
Valid data, start of packet	1	0
Valid data	1	1
Valid data, end of packet	0	1
Data not valid	0	0

In this mode flow control can cause an indefinite number of invalid data bytes to be transferred. This is the only mode in which an empty eTSEC Tx FIFO also causes a string of invalid data bytes to be transmitted rather than causing an underrun error.

13.6.2.5 16-Bit GMII-Style Packet FIFO Mode

Figure 13-125 depicts the signals required to establish eTSEC module connection with an external device using the 16-bit FIFO interface. The controlling eTSEC is marked as eTSEC*n*, with eTSEC*n+1* being disabled in this mode. Data is transferred simultaneously on both pairs of RXD[7:0] and TXD[7:0]. The ordering of bytes in each 16 bits is such that eTSEC*n* receives/transmits the earlier byte ahead of the byte in the packet transferred by eTSEC*n+1*. In the case where an odd number of bytes are being transferred, the last byte on eTSEC*n+1* data signals are undefined.



1 The flow control signals (TSEC*n*_CRS and TSEC*n*_COL) are common to all of the FIFO modes. TSEC*n*_CRS becomes an output signal in FIFO modes only.

Figure 13-125. eTSEC-FIFO (16-Bit) Connection

The 16-bit FIFO interface has 44 signals (including the flow control signals). Illustrative timing of the GMII-style FIFO mode is shown in Figure 13-126. The eTSEC*n* control signals mark the extent of valid data, while the eTSEC*n+1* signals mark whether the paired byte is also valid. Any errors are indicated by the eTSEC*n* error signals, and held until the end of the packet.

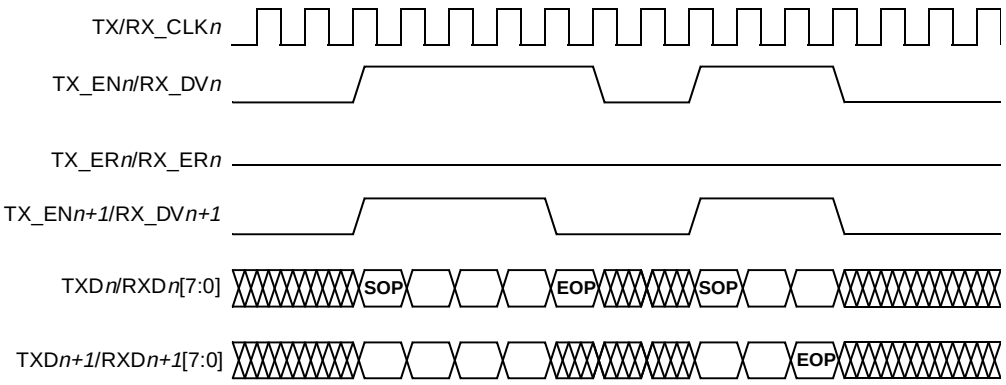


Figure 13-126. 16-Bit GMII-Style Packet FIFO Timing

The encoding of the paired eTSEC GMII signals in 16-bit GMII-style packet FIFO mode is shown in Table 13-130.

Table 13-130. Signal Encoding for GMII-Style 16-Bit FIFO

Condition	TX_EN*n*/RX_DV*n*	TX_ER*n*/RX_ER*n*	TX_EN*n*+1/RX_DV*n*+1
Data not valid	0	0	0
Start of packet, 2 bytes valid	0 to 1 transition at start cycle	0	0 to 1 transition at start cycle
Middle of packet, 2 bytes valid	1	0	1
End of packet, 1 byte valid	1 to 0 transition at end cycle	0	0
End of packet, 2 bytes valid	1 to 0 transition at end cycle	0	1 to 0 transition at end cycle
Error	1	1 held until TX_EN/RX_DV = 0 for both eTSECs	1

In 16-bit FIFO mode, one invalid byte and one valid byte received prior to the end of the packet is a signaling error which results in the CR bit of the RxBD being set for the frame in question (regardless of the state of FIFOCFG[CRCAPP]) and should be recognized by software as an indication that the receive data contains error(s).

In this mode flow control can control only the decision to continue transmitting packets, as packet transfers cannot be suspended once started.

13.6.2.6 16-Bit Encoded Packet FIFO Mode

The 16-bit encoded packet FIFO mode uses the control signals from both eTSECs to mark the status of data. The encoding scheme facilitates easy conversion to POS PHY Level 3 signaling through a PLD.

Illustrative timing of the encoded FIFO mode is shown in Figure 13-127. Note that GMII control signals from both participating eTSECs are combined into a 3-bit code, with the usual FIFO flow control signals operating in parallel.

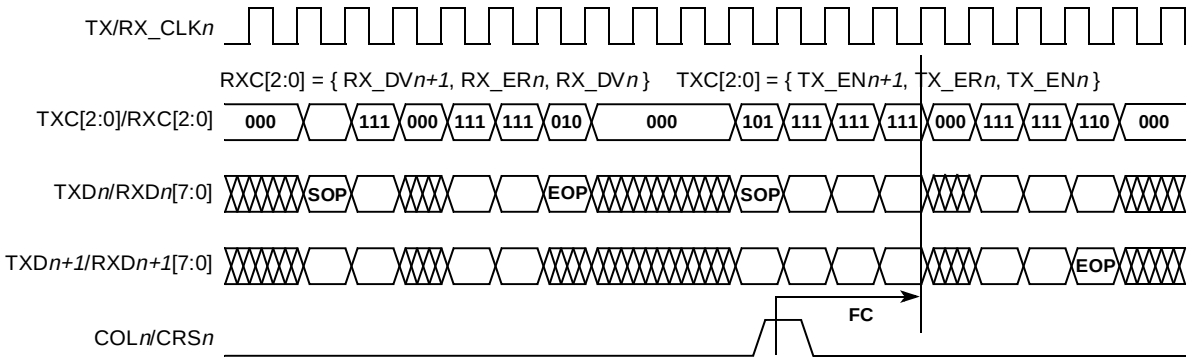


Figure 13-127. 16-Bit Encoded Packet FIFO Timing

The encoding of the paired eTSEC GMII signals in 16-bit encoded packet FIFO mode is shown in Table 13-131.

Table 13-131. Signal Encoding for Encoded 16-Bit FIFO

Condition	TXC[2:0][1]/RXC[2:0][2]
Data not valid	000
Reserved	001
End of packet, only eTSEC*n* byte valid	010
End of packet with error	011
Reserved	100
Start of packet, both eTSEC*n* and eTSEC*n+1* bytes valid	101
End of packet, both eTSEC*n* and eTSEC*n+1* bytes valid	110
Middle of packet, both eTSEC*n* and eTSEC*n+1* bytes valid	111

1 TXC[2:0] = {TX_EN*n+1*, TX_ER*n*, TX_EN*n*}
2 RXC[2:0] = {RX_DV*n+1*, RX_ER*n*, RX_DV*n*}

In this mode flow control can control either the decision to continue transmitting packets or insert invalid data (TXC/RXC = 000) into the data stream. Once asserted by the external receiver, transmit flow control takes effect after several (typically six) rising edges of TX_CLK. Should eTSEC momentarily run out of transmit data, it sends an indefinite stream of invalid data until normal transmission can resume; no transmit underrun can occur. In the event where the transmitter halts due to a lack of valid transmit BDs, the packet is terminated with an error.

13.6.2.7 FIFO Interface Signal Summary

Refer to Section 13.7.1.7, “8-Bit FIFO Mode” and Section 13.7.1.8, “16-Bit FIFO Mode,” for interface signal details.

13.6.3 Gigabit Ethernet Controller Channel Operation

This section describes the operation of the eTSEC. First, the software initialization sequence is described. Next, the software (Ethernet driver) interface for transmitting and receiving frames is reviewed. Frame filtering and receive filing algorithm features are also discussed. The section concludes with interrupt handling, inter-packet gap time, and loop back descriptions.

13.6.3.1 Initialization Sequence

This sections describes which registers are reset due to a hard or software reset and what registers the user must initialize prior to enabling the eTSEC.

13.6.3.1.1 Hardware Controlled Initialization

A hard reset occurs when the system powers up. All eTSEC’s registers and control logic are reset to their default states after a hard reset has occurred. In this state, each eTSEC behaves like a PowerQUICC III

device, except for the absence of out-of-sequence TxBD features. That is, initially TCP/IP off-load is disabled and only single RxBD and TxBD rings are accessible.

13.6.3.1.2 User Initialization

After the system has undergone a hard reset, software must initialize certain basic eTSEC registers. Other registers can also be initialized during this time, but they are optional and must be determined based on the requirements of the system. See Table 13-3 for the register list. Table 13-132 describes the minimum steps for register initialization.

Table 13-132. Steps for Minimum Register Initialization

Description
1. Set and clear MACCFG1 [Soft_Reset]
2. Initialize MACCFG2
3. Initialize MAC station address
4. Set up the PHY using the MII Mgmt Interface
5. Configure the TBI control to TBI or GMII
6. Clear IEVENT
7. Initialize IMASK
8. Initialize RCTRL
9. Initialize DMACTRL

After the initialization of registers is performed, the user must execute the following steps in the order described below to bring the eTSEC into a functional state (out of reset):

1. Write to the MACCFG1 register and set the appropriate bits. These need to include RX_EN and TX_EN. To enable flow control, Rx_Flow and Tx_Flow should also be set.
2. For the transmission of Ethernet frames, TxBDs must first be built in memory, linked together as a ring, and pointed to by the TBASE*n* registers. A minimum of two buffer descriptors per ring is required, unless the ring is disabled. Setting the ring to a size of one causes the same frame to be transmitted twice. If TCP/IP off-load is to be enabled, the TxBD[TOE] bit must be set for each frame.
3. Likewise, for the reception of Ethernet frames, the receive queue (or queues) must be ready, with its RxBD pointed to by the RBASE*n* registers. If TCP/IP off-load is to be enabled, RCTRL[PRSDEP] must be set to the required off-load level. Both transmit and receive can be gracefully stopped after transmission and reception begins.
4. Clearing DMACTRL[GTS] triggers the transmission of frame data if the transmitter had been previously stopped. The DMACTRL[GRS] must be cleared if the receiver had been previously stopped. Refer to the DMACTRL register section, and Section 13.6.7.1, “Data Buffer Descriptors,” for more information.

13.6.3.2 Soft Reset and Reconfiguring Procedure

Before issuing a soft-reset to and/or reconfiguring the MAC with new parameters, the user must properly shutdown the DMA and make sure it is in an idle state for the entire duration. User must gracefully stop the DMA by setting both GRS and GTS bits in the DMACTRL register, then wait for both GRSC and GTSC bits to be set in the IEVENT register before resetting the MAC or changing parameters. Both GRS and GTS bits must be cleared before re-enabling the MAC to resume the DMA.

During the MAC configuration, if a new set of Tx buffer descriptors are used, the user must load the pointers into the TBASE registers. Likewise if a new set of Rx buffer descriptors are used, the RBASE registers must be written with new pointers.

Following is a procedure to gracefully reset and reconfigure the MAC:

1. Set GRS/GTS bits in DMACTRL register
2. Poll GRSC/GTSC bits in IEVENT register until both are set
3. Set SOFT_RESET bit in MACCFG1 register (Note that SOFT_RESET must remain set for at least 3 TX clocks before proceeding.)
4. Clear SOFT_RESET bit in MACCFG1 register
5. Load TDBPH, TBASEH, TBASE0–TBASE7 with new Tx BD pointers
6. Load RDBPH, RBASEH, RBASE0–RBASE7 with new Rx BD pointers
7. Setup other MAC registers (MACCFG2, MAXFRM, and so on)
8. Setup group address hash table (GADDR0–GADDR15) if address filtering is required
9. Setup receive frame filer table (through RQFAR, RQFCR, and RQFPR) if filing to multiple RxBD rings is required
10. Setup WWR, WOP, TOD bits in DMACTRL register
11. Enable transmit queues in TQUEUE, and ensure that the transmit scheduling mode is correctly set in TCTRL.
12. Enable receive queues in RQUEUE, and optionally set TOE functionality in RCTRL.
13. Clear THLT and TXF bits in TSTAT register by writing 1 to them
14. Clear QHLT and RXF bits in RSTAT register by writing 1 to them.
15. Clear GRS/GTS bits in DMACTRL (do not change other bits)
16. Enable Tx_EN/Rx_EN in MACCFG1 register

13.6.3.3 Gigabit Ethernet Frame Transmission

The Ethernet transmitter requires little core intervention. After the software driver initializes the system, the eTSEC begins to poll the first transmit buffer descriptor (TxBD) in TxBD ring 0 every 512 transmit clocks. If TxBD[R] is set, and the TxBD ring is scheduled for transmission, the eTSEC begins copying the associated transmit buffer from memory to its Tx FIFO. The transmitter takes data from the Tx FIFO and transmits data to the MAC. The MAC transmits the data through the GMII interface to the physical media. The transmitter, once initialized, runs until the end-of-frame (EOF) condition is detected unless a collision within the collision window occurs (half-duplex mode) or an abort condition is encountered.

If the user has a frame ready to transmit, setting the DMACTRL[TOD] eliminates waiting for the next poll and a DMA transfer of the transmit data buffers can begin immediately. The transmission begins once all data for the frame is loaded into the Tx FIFO or sufficient transmit data (determined by the Tx FIFO threshold register) is in the Tx FIFO. If the line is not busy, the MAC transmit logic asserts TX_EN and sends the 7-octet preamble sequence, 1-octet start of frame delimiter, and frame information in that order. If the line is busy, the controller waits for the carrier sense signal, CRS, to remain inactive for 60 bit times (60 clocks) and transmission begins after an additional 36 bit times (96 bit times after CRS became active). In full-duplex mode, because collisions are ignored, frame transmission maintains only the interframe gap (96 bit times) regardless of CRS.

In half-duplex mode (MACCFG2[Full Duplex] is cleared) the MAC defers transmission if the line is busy (CRS asserted). Before transmitting, the MAC waits for carrier sense to become inactive, at which point it then determines if CRS remains negated for 60 clocks. If so, transmission begins after an additional 36 bit times (96 bit times after CRS originally became negated). If CRS continues to be asserted, the MAC follows a specified back-off procedure and tries to retransmit the frame until the retry limit is reached. Data stored in the Tx FIFO is re-transmitted in case of a collision. This avoids unnecessary memory traffic.

The transmitter also monitors for an abort condition and terminates the current frame if an abort condition is encountered. In full-duplex mode the protocol is independent of network activity, and only the transmit inter-frame gap must be enforced.

The transmitter implements full-duplex flow control. If a flow control frame is received, the MAC does not service the transmitter’s request to send data until the pause duration is over. If the MAC is currently sending data after a pause frame has been received and processed, the MAC finishes sending the current frame, then suspends subsequent frames (except a pause frame) until the pause duration is over. In addition, the transmitter supports transmission of flow control frames through TCTRL[TFC_PAUSE]. The transmit pause frame is generated internally based on the PAUSE register that defines the pause value to be sent. Note that it is possible to send a pause frame while the pause timer has not expired.

The MAC automatically appends FCS (32-bit CRC) bytes to the frame if any of the following values are set:

- TxBD[PAD/CRC] is set in first TxBD
- TxBD[TC] is set in first TxBD
- MACCFG2[PAD/CRC] is set
- MACCFG2[CRC] is set

The Tx_EN is negated after the FCS is sent. This notifies the PHY of the need to generate the illegal Manchester encoding that signifies the end of an Ethernet frame. Following the transmission of the FCS, the Ethernet controller writes the frame status bits into the BD and clears TxBD[R]. If the end of the current buffer is reached and TxBD[L] is cleared (a frame is comprised of multiple buffer descriptors), only TxBD[R] is cleared.

For both half- and full-duplex modes, an interrupt can be issued depending on TxBD[I]. The Ethernet controller then proceeds to the next TxBD in the table. In this way, the core can be interrupted after each frame, after each buffer, or after a specific buffer is sent. If TxBD[PAD/CRC] is set, the Ethernet controller pads any frame shorter than 64 bytes with zero bytes to make up the minimum length.

To pause transmission, or rearrange the transmit queue, set DMACTRL[GTS]. This can be useful for transmitting expedited data ahead of previously-linked buffers or for error situations. If this bit is set, the eTSEC transmitter performs a graceful transmit stop. The Ethernet controller stops immediately if no transmission is in progress or continues transmission until all queued frames in the Tx FIFO have been disposed of. The IEVENT[GTSC] interrupt occurs once the graceful transmit stop operation is completed. After the DMACTRL[GTS] is cleared, the eTSEC resumes transmission with the next frame.

While the eTSEC is in 10/100Mbps mode it sends bytes least-significant nibble first and each nibble is sent lsb first. While it is in 1000Mbps mode it sends bytes LSB first.

13.6.3.4 Gigabit Ethernet Frame Reception

The eTSEC Ethernet receiver is designed to work with little core intervention and can perform data extraction, address recognition, CRC checking, short frame checking, and maximum frame-length checking.

After a hardware reset, the software driver clears the RSTAT register and sets MACCFG1[RX_EN]. The Ethernet receiver is enabled and immediately starts processing receive frames. The MAC checks for when TSEC*n*_RX_DV is asserted and as long as TSEC*n*_COL remains negated (full-duplex mode ignores TSEC*n*_COL), the MAC looks for the start of a frame by searching for a valid preamble/SFD (start of frame delimiter) header, which is stripped (unless MACCFG2[PreAM RxEN] is set) and the frame begins to be processed. If a valid header is not found, the frame is ignored.

If the receiver detects the first bytes of a frame, the eTSEC controller begins to perform the frame recognition function through destination address (DA) recognition (see Section 13.6.3.7, “Frame Recognition”). Based on this match the frame can be accepted or rejected. The receiver can filter frames based on individual (unicast), group (multicast), and broadcast addresses. Because Ethernet receive frame data is not written to memory until the internal frame recognition algorithm is complete, system bus usage is not wasted on frames unwanted by this station.

If a frame is accepted, the Ethernet controller fetches the receive buffer descriptor (RxBD) from either queue 0 or the queue determined by the filer. If the RxBD is not being used by software (RxBD[E] is set), the eTSEC starts transferring the incoming frame. RxBD[F] is set for the first RxBD used for any particular receive frame. If the current RxBD is not available for the received frame, a receive busy error condition is raised in IEVENT[BSY].

After the buffer is filled, the eTSEC clears RxBD[E] and, if RxBD[I] is set, generates an interrupt. If the incoming frame is larger than the buffer, the Ethernet controller fetches the next RxBD in the table. If it is empty, the controller continues receiving the rest of the frame. In half-duplex mode, if a collision is detected during the frame, no RxBDs are used; thus, no collision frames are presented to the user except late collisions, which indicate LAN problems.

The RxBD length is determined by the MRBL field in the maximum receive buffer length register (MRBLR). The smallest valid value is 64 bytes, with larger values being be some integral multiple of 64 bytes. During reception, the Ethernet controller checks for frames that are too short or too long. After the frame ends (CRS is negated), the receive CRC field is checked and written to the data buffer. The data length written to the last RxBD in the Ethernet frame is the length of the entire frame, which enables the software to recognize an oversized frame condition.

Receive frames are not truncated when they exceed maximum frame bytes in the MAC's maximum frame register if MACCFG2[Huge Frame] is set, yet the babbling receiver error interrupt occurs (IEVENT[BABR] is set) and RxBD[LG] is set.

After the receive frame is complete, the Ethernet controller sets RxBD[L], updates the frame status bits in the RxBD, and clears RxBD[E]. If RxBD[I] is set, the Ethernet controller next generates an interrupt (that can be masked) indicating that a frame was received and is in memory. The Ethernet controller then waits for a new frame.

To interrupt reception or rearrange the receive queue, DMACTRL[GRS] must be set. If this bit is set, the eTSEC receiver performs a graceful receive stop. The Ethernet controller stops immediately if no frames are being received or continues receiving until the current frame either finishes or an error condition occurs. The IEVENT[GRSC] interrupt event is signaled after the graceful receive stop operation is completed. While in this mode the user can write to registers that are accessible to both the user and the eTSEC hardware without fear of conflict, and finally clear IEVENT[GRSC]. After DMACTRL[GRS] is cleared, the eTSEC scans the input data stream for the start of a new frame (preamble sequence and start of frame delimiter), it resumes receiving, and the first valid frame received is placed in the next available RxBD.

13.6.3.5 Ethernet Preamble Customization

By default eTSEC generates a standard Ethernet preamble sequence prior to transmitting frames. However, the user can substitute a custom preamble sequence for the purpose of controlling switching equipment at the receiver, particularly at 100/1000Mbps speeds. In FIFO mode preamble customization is ignored; in any RMII mode only the standard preamble can be transmitted.

eTSEC normally searches for and discards the standard Ethernet preamble sequence upon receiving frames. Part of the received preamble sequence can be optionally recovered and returned as part of the frame data, making it visible to user software. Note however, that no preamble is received in FIFO mode, and preamble cannot be recovered in any RMII mode. Note that it is also possible for the first two bytes of custom preamble (PreOct0 and PreOct1) to be lost in during conversion to ten-bit code groups in the PCS sub-layer. Thus is it recommended that any custom preamble start at PreOct2.

13.6.3.5.1 User-Defined Preamble Transmission

To substitute a custom preamble, the user must ensure that:

- MACCFG2[PreAm TxEN] bit is set
- The first TxBD of every frame containing a custom preamble has its PRE bit set
- An 8-byte custom preamble sequence appears before the Ethernet DA field in the first transmit data buffer

The definition of the 8-byte custom preamble sequence is shown in Figure 13-128.

Byte Offsets	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0–1	PreOct0								PreOct1							
2–3	PreOct2								PreOct3							
4–5	PreOct4								PreOct5							
6–7	PreOct6															

Figure 13-128. Definition of Custom Preamble Sequence

The fields of the custom preamble sequence are described in Table 13-133. It should be noted that use of preamble octets matching the standard start of frame delimiter (0xD5) can be expected to trigger premature frame reception by the receiving station.

Table 13-133. Custom Preamble Field Descriptions

Bytes	Bits	Name	Description
0–1	0–7	PreOct0	Octet #0 of custom transmit preamble. This is the first octet of preamble sent.
	8–15	PreOct1	Octet #1 of custom transmit preamble. This is the second octet of preamble sent.
2–3	0–7	PreOct2	Octet #2 of custom transmit preamble. This is the third octet of preamble sent.
	8–15	PreOct3	Octet #3 of custom transmit preamble. This is the fourth octet of preamble sent.
4–5	0–7	PreOct4	Octet #4 of custom transmit preamble. This is the fifth octet of preamble sent.
	8–15	PreOct5	Octet #5 of custom transmit preamble. This is the sixth octet of preamble sent.
6–7	0–7	PreOct6	Octet #6 of custom transmit preamble. This is the seventh octet of preamble sent. The last octet (the start of frame delimiter) is generated by the MAC automatically.
	8–15	—	Reserved; should be cleared.

13.6.3.5.2 User-Visible Preamble Reception

To return the received preamble, the user must ensure that:

- MACCFG2[PreAm RxEN] bit is set
- Space for an 8-byte preamble sequence is allowed before the Ethernet DA field in the first receive data buffer of each frame

The definition of the 8-byte received preamble sequence is shown in Figure 13-129.

Byte Offsets	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0–1	PreOct0								PreOct1							
2–3	PreOct2								PreOct3							
4–5	PreOct4								PreOct5							
6–7	PreOct6															

Figure 13-129. Definition of Received Preamble Sequence

The fields of the received preamble sequence are described in Table 13-134. Should the received preamble be shorter than the 7-octet sequence defined by IEEE Std. 802.3, initial bytes of the received preamble sequence hold undefined values. The standard start of frame delimiter (0xD5) is always omitted. Note that preamble extraction is not possible in RMII mode.

Table 13-134. Received Preamble Field Descriptions

Bytes	Bits	Name	Description
0–1	0–7	PreOct0	Octet #0 of received preamble. This is the first octet of preamble received.
	8–15	PreOct1	Octet #1 of received preamble. This is the second octet of preamble received.
2–3	0–7	PreOct2	Octet #2 of received preamble. This is the third octet of preamble received.
	8–15	PreOct3	Octet #3 of received preamble. This is the fourth octet of preamble received.
4–5	0–7	PreOct4	Octet #4 of received preamble. This is the fifth octet of preamble received.
	8–15	PreOct5	Octet #5 of received preamble. This is the sixth octet of preamble received.
6–7	0–7	PreOct6	Octet #6 of received preamble. This is the seventh octet of preamble received. The last octet (the start of frame delimiter) is discarded.
	8–15	—	Reserved

13.6.3.6 RMON Support

Using promiscuous mode, the eTSEC can automatically gather network statistics required for remote network interface monitoring. The RMON MIB group 1, RMON MIB group 2, RMON MIB group 3, RMON MIB group 9, RMON MIB2, and the 802.3 Ethernet MIB are supported. For RMON statistics and their corresponding counters, see the memory map.

13.6.3.7 Frame Recognition

The Ethernet controller performs frame recognition using destination address (DA) recognition. A frame can be rejected or accepted based on the outcome.

13.6.3.7.1 Destination Address Recognition and Frame Filtering

The eTSEC can perform layer 2 frame filtering on the basis of destination Ethernet address (DA), as illustrated by the flowchart in Figure 13-130.

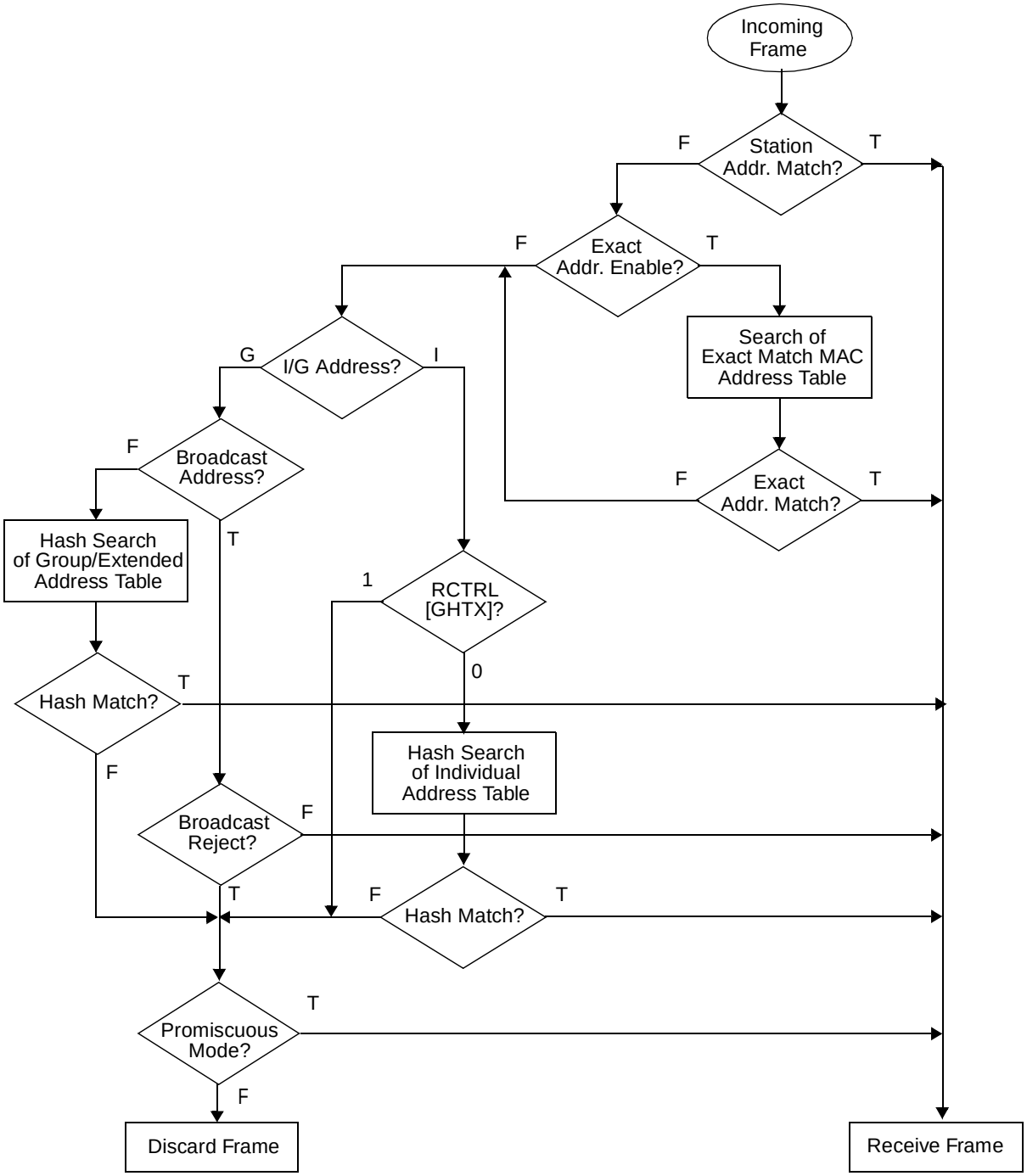


Figure 13-130. Ethernet Address Recognition Flowchart

In promiscuous mode, the eTSEC accepts all received frames regardless of DA. Note, however, that Ethernet frame filtering simply restricts the traffic seen by the receive queue filer. Therefore even in

promiscuous mode it remains possible to program the filer to reject frames based on their higher-layer header contents.

In the case of an individual address, the DA field of the received frame is compared with the physical address that the user programs in the station address registers (MACSTNADDR1 and MACSTNADDR2). If the DA does not match the station address, and exact MAC address matching is enabled through RCTRL[EMEN], the controller performs address recognition on the multiple MAC addresses written to the MAC*x*ADDR1 and MAC*x*ADDR2 registers. These virtual addresses give a particular eTSEC the ability to mirror other MACs on the network, which caters for router redundancy protocols, such as HSRP and VRRP.

If exact MAC address matching is not enabled, the eTSEC determines whether DA is a group or individual address. If DA is the standard broadcast address, and broadcast addresses are not rejected, the frame is accepted. If any other group address is received, the eTSEC looks-up the DA by means of the group hash table. The group hash table may be extended to 512 entries if RCTRL[GHTX] = 1. Otherwise, an individual address is hashed into the 256-entry individual hash table when RCTRL[GHTX] = 0.

13.6.3.7.2 Hash Table Algorithm

The hash table process used in the group hash filtering operates as follows. By default, the Ethernet controller maps any 48-bit destination address into one of 256 bins, represented by the 256 bits in IGADDR0–IGADDR7 for individual addresses, and the 256 bits in GADDR0–GADDR7 for group addresses. But in the case where RCTRL[GHTX] is set, both sets of registers are combined into an extended group-only hash table of 512 bits, where IGADDR0–IGADDR7 contain the first 256 bits and GADDR0–GADDR7 contain the last 256 bits. No individual-address table exists in extended mode.

The 48-bit destination address received by the MAC is passed through the Ethernet CRC-32 algorithm to produce a hash value. The CRC polynomial used is:

$$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

The MAC initializes its CRC register to 0xFFFFFFFF before computing a CRC on the 6 bit-reversed octets of the DA. A non-optimized sample of C code for computing the DA hash is listed in Figure 13-131. The 9 most significant bits of the raw, uninverted CRC are used as the hash table index, H[8:0]. If RCTRL[GHTX] = 0, bits H[8:6] select one of the 8 IGADDR or GADDR registers, while bits H[5:1] select a bit within the 32-bit register. If RCTRL[GHTX] = 1, bits H[8:5] select one of the 16 registers in the {IGADDR, GADDR} set, while bits H[4:0] select a bit within the 32-bit register. For example, if H[8:5] = 7, IGADDR7 is selected, whereas H[8:5] = 9 selects GADDR1.

```
/* Wrapper macros for 256-bucket and 512-bucket hash tables:
   Pass 6-byte Ethernet MAC address as parameter. */
#define TSEC_HASH256(macaddr) ((crc32(macaddr) >> 24) & 0xff)
#define TSEC_HASH512(macaddr) ((crc32(macaddr) >> 23) & 0x1ff)

/* CRC constants.  Note: CRC-32 polynomial is bit-reversed. */
#define CRC_POLYNOMIAL 0xedb88320
#define CRC_INITIAL    0xffffffff
#define MAC_ADDRLEN    6
#define BITS_PER_BYTE  8

/* crc32() Takes the array of bytes, macaddr[], representing an
   Ethernet MAC address and returns the CRC-32 result over these bytes,
   where each byte is used in bit-reversed form (Ethernet bit order).
   Index 0 of macaddr[] is the first byte of the address on the wire.
   Test case: the result of crc32 on {0x00, 0x01, 0x02, 0x03, 0x04, 0x05}
   should be 0xad0c28f3.
 */
unsigned long crc32(unsigned char macaddr[MAC_ADDRLEN])
{
  unsigned long crc, result;
  int byte, i;

  /* CRC-32 algorithm starts by inverting first 4 bytes */
  crc = CRC_INITIAL;
  /* add each byte to running CRC accumulator */
  for (byte = 0; byte < MAC_ADDRLEN; ++byte) {
    crc ^= macaddr[byte];
    /* shift CRC right to perform but reversal on byte of address */
    for (i = 0; i < BITS_PER_BYTE; ++i)
      if (crc & 1)
     crc = (crc >> 1) ^ CRC_POLYNOMIAL;
      else
      crc >>= 1;
  }
  /* finally, reverse bits of result to get CRC in normal bit order */
  for (result = 0, i = 4*BITS_PER_BYTE-1; i >= 0; crc >>= 1, --i)
    result |= (crc & 1) << i;
  return result;
}
```

Figure 13-131. Sample C Code for Computing eTSEC Hash Table Indices

If the CRC hash table index selects a bit that is set in the hash table, the frame is accepted. If 32 group addresses are stored in the hash table and random group addresses are received, the extended hash table prevents roughly 480/512 (93.8%) of the group address frames from reaching memory. Software must further filter those that reach memory to determine if they contain the correct addresses. Alternatively, small multicast groups can be held in the exact match MAC address registers, which guarantees that only correct frames are admitted.

The effectiveness of the hash table declines as the number of addresses increases. For instance, as the number of addresses stored in the 512-bin hash table increases, the vast majority of the hash table bits are set, preventing only a small fraction of frames from reaching memory.

NOTE

The hash table cannot be used to reject frames that match a set of selected addresses because unintended addresses can map to the same bit in the hash table. The receive queue filer may be used to reject frames with unintended address hits in the hash table.

13.6.3.8 Magic Packet Mode

eTSEC implements the AMD Magic Packet™ specification for LAN-initiated power management. This mode is normally entered with the rest of the system in a low-power sleep mode. Software must enable normal receive function in the Ethernet MAC, and then finally set the MACCFG2[MPEN] bit to enable Magic Packet detection before the system enters a reduced mode. While the rest of the system is operating in low-power mode, the enabled eTSEC continues to receive Ethernet frames, but discards them immediately. Upon receipt of any frame whose contents contain the valid Magic Packet sequence, the eTSEC exits out of Magic Packet mode, thus clearing MACCFG2[MPEN], and raises an error/diagnostic interrupt through IEVENT[MAG], which causes the surrounding system to wake-up. Frames received after Magic Packet mode has exited are received into software buffers as usual. Software can abort Magic Packet mode by writing 0 to MACCFG2[MPEN] at any time.

AMD specify a Magic Packet™ to be any Ethernet frame containing a valid Ethernet header (Destination and Source Addresses) and valid FCS (CRC-32), and whose payload includes the specific Magic Packet byte sequence at any offset from the start of frame. The specific byte sequence comprises an unbroken stream of 102 bytes, the first 6 bytes of which are 0xFFFFFF_FFFFFF, followed by 16 copies of the MAC's unique IEEE station address in the normal byte order for Ethernet addresses. For example, if the station address were 0x112233_445566, then the MAC would have to receive 0xFFFFFF_FFFFFF, 0x112233_445566, ..., 0x112233_445566 in any payload to detect a Magic Packet. Only frames addressed specifically to the MAC's station address or a valid multicast or broadcast address can be examined for the Magic Packet sequence.

13.6.3.9 Flow Control

Because collisions cannot occur in full-duplex mode, gigabit Ethernet can operate at the maximum rate. If the rate becomes too fast for a station's receiver, the station's transmitter can send flow-control frames to reduce the rate. Flow-control instructions are transferred by special frames of minimum frame size. The length/type fields of these frames have a special value.

Table 13-135 lists the flow-control frame structure.

Table 13-135. Flow Control Frame Structure

Size [Octets]	Description	Value	Comment
7	Preamble	—	—
1	SFD	—	Start frame delimiter
6	Destination address	01-80-C2-00-00-01	Multicast address reserved for use in MAC frames (or MAC station address)
6	Source address		—
2	Length/type	88-08	Control frame type

Table 13-135. Flow Control Frame Structure (continued)

Size [Octets]	Description	Value	Comment
2	MAC opcode	00-01	Pause command
2	MAC parameter	—	Pause time as defined by the PTV[PT] field. The pause period is measured in pause_quanta, a speed independent constant of 512 bit-times (unlike slot time). The most-significant octet is transmitted first.
2	Extended MAC parameter	—	Pause time extended as defined by the PTV[PTE] field. The most significant octet is transmitted first.
40	Reserved	—	—
4	FCS	—	Frame check sequence (CRC)

If flow-control mode is enabled (MACCFG1[Rx_Flow] is set) and the receiver identifies a pause-flow control frame, transmission stops for the time specified in the control frame. Since the pause timer commences counting immediately upon receipt of a PAUSE frame, regardless of whether transmission is currently in progress, a sufficiently large pause time must be received to stop transmission past a frame of MTU size. During a pause, only a control frame can be sent (TCTRL[TFC_PAUSE] is set). Normal transmission resumes after the pause timer stops counting, or resumes immediately if a pause frame with a zero time-out is received. If another pause-control frame is received during the pause, the period changes to the new value received.

13.6.3.10 Interrupt Handling

The following describes what usually occurs within a eTSEC interrupt handler:

- If an interrupt occurs, read IEVENT to determine interrupt sources. IEVENT bits to be handled in this interrupt handler are normally cleared at this time. There are three kinds of interrupt:
  - Errors (all bits in IEVENT other than RXB, RXF, TXB, or TXF)
  - Receive interrupts, when bits RXB or RXF in IEVENT are set
  - Transmit interrupts, when bits TXB or TXF in IEVENT are set
- Process the TxBDs to reuse them if the IEVENT[TXB, TXF or TXE] were set. Consult register bits TSTAT[TXF0–TXF7] to determine which TxBD rings gave rise to the transmit interrupt in the case of TXF. If the transmit speed is fast or the interrupt delay is long, more than one transmit buffer may have been sent by the eTSEC; thus, it is important to check more than just one TxBD during the interrupt handler. One common practice is to process all TxBDs in the interrupt handler until one is found with R set.
- Obtain data from RxBD rings if IEVENT[RXC, RXB or RXF] is set. Consult register bits RSTAT[RXF0–RXF7] to determine which RxBD rings gave rise to the receive interrupt in the case of RXF. If the receive speed is fast or the interrupt delay is long, the eTSEC may have received more than one RxBD; thus, it is important to check more than just one RxBD during interrupt handling. Typically, all RxBDs in the interrupt handler are processed until one is found with E set. Because the eTSEC pre-fetches BDs, the BD table must be big enough so that there is always another empty BD to pre-fetch, otherwise a BSY error occurs.

- Clear any set halt or frame interrupt bits in TSTAT and RSTAT registers, or DMACTRL[GTS] and DMACTRL[GRS] by writing 1s to these bits.
- Continue normal execution.

Table 13-136. Non-Error Transmit Interrupts

Interrupt	Description	Action Taken by the eTSEC
GTSC	Graceful transmit stop complete: transmitter is put into a pause state after completion of the frame currently being transmitted.	None
TXC	Transmit control: Instead of the next transmit frame, a control frame was sent.	None
TXB	Transmit buffer: A transmit buffer descriptor, that is not the last one in the frame, was updated in one of the enabled TxBD rings.	Programmable ‘write with response’ TxBD to memory before setting IEVENT[TXB].
TXF	Transmit frame: A frame from an enabled TxBD ring was transmitted and the last transmit buffer descriptor (TxBD) of that frame was updated.	Programmable ‘write with response’ to memory on the last TxBD before setting IEVENT[TXF].

Table 13-137. Non-Error Receive Interrupts

Interrupt	Description	Action Taken by the eTSEC
GRSC	Graceful receive stop complete: Receiver is put into a pause state after completion of the frame currently being received.	None
RXC	Receive control: A control frame was received. As soon as the transmitter finishes sending the current frame, a pause operation is performed.	None
RXB	Receive buffer: A receive buffer descriptor, that is not the last one of the frame, was updated in one of the enabled RxBD rings.	Programmable ‘write with response’ RxBD to memory before setting IEVENT[RXB].
RXF	Receive frame: A frame was received to an enabled RxBD ring and the last receive buffer descriptor (RxBD) of that frame was updated.	Programmable ‘write with response’ to memory on the last RxBD before setting IEVENT[RXF].

13.6.3.10.1 Interrupt Coalescing

Interrupt coalescing offers the user the ability to contour the behavior of the eTSEC with regard to frame interrupts. Separate but identical mechanisms exist for both transmitted frames and received frames. In either case, frame interrupts require that software set the I-bit in RxBDs or TxBDs, and disable buffer interrupts (IEVENT[RXB] or IEVENT[TXB]). Particular rings can remain free of interrupts by ensuring that the I-bit is consistently cleared in all BDs. While interrupt coalescing is enabled, a transmit or receive frame interrupt is raised either when a counter threshold-defined number of frames is received/transmitted or the timer threshold-defined period of time has elapsed, whichever occurs first. Disabling and then re-enabling interrupt coalescing forces reset of the coalescing timers and counters to reflect changes made to the threshold registers.

13.6.3.10.2 Interrupt Coalescing By Frame Count Threshold

To avoid interrupt bandwidth congestion due to frequent, consecutive interrupts, the user may enable and configure interrupt coalescing to deliberately group frame interrupts, reducing the total number of

interrupts raised. The number of frames received or transmitted prior to an interrupt being raised is determined by the frame threshold field (ICFT) in the appropriate interrupt coalescing configuration register (RXIC or TXIC). The frame threshold field may be assigned a value between 1 and 255. The internal transmit or receive frame counter decrements from this initial value each time a frame is transmitted or received. Upon reaching zero, an interrupt is raised, the appropriate threshold counter is reset to the value in the ICFT field, and then eTSEC continues counting frames while the interrupt is active. The appropriate threshold counter is also reset to the value in the ICFT field if an interrupt is raised subject to the corresponding threshold timer.

13.6.3.10.3 Interrupt Coalescing By Timer Threshold

To avoid stale frame interrupts, the user may also assign a timer threshold, beyond which any frame interrupts not yet raised are forced. The timer threshold fields of the receive and transmit interrupt coalescing configuration registers (RXIC[ICTT] and TXIC[ICTT]) are defined in units equivalent to 64 interface clocks or system clocks, depending on the setting of the ICCS field in RXIC and TXIC.

After transmitting a frame, the transmit interrupt coalescing threshold time begins counting down from the value in TXIC[ICTT]. An interrupt is raised when the counter reaches zero. In the event of graceful transmit stop completion before the coalescing timer expires, the eTSEC issues two interrupts, the first for GTS, the second for TXF (due to timer expiration of a pending event). To prevent the second interrupt from affecting servicing of the GTS event, it is recommended that the user mask out the TXF event during execution of the service routine. After receiving a frame, the receive interrupt coalescing threshold time begins counting down from the value in RXIC[ICTT]. An interrupt is raised when the counter reaches zero. In the event of graceful receive stop completion before the coalescing timer expires, the eTSEC issues two interrupts, the first for GRS, the second for RXF (due to timer expiration of a pending event). To prevent the second interrupt from affecting servicing of the GRS event, it is recommended that the user mask out the RXF event during execution of the service routine.

The interrupt coalescing timer thresholds (transmit and receive, operating independently) may be values ranging from 0x0001 to 0xFFFF. Table 13-138 specifies the range of possible timing thresholds subject to timer clock source, the interface or system frequency, and the value of the RXIC[ICTT] or TXIC[ICTT] field.

Table 13-138. Interrupt Coalescing Timing Threshold Ranges

ICCS (Clock Source)	eTSEC Interface Format and Frequency or eTSEC System Frequency	Interrupt Coalescing Threshold Time	
		Minimum (ICTT = 0x0001)	Maximum (ICTT = 0xFFFF)
0 (I/F clock)	10Base-T at 2.5 MHz	25.6 µs	1.68 s
0 (I/F clock)	100Base-T at 25 MHz	2.56 µs	168 ms
0 (I/F clock)	1000Base-T at 125 MHz	0.51 µs	33.6 ms
1 (sys. clock)	eTSEC operating at 266 MHz	0.24 µs	15.7 ms
1 (sys. clock)	eTSEC operating at 333 MHz	0.19 µs	12.6 ms

The transmit timer threshold counter is reset to the value in TXIC[ICTT] and begins counting down on transmission of the frame following an interrupt.

The receive timer threshold counter is reset to the value in RXIC[ICTT] and begins counting down on receiving the frame following an interrupt.

13.6.3.11 Inter-Frame Gap Time

If a station must transmit, it waits until the LAN becomes silent for a specified period (inter-frame gap, or IFG). The minimum inter-packet gap (IPG) time for back-to-back transmission is set by IPGIFG[Back-to-Back Inter-Packet-Gap]. The receiver receives back-to-back frames with the minimum interframe gap (IFG) as set in IPGIFG[Minimum IFG Enforcement]. If multiple frames are ready to transmit, the ethernet controller follows the minimum IPG as long as the following restrictions are met:

- The first TxBD pointer, TBPTR*n*, of any given frame is located at a 16-byte aligned address.
- Each TxBD[Data Length] is greater-than or equal to 64 bytes.

If the first TxBD alignment restriction is not met, the back-to-back IPG may be as many as 32 cycles. If the TxBD size restriction is not met, the back-to-back IPG may be significantly longer.

In half-duplex mode, after a station begins sending, it continually checks for collisions on the LAN. If a collision is detected, the station forces a jam signal (all ones) on its frame and stops transmitting. Collisions usually occur close to the beginning of a packet. The station then waits a random time period (back-off) before attempting to send again. After the back-off completes, the station waits for silence on the LAN (carrier sense negated) and then begins retransmission (retry) on the LAN. Retransmission begins 36 bit times after carrier sense is negated for at least 60 bit times. If the frame is not successfully sent within a specified number of retries, an error is indicated (collision retry limit exceeded).

13.6.3.12 Internal and External Loop Back

Setting MACCFG1[Loop Back] causes the MAC transmit outputs to be looped back to the MAC receive inputs. Clearing this bit results in normal operation. This bit is cleared by default. Clearing this bit results in normal operation.

13.6.3.13 Error-Handling Procedure

The eTSEC reports frame reception and transmission error conditions using the channel BDs, the error counters, and the IEVENT register.

Transmission errors are described in Table 13-139.

Table 13-139. Transmission Errors

Error	Response
Transmitter underrun	Transmitter underrun can occur either after frame transmission has commenced, or in response to an incomplete sequence of TxBDs. In the former case, the controller sends 32 bits that ensure a CRC error, and terminates buffer transmission. In the latter case, the relevant transmit queue is halted. In all cases, the eTSEC closes the buffer, sets TxBD[UN], IEVENT[XFUN], and IEVENT[TXE]. The controller resumes transmission after TSTAT[THLT] is cleared (and DMACTRL[GTS] is cleared).
Retransmission attempts limit expired	The controller terminates buffer transmission, sets TxBD[RL], closes the buffer, IEVENT[CRL], and IEVENT[TXE]. Transmission resumes after TSTAT[THLT] is cleared (and DMACTRL[GTS] is cleared).

Table 13-139. Transmission Errors (continued)

Error	Response
Late collision	The controller terminates buffer transmission, sets TxBD[LC], closes the buffer, IEVENT[LC], and IEVENT[TXE]. The controller resumes transmission after TSTAT[THLT] is cleared (and DMACTRL[GTS] is cleared).
Memory read error	A system bus error occurred during a DMA transaction. The controller sets IEVENT[EBERR], DMA stops sending data to the FIFO which causes an underrun error, and therefore TxBD[UN] is set, but IEVENT[XFUN] is not set. The TSTAT[THLT] is set. Transmits are continued once TSTAT[THLT] is cleared.
Data parity error	Data in the transmit FIFO was potentially corrupted. The controller sets IEVENT[DPE], but otherwise continues transmission until halted explicitly.
Babbling transmit error	A frame is transmitted which exceeds the MAC's Maximum Frame Length and MACCFG2[Huge Frame] is a 0. The controller sets IEVENT[BABT] and continues without interruption.

Reception errors are described in Table 13-140.

Table 13-140. Reception Errors

Error	Description
Overrun error	The Ethernet controller maintains an internal FIFO buffer for receiving data. If a receiver FIFO buffer overrun occurs, the controller sets RxBD[OV], sets RxBD[L], closes the buffer, increments the discarded frame counter (RDRP), and sets IEVENT[RXF], The receiver then enters hunt mode (seeking start of a new frame).
Busy error	A frame is received and discarded due to a lack of buffers. The controller sets IEVENT[BSY] and increments the discarded frame counter (RDRP). In addition, the RSTAT[QHLT*n*] bit is set. RDRP increments for each frame that is received while the receiver is halted due to a busy condition. The halted queue resumes reception once the RSTAT[QHLT*n*] bit is cleared.
Filed frame to invalid queue error	A frame is received and discarded as a result of the filer directing it to an RxBD ring that is currently not enabled. The controller sets IEVENT[FIQ] and increments the discarded frame counter (RDRP).
Parser error	If the receive frame parser is enabled, a parse error can be flagged as a result of inconsistencies discovered between fields of the embedded packet headers. For example, the L2 header may indicate an IPv4 header, but the IP version number fails to match. In the event of a parse error, parsing is terminated at the inconsistent header, and the RxFCB[PERR] field indicates at which layer of the protocol stack the error was discovered. Receiver function continues regardless of parse errors, but IEVENT[PERR] is set. The receive queue filer may operate with reduced or default information in some cases; therefore, filer rule sets should be constructed so as to be tolerant of misformed frames. **Note:** Any values in the length/type field between 1500 and 1536 is treated as a length, however, only illegal packets exist with this length/type since these are not valid lengths and not valid types. These are treated by the MAC logic as out of range. Software must confirm the parser and filer results by checking the type/length field after the packet has been written to memory to see if it falls in this range.
Non-octet error (dribbling bits)	The Ethernet controller handles a nibble of dribbling bits if the receive frame terminates as non-octet aligned and it checks the CRC of the frame on the last octet boundary. If there is a CRC error, the frame non-octet aligned (RxBD[NO]) error is reported, IEVENT[RXF] is set, and the alignment error counter increments. The eTSEC relies on the statistics collector block to increment the receive alignment error counter (RALN). If there is no CRC error, no error is reported.

Table 13-140. Reception Errors (continued)

Error	Description
CRC error	If a CRC error occurs, the controller sets RxBD[CR], closes the buffer, and sets IEVENT[RXF]. This eTSEC relies on the statistics collector block to record the event. After receiving a frame with a CRC error, the receiver then enters hunt mode.
Memory read error	A system bus error occurred during a DMA transaction. The controller sets IEVENT[EBERR] and discards the frame and increments the discarded frame counter (RDRP). In addition the RSTAT[QHLT*n*] bit is set. The halted queue resumes reception once the RSTAT[QHLT*n*] bit is cleared.
Data parity error	Data in the receive FIFO or filer table was potentially corrupted. The controller sets IEVENT[DPE], but otherwise continues reception until halted explicitly.
Babbling receive error	A frame is received that exceeds the MAC's maximum frame length. The controller sets IEVENT[BABR] and continues.

13.6.4 TCP/IP Off-Load

Each eTSEC provides hardware support for accelerating the basic functions of TCP/IP packet transmission and reception. By default, these features are disabled and must be explicitly enabled through RCTRL and TCTRL. In this configuration, the eTSEC processes frames as vanilla Ethernet frames and none of the multi-ring QoS/CoS receive services or per-frame VLAN insertion and deletion are available. Operate eTSEC in this default configuration when using existing TCP/IP stack software that has not been modified to take advantage of TOE.

TOE can be enabled independently for Rx and Tx and at various levels. Receive TOE functions are controlled by RCTRL and transmit functions through a combination of TCTRL[TUCSEN] and the Tx frame control block.

On receive, according to RCTRL[PRSDEP], eTSEC can parse frames at layer 2 of the stack only (Ethernet headers and switching headers), layers 2 to 3 (including IPv4 or IPv6), or layers 2 to 4 (including TCP and UDP). TOE provides protocol header recognition, header verification (IPv4 header checksum verification), and TCP/UDP payload checksum verification including verification of associated pseudo-header checksums. For large frames off-load of checksum verification saves a significant fraction of the CPU cycles that would otherwise be spent by the TCP/IP stack. IP packet fragmentation and re-assembly, and TCP stream establishment and tear-down are not performed in hardware. The frame parser sets RQFPR[IPF] status flag encountering a fragmented frame. The frame parser in eTSEC searches a maximum of 512 bytes from the start of a received frame when attempting to locate headers; headers deeper than 512 bytes are assumed not to exist, and any associated receive status flags in the frame control block remain cleared.

On transmit, TOE provides IPv4 and TCP/UDP header checksum generation. Like receive TOE, checksum generation reduces CPU load significantly for TCP/IP stacks modified to exploit eTSEC TOE functions. The eTSEC does not checksum transmitted packets with IPv6 routing headers or calculate TCP/UDP checksums from IP fragments. If a transmitted TCP segment requires checksum generation but IPv6 extension headers would prevent eTSEC from calculating the pseudo-header checksum, software can calculate just the pseudo-header checksum in advance and supply it to the eTSEC as part of per-frame TOE configuration.

13.6.4.1 Frame Control Blocks

Frame control blocks (FCBs) are 8-byte blocks of TOE control and/or status data that are passed between software (driver and TCP/IP stack) and each eTSEC. A FCB always precedes the frame it applies to, and is present only when TOE functions are being used. As Figure 13-132 shows, the first BD of each frame points to the initial data buffer and the FCB. The initial data buffer must be at least 8 bytes long to contain the FCB without breaking it. Custom or received Ethernet preamble sequences also follow the FCB if preambles are visible.

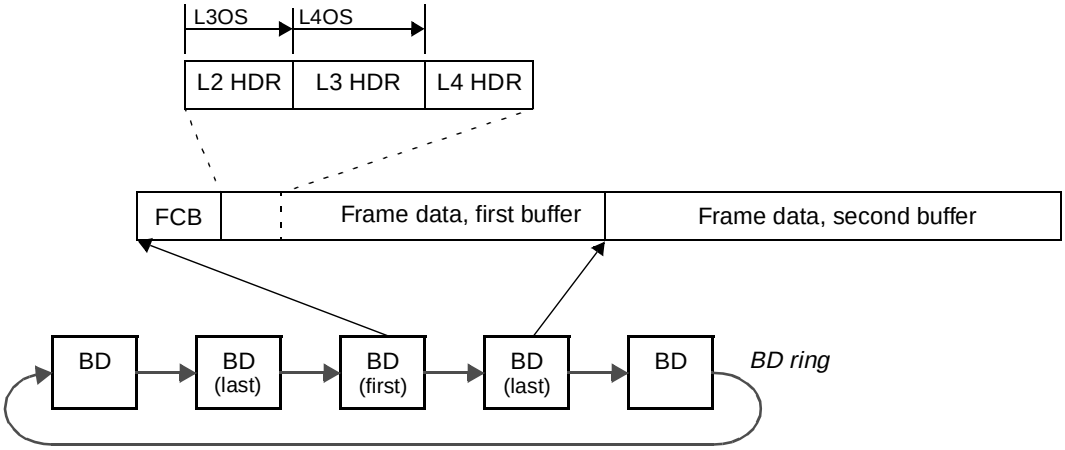


Figure 13-132. Location of Frame Control Blocks for TOE Parameters

For TxBD rings, FCBs are assumed present when the TxBD[TOE/UN] bit is set by user software. The eTSEC ignores the TxBD[TOE/UN] bit in all BDs other than those pointing to initial data buffers, therefore FCBs must not be inserted in second and subsequent data buffers. Since TxBD[TOE/UN] can be set under software discretion, TOE acceleration for transmit may be applied on a frame-by-frame basis.

In the case of RxBD rings, FCBs are inserted by the eTSEC whenever RCTRL[PRSDEP] is set to a non-zero value. Only one FCB is inserted per frame, in the buffer pointed to by the RxBD with bit F set. TOE acceleration for receive is enabled for all frames in this case.

13.6.4.2 Transmit Path Off-Load and Tx PTP Packet Parsing

TOE functions for transmit are defined by the contents of the Tx FCB. Figure 13-133 describes the definition for the Tx FCB.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	VLN	IP	IP6	TUP	UDP	CIP	CTU	NPH								
Offset + 2	L4OS								L3OS							
Offset + 4	PHCS															
Offset + 6	VLCTL															

Figure 13-133. Transmit Frame Control Block

13.6.4.3 Receive Path Off-Load

Upon receive, the Rx FCB returns the status of frame parse and TOE functions applied to the accompanying frame. Figure 13-134 describes the definition for the Rx FCB.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	VLN	IP	IP6	TUP	CIP	CTU	EIP	ETU					PERR			
Offset + 2			RQ						PRO							
Offset + 4																
Offset + 6	VLCTL															

Figure 13-134. Receive Frame Control Block

The contents of the Rx FCB are defined in Table 13-141.

Table 13-141. Rx Frame Control Block Descriptions

Bytes	Bits	Name	Description
0–1	0	VLN	VLAN tag recognized. This bit is set only if RCTRL[VLEX] is set. 0 No VLAN tag recognized. 1 IEEE Std. 802.1Q VLAN tag found; VLAN control word in VLCTL is valid.
	1	IP	IP header found at layer 3.RCTRL[PRSDEP] must be set to 10 or 11 in order to enable IP discovery. See also IP6 bit of FCB. 0 No layer 3 header recognized. 1 An IP header was recognized at layer 3; the IANA protocol identifier for the next header can be found in PRO; see PRO for more information. If S/W is relying on the RxFCB for the parse results, any RxFCB[IP] bits set with the corresponding RxFCB[PRO] = 0xFF indicates a fragmented packet (or that this packet had a back-to-back IPv6 routing extension header). Additionally, RQFPR[IPF] (see Section 13.5.3.3.8, “Receive Queue Filer Table Property Register (RQFPR)”) indicates that the packet was fragmented.
	2	IP6	IP version 6 header found at layer 3. 0 No IPv6 header was found. 1 The layer 3 header was an IPv6 header provided IP = 1.
	3	TUP	TCP or UDP header found at layer 4. RCTRL[PRSDEP] must be set to 10 or 11 in order to enable TCP/UDP discovery. 0 No layer 4 header recognized. 1 The layer 4 header was recognized as either TCP (PRO = 0x06) or UDP (PRO = 0x11).
	4	CIP	IPv4 header checksum checked. RCTRL[PRSDEP] must be set to 10 or 11 in order to enable IPv4 checksum verification. 0 IPv4 header checksum not verified, either because verification was disabled or a valid IPv4 header could not be located. 1 IPv4 header checksum was verified by the eTSEC, and bit EIP indicates result.
	5	CTU	TCP or UDP header checksum checked. RCTRL[PRSDEP] must be set to 11 in order to enable layer 4 checksum verification. 0 TCP or UDP header checksum not verified, either because verification was disabled or a valid TCP or UDP header could not be located. If a UDP header with zero checksum was located, this bit is cleared in accordance with RFC 768. 1 TCP or UDP header checksum was verified by the eTSEC, and ETU indicates result.
	6	EIP	IPv4 header checksum verification error. Not valid unless CIP = 1. 0 No checksum error in IPv4 header. 1 Error in header checksum only if IP = 1 and IP6 = 0.
0–1	7	ETU	TCP or UDP header checksum verification error. Not valid unless CTU = 1. 0 No checksum error in TCP or UDP header. 1 Error in header checksum only if PRO = 0x06 or PRO = 0x11.
	8–11	—	Reserved
	12–13	PER	Parse error. 00 No error in L2 to L4 parse 01 Reserved 10 Inconsistent or unsupported L3 header sequence 11 Reserved
	14–15	—	Reserved

Table 13-141. Rx Frame Control Block Descriptions (continued)

Bytes	Bits	Name	Description
2–3	0–1	—	Reserved
	2–7	RQ	Receive queue index. This index was selected by the eTSEC Rx Filer (from a matching Filer rule's RQCTRL[Q] field) when it accepted the associated frame. If filing is not enabled, RQ is zero. Note that the 3 least significant bits of RQ correspond with the RxBD ring index whenever RCTRL[FSQEN] = 0.
	8–15	PRO	If IP = 1, PRO is set as follows: • PRO=0xFF for a fragment header or a back to back route header • PRO=0x*nn* for an unrecognized header, where *nn* is the next protocol field • PRO=(TCP/UDP header), as defined in the IANA specification, if TCP or UDP header is found If IP = 0, PRO is undefined. Note that the eTSEC parser logic stops further parsing when encountering an IP datagram that has indicated that it has fragmented the upper layer protocol. This in general means that there is likely no layer 4 header following the IP header and extension headers. eTSEC leaves the RxFCB[PRO] and RQFPR[L4P] fields 0xFF in this case, which usually means that there was no IP header seen. In this case RxFCB[IP] and optionally RxFCB[IP6] is set. IP header checksumming operates and performs as intended. Most of the time, the eTSEC updates the RxFCB[PRO] field and RQFPR[L4P] fileds with whatever value was found in the protocol field of the IP header. See Section 13.5.3.3.8, "Receive Queue Filer Table Property Register (RQFPR)," for a description of RQFPR.
4–5	0–15	—	Reserved
6–7	0–15	VLCTL	VLAN control word as per IEEE Std. 802.1Q. The lower 12 bits comprise the VLAN identifier. Valid only if VLN = 1.

13.6.5 Quality of Service (QoS) Provision

This section describes the quality of service support features of this device. It includes a parser which extracts vital packet properties and passes then to the filer which essentially acts as a frame classifier.

13.6.5.1 Receive Parser

The receive parser parses the incoming frame data and generates filer properties and frame control block (FCB). The receive parser composes of the Ethernet header parser and L3/L4 parser.

The Ethernet header parser parses only L2 (ethertype) headers. It is enabled by RCTRL[PRSDEP] != 0. It has the following key features:

- Extraction of 48-bit MAC destination and source addresses
- Extraction and recognition of the first 2-byte ethertype field
- Extraction and recognition of the final 2-byte ethertype field
- Extraction of 2-byte VLAN control field
- Walk through MPLS stack and find layer 3 protocol
- Walk through VLAN stack and find layer 3 protocol
- Recognition of the following ethertypes for inner layer parsing
  - LLC and SNAP header

— JUMBO and SNAP header
— IPV4
— IPV6
— VLAN
— MPLSU/MPLSM
— PPPOES

For stack L2 (that is, more than one ethertypes) header, the Ethernet parser traverses through the header until it finds the last valid ethertype or the ethertype is unsupported. Below is a description of what the Ethernet header parser recognizes for stack L2 header.

Table 13-142. Supported Stack L2 Ethernet Headers

Column—Current L2 Ethertype Row—Next Supported L2 Ethertype	LLC/ SNAP	JUMBO/ SNAP	IPV4	IPV6	VLAN	MPLSU	MPLSM	PPOES
LLC/SNAP	N	N	Y	Y	Y	Y	Y	Y
JUMBO/SNAP	N	N	Y	Y	Y	Y	Y	Y
IPV4	N	N	N	N	N	N	N	N
IPV6	N	N	N	N	N	N	N	N
VLAN	Y	Y	Y	Y	Y	Y	Y	Y
MPLSU	N	N	Y*	Y*	N	y	Y	N
MPLSM	N	N	Y*	Y*	N	Y	Y	N
PPOES	N	N	Y	Y	N	Y	Y	N

Note: * means that it is the next protocol

The L3 parser is enabled by RCTRL[PRSDEP] = 10 or 11. It begins when the Ethernet parser ends and a valid IPv4/v6 ethertype is found. The L4 header is enabled by RCTRL[PRSDEP] = 11. It begins when the L3 parser ends and a valid TCP/UDP next protocol is found and no fragment frame is found. The primary functionalities of L3(IPv4/6) and L4(TCP/UDP) parsers are as follows:

- IP recognition (v4/v6, encapsulated protocol)
- IP header checksum verification
- IPv4/6 over IPv4/6 (tunneling)—parse headers and find layer 4 protocol
- IP layer 4 protocol/next header extraction
- Stop parsing on unrecognized next header/protocol
- IPv4 support

- IPv4 source and destination addresses
- 8-bit IPv4 type of service
- IP layer 4 protocol / next header support
  - IPV4
  - IPV4 Fragment. Parser stops after a fragment is found
  - TCP/UDP

- IPv6 support
  - The first 4 bytes of the IPv6 source address extraction
  - The first 4 bytes of the IPv6 destination address extraction
  - IPv6 source address hash for pseudo header calculation
  - IPv6 destination address hash for pseudo header calculation
  - 8-bit IPv6 traffic class field extraction
  - Payload length field extraction
  - IP layer 4 protocol/next header support
    - IPV6
    - IPV6 fragment. Parser stops after a fragment is found
    - IPV6 route
    - IPV6 hop/destination
    - TCP/UDP

- L4 (TCP/UDP) support
  - Extraction of 16-bit source port number extraction
  - Extraction of 16-bit destination port number extraction
  - TCP checksum calculation (including pseudo header)
  - UDP checksum calculation if the checksum field is not zero (including pseudo header)

13.6.5.2 Receive Queue Filer

The receive queue filer receives protocol header properties extracted from the incoming frame by the eTSEC frame parse engine. A property is defined to be a field extracted from a packet header, such as a TCP port number or VLAN identifier. As soon as the last identifiable header has been recognized, the filer commences searching the receive queue filer table, comparing properties in the table against properties extracted from the frame. This table is illustrated in Figure 13-135. Software populates the table with property values, stored to the RQPROP field, and indicates how to match and interpret the properties by setting flags in the RQCTRL field. The eTSEC memory map provides access to these fields by way of an address register (RQFAR) and two porthole registers (RQFCR and RQFPR).

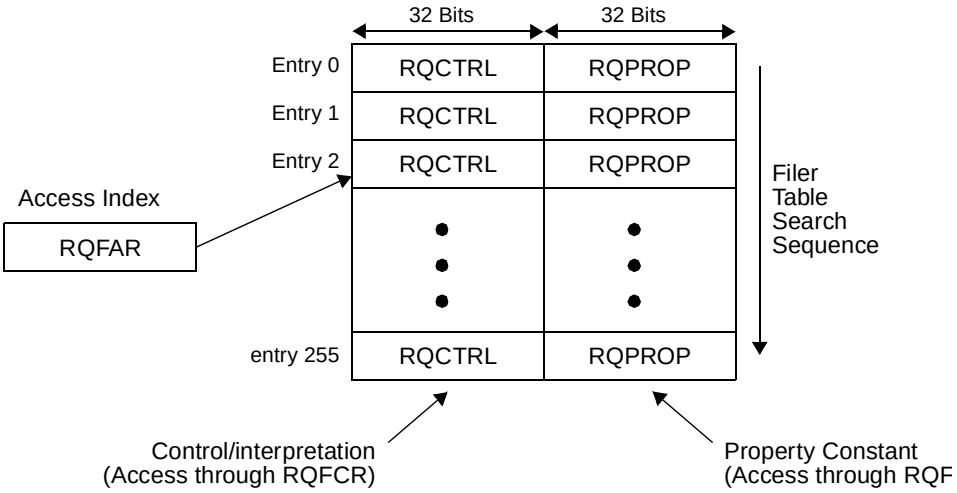


Figure 13-135. Structure of the Receive Queue Filer Table

13.6.5.2.1 Filing Rules

Unless the filer is disabled, every received frame from the Ethernet MAC or FIFO interface initiates a search of the receive queue filer table, starting at entry 0. The table search is terminated as soon as an entry is found whose contents match a property of the frame. Accordingly, software must guarantee that at least one entry results in a match—even if only to set a default receive queue index.

Since eTSEC searches the table at a rate of two entries every system clock cycle, all 256 entries can be searched in the time taken to receive a 64-byte Ethernet frame. However, for frames received through the 16-bit FIFO interface, fewer than 256 entries may be the maximum that can be searched while receiving a 40-byte packet. For example, with the 16-bit FIFO operating at 155 MHz, a 333-MHz eTSEC is limited to a maximum of 88 filing table entries.

Each entry of the receive queue filer table specifies a simple match rule for determining how to process the received frame. The elements of a filing rule, expressed in the RQCTRL and RQPROP fields, are summarized as follows:

- The PID field in RQCTRL identifies what property is being matched against RQPROP. The eTSEC supports 16 properties, some of which are different portions of the same header field. Reserved or unused bits in RQPROP are read as zero. See Section 13.5.3.3.8, “Receive Queue Filer Table Property Register (RQFPR),” on page 13-59 for a list of all properties and their associated PID values.
- The Q field in RQCTRL identifies which one of 64 virtual receive queues the frame should be filed to (sent through DMA) in the event of a filing rule match that accepts the frame. The physical RxBD ring this queue maps to is controlled by the RCTRL[FSQEN] bit. If RCTRL[FSQEN] = 0, the three least significant bits of the Q field indicate which physical RxBD ring hosts the queue. If RCTRL[FSQEN] = 1, RxBD ring 0 hosts all receive queues, but the RxFCB[RQ] field allows software to distinguish queues by ID. In all cases if Q maps to a RxBD ring that is not currently enabled, the frame is discarded with an IEVENT[FIQ] error.

- The REJ field in RQCTRL controls whether the frame is to be rejected (REJ = 1) or filed (REJ = 0) upon a filing rule match. Rejected frames occupy Rx FIFO space, but do not consume memory bus cycles.
- The CMP field in RQCTRL determines how property PID is compared against RQPROP. Equality, inequality, greater-or-equal, and less-than compares are available.
- The AND field in RQCTRL allows more than one comparison in a sequence to be chained together as a Boolean AND condition. Setting AND = 1 defers evaluation of the rule until the next entry has been matched, which may, in turn, have AND set. If any comparison involving AND = 1 fails, the entire chained sequence fails. A typical use for AND is to combine a pair of comparisons in a range match; the first such entry has AND = 1, the second has AND = 0 and its values of Q and REJ take effect.
- The CLE field in RQCTRL offers a way to bracket a set of consecutive—perhaps related—rules into a rule cluster. A cluster must be preceded by a guard rule, which simply determines whether the cluster rules can be evaluated. If the guard rule succeeds and its last entry has both CLE = 1 and AND = 1, the cluster rules that follow are enabled. The cluster ends at the first entry where CLE = 1 and AND = 0, which may also belong to a rule that files or rejects a frame. If the guard rule fails, all rules in the cluster are skipped, including mask_register assignments. Clusters must not be nested.

13.6.5.2.2 Comparing Properties with Bit Masks

By default, extracted properties are compared arithmetically according to the CMP field in each RQCTRL word. This permits point value matches in each table entry, and range checks across a pair of table entries combined with the AND attribute in RQCTRL. However, inspection of the parse flags, Ethernet preamble, and IP addresses typically requires ‘don’t care’ bit fields in the properties to be cleared as part of the comparison. The eTSEC provides a dedicated 32-bit register, known as the mask_register, for performing such masking operations. At the start of each table search by the filer, mask_register is reset to 0xFFFF_FFFF, which ensures that no masking occurs.

Filer rules may be configured to assign specific bit patterns to mask_register. Such rules can be configured to either match always (useful for implementing a default rule and specifying an associated receive queue), or fail always (which prevents termination of the filer table search). Once mask_register has been assigned, it retains its value until it is reassigned or the table search terminates. All properties are non-destructively bit-wise ANDed with mask_register prior to comparison in subsequent rules, which allows an entire cluster of rules to make use of a common mask. Individual masks for specific rules can also be created simply by combining a mask_register assignment (match always form) with a regular rule using the AND attribute.

To create a mask_register assignment rule, it is necessary to select PID = 0 in RQCTRL, and choose CMP such that the rule either matches (CMP = 01) or fails (CMP = 11). In this entry, RQPROP is then considered to be the assigned bit vector.

13.6.5.2.3 Special-Case Rules

It is frequently useful to create rules that are guaranteed to succeed or fail, specifically to enforce a default filing decision or act as null entries. Suggested constructions for such rules are shown in Table 13-143.

Table 13-143. Special Filer Rules

Rule Description	RQCTRL Fields						RQPROP Word	RQCTRL Word[1]
	CLE	REJ	AND	Q	CMP	PID		
Default file—Always file frame to ring Q	0	0	0	Q	01	0000	0x0000_0000	0x0000_*qq*20
Default reject—Always discard frame	0	1	0	000_000	01	0000	0x0000_0000	0x0000_0120
Empty rule in AND—Always matches	0/1[2]	0	1	000_000	01	0000	0xFFFF_FFFF	0x0000_00A0
Empty rule in rule set—Always fails	0/1[3]	0	0	000_000	11	0000	0xFFFF_FFFF	0x0000_0060

1 Hexadecimal digits *qq* denotes field Q shifted left 2 bits.

2 Set CLE = 1 if the empty rule guards a cluster.

3 Set CLE = 1 if the empty rule occurs at the end of a cluster.

13.6.5.2.4 Filer Interrupt Events

The filer can produce two interrupt events in IEVENT. Event FIR indicates an error condition where the filer was unable to provide a definite result, either because no rule in the table succeeded, or because frames arrived too rapidly to complete searching of the table. Event FIQ indicates that the filer accepted a frame to a RxBD ring that was not enabled in RQCTRL (this can also occur if the filer is disabled, but RxBD ring 0—default queue or FSQEN mode queue—is not enabled). FIQ is also asserted in the case where no rule in the entire table succeeded. The various combinations of these interrupt events and their interpretation appear in Table 13-144.

Table 13-144. Receive Queue Filer Interrupt Events

IEVENT[FIR]	IEVENT[FIQ]	Description
0	0	No error. The filer successfully rejected or filed a frame.
0	1	Illegal queue error. The filer accepted a frame to a RxBD ring that is disabled (including ring 0 if filing is disabled).
1	0	Partial search error. The filer did not have sufficient time to complete its search of the filer table.
1	1	No matching rule error. The filer searched all 256 entries of the filer table without finding a rule that succeeds.

13.6.5.2.5 Setting Up the Receive Queue Filer Table

The eTSEC frame parser always provides values for all properties, even where the relevant headers are not available. In the latter case, the filer is given default properties that can be used to avoid conflict with normal, defined property values. Accordingly, the rules in the filer table can be partitioned into rule sets such that if all rules in a given set fail (due to headers being unavailable), lower priority rule sets can be subsequently searched until either a rule set provides a match or a single default—catch-all—rule specifies a definite receive queue. For example, an 802.1p priority rule set may be followed by an IP TOS rule set,

followed by a default rule; thus, if no VLAN tag appears in the received frame, the TOS rules are checked, or the default is activated should no IP header be present.

The rule cluster feature is used to conditionalize evaluation of rule sets. Typically, this avoids evaluating rules based on properties that may not be valid or relevant to the filing or filtering decision. For example, TCP-related rules might be clustered behind a guard rule that checks that a TCP header has appeared and the IP address matches our home address. Property 1—the parse flags property—is provided specifically to check the characteristics of the received frame and the parser error status. The mask_register is typically assigned beforehand to extract specific flags, in which case care should be taken that mask_register be reassigned an appropriate mask vector for following comparisons.

In many cases it is possible to write the entire filer table before using eTSEC, as the rule set is static. However, dynamic rule updates can be supported by pre-allocating partially instantiated rule sets, which software rewrites as necessary. Rules that are not instantiated should be composed of empty entries, as indicated in Table 13-143. In many cases empty entries can be overwritten by software without stopping eTSEC's receive function.

13.6.5.2.6 Filer Example—802.1p Priority Filing

This example, shown in Table 13-145, illustrates how to file frames according to layer 2 802.1p priority. This matches against property 1001, comparing each specific priority level in order to associate them with a RxBD ring index. Note that if a VLAN tag does not appear in the frame, the parser passes priority 0 to the filer, which always matches the rule at entry 7 and terminate the table search.

Table 13-145. Filer Table Example—802.1p Priority Filing

Table Entry	RQCTRL Fields						RQPROP	Comment	RQCTRL Word
	CLE	REJ	AND	Q	CMP	PID			
0	0	0	0	000_000	00	1001	0x0000_0007	File priority 7 to ring 0	0x0000_0009
1	0	0	0	000_001	00	1001	0x0000_0006	File priority 6 to ring 1	0x0000_0409
2	0	0	0	000_010	00	1001	0x0000_0005	File priority 5 to ring 2	0x0000_0809
3	0	0	0	000_011	00	1001	0x0000_0004	File priority 4 to ring 3	0x0000_0C09
4	0	0	0	000_100	00	1001	0x0000_0003	File priority 3 to ring 4	0x0000_1009
5	0	0	0	000_101	00	1001	0x0000_0002	File priority 2 to ring 5	0x0000_1409
6	0	0	0	000_110	00	1001	0x0000_0001	File priority 1 to ring 6	0x0000_1809
7	0	0	0	000_111	00	1001	0x0000_0000	File undefined 802.1p or priority 0 to ring 7—Default always matches	0x0000_1C09

13.6.5.2.7 Filer Example—IP Diff-Serv Code Points Filing

This example demonstrates use of rule priority for determining class selector codepoints (RFC 2474) from the IP TOS property. An example filer table is shown in Table 13-146. The example relies on the fact that the first rule matched terminates the search, hence successively lower Diff-Serv codepoint ranges can be compared in each step until the default (zero or greater) range is reached. By default, property 1010 (IP TOS) takes the value 0x00 if no IP headers were recognized, therefore the table search always terminates.

Table 13-146. Filer Table Example—IP Diff-Serv Code Points Filing

Table Entry	RQCTRL Fields						RQPROP	Comment	RQCTRL Word
	CLE	REJ	AND	Q	CMP	PID			
0	0	0	0	001_000	01	1010	0x0000_00E0	File class 7 to queue 8 (TOS >= 0xE0)	0x0000_202A
1	0	0	0	001_001	01	1010	0x0000_00C0	File class 6 to queue 9 (TOS >= 0xC0)	0x0000_242A
2	0	0	0	001_010	01	1010	0x0000_00A0	File class 5 to queue 10 (TOS >= 0xA0)	0x0000_282A
3	0	0	0	001_011	01	1010	0x0000_0080	File class 4 to queue 11 (TOS >= 0x80)	0x0000_2C2A
4	0	0	0	000_100	01	1010	0x0000_0060	File class 3 to queue 4 (TOS >= 0x60)	0x0000_102A
5	0	0	0	001_100	01	1010	0x0000_0040	File class 2 to queue 12 (TOS >= 0x40)	0x0000_302A
6	0	0	0	010_100	01	1010	0x0000_0020	File class 1 to queue 20 (TOS >= 0x20)	0x0000_502A
7	0	0	0	011_100	01	1010	0x0000_0000	File class 0 to queue 28 (TOS >= 0x00) or file to ring 4 by default	0x0000_702A

13.6.5.2.8 Filer Example—TCP and UDP Port Filing

This example demonstrates rule clusters and AND-combined entries for filing packets based on transport protocol and well-known port numbers in a termination application. An example filer table is shown in Table 13-147. The example contains two clusters; the first is entered only for TCP packets, the second is entered only for UDP packets. A default filing rule catches the case where neither TCP nor UDP headers are found. Each cluster compares source port number (property 1111) against a list of server ports, and files the packets accordingly. Note that entries 1 and 2 form an AND rule for checking that the port number >= 20 and port number < 22. Entries 4 and 5 are initially set up to always fail (zero port number), and thus comprise empty entries that can be used at a later time.

Table 13-147. Filer Table Example—TCP and UDP Port Filing

Table Entry	RQCTRL Fields						RQPROP	Comment	RQCTRL Word
	CLE	REJ	AND	Q	CMP	PID			
0	1	0	1	000_000	00	1011	0x0000_0006	Enter cluster if layer 4 is TCP	0x0000_028B
1	0	0	1	000_000	01	1111	0x0000_0014	AND rule—FTP from TCP ports 20 and 21: file to ring 2	0x0000_00AF
2	0	0	0	000_010	11	1111	0x0000_0016		0x0000_086F
3	0	0	0	000_011	00	1111	0x0000_0017	telnet from TCP port 23: file to ring 3	0x0000_0C0F
4	0	0	0	000_000	00	1111	0x0000_0000	*empty entry reserved for future use*	0x0000_000F
5	0	0	0	000_000	00	1111	0x0000_0000	*empty entry reserved for future use*	0x0000_000F
6	1	0	0	000_001	01	0000	0x0000_0000	end cluster; default TCP: file to ring 1	0x0000_0620
7	1	0	1	000_000	00	1011	0x0000_0011	Enter cluster if layer 4 is UDP	0x0000_028B
8	0	0	0	000_101	00	1111	0x0000_0801	NFS from UDP port 2049	0x0000_140F
9	0	0	0	000_111	00	1111	0x0000_0208	Route from UDP port 520	0x0000_000F
10	0	0	0	000_110	00	1111	0x0000_0045	TFTP from UDP port 69	0x0000_180F

Table 13-147. Filer Table Example—TCP and UDP Port Filing (continued)

Table Entry	RQCTRL Fields						RQPROP	Comment	RQCTRL Word
	CLE	REJ	AND	Q	CMP	PID			
11	1	0	0	000_100	01	0000	0x0000_0000	End cluster; default UDP: file to ring 4	0x0000_1220
12	0	0	0	000_000	01	0000	0x0000_0000	By default, file to ring 0	0x0000_0020

13.6.5.3 Transmission Scheduling

Each eTSEC can maintain multiple TxBD rings (or transmission queues) to satisfy QoS requirements. The ability to choose from a number of transmission streams dynamically is especially important during periods of network congestion. Certain application such as voice and video streaming are delay sensitive, but loss insensitive. For instance, VoIP applications require little bandwidth, but are highly sensitive to latency. Conversely, FTP or SMTP protocols are delay insensitive, but loss sensitive.

eTSEC has a transmission scheduler that implements a programmable QoS regime. The scheduler is responsible for choosing which of the prefetched TxBDs shall be processed next, and accordingly issuing DMA requests to service the data stream described by the chosen BD(s). The scheduler cycle is one of:

1. decide on a TxBD queue,
2. transmit exactly one frame from that queue, and
3. return to deciding on another queue, in step 1.

If TCTRL[TXSCHED] is set to 00, no transmission scheduling occurs, and only TxBD ring 0 is polled for new data to transmit, with DMACTRL controlling waiting or polling. TCTRL[TXSCHED], if not zero, can be programmed to invoke one of two scheduling algorithms, namely priority-based queuing (PBQ), and modified weighted round-robin queuing (MWRR). In all cases where TCTRL[TXSCHED] is not zero, the scheduler can choose from among 1 to 8 TxBD rings per eTSEC, with individual rings being enabled by the setting of TQUEUE[EN0–EN7] bits. For example, TxBD rings 3, 4, and 7 may be enabled for scheduling by setting EN3, EN4, and EN7, and clearing all other EN bits.

13.6.5.3.1 Priority-Based Queuing (PBQ)

PBQ is the simplest scheduler decision policy. The enabled TxBD rings are assigned a priority value based on their index. Rings with a lower index have precedence over rings with higher indices. For example, TxBD ring 0 has higher priority than TxBD ring 1, and TxBD ring 1 has higher priority than TxBD ring 2, and so on.

The scheduling decision is then achieved as follows:

```
loop
      priority_ring = null;
      ring = 0;
      while priority_ring == null and ring <= 7 loop
       if enabled(ring) and not ring_empty(ring) then
              priority_ring = ring;
       endif
       ring = ring + 1;
      endloop
      if priority_ring >= 0 then
```

```
        while not ring_empty(priority_ring) loop
               transmit_frame(priority_ring);
        endloop
      endif
endloop
```

In practice a protocol stack or device driver can abuse PBQ by attempting to queue too much traffic onto high priority rings. It is recommended that the highest priority ring should normally not be used at all except for frames requiring the utmost urgent transmission. This allows emergency traffic to overtake backlogged queues out of sequence.

13.6.5.3.2 Modified Weighted Round-Robin Queuing (MWRR)

eTSEC implements a modified weighted round-robin scheduling algorithm across all enabled TxBD rings when TCTRL[TXSCHED] = 10. In MWRR, the weights in the TR03WT and TR47WT registers determine the ideal size of each transmit slot, as measured in multiples of 64 bytes. Thus, to set a transmit slot of 512 bytes, a weight of 512/64 or 8 needs to be set for the ring. In this mode TxBD rings 1–7 are selected in round-robin fashion, whereas TxBD ring 0, if enabled with ready data for transmission, is always selected in between other rings so as to expedite transmission from ring 0.

The scheduling decision is then achieved as follows:

```
for ring = 1..7 and enabled(ring) loop
      credit[ring] = 0;
endloop
for ring = 1..7 and enabled(ring) loop
      if not ring_empty(0) then
        credit[0] = credit[0] + weight[0];
        while credit[0] > 0 loop
               transmit_frame(0);
               credit[0] = credit[0] - frame_size;
               if ring_empty(0) then
                credit[0] = 0;
               endif
        endloop
      endif
      if not ring_empty(ring) then
        credit[ring] = credit[ring] + weight[ring];
      endif
      while credit[ring] > 0 loop
        transmit_frame(ring);
        credit[ring] = credit[ring] - frame_size;
        if ring_empty(ring) then
               credit[ring] = 0;
        endif
      endloop
endloop
```

The algorithm checks registers TQUEUE[EN0–EN7] for `enabled()`, TSTAT[THLT0–THLT7] for `ring_empty()`, and TR*x*WT for `weight()`. For TxBD ring *k*, having a weight WT*k*, the long term average throughput for that ring is:

rate of queue[k] (K = 1 to 7) = (available bandwidth) * WT*k*/(sum(WT*i*) + 6WT0)

rate of queue(0) = (available bandwidth) * 7 * WT0/(sum(WT*i*) + 6WT0)
where *i* = 0 to 7

13.6.6 Lossless Flow Control

The eTSEC DMA subsystem is designed to be able to support simultaneous receive and transmit traffic at gigabit line rates. If the host memory has sufficient bandwidth to support such line rates, then the principle cause of overflow on receive traffic is due to a lack of Rx BDs. Thus, the long term receive throughput is determined by the rate at which software can process receive traffic. If a user desires to prevent dropped packets, they can inform the far-end link to stop transmission while the software processing catches up with the backlog.

To avoid overflow in the latter case, back pressure must be applied to the far-end transmitter before the Rx descriptor controller encounters a non-empty BD and halts with a BSY error. As there is lag between application of back-pressure and response of the far-end, the pause request must be issued while there are still BDs free in the ring. In the traditional eTSEC descriptor ring programming model, there is no way for hardware to know how many free BDs are available, so software must initiate any pause requests required during operation. If software is backlogged, the request may be not be issued in time to prevent BSY errors. To allow the eTSEC to generate the pause request automatically, additional information (a pointer the last free BD and ring length) is required.

13.6.6.1 Back Pressure Determination through Free Buffers

Ultimately, the rate of data reception is determined by how quickly software can release buffers back into the receive ring(s). Each time a buffer is freed, the associated BD has its empty bit set and hardware is free to consume both. Thus the number of free BDs in a given Rx ring indicates how close hardware is to the end of that ring. To prevent data loss, back pressure should be applied when the number of free BDs drops below some critical level. The number of BDs that can be consumed by an incoming packet stream while back-pressure takes effect is determined by several factors, such as: receive traffic profile, transmit traffic profile, Rx buffer size, physical transmission time between eTSEC and far-end device and intra-device latency. Theoretically, the worst case is as follows:

$$\text{FreeBDsRequired} = \frac{\text{MaxFrameSize}}{\text{MinFrameSize} + \text{IFG}} + \frac{\text{MaxFrameSize}}{\text{RxBufferSize}} + \text{LinkDelay}$$

This case comes about when:

- The eTSEC has just started transmitting a large frame and thus cannot send out a pause frame
- Upon reception of the pause request the far-end has just started transmission of a large frame
- The eTSEC receives a burst of short frames with minimum inter-frame-gap (96bit times for ethernet)

Once the user has determined the worst case scenario for their application, they program the required free BD threshold into the eTSEC (through RQPRM[PBTHR]). Since different BD rings may have different sizes and expected packet arrival rates, a separate threshold is provided for each active ring. It is recommended that a threshold of at least fourBDs is the practical minimum for gigabit ethernet links.

For the Rx descriptor controller to determine the number of free BDs remaining in the ring, it needs to know the following:

1. The location of the current BD being used by hardware
2. The location of the last BD that was released (freed) by software
3. The length of the Rx BD ring.

For each active ring, the current BD pointer (RBPTR*n*) is maintained by the eTSEC. Software knows both the size of the Rx ring and the location of the last freed BD. By providing the eTSEC with those values (through RQPRM[LEN] and RFBPTR respectively) the eTSEC always know how many receive buffers are available to be consumed by incoming data.

The number of guaranteed free BDs in the ring is then determined by:

When RFBPTRn < RBPTR*n*

$$FreeBDs = RQPRMn[LEN] - RBPTRn + RFBPTRn$$

When RFBPTRn > RBPTR*n*

$$FreeBDs = RFBPTRn - RBPTRn$$

When RBPTR*n* = RFBPTR*n* the number of free BDs in the ring is either one (since RFBPTR*n* points to a free BD) or equal to the ring length. Since the BD pointed to by RBPTR*n* may be either in use or about to be used, it is not considered in the free BD count. To resolve the case where the two pointers collide, the following logic applies:

If RBASE*n* was updated and thus initializes both RBPTR*n* and RFBPTR*n*, the ring is deemed empty.

If RFBPTR*n* is updated by a software write and matches RBPTR*n*, the ring is deemed empty.

If HW updates RBPTR*n* and the result matches RFBPTR*n*, the ring is deemed to have one BD remaining. Upon writing this BD back to memory (indicating the buffer is occupied) the ring is deemed to be full.

Important. There is a possibility that if software is severely backlogged in updating RFBPTR*n*, the hardware could wrap around the ring entirely, consume exactly the remaining number of BDs and not halt with a BSY error. If software then increments RFBPTR*n* to the next address (thereby equalling RBPTR*n*), the hardware assumes the ring is now empty (when in fact there is only a single BD freed up). This results in the hardware failing to maintain back pressure on the far end. Upon software incrementing RFBPTR*n* a subsequent time, the wrap condition is successfully detected and hardware recognizes a nearly full ring (rather than a nearly empty one). Since software can increment RFBPTR*n* by any amount, it is not possible for hardware to determine in this case whether the user has cleared the entire ring or just one BD. Users can eliminate the possibility of this condition occurring by ensuring that RFBPTR*n* is incremented by at least two BDs each time (that is, clear at least two buffers whenever the RxBD unload routine is called).

Once the eTSEC determines that this threshold has been reached, back pressure is applied accordingly. The type of back pressure that is applied varies according to the physical interface that is used.

- **Half duplex Ethernet:** No support in this mode.

- **Full duplex Ethernet:** An IEEE 802.3 PAUSE frame (see sect. 13.6.3.9/13-154) is issued as if the TCTRL[TFC_PAUSE] bit was set. An internal counter tracks the time the far end controller is expected to remain in pause (based on the setting of PTV[PT]). When that counter reaches half the value of PTV[PT], the eTSEC reissues a pause frame if the free BD calculation for any ring is below the threshold for that ring. For example, if PTV[PT] is set to 10 quanta, a pause frame is re-issued when five quanta have elapsed if the free BD threshold is still not met. A practical minimum for PTV[PT] of 4 quanta is recommended.
- **FIFO packet interface:** Link layer flow control is asserted through use of the RFC signal (CRS pin). Flow control is asserted for the entire time that free BD threshold is not met. The same mechanism is used for both GMII-style and encoded packet modes.

13.6.6.2 Software Use of Hardware-Initiated Back Pressure

13.6.6.2.1 Initialization

Software configures RBASE*n* and RQPRM*n*[LEN] according to the parameters for that ring. Then the number of free BDs that are required to prevent the eTSEC from automatically asserting flow control are programmed in RQPRM[FBTHR]. The receiver is then enabled.

Note: the act of programming RBASE*n* initializes RFBPTR*n* to the start of the of the ring. When the ring is in this initial empty state, there is no concept of a last freed BD. In this case, the calculated number of free BDs is the size of the ring. Since the BD that the hardware is currently pointing to is to be considered in-use, the free BD count is actually one higher than the total available. As soon as the hardware consumes a BD (by writing it back to memory), RBPTR*n* advances and the free BD count reflects the correct number of available free BDs.

13.6.6.2.2 Operation

As software frees BDs from the ring, it writes the physical address of the BD just freed to RFBPTR*n*. The eTSEC asserts flow control if the distance (using modulo arithmetic) between RBPTR*n* and RFBPTR*n* is < RQPRM*n*[FBTHR]. In multi-ring operation, if the free BD count of **any** active ring drops below the threshold for that ring, flow control is asserted. Once enough BDs are freed for **all** active rings to meet their respective free BD thresholds, application of back pressure cases.

Note: The eTSEC does not issue an exit pause frame (that is, pause frame with PTV of 0x0000) once all active rings have sufficient BDs. Instead, it waits for the far-end pause timer to expire and start re-transmission.

13.6.7 Buffer Descriptors

The eTSEC buffer descriptor (BD) is modeled after the MPC8260 Fast Ethernet controller BD for ease of reuse across the PowerQUICC network processor family. Drawing from the MPC8260 FEC BD programming model, the eTSEC descriptor base registers point to the beginning of BD rings. The eTSEC BD also expands upon the MPC8260 BD model to accommodate the eTSEC's unique features. However, the 8-byte data BD format is designed to be compatible with the existing MPC8260 BD model.

13.6.7.1 Data Buffer Descriptors

Data buffers are used in the transmission and reception of Ethernet frames (see Figure 13-136). Data BDs encapsulate all information necessary for the eTSEC to transmit or receive an Ethernet frame. Within each data BD there is a status field, a data length field, and a data pointer. The BD completely describes an Ethernet packet by centralizing status information for the data packet in the status field of the BD and by containing a data BD pointer to the location of the data buffer. Software is responsible for setting up the BDs in memory. Because of pre-fetching, a minimum of four buffer descriptors per ring are required. This applies to both the transmit and the receive descriptor rings. Transmit rings are limited to a maximum size of 65536 BDs due to BD and frame data prefetching. Software also must have the data pointer pointing to a legal memory location. Within the status field, there exists an ownership bit which defines the current state of the buffer (pointed to by the data pointer). Other bits in the status field of the buffer descriptor are used to communicate status/control information between the eTSEC and the software driver.

Because there is no next BD pointer in the transmit/receive BD (see Figure 13-137), all BDs must reside sequentially in memory. The eTSEC increments the current BD location appropriately to the next BD location to be processed. There is a wrap bit in the last BD that informs the eTSEC to loop back to the beginning of the BD chain. Software must initialize the TBASE and RBASE registers that point to the beginning transmit and receive BDs for eTSEC.

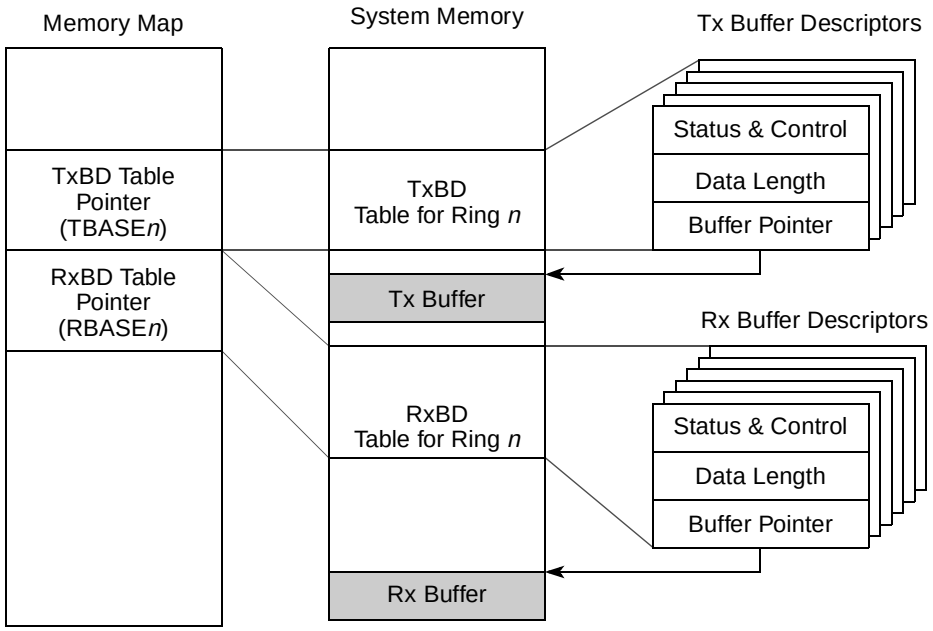


Figure 13-136. Example of eTSEC Memory Structure for BDs

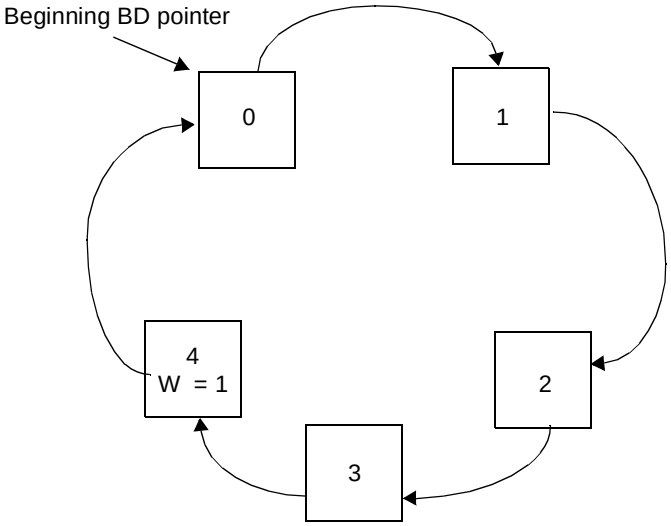


Figure 13-137. Buffer Descriptor Ring

13.6.7.2 Transmit Data Buffer Descriptors (TxBD)

Data is presented to the eTSEC for transmission by arranging it in memory buffers referenced by the TxBDs. In the TxBD the user initializes the R, PAD, W, I, L, TC, PRE, HFE, CF, and TOE bits and the length (in bytes) in the first word, and the buffer pointer in the second word. Unused fields or fields written by the eTSEC must be initialized to zero. For transmission over the FIFO interface the Ethernet specific bits (PRE, DEF, HFE, LC, CF, RL and RC) have no meaning.

The eTSEC clears the R bit in the first word of the BD after it finishes using the data buffer. The transfer status bits are then updated. Additional transmit frame status can be found in statistic counters in the MIB block.

Software must expect eTSEC to prefetch multiple TxBDs, and for TCP/IP checksumming an entire frame must be read from memory before a checksum can be computed. Accordingly, the R bit of the first TxBD in a frame must not be set until at least one entire frame can be fetched from this TxBD onwards. If eTSEC prefetches TxBDs and fails to reach a last TxBD (with bit L set), it halts further transmission from the current TxBD ring and report an underrun error as IEVENT[XFUN]; this indicates that an incomplete frame was fetched, but remained unprocessed. The relevant TBPTR register points to the next unread TxBD following the error.

Figure 13-138 defines the TxBD.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	R	PAD/CRC	W	I	L	TC	PRE/DEF	0	HFE/LC	CF/RL	RC				TOE/UN	TR
Offset + 2	DATA LENGTH															
Offset + 4	TX DATA BUFFER POINTER															
Offset + 6																

Figure 13-138. Transmit Buffer Descriptor

The TxBD definition is interpreted by eTSEC hardware as if TxBDs mapped to C data structures in the manner illustrated by Figure 13-139.

```
typedef unsigned short uint_16; /* choose 16-bit native type */
typedef unsigned int uint_32; /* choose 32-bit native type */
typedef struct txbd_struct {
       uint_16 flags;
       uint_16 length;
       uint_32 bufptr;
} txbd;
```

Figure 13-139. Mapping of TxBDs to a C Data Structure

The TxBD fields are detailed in Table 13-148.

Table 13-148. Transmit Data Buffer Descriptor (TxBD) Field Descriptions

Offset	Bits	Name	Description
0–1	0	R	Ready, written by eTSEC and user. 0 The data buffer associated with this BD is not ready for transmission. The user is free to manipulate this BD or its associated data buffer. The eTSEC clears this bit after the buffer is transmitted or after an error condition is encountered. 1 The data buffer, which is prepared for transmission by the user, was not transmitted or is currently being transmitted. No fields of this BD may be written by the user once this bit is set.
	1	PAD/CRC	Padding for frames. (Valid only while it is set in the first BD and MACCFG2[PAD enable] is cleared). If MACCFG2[PAD enable] is set, this bit is ignored. 0 Do not add padding to short frames. 1 Add PAD to frames. PAD bytes are inserted until the length of the transmitted frame equals 64 bytes. Unlike the MPC8260 which PADs up to MINFLR value, the eTSEC PADs always up to the IEEE minimum frame length of 64 bytes. CRC is always appended to frames.
	2	W	Wrap. Written by user. 0 The next buffer descriptor is found in the consecutive location. 1 The next buffer descriptor is found at the location defined in TBASE.
	3	I	Interrupt. Written by user. 0 No interrupt is generated after this buffer is serviced. 1 IEVENT[TXB] or IEVENT[TXF] are set after this buffer is serviced. These bits can cause an interrupt if they are enabled (That is, IEVENT[TXBEN] or IEVENT[TXFEN] are set).
	4	L	Last in frame. Written by user. 0 The buffer is not the last in the transmit frame. 1 The buffer is the last in the transmit frame.

Table 13-148. Transmit Data Buffer Descriptor (TxBD) Field Descriptions (continued)

Offset	Bits	Name	Description
0–1	5	TC	Tx CRC. Written by user. (Valid only while it is set in first BD and TxBD[PAD/CRC] is cleared and MACCFG2[PAD/CRC enable] is cleared and MACCFG2[CRC enable] is cleared.) If MACCFG2[PAD/CRC enable] is set or MACCFG2[CRC enable] is set, this bit is ignored in ethernet modes. If FIFOCFG[CRCAPP] is set, this bit is ignored in FIFO modes 0 End transmission immediately after the last data byte with no hardware generated CRC appended, unless TxBD[PAD/CRC] is set. 1 Transmit the CRC sequence after the last data byte.
	6	PRE	Transmit user-defined Ethernet preamble. Written by user. Valid only if set in the first BD of a frame, and MACCFG2[PreAm TxEN] is set. 0 This frame does not contain Ethernet preamble bytes for transmission. 1 This frame includes a user-defined Ethernet preamble sequence prior to the destination address in the data buffer.
		DEF	Defer indication. The eTSEC updates this bit after transmitting a frame (TxBD[L] is set) 0 This frame was not deferred. 1 This frame did not have a collision before it was sent but it was sent late because of deferring
	7	—	Reserved
	8	HFE	Huge frame enable. Written by user. Valid only if set in the first BD of a frame and MACCFG2[Huge Frame] is cleared. If MACCFG2[Huge Frame] is set, this bit is ignored. 0 Truncate transmit frame if its length is greater than the MAC's maximum frame length. 1 Allow large frames to be transmitted without truncation.
		LC	Late collision. Written by the eTSEC. 0 No late collision. 1 A collision occurred after 64 bytes are sent. The eTSEC terminates the transmission and updates LC.
	9	CF	Control Frame. Written by user. Valid only if set in the first BD of a frame. 0 Regular frame; transmission is deferred when eTSEC is in PAUSE. 1 Control frame; transmission starts even if eTSEC is in PAUSE.
		RL	Retransmission Limit. Written by the eTSEC. 0 Transmission before maximum retry limit is hit. 1 The transmitter failed (max. retry limit + 1) attempts to successfully send a message due to repeated collisions. The eTSEC terminates the transmission and updates RL.
	10–13	RC	Retry Count. Written by the eTSEC. 0 The frame is sent correctly the first time. x One or more attempts where needed to send the transmit frame. If this field is 15, then 15 or more retries were needed. The Ethernet controller updates RC after sending the buffer.

Table 13-148. Transmit Data Buffer Descriptor (TxBD) Field Descriptions (continued)

Offset	Bits	Name	Description
0–1	14	UN	Underrun. Written by the eTSEC. 0 No underrun encountered (data was retrieved from external memory in time to send a complete frame). 1 The Ethernet controller encountered a transmitter underrun condition while sending the associated buffer. This could also have occurred in relation to a bus error causing IEVENT[EBERR]. The eTSEC terminates the transmission and updates UN.
		TOE	TCP/IP off-load enable. Written by user. Valid only if set in the first BD of a frame. 0 No TCP/IP off-load acceleration is applied to the frame prior to transmission. 1 eTSEC looks for a TOE Frame Control Block preceding the frame, and applies TCP/IP off-load acceleration as controlled by the FCB.
	15	TR	Truncation. Written by the eTSEC. Set in the last TxBD (TxBD[L] is set) when IEVENT[BABT] occurs for a frame (a frame length greater than or equal to the value set in the maximum frame length register is encountered, the HFE bit in the BD is cleared, and MACCFG2[Huge Frame] is cleared). The frame is sent truncated.
2–3	0–15	Data Length	Data length is the number of octets the eTSEC should transmit from this BD's data buffer. It is never modified by the eTSEC. This field must be greater than zero, as zero indicates a BD not ready.
4–7	0–31	TX Data Buffer Pointer	The transmit buffer pointer contains the address of the associated data buffer. The data buffer pointer for the first BD of a TxPAL-enabled frame must be aligned on an 8-byte boundary. There are no alignment restrictions for the data buffer pointers of the second or subsequent BDs of a TxPAL-enabled frame, or for non-TxPAL frames.

13.6.7.3 Receive Buffer Descriptors (RxBD)

In the RxBD the user initializes the E, I, and W bits in the first word and the pointer in second word. If the data buffer is used, the eTSEC modifies the E, L, F, M, BC, MC, LG, NO, CR, OV, and TR bits and writes the length of the used portion of the buffer in the first word. The M, BC, MC, LG, NO, CR, OV, and TR bits in the first word of the buffer descriptor are only modified by the eTSEC if the L (last BD in frame) bit is set. The first word of the RxBD contains control and status bits. For packets received over the FIFO interface, those bits which are ethernet specific (M, BC, MC, LG, NO, and TR) should be ignored. Its formats are detailed below.

The number of buffer descriptors in a ring is set using the W bit to indicate that the next buffer wraps back to the beginning of the ring. See Section 13.5.3.5.5, "Maximum Frame Length Register (MAXFRM)," for information on setting the size of the buffer ring.

Figure 13-140 defines the RxBD.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	E	RO1	W	I	L	F	0	M	BC	MC	LG	NO	SH	CR	OV	TR
Offset + 2	DATA LENGTH															
Offset + 4	RX DATA BUFFER POINTER															
Offset + 6																

Figure 13-140. Receive Buffer Descriptor

The RxBD definition is interpreted by eTSEC hardware as if RxBDs mapped to C data structures in the manner illustrated by Figure 13-141.

```
typedef unsigned short uint_16; /* choose 16-bit native type */
typedef unsigned int uint_32; /* choose 32-bit native type */
typedef struct rxbd_struct {
       uint_16 flags;
       uint_16 length;
       uint_32 bufptr;
} rxbd;
```

Figure 13-141. Mapping of RxBDs to a C Data Structure

Table 13-149 describes the fields of the RxBD.

Table 13-149. Receive Buffer Descriptor Field Descriptions

Offset	Bits	Name	Description
0–1	0	E	Empty, written by the eTSEC (when cleared) and by the user (when set). 0 The data buffer associated with this BD is filled with received data, or data reception is aborted due to an error condition. The status and length fields have been updated as required. 1 The data buffer associated with this BD is empty, or reception is currently in progress.
	1	RO1	Receive software ownership bit. This field is reserved for use by software. This read/write bit is not modified by hardware, nor does its value affect hardware.
	2	W	Wrap, written by user. 0 The next buffer descriptor is found in the consecutive location. 1 The next buffer descriptor is found at the location defined in RBASE.
	3	I	Interrupt, written by user. 0 No interrupt is generated after this buffer is serviced. 1 IEVENT[RXB] or IEVENT[RXF] are set after this buffer is serviced. This bit can cause an interrupt if enabled (IMASK[RXBEN] or IMASK[RXFEN]). If the user wants to be interrupted only if RXF occurs, then the user must disable RXB (IMASK[RXBEN] is cleared) and enable RXF (IMASK[RXFEN] is set).
	4	L	Last in frame, written by the eTSEC. 0 The buffer is not the last in a frame. 1 The buffer is the last in a frame.
	5	F	First in frame, written by the eTSEC. 0 The buffer is not the first in a frame. 1 The buffer is the first in a frame.
	6	—	Reserved
	7	M	Miss, written by the eTSEC. (This bit is valid only if the L-bit is set and eTSEC is in promiscuous mode.) This bit is set by the eTSEC for frames that were accepted in promiscuous mode, but were flagged as a “miss” by the internal address recognition; thus, while in promiscuous mode, the user can use the M-bit to quickly determine whether the frame was destined to this station. 0 The frame was received because of an address recognition hit. 1 The frame was received because of promiscuous mode.

Table 13-149. Receive Buffer Descriptor Field Descriptions (continued)

Offset	Bits	Name	Description
0–1	8	BC	Broadcast. Written by the eTSEC. (Only valid if L is set.) Is set if the DA is broadcast (FF-FF-FF-FF-FF-FF).
	9	MC	Multicast. Written by the eTSEC. (Only valid if L is set.) Is set if the DA is multicast and not BC.
	10	LG	Rx frame length violation, written by the eTSEC (only valid if L is set). A frame length greater than or equal to the maximum frame length was recognized; in this case LG is set regardless of the setting of MACCFG2[Huge Frame]. If MACCFG2[Huge Frame] is cleared, the frame is truncated to the value programmed in the maximum frame length register. This bit is valid only if the L bit is set. This bit is not set when in FIFO mode as truncation cannot occur.
	11	NO	Rx non-octet aligned frame, written by the eTSEC (only valid if L is set). A frame that contained a number of bits not divisible by eight was received. This bit cannot be set in FIFO mode.
	12	SH	Short frame, written by the eTSEC (only valid if L is set). A frame length less than the minimum 64B that is defined for ethernet. was recognized, provided RCTRL[RSF] is set. This bit should be disregarded in FIFO mode.
	13	CR	Rx CRC error, written by the eTSEC (only valid if L is set). This frame contains a CRC error and is an integral number of octets in length.This bit is also set if a receive code group error is detected.
	14	OV	Overrun, written by the eTSEC (only valid if L is set). A receive FIFO overrun occurred during frame reception. If this bit is set, the other status bits, M, LG, NO, CR and TR lose their normal meaning and are zero.
	15	TR	Truncation, written by the eTSEC (only valid if L is set). Set if the receive frame is truncated. This can happen if a frame length greater than the maximum frame length is received and MACCFG2[Huge Frame] is cleared. If this bit is set, the frame must be discarded and the other error bits must be ignored as they may be incorrect. This bit is not set when in FIFO mode as truncation cannot occur.
2–3	0–15	Data Length	Data length, written by the eTSEC. Data length is the number of octets written by the eTSEC into this BD's data buffer if L is cleared (the value is equal to MRBLR), or, if L is set, the length of the frame including CRC, FCB (if RCTRL[PRSDEP > 00), preamble (if MACCFG2[PreAmRxEn]=1), and any padding (RCTRL[PAL]).
4–7	0–31	RX Data Buffer Pointer	Receive buffer pointer, written by the user. The receive buffer pointer, which always points to the first location of the associated data buffer, must be 8-byte aligned. For best performance, use 64-byte aligned receive buffer pointer addresses. The buffer must reside in memory external to the eTSEC.

13.7 Initialization/Application Information

13.7.1 Interface Mode Configuration

This section describes how to configure the eTSEC in different supported interface modes. These include:

- MII
- RMII
- GMII

- RGMII
- TBI
- RTBI
- 8-bit FIFO
- 16-bit FIFO

The pinout, the data registers that must be initialized, as well as speed selection options are described.

13.7.1.1 MII Interface Mode

Table 13-150 describes the signal configurations required for MII interface mode.

Table 13-150. MII Interface Mode Signal Configuration

eTSEC Signals			MII Interface		
			Frequency [MHz] 25		
			Voltage [V] 3.3		
Signals	**I/O**	**No. of Signals**	**Signals**	**I/O**	**No. of Signals**
GTX_CLK	O	1	leave unconnected		
TX_CLK	I	1	TX_CLK	I	1
TxD[0]	O	1	TxD[0]	O	1
TxD[1]	O	1	TxD[1]	O	1
TxD[2]	O	1	TxD[2]	O	1
TxD[3]	O	1	TxD[3]	O	1
TxD[4]	O	1	leave unconnected		
TxD[5]	O	1	leave unconnected		
TxD[6]	O	1	leave unconnected		
TxD[7]	O	1	leave unconnected		
TX_EN	O	1	TX_EN	O	1
TX_ER	O	1	TX_ER	O	1
RX_CLK	I	1	RX_CLK	I	1
RxD[0]	I	1	RxD[0]	I	1
RxD[1]	I	1	RxD[1]	I	1
RxD[2]	I	1	RxD[2]	I	1
RxD[3]	I	1	RxD[3]	I	1
RxD[4]	I	1	not used		
RxD[5]	I	1	not used		
RxD[6]	I	1	not used		
RxD[7]	I	1	not used		

Table 13-150. MII Interface Mode Signal Configuration (continued)

eTSEC Signals			MII Interface		
			Frequency [MHz] 25		
			Voltage [V] 3.3		
Signals	**I/O**	**No. of Signals**	**Signals**	**I/O**	**No. of Signals**
RX_DV	I	1	RX_DV	I	1
RX_ER	I	1	RX_ER	I	1
COL	I	1	COL	I	1
CRS	I	1	CRS	I	1
Sum		25	**Sum**		16

Table 13-151 describes the shared signals of the MII interface.

Table 13-151. Shared MII Signals

eTSEC Signals	I/O	No. of Signals	MII Signals	I/O	No. of Signals	Function
MDIO	I/O	1	MDIO	I/O	1	Management interface I/O
MDC	O	1	MDC	O	1	Management interface clock
EC1_GTX_CLK125	I	1	not used	I	0	Reference clock for eTSEC1/2
EC2_GTX_CLK125	I	1	not used	I	0	Reference clock for eTSEC3/4
Sum		4	**Sum**		3	—

Table 13-152 describes the register initializations required to configure the eTSEC in MII mode.

Table 13-152. MII Mode Register Initialization Steps

Set Soft_Reset, MACCFG1[1000_0000_0000_0000_0000_0000_0000_0000]
Clear Soft_Reset, MACCFG1[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize MACCFG2, for MII, half duplex operation. Set I/F Mode bit, MACCFG2[0000_0000_0000_0000_0111_0001_0000_0100] (This example has Full Duplex = 0, Preamble count = 7, PAD/CRC append = 1)
Initialize ECNTRL, ECNTRL[0000_0000_0000_0000_0001_0000_0000_0000] (This example has Statistics Enable = 1)
Initialize MAC Station Address, MACSTNADDR2[0110_0000_0000_0010_0000_0000_0000_0000] Set station address to 02_60_8C_87_65_43, for example.

Table 13-152. MII Mode Register Initialization Steps (continued)

Initialize MAC Station Address, MACSTNADDR1[0100_0011_0110_0101_1000_0111_1000_1100] Set station address to 02_60_8C_87_65_43, for example.
Assign a Physical address to the TBI so as to not conflict with the external PHY Physical address, TBIPA[0000_0000_0000_0000_0000_0000_0000_0101] Set to 05, for example.
Reset the management interface. MIIMCFG[1000_0000_0000_0000_0000_0000_0000_0111]
Setup the MII Mgmt clock speed, MIIMCFG[0000_0000_0000_0000_0000_0000_0000_0101] set source clock divide by 14 for example to insure that MDC clock speed is not greater than 2.5 MHz
Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the eTSEC MII Mgmt bus is idle.
Set up the MII Mgmt for a write cycle to the external PHY Auxiliary Control and Status Register to configure the PHY through the Management interface (overrides configuration signals of the PHY). MIIMADD[0000_0000_0000_0000_0000_0000_0001_1100]
Perform an MII Mgmt write cycle to the external PHY Writing to MII Mgmt Control with 16-bit data intended for the external PHY register, MIIMCON[0000_0000_0000_0000_0000_0000_0000_0100]
Check to see if MII Mgmt write is complete Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.
Set up the MII Mgmt for a write cycle to the external PHY Extended PHY control register #1 to set up the interface mode selection. MIIMADD[0000_0000_0000_0000_0000_0000_0001_0111]
Perform an MII Mgmt write cycle to the external PHY. Write to MII Mgmt Control with 16-bit data intended for the external PHY register, MIIMCON[0000_0000_0000_0000_0000_0000_0000_0000]
Check to see if MII Mgmt write is complete. Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.
Set up the MII Mgmt for a write cycle to the external PHY Mode control register to set up the interface mode selection. MIIMADD[0000_0000_0000_0000_0000_0000_0000_0000]
Perform an MII Mgmt write cycle to the external PHY. Write to MII Mgmt Control with 16-bit data intended for the external PHY register, MIIMCON[0000_0000_0000_0000_00uu_00uu_0u00_0000] where u is user defined based on desired configuration.
Check to see if MII Mgmt write is complete Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.
If auto-negotiation was enabled in the PHY, check to see if PHY has completed Auto-Negotiation. Set up the MII Mgmt for a read cycle to PHY MII Mgmt register (write the PHY address and Register address), MIIMADD[0000_0000_0000_0000_0000_0000_0000_0001] The PHY Status register is at address 0x1 and in this case the PHY Address is 0x00.

Table 13-152. MII Mode Register Initialization Steps (continued)

Perform an MII Mgmt read cycle of Status Register. Clear MIIMCOM[Read Cycle]. Set MIIMCOM[Read Cycle]. (Uses the PHY address (0) and Register address (1) placed in MIIMADD register), When MIIMIND[BUSY]=0, read the MIIMSTAT register and check bit 10 (AN Done and Link is up) MIIMSTAT ---> [0000_0000_0000_0000_0000_0000_0010_0100] Other information about the link is also returned.(Extend Status, No pre, Remote Fault, An Ability, Link status, extend Ability)
Check auto-negotiation attributes in the PHY as necessary.
Clear IEVENT register, IEVENT[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize IMASK (Optional) IMASK[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize MACnADDR1/2 (Optional) MACnADDR1/2[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize GADDR*n* (Optional) GADDR*n*[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize RCTRL (Optional) RCTRL[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize DMACTRL (Optional) DMACTRL[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize (Empty) Transmit Descriptor ring and fill buffers with Data Initialize TBASE0–TBASE7, TBASE0–TBASE7[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]
Initialize (Empty) Receive Descriptor ring and fill with empty buffers Initialize RBASE0–RBASE7, RBASE0–RBASE7[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]
Enable Transmit Queues Initialize TQUEUE
Enable Receive Queues Initialize RQUEUE
Enable Rx and Tx, MACCFG1[0000_0000_0000_0000_0000_0000_0000_0101]

13.7.1.2 GMII Interface Mode

Table 13-153 describes the signal configurations required for GMII interface mode.

Table 13-153. GMII Interface Mode Signal Configuration

eTSEC Signal s			GMII Interface		
			Frequency [MHz] 125		
			Voltage [V] 3.3		
Signals	**I/O**	**No. of Signals**	**Signals**	**I/O**	**No. of Signals**
GTX_CLK	O	1	GTX_CLK	O	1
TX_CLK	I	1	TX_CLK	I	1
TxD[0]	O	1	TxD[0]	O	1
TxD[1]	O	1	TxD[1]	O	1
TxD[2]	O	1	TxD[2]	O	1
TxD[3]	O	1	TxD[3]	O	1
TxD[4]	O	1	TxD[4]	O	1
TxD[5]	O	1	TxD[5]	O	1
TxD[6]	O	1	TxD[6]	O	1
TxD[7]	O	1	TxD[7]	O	1
TX_EN	O	1	TX_EN	O	1
TX_ER	O	1	TX_ER	O	1
RX_CLK	I	1	RX_CLK	I	1
RxD[0]	I	1	RxD[0]	I	1
RxD[1]	I	1	RxD[1]	I	1
RxD[2]	I	1	RxD[2]	I	1
RxD[3]	I	1	RxD[3]	I	1
RxD[4]	I	1	RxD[4]	I	1
RxD[5]	I	1	RxD[5]	I	1
RxD[6]	I	1	RxD[6]	I	1
RxD[7]	I	1	RxD[7]	I	1
RX_DV	I	1	RX_DV	I	1
RX_ER	I	1	RX_ER	I	1
COL	I	1	not used		
CRS	I	1	not used		
Sum		25	**Sum**		23

Table 13-154 describes the shared signals of the GMII interface.

Table 13-154. Shared GMII Signals

eTSEC Signals	I/O	No. of Signals	GMII Signals	I/O	No. of Signals	Function
MDIO	I/O	1	MDIO	I/O	1	Management interface I/O
MDC	O	1	MDC	O	1	Management interface clock
EC1_GTX_CLK125	I	1	GTX_CLK125	I	1	Reference clock for eTSEC1/2
EC2_GTX_CLK125	I	1	GTX_CLK125	I	1	Reference clock for eTSEC3/4
Sum		4	**Sum**		4	—

Table 13-155 describes the register initializations required to configure the eTSEC in GMII mode.

Table 13-155. GMII Mode Register Initialization Steps

Set Soft_Reset, MACCFG1[1000_0000_0000_0000_0000_0000_0000_0000]
Clear Soft_Reset, MACCFG1[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize MACCFG2, for GMII, Full duplex operation. Set I/F Mode bit. MACCFG2[0000_0000_0000_0000_0111_0010_0000_0101] (This example has Full Duplex = 1, Preamble count = 7, PAD/CRC append = 1)
Initialize ECNTRL, ECNTRL[0000_0000_0000_0000_0001_0000_0000_0000] (This example has Statistics Enable = 1)
Initialize MAC Station Address, MACSTNADDR2[0110_0000_0000_0010_0000_0000_0000_0000] Set station address to 02_60_8C_87_65_43, for example.
Initialize MAC Station Address, MACSTNADDR1[0100_0011_0110_0101_1000_0111_1000_1100] Set station address to 02_60_8C_87_65_43, for example.
Assign a Physical address to the TBI so as to not conflict with the external PHY Physical address, TBIPA[0000_0000_0000_0000_0000_0000_0000_0101] Set to 05, for example.
Reset the management interface, MIIMCFG[1000_0000_0000_0000_0000_0000_0000_0111]
Setup the MII Mgmt clock speed, MIIMCFG[0000_0000_0000_0000_0000_0000_0000_0101] set source clock divide by 14 for example to insure that MDC clock speed is not greater than 2.5 MHz
Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the eTSEC MII Mgmt bus is idle.
Set up the MII Mgmt for a write cycle to the external PHY Auxiliary Control and Status Register to configure the PHY through the Management interface (overrides configuration signals of the PHY), MIIMADD[0000_0000_0000_0000_0000_0000_0001_1100]
Perform an MII Mgmt write cycle to the external PHY. Write to MII Mgmt Control with 16-bit data intended for the external PHY register, MIIMCON[0000_0000_0000_0000_0000_0000_0000_0100]

Table 13-155. GMII Mode Register Initialization Steps (continued)

Check to see if MII Mgmt write is complete. Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed
Set up the MII Mgmt for a write cycle to the external PHY Extended PHY control register #1 to set up the interface mode selection MIIMADD[0000_0000_0000_0000_0000_0000_0001_0111]
Perform an MII Mgmt write cycle to the external PHY. Write to MII Mgmt Control with 16-bit data intended for the external PHY register, MIIMCON[0000_0000_0000_0000_0000_0000_0000_0000]
Check to see if MII Mgmt write is complete. Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.
Set up the MII Mgmt for a write cycle to the external PHY Mode control register to set up the interface mode selection, MIIMADD[0000_0000_0000_0000_0000_0000_0000_0000]
Perform an MII Mgmt write cycle to the external PHY. Write to MII Mgmt Control with 16-bit data intended for the external PHY register, MIIMCON[0000_0000_0000_0000_000u_00u1_0100_0000] where u is user defined based on desired configuration.
Check to see if MII Mgmt write is complete. Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.
If auto-negotiation was enabled in the PHY, check to see if PHY has completed Auto-Negotiation. Set up the MII Mgmt for a read cycle to PHY MII Mgmt register (write the PHY address and Register address), MIIMADD[0000_0000_0000_0000_0000_0000_0000_0001] The PHY Status register is at address 0x1 and in this case the PHY Address is 0x00
Perform an MII Mgmt read cycle of Status Register. Clear MIIMCOM[Read Cycle]. Set MIIMCOM[Read Cycle]. (Uses the PHY address (0) and Register address (1) placed in MIIMADD register), When MIIMIND[BUSY]=0, Read the MIIMSTAT register and check bit 10 (AN Done and Link is up), MIIMSTAT ---> [0000_0000_0000_0000_0000_0000_0010_0100] Other information about the link is also returned.(Extend Status, No pre, Remote Fault, An Ability, Link status, extend Ability)
Check auto-negotiation attributes in the PHY as necessary.
Clear IEVENT register, IEVENT[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize IMASK (Optional) IMASK[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize MACnADDR1/2 (Optional) MACnADDR1/2[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize GADDR*n* (Optional) GADDR*n*[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize RCTRL (Optional) RCTRL[0000_0000_0000_0000_0000_0000_0000_0000]

Table 13-155. GMII Mode Register Initialization Steps (continued)

Initialize DMACTRL (Optional) DMACTRL[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize (Empty) Transmit Descriptor ring and fill buffers with Data Initialize TBASE0–TBASE7, TBASE0–TBASE7[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]
Initialize (Empty) Receive Descriptor ring and fill with empty buffers Initialize RBASE0–RBASE7, RBASE0–RBASE7[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]
Enable Transmit Queues Initialize TQUEUE
Enable Receive Queues Initialize RQUEUE
Enable Rx and Tx, MACCFG1[0000_0000_0000_0000_0000_0000_0000_0101]

13.7.1.3 TBI Interface Mode

Table 13-156 describes the signal configurations required for TBI interface mode.

Table 13-156. TBI Interface Mode Signal Configuration

eTSEC Signals			TBI Interface		
			Frequency [MHz] 62.5		
			Voltage [V] 3.3		
Signals	**I/O**	**No. of Signals**	**Signals**	**I/O**	**No. of Signals**
GTX_CLK	O	1	GTX_CLK	O	1
TX_CLK	I	1	RX_CLK1	I	1
TxD[0]	O	1	TCG[0]	O	1
TxD[1]	O	1	TCG[1]	O	1
TxD[2]	O	1	TCG[2]	O	1
TxD[3]	O	1	TCG[3]	O	1
TxD[4]	O	1	TCG[4]	O	1
TxD[5]	O	1	TCG[5]	O	1
TxD[6]	O	1	TCG[6]	O	1
TxD[7]	O	1	TCG[7]	O	1
TX_EN	O	1	TCG[8]	O	1
TX_ER	O	1	TCG[9]	O	1
RX_CLK	I	1	RX_CLK0	I	1
RxD[0]	I	1	RCG[0]	I	1
RxD[1]	I	1	RCG[1]	I	1
RxD[2]	I	1	RCG[2]	I	1
RxD[3]	I	1	RCG[3]	I	1
RxD[4]	I	1	RCG[4]	I	1
RxD[5]	I	1	RCG[5]	I	1
RxD[6]	I	1	RCG[6]	I	1
RxD[7]	I	1	RCG[7]	I	1
RX_DV	I	1	RCG[8]	I	1
RX_ER	I	1	RCG[9]	I	1
COL	I	1	not used		
CRS	I	1	SDET	I	1
Sum		25	**Sum**		24

Table 13-157 describes the shared signals for the TBI interface.

Table 13-157. Shared TBI Signals

eTSEC Signals	I/O	No. of Signals	GMII Signals	I/O	No. of Signals	Function
MDIO	I/O	1	MDIO	I/O	1	Management interface I/O
MDC	O	1	MDC	O	1	Management interface clock
EC1_GTX_CLK125	I	1	GTX_CLK125	I	1	Reference clock for eTSEC1/2
EC2_GTX_CLK125	I	1	GTX_CLK125	I	1	Reference clock for eTSEC3/4
Sum		4	**Sum**		4	—

Table 13-158 describes the register initializations required to configure the eTSEC in TBI mode.

Table 13-158. TBI Mode Register Initialization Steps

Set Soft_Reset, MACCFG1[1000_0000_0000_0000_0000_0000_0000_0000]
Clear Soft_Reset, MACCFG1[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize MACCFG2, MACCFG2[0000_0000_0000_0000_0111_0010_0000_0101] (I/F Mode = 2, Full Duplex = 1)
Initialize ECNTRL, ECNTRL[0000_0000_0000_0000_0001_0000_0000_0000] (This example has Statistics Enable = 1)
Initialize MAC Station Address MACSTNADDR2[0110_0000_0000_0010_0000_0000_0000_0000] to 02608C:876543, for example.
Initialize MAC Station Address MACSTNADDR1[0100_0011_0110_0101_1000_0111_1000_1100] to 02608C:876543, for example.
Assign a Physical address to the TBI, TBIPA[0000_0000_0000_0000_0000_0000_0001_0000] set to 16, for example.
Setup the MII Mgmt clock speed, MIIMCFG[0000_0000_0000_0000_0000_0000_0000_0101] set source clock divide by 14 for example to insure that MDC clock speed is not greater than 2.5 MHz
Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the eTSEC MII Mgmt bus is idle.
Set up the MII Mgmt for a read cycle to TBI's Control register (write the TBI address and Register address), MIIMADD[0000_0000_0000_0000_0001_0000_0000_0000] The control register (CR) is at offset address 0x0 from TBIPA.

Table 13-158. TBI Mode Register Initialization Steps (continued)

Perform an MII Mgmt read cycle to verify state of TBI Control Register(0ptional) Clear MIIMCOM[Read Cycle] Set MIIMCOM[Read Cycle] (Uses the TBI address and Register address placed in MIIMADD register), When MIIMIND[BUSY]=0, read the MIIMSTAT and look for AN Enable and other bit information.
Set up the MII Mgmt for a write cycle to TBI's AN Advertisement register (write the PHY address and Register address), MIIMADD[0000_0000_0000_0000_0001_0000_0000_0100] The AN Advertisement register is at offset address 0x04 from the TBI's address. (in this case 0x10)
Perform an MII Mgmt write cycle to TBI. Writing to MII Mgmt Control with 16-bit data intended for TBI's AN Advertisement register, MIIMCON[0000_0000_0000_0000_0000_0001_1010_0000] This advertises to the Link Partner that the TBI supports PAUSE and Full Duplex mode and does not support Half Duplex mode.
Check to see if MII Mgmt write is complete. Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.
Set up the MII Mgmt for a write cycle to TBI's Control register (write the PHY address and Register address), MIIMADD[0000_0000_0000_0000_0001_0000_0000_0000] the control register (CR) is at offset address 0x00 from the TBI's address. (in this case 0x10)
Perform an MII Mgmt write cycle to TBI. Writing to MII Mgmt Control with 16-bit data intended for TBI's Control register, MIIMCON[0000_0000_0000_0000_0001_0010_0000_0000] This enables the TBI to restart Auto-Negotiations using the configuration set in the AN Advertisement register.
Check to see if MII Mgmt write is complete. Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.
Check to see if PHY has completed Auto-Negotiation. Set up the MII Mgmt for a read cycle to PHY MII Mgmt register (write the PHY address and Register address), MIIMADD[0000_0000_0000_0000_0001_0000_0000_0001] The PHY Status control register is at address 0x1 and in this case the PHY Address is 0x10.
Perform an MII Mgmt read cycle of Status Register. Clear MIIMCOM[Read Cycle] Set MIIMCOM[Read Cycle] (Uses the PHY address (2) and Register address (2) placed in MIIMADD register), When MIIMIND[BUSY]=0, read the MIIMSTAT register and check bit 10 (AN Done) MIIMSTAT ---> [0000_0000_0000_0000_0000_0000_0010_0000] Other information about the link is also returned. (Extend Status, No pre, Remote Fault, An Ability, Link status, extend Ability)

Table 13-158. TBI Mode Register Initialization Steps (continued)

Perform an MII Mgmt read cycle of AN Expansion Register. Setup MIIMADD[0000_0000_0000_0000_0001_0000_0000_0110] Clear MIIMCOM[Read Cycle] Set MIIMCOM[Read Cycle] (Uses the PHY address (0x10) and Register address (6) placed in MIIMADD register), When MIIMIND[BUSY]=0, read the MII Mgmt AN Expansion register and check bits 13 and 14 (NP Able and Page Rx'd) MII Mgmt AN Expansion ---> [0000_0000_0000_0000_0000_0000_0000_0110]
Perform an MII Mgmt read cycle of AN Link Partner Base Page Ability Register. (Optional) Setup MIIMADD[0000_0000_0000_0000_0001_0000_0000_0101] Clear MIIMCOM[Read Cycle] Set MIIMCOM[Read Cycle] (Uses the PHY address (0x10) and Register address (5) placed in MIIMADD register), When MIIMIND[BUSY]=0, read the MII Mgmt AN Link Partner Base Page Ability register and check bits 9 and 10. (Half and Full Duplex) MII Mgmt AN Link Partner Base Page Ability ---> [0000_0000_0000_0000_0000_000x_x110_0000]
Clear IEVENT register, IEVENT[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize IMASK (Optional) IMASK[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize MACnADDR1/2 (Optional) MACnADDR1/2[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize GADDR*n* (Optional) GADDR*n*[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize RCTRL (Optional) RCTRL[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize DMACTRL (Optional) DMACTRL[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize (Empty) Transmit Descriptor ring and fill buffers with Data Initialize TBASE0–TBASE7, TBASE0–TBASE7[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]
Initialize (Empty) Receive Descriptor ring and fill with empty buffers Initialize RBASE0–RBASE7, RBASE0–RBASE7[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]
Enable Transmit Queues Initialize TQUEUE
Enable Receive Queues Initialize RQUEUE
Enable Rx and Tx, MACCFG1[0000_0000_0000_0000_0000_0000_0000_0101]

13.7.1.4 RGMII Interface Mode

Table 13-159 shows the signals configurations required for RGMII interface mode.

Table 13-159. RGMII Interface Mode Signal Configuration

eTSEC Signals			RGMII Interface		
			Frequency [MHz] 125		
			Voltage [V] 2.5		
Signals	**I/O**	**No. of Signals**	**Signals**	**I/O**	**No. of Signals**
GTX_CLK	O	1	GTX_CLK	O	1
TX_CLK	I	1	not used		
TxD[0]	O	1	TxD[0]/TxD[4]	O	1
TxD[1]	O	1	TxD[1]/TxD[5]	O	1
TxD[2]	O	1	TxD[2]/TxD[6]	O	1
TxD[3]	O	1	TxD[3]/TxD[7]	O	1
TxD[4]	O	1	leave unconnected		
TxD[5]	O	1	leave unconnected		
TxD[6]	O	1	leave unconnected		
TxD[7]	O	1	leave unconnected		
TX_EN	O	1	TX_CTL (TX_EN/TX_ERR)	O	1
TX_ER	O	1	leave unconnected		
RX_CLK	I	1	RX_CLK	I	1
RxD[0]	I	1	RxD[0]/RxD[4]	I	1
RxD[1]	I	1	RxD[1]/RxD[5]	I	1
RxD[2]	I	1	RxD[2]/RxD[6]	I	1
RxD[3]	I	1	RxD[3]/RxD[7]	I	1
RxD[4]	I	1	not used		
RxD[5]	I	1	not used		
RxD[6]	I	1	not used		
RxD[7]	I	1	not used		
RX_DV	I	1	RX_CTL (RX_DV/RX_ERR)	I	1
RX_ER	I	1	not used		
COL	I	1	not used		
CRS	I	1	not used		
Sum		25	**Sum**		12

Table 13-160 describes the shared signals for the RGMII interface.

Table 13-160. Shared RGMII Signals

eTSEC Signals	I/O	No. of Signals	GMII Signals	I/O	No. of Signals	Function
MDIO	I/O	1	MDIO	I/O	1	Management interface I/O
MDC	O	1	MDC	O	1	Management interface clock
EC1_GTX_CLK125	I	1	GTX_CLK125	I	1	Reference clock for eTSEC1/2
EC2_GTX_CLK125	I	1	GTX_CLK125	I	1	Reference clock for eTSEC3/4
Sum		4	**Sum**		4	—

Table 13-161 describes the register initializations required to configure the eTSEC in RGMII mode.

Table 13-161. RGMII Mode Register Initialization Steps

Set Soft_Reset, MACCFG1[1000_0000_0000_0000_0000_0000_0000_0000]
Clear Soft_Reset, MACCFG1[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize MACCFG2, MACCFG2[0000_0000_0000_0000_0111_0010_0000_0101] (I/F Mode = 2, Full Duplex = 1)
Initialize ECNTRL, ECNTRL[0000_0000_0000_0000_0001_0000_0000_0000] (This example has RGMII 10Mbps mode, Statistics Enable = 1)
Initialize MAC Station Address, MACSTNADDR2[0110_0000_0000_0010_0000_0000_0000_0000] to 02608C:876543, for example.
Initialize MAC Station Address, MACSTNADDR1[0100_0011_0110_0101_1000_0111_1000_1100] to 02608C:876543, for example.
Assign a Physical address to the TBI, TBIPA[0000_0000_0000_0000_0000_0000_0001_0000] set to 16, for example.
Setup the MII Mgmt clock speed, MIIMCFG[0000_0000_0000_0000_0000_0000_0000_0101] Set source clock divide by 14, for example, to insure that TSEC_MDC clock speed is not greater than 2.5 MHz.
Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the eTSEC MII Mgmt bus is idle.
Set up the MII Mgmt for a write cycle to external the PHY AN Advertisement register (write the PHY address and Register address), MIIMADD[0000_0000_0000_0000_0001_0001_0000_0100] The AN Advertisement register is at offset address 0x04 from the external PHY address. (in this case 0x11)

Table 13-161. RGMII Mode Register Initialization Steps (continued)

Perform an MII Mgmt write cycle to the external PHY. Write to MII Mgmt Control with 16-bit data intended for the external PHY AN Advertisement register, MIIMCON[0000_0000_0000_0000_u0uu_uuuu_uuuu_uuuu] Where u must be selected by the user for proper system configuration.
Check to see if MII Mgmt write is complete. Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.
Set up the MII Mgmt for a write cycle to the external PHY Control register (write the PHY address and Register address), MIIMADD[0000_0000_0000_0000_0001_0001_0000_0000] The control register (CR) is at offset address 0x00 from the external PHY address. (in this case 0x11)
Perform an MII Mgmt write cycle to the external PHY. Write to MII Mgmt Control with 16-bit data intended for the external PHY Control register, MIIMCON[0000_0000_0000_0000_0001_0010_0000_0000] This enables the external PHY to restart Auto-Negotiations using the configuration set in the AN Advertisement register.
Check to see if MII Mgmt write is complete. Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.
Check to see if PHY has completed Auto-Negotiation. Set up the MII Mgmt for a read cycle to the PHY MII Mgmt register (write the PHY address and Register address), MIIMADD[0000_0000_0000_0000_0000_0010_0000_0001] The PHY Status register is at address 0x1 and in this case the PHY Address is 0x2.
Perform an MII Mgmt read cycle of Status Register. Clear MIIMCOM[Read Cycle] Set MIIMCOM[Read Cycle] (Uses the PHY address (2) and Register address (2) placed in MIIMADD register) When MIIMIND[BUSY]=0, read the MIIMSTAT register and check bit 10. (AN Done) MIIMSTAT ---> [0000_0000_0000_0000_0000_0000_0010_0000] Other information about the link is also returned. (Extend Status, No pre, Remote Fault, An Ability, Link status, extend Ability)
Perform an MII Mgmt read cycle of AN Expansion Register. Setup MIIMADD[0000_0000_0000_0000_0001_0001_0000_0110] Clear MIIMCOM[Read Cycle] Set MIIMCOM[Read Cycle] (Uses the PHY address (0x11) and Register address (6) placed in MIIMADD register) When MIIMIND[BUSY]=0, read the MII Mgmt AN Expansion register and check bits 13 and 14. (NP Able and Page Rx'd) MII Mgmt AN Expansion ---> [0000_0000_0000_0000_0000_0000_0000_0110]
Perform an MII Mgmt read cycle of AN Link Partner Base Page Ability Register. (Optional) Setup MIIMADD[0000_0000_0000_0000_0001_0001_0000_0101] Clear MIIMCOM[Read Cycle] Set MIIMCOM[Read Cycle] (Uses the PHY address (0x11) and Register address (5) placed in MIIMADD register) When MIIMIND[BUSY]=0, read the MII Mgmt AN Link Partner Base Page Ability register and check bits 9 and 10. (Half and Full Duplex) MII Mgmt AN Link Partner Base Page Ability ---> [0000_0000_0000_0000_0000_000x_1x10_0000]

Table 13-161. RGMII Mode Register Initialization Steps (continued)

Clear IEVENT register, IEVENT[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize IMASK (Optional) IMASK[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize MACnADDR1/2 (Optional) MACnADDR1/2[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize GADDR*n* (Optional) GADDR*n*[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize RCTRL (Optional) RCTRL[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize DMACTRL (Optional) DMACTRL[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize (Empty) Transmit Descriptor ring and fill buffers with Data Initialize TBASE0–TBASE7, TBASE0–TBASE7[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]
Initialize (Empty) Receive Descriptor ring and fill with empty buffers Initialize RBASE0–RBASE7, RBASE0–RBASE7[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]
Enable Transmit Queues Initialize TQUEUE
Enable Receive Queues Initialize RQUEUE
Enable Rx and Tx, MACCFG1[0000_0000_0000_0000_0000_0000_0000_0101]

13.7.1.5 RMII Interface Mode

Table 13-162 shows the signals configurations required for RMII interface mode.

Table 13-162. RMII Interface Mode Signal Configuration

eTSEC Signals			RMII Interface		
			Frequency [MHz] 50		
			Voltage [V] 3.3		
Signals	**I/O**	**No. of Signals**	**Signals**	**I/O**	**No. of Signals**
GTX_CLK	O	1	leave unconnected		
TX_CLK	I	1	REF_CLK	I	1
TxD[0]	O	1	TxD[0]	O	1
TxD[1]	O	1	TxD[1]	O	1
TxD[2]	O	1	leave unconnected		
TxD[3]	O	1	leave unconnected		
TxD[4]	O	1	leave unconnected		
TxD[5]	O	1	leave unconnected		
TxD[6]	O	1	leave unconnected		
TxD[7]	O	1	leave unconnected		
TX_EN	O	1	TX_EN	O	1
TX_ER	O	1	leave unconnected		
RX_CLK	I	1	leave unconnected		
RxD[0]	I	1	RxD[0]	I	1
RxD[1]	I	1	RxD[1]	I	1
RxD[2]	I	1	not used		
RxD[3]	I	1	not used		
RxD[4]	I	1	not used		
RxD[5]	I	1	not used		
RxD[6]	I	1	not used		
RxD[7]	I	1	not used		
RX_DV	I	1	CRS_DV	I	1
RX_ER	I	1	RX_ER	I	1
COL	I	1	not used		
CRS	I	1	not used		
Sum		25	**Sum**		8

Table 13-163 describes the shared signals for the RMII interface.

Table 13-163. Shared RMII Signals

eTSEC Signals	I/O	No. of Signals	GMII Signals	I/O	No. of Signals	Function
MDIO	I/O	1	MDIO	I/O	1	Management interface I/O
MDC	O	1	MDC	O	1	Management interface clock
TX_CLK	I	1	REF_CLK	I	1	Reference clock
Sum		3	**Sum**		3	—

Table 13-164 describes the register initializations required to configure the eTSEC in RMII mode.

Table 13-164. RMII Mode Register Initialization Steps

Set Soft_Reset, MACCFG1[1000_0000_0000_0000_0000_0000_0000_0000]
Clear Soft_Reset, MACCFG1[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize MACCFG2, MACCFG2[0000_0000_0000_0000_0111_0010_0000_0101] (I/F Mode = 2, Full Duplex = 1)
Initialize ECNTRL, ECNTRL[0000_0000_0000_0000_0001_0000_0001_0000] (Used to setup Reduced-Pin mode = 1, and TBIM = 0,statistics enable = 1)
Initialize MAC Station Address MACSTNADDR2[0110_0000_0000_0010_0000_0000_0000_0000] to 02608C:876543 for example
Initialize MAC Station Address MACSTNADDR1[0100_0011_0110_0101_1000_0111_1000_1100] to 02608C:876543 for example
Setup the MII Mgmt clock speed, MIIMCFG[0000_0000_0000_0000_0000_0000_0000_1101] set system clock divide by 14 for example to insure that MDC clock speed = 2.5 MHz
Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the eTSEC MII Mgmt bus is idle.
Set up the MII Mgmt for a write cycle to external the PHY AN Advertisement register (write the PHY address and Register address), MIIMADD[0000_0000_0000_0000_0001_0001_0000_0100] The AN Advertisement register is at offset address 0x04 from the external PHY address. (in this case 0x11)
Perform an MII Mgmt write cycle to the external PHY. Write to MII Mgmt Control with 16-bit data intended for the external PHY AN Advertisement register, MIIMCON[0000_0000_0000_0000_u0uu_uuuu_uuuu_uuuu] Where u must be selected by the user for proper system configuration.
Check to see if MII Mgmt write is complete. Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.

Table 13-164. RMII Mode Register Initialization Steps (continued)

Set up the MII Mgmt for a write cycle to the external PHY Control register (write the PHY address and Register address), MIIMADD[0000_0000_0000_0000_0001_0001_0000_0000] The control register is at offset address 0x00 from the external PHY address. (in this case 0x11)
Perform an MII Mgmt write cycle to the external PHY. Write to MII Mgmt Control with 16-bit data intended for the external PHY Control register, MIIMCON[0000_0000_0000_0000_0001_0010_0000_0000] This enables the external PHY to restart Auto-Negotiations using the configuration set in the AN Advertisement register.
Check to see if MII Mgmt write is complete. Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.
Check to see if PHY has completed Auto-Negotiation. Set up the MII Mgmt for a read cycle to the PHY MII Mgmt register (write the PHY address and Register address), MIIMADD[0000_0000_0000_0000_0000_0010_0000_0001] The PHY Status register is at address 0x1 and in this case the PHY Address is 0x2.
Perform an MII Mgmt read cycle of Status Register. Clear MIIMCOM[Read Cycle] Set MIIMCOM[Read Cycle] (Uses the PHY address (2) and Register address (1) placed in MIIMADD register) When MIIMIND[BUSY]=0, read the MIIMSTAT register and check bit 10. (AN Done) MIIMSTAT ---> [0000_0000_0000_0000_0000_0000_0010_0000] Other information about the link is also returned. (Extend Status, No pre, Remote Fault, An Ability, Link status, extend Ability)
Perform an MII Mgmt read cycle of AN Expansion Register. Setup MIIMADD[0000_0000_0000_0000_0001_0001_0000_0110] Clear MIIMCOM[Read Cycle] Set MIIMCOM[Read Cycle] (Uses the PHY address (0x11) and Register address (6) placed in MIIMADD register) When MIIMIND[BUSY]=0, read the MII Mgmt AN Expansion register and check bits 13 and 14. (NP Able and Page Rx'd) MII Mgmt AN Expansion ---> [0000_0000_0000_0000_0000_0000_0000_0110]
Perform an MII Mgmt read cycle of AN Link Partner Base Page Ability Register. (Optional) Setup MIIMADD[0000_0000_0000_0000_0001_0001_0000_0101] Clear MIIMCOM[Read Cycle] Set MIIMCOM[Read Cycle] (Uses the PHY address (0x11) and Register address (5) placed in MIIMADD register) When MIIMIND[BUSY]=0, read the MII Mgmt AN Link Partner Base Page Ability register and check bits 9 and 10. (Half and Full Duplex) MII Mgmt AN Link Partner Base Page Ability ---> [0000_0000_0000_0000_0000_000x_x110_0000]
Setting up the MII Mgmt for a write cycle to TBI MII Mgmt register (write the TBI's address and Register address), MIIMADD[0000_0000_0000_0000_0001_0000_0000_1011] the TBI control register is at offset address 0x11 from TBIPA
Perform an MII Mgmt write cycle Writing to MII Mgmt Control with 16-bit data intended for TBI's MII Mgmt control register (TBI control), MIIMCON[0000_0000_0000_0000_0000_0010_0001_0000] This configures the TBI control to GMII mode and AN sense
Check to see if MII Mgmt write is complete Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicate that the write cycle was completed

Table 13-164. RMII Mode Register Initialization Steps (continued)

Perform an MII Mgmt read cycle (0ptional) Set MIIMCOM[Read Cycle] (Uses the TBI address and Register address placed in MIIMADD register), read the MIIMSTAT register and verify that MIIMSTAT ---> [0000_0000_0000_0000_0000_0010_0001_0000]
Check to see if PHY has completed Auto-Negotiation Setting up the MII Mgmt for a read cycle to PHY's MII Mgmt register (write the PHY's address and Register address), MIIMADD[0000_0000_0000_0000_0000_0010_0000_0010] the PHY Status control register is at address 0x2 and lets say the PHY Address is 0x2
Perform an MII Mgmt read cycle of Status Register Set MIIMCOM[Read Cycle] (Uses the PHY address (2) and Register address (2) placed in MIIMADD register), read the MIIMSTAT register and check bit 10 (AN Done) MIIMSTAT ---> [0000_0000_0000_0000_0000_0000_0010_0000] other information about the link is also returned (Extend Status, No pre, Remote Fault, An Ability, Link status, extend Ability)
Perform an MII Mgmt read cycle of AN Expansion Register MIIMADD[0000_0000_0000_0000_0000_0010_0000_0110] Set MIIMCOM[Read Cycle] (Uses the PHY address (2) and Register address (6) placed in MIIMADD register), read the MII Mgmt AN Expansion register and check bits 13 and 14 (NP Able and Page Rx'd) MII Mgmt AN Expansion ---> [0000_0000_0000_0000_0000_0000_0000_0110]
Perform an MII Mgmt read cycle of AN Link Partner Base Page Ability Register (Optional) MIIMADD[0000_0000_0000_0000_0000_0010_0000_0101] Set MIIMCOM[Read Cycle] (Uses the PHY address (2) and Register address (5) placed in MIIMADD register), read the MII Mgmt AN Link Partner Base Page Ability register and check bits 9 and 10 (Half and Full Duplex) MII Mgmt AN Link Partner Base Page Ability ---> [0000_0000_0000_0000_0000_00X_1110_0000]
Clear IEVENT register, IEVENT[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize IMASK (Optional) IMASK[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize GADDR*n* (Optional) GADDR*n*[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize RCTRL (Optional) RCTRL[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize DMACTRL (Optional) DMACTRL[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize (Empty) Transmit Descriptor ring and fill buffers with Data Initialize TBASE0–TBASE7, TBASE0–TBASE7[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]
Initialize (Empty) Receive Descriptor ring and fill with empty buffers Initialize RBASE0–RBASE7, RBASE0–RBASE7[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]
Enable Transmit Queues Initialize TQUEUE
Enable Receive Queues Initialize RQUEUE
Enable Rx and Tx, MACCFG1[0000_0000_0000_0000_0000_0000_0000_0101]

13.7.1.6 RTBI Interface Mode

Table 13-165 describes the signal configurations required for RTBI interface mode.

Table 13-165. RTBI Interface Mode Signal Configuration

eTSEC Signal s			RTBI Interface		
			Frequency [MHz] 125		
			Voltage [V] 2.5		
Signals	**I/O**	**No. of Signals**	**Signals**	**I/O**	**No. of Signals**
GTX_CLK	O	1	GTX_CLK	O	1
TX_CLK	I	1	not used		
TxD[0]	O	1	TCG[0]/TCG[5]	O	1
TxD[1]	O	1	TCG[1]/TCG[6]	O	1
TxD[2]	O	1	TCG[2]/TCG[7]	O	1
TxD[3]	O	1	TCG[3]/TCG[8]	O	1
TxD[4]	O	1	leave unconnected		
TxD[5]	O	1	leave unconnected		
TxD[6]	O	1	leave unconnected		
TxD[7]	O	1	leave unconnected		
TX_EN	O	1	TCG[4]/TCG[9]	O	1
TX_ER	O	1	leave unconnected		
RX_CLK	I	1	RX_CLK	I	1
RxD[0]	I	1	RCG[0]/RCG[5]	I	1
RxD[1]	I	1	RCG[1]/RCG[6]	I	1
RxD[2]	I	1	RCG[2]/RCG[7]	I	1
RxD[3]	I	1	RCG[3]/RCG[8]	I	1
RxD[4]	I	1	not used		
RxD[5]	I	1	not used		
RxD[6]	I	1	not used		
RxD[7]	I	1	not used		
RX_DV	I	1	RCG[4]/RCG[9]	I	1
RX_ER	I	1	not used		
COL	I	1	not used		
CRS	I	1	not used		
Sum		25	**sum**	—	12

Table 13-166 describes the shared signals for the RTBI interface.

Table 13-166. Shared RTBI Signals

eTSEC Signals	I/O	No. of Signals	GMII Signals	I/O	No. of Signals	Function
MDIO	I/O	1	MDIO	I/O	1	Management interface I/O
MDC	O	1	MDC	O	1	Management interface clock
EC1_GTX_CLK125	I	1	GTX_CLK125	I	1	Reference clock for eTSEC1/2
EC2_GTX_CLK125	I	1	GTX_CLK125	I	1	Reference clock for eTSEC3/4
Sum		4	Sum		4	—

Table 13-167 describes the register initializations required to configure the eTSEC in RTBI mode.

Table 13-167. RTBI Mode Register Initialization Steps

Set Soft_Reset, MACCFG1[1000_0000_0000_0000_0000_0000_0000_0000]
Clear Soft_Reset, MACCFG1[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize MACCFG2, MACCFG2[0000_0000_0000_0000_0111_0010_0000_0101] (I/F Mode = 2, Full Duplex = 1)
Initialize ECNTRL, ECNTRL[0000_0000_0000_0000_0001_0000_0000_0000] (This example has Statistics Enable = 1)
Initialize MAC Station Address, MACSTNADDR2[0110_0000_0000_0010_0000_0000_0000_0000] to 02608C:876543, for example.
Initialize MAC Station Address, MACSTNADDR1[0100_0011_0110_0101_1000_0111_1000_1100] to 02608C:876543, for example.
Assign a Physical address to the TBI, TBIPA[0000_0000_0000_0000_0000_0000_0001_0000] set to 16, for example.
Setup the MII Mgmt clock speed, MIIMCFG[0000_0000_0000_0000_0000_0000_0000_0101] Set source clock divide by 14, for example, to insure that TSEC_MDC clock speed is not greater than 2.5 MHz.
Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the eTSEC MII Mgmt bus is idle.
Set up the MII Mgmt for a read cycle to TBI's Control register (write the TBI's address and Register address), MIIMADD[0000_0000_0000_0000_0001_0000_0000_0000] The control register (CR) is at offset address 0x0 from TBIPA.

Table 13-167. RTBI Mode Register Initialization Steps (continued)

Perform an MII Mgmt read cycle to verify state of TBI Control Register(0ptional) Clear MIIMCOM[Read Cycle] Set MIIMCOM[Read Cycle] (Uses the TBI address and Register address placed in MIIMADD register), When MIIMIND[BUSY]=0, read the MIIMSTAT and look for AN Enable and other bit information.
Set up the MII Mgmt for a write cycle to TBI's AN Advertisement register (write the PHY address and Register address), MIIMADD[0000_0000_0000_0000_0001_0000_0000_0100] The AN Advertisement register is at offset address 0x04 from the TBI's address. (in this case 0x10)
Perform an MII Mgmt write cycle to TBI. Write to MII Mgmt Control with 16-bit data intended for TBI's AN Advertisement register, MIIMCON[0000_0000_0000_0000_0000_0001_1010_0000] This advertises to the Link Partner that the TBI supports PAUSE and Full Duplex mode and does not support Half Duplex mode.
Check to see if MII Mgmt write is complete. Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.
Set up the MII Mgmt for a write cycle to TBI's Control register (write the PHY address and Register address), MIIMADD[0000_0000_0000_0000_0001_0000_0000_0000] The control register (CR) is at offset address 0x00 from the TBI's address. (in this case 0x10)
Perform an MII Mgmt write cycle to TBI. Writing to MII Mgmt Control with 16-bit data intended for TBI's Control register, MIIMCON[0000_0000_0000_0000_0001_0010_0000_0000] This enables the TBI to restart Auto-Negotiations using the configuration set in the AN Advertisement register.
Check to see if MII Mgmt write is complete. Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.
Check to see if PHY has completed Auto-Negotiation. Set up the MII Mgmt for a read cycle to the PHY MII Mgmt register (write the PHY address and Register address), MIIMADD[0000_0000_0000_0000_0001_0000_0000_0001] The PHY Status control register is at address 0x1 and in this case the PHY Address is 0x10.
Perform an MII Mgmt read cycle of Status Register. Clear MIIMCOM[Read Cycle] Set MIIMCOM[Read Cycle] (Uses the PHY address (2) and Register address (2) placed in MIIMADD register), When MIIMIND[BUSY]=0, read the MIIMSTAT register and check bit 10 (AN Done) MIIMSTAT ---> [0000_0000_0000_0000_0000_0000_0010_0000] Other information about the link is also returned. (Extend Status, No pre, Remote Fault, An Ability, Link status, extend Ability)

Table 13-167. RTBI Mode Register Initialization Steps (continued)

Perform an MII Mgmt read cycle of AN Expansion Register. Setup MIIMADD[0000_0000_0000_0000_0001_0000_0000_0110] Clear MIIMCOM[Read Cycle] Set MIIMCOM[Read Cycle] (Uses the PHY address (0x10) and Register address (6) placed in MIIMADD register), When MIIMIND[BUSY]=0, read the MII Mgmt AN Expansion register and check bits 13 and 14. (NP Able and Page Rx'd) MII Mgmt AN Expansion ---> [0000_0000_0000_0000_0000_0000_0000_0110]
Perform an MII Mgmt read cycle of AN Link Partner Base Page Ability Register. (Optional) Setup MIIMADD[0000_0000_0000_0000_0001_0000_0000_0101] Clear MIIMCOM[Read Cycle] Set MIIMCOM[Read Cycle] (Uses the PHY address (0x10) and Register address (5) placed in MIIMADD register), When MIIMIND[BUSY]=0, read the MII Mgmt AN Link Partner Base Page Ability register and check bits 9 and 10. (Half and Full Duplex) MII Mgmt AN Link Partner Base Page Ability ---> [0000_0000_0000_0000_0000_00x_x110_0000]
Clear IEVENT register, IEVENT[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize IMASK (Optional) IMASK[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize MACnADDR1/2 (Optional) MACnADDR1/2[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize GADDR*n* (Optional) GADDR*n*[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize RCTRL (Optional) RCTRL[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize DMACTRL (Optional) DMACTRL[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize (Empty) Transmit Descriptor ring and fill buffers with Data Initialize TBASE0–TBASE7, TBASE0–TBASE7[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]
Initialize (Empty) Receive Descriptor ring and fill with empty buffers Initialize RBASE0–RBASE7, RBASE0–RBASE7[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]
Enable Transmit Queues Initialize TQUEUE
Enable Receive Queues Initialize RQUEUE
Enable Rx and Tx, MACCFG1[0000_0000_0000_0000_0000_0000_0000_0101]

13.7.1.7 8-Bit FIFO Mode

Table 13-168 describes the signal configurations required for 8-bit FIFO interface mode.

Table 13-168. 8-Bit FIFO Interface Mode Signal Configurations

eTSEC Signals			8-Bit FIFO Interface		
			Frequency [MHz] 155		
			Voltage [V] 2.5		
Signals	**I/O**	**No. of Signals**	**Signals**	**I/O**	**No. of Signals**
TSEC*n*_GTX_CLK	O	1	FIFO*n*_GTX_CLK	O	1
TSEC*n*_TX_CLK	I	1	FIFO*n*_TX_CLK	I	1
TSEC*n*_TxD[0]	O	1	FIFO*n*_TxD[0]	O	1
TSEC*n*_TxD[1]	O	1	FIFO*n*_TxD[1]	O	1
TSEC*n*_TxD[2]	O	1	FIFO*n*_TxD[2]	O	1
TSEC*n*_TxD[3]	O	1	FIFO*n*_TxD[3]	O	1
TSEC*n*_TxD[4]	O	1	FIFO*n*_TxD[4]	O	1
TSEC*n*_TxD[5]	O	1	FIFO*n*_TxD[5]	O	1
TSEC*n*_TxD[6]	O	1	FIFO*n*_TxD[6]	O	1
TSEC*n*_TxD[7]	O	1	FIFO*n*_TxD[7]	O	1
TSEC*n*_TX_EN	O	1	FIFO*n*_TX_EN	O	1
TSEC*n*_TX_ER	O	1	FIFO*n*_TX_ER	O	1
TSEC*n*_RX_CLK	I	1	FIFO*n*_RX_CLK	I	1
TSEC*n*_RxD[0]	I	1	FIFO*n*_RxD[0]	I	1
TSEC*n*_RxD[1]	I	1	FIFO*n*_RxD[1]	I	1
TSEC*n*_RxD[2]	I	1	FIFO*n*_RxD[2]	I	1
TSEC*n*_RxD[3]	I	1	FIFO*n*_RxD[3]	I	1
TSEC*n*_RxD[4]	I	1	FIFO*n*_RxD[4]	I	1
TSEC*n*_RxD[5]	I	1	FIFO*n*_RxD[5]	I	1
TSEC*n*_RxD[6]	I	1	FIFO*n*_RxD[6]	I	1
TSEC*n*_RxD[7]	I	1	FIFO*n*_RxD[7]	I	1
TSEC*n*_RX_DV	I	1	FIFO*n*_RX_DV	I	1
TSEC*n*_RX_ER	I	1	FIFO*n*_RX_ER	I	1
TSEC*n*_COL	I	1	FIFO*n*_TX_FC	I	1
TSEC*n*_CRS	I/O	1	FIFO*n*_RX_FC	O	1
MDIO	I/O	1	leave unconnected		
MDC	O	1	leave unconnected		
Sum		27	**Sum**		25

Table 13-169 describes the register initializations required to configure the eTSEC in 8-bit FIFO mode.

Table 13-169. 8-Bit FIFO Mode Register Initialization Steps

Set FIFO Soft_Reset, FIFOCFG[0000_0000_0000_0000_1100_0000_0000_0000] (Reset RX = 1, reset Tx = 1, Rx enable = 0, Tx enable = 0)
Clear FIFO Soft_Reset, FIFOCFG[0000_0000_0000_0000_0000_0000_0000_1000] (Reset RX = 0, reset Tx = 0, Rx enable = 0, Tx enable = 0)
Ensure MACCFG2 is set to default values. MACCFG2[0000_0000_0000_0000_0111_0000_0000_0000]
Initialize ECNTRL, ECNTRL[0000_0000_0000_0000_1000_0000_0000_0000] (Used to set up FIFO mode = 1, and statistics enable = 0)
Clear IEVENT register, IEVENT[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize IMASK (Optional) IMASK[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize RCTRL (Optional) RCTRL[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize DMACTRL (Optional) DMACTRL[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize (Empty) transmit descriptor ring and fill buffers with data Initialize TBASE0–TBASE7, TBASE0–TBASE7[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]
Initialize (Empty) Receive Descriptor ring and fill with empty buffers Initialize RBASE0–RBASE7, RBASE0–RBASE7[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]
Enable Transmit Queues Initialize TQUEUE
Enable Receive Queues Initialize RQUEUE
Enable Rx and Tx over FIFO, FIFOCFG[0000_0000_0000_0000_0011_0000_1101_1000] (Rx enable = 1, Tx enable = 1, enable flow control and CRC, 8-bit mode)

13.7.1.8 16-Bit FIFO Mode

Table 13-170 describes the signal configurations required when eTSECs 1 and 2 are placed into 16-bit FIFO interface mode.

Table 13-170. 16-Bit FIFO Interface Mode Signal Configuration (eTSECs 1 and 2)

eTSEC Signals			16-Bit FIFO Interface		
			Frequency [MHz] 155		
			Voltage [V] 2.5		
Signals	**I/O**	**No. of Signals**	**Signals**	**I/O**	**No. of Signals**
TSEC1_GTX_CLK	O	1	FIFO1_GTX_CLK	O	1
TSEC1_TX_CLK	I	1	FIFO1_TX_CLK	I	1
TSEC1_TxD[0]	O	1	FIFO1_TxD[0]	O	1
TSEC1_TxD[1]	O	1	FIFO1_TxD[1]	O	1
TSEC1_TxD[2]	O	1	FIFO1_TxD[2]	O	1
TSEC1_TxD[3]	O	1	FIFO1_TxD[3]	O	1
TSEC1_TxD[4]	O	1	FIFO1_TxD[4]	O	1
TSEC1_TxD[5]	O	1	FIFO1_TxD[5]	O	1
TSEC1_TxD[6]	O	1	FIFO1_TxD[6]	O	1
TSEC1_TxD[7]	O	1	FIFO1_TxD[7]	O	1
TSEC1_TX_EN	O	1	FIFO1_TXC[0]	O	1
TSEC1_TX_ER	O	1	FIFO1_TXC[1]	O	1
TSEC1_RX_CLK	I	1	FIFO1_RX_CLK	I	1
TSEC1_RxD[0]	I	1	FIFO1_RxD[0]	I	1
TSEC1_RxD[1]	I	1	FIFO1_RxD[1]	I	1
TSEC1_RxD[2]	I	1	FIFO1_RxD[2]	I	1
TSEC1_RxD[3]	I	1	FIFO1_RxD[3]	I	1
TSEC1_RxD[4]	I	1	FIFO1_RxD[4]	I	1
TSEC1_RxD[5]	I	1	FIFO1_RxD[5]	I	1
TSEC1_RxD[6]	I	1	FIFO1_RxD[6]	I	1
TSEC1_RxD[7]	I	1	FIFO1_RxD[7]	I	1
TSEC1_RX_DV	I	1	FIFO1_RXC[0]	I	1
TSEC1_RX_ER	I	1	FIFO1_RXC[1]	I	1
TSEC1_COL	I	1	FIFO1_TX_FC	I	1
TSEC1_CRS	I/O	1	FIFO1_RX_FC	O	1

Table 13-170. 16-Bit FIFO Interface Mode Signal Configuration (eTSECs 1 and 2) (continued)

eTSEC Signals			16-Bit FIFO Interface		
			Frequency [MHz] 155		
			Voltage [V] 2.5		
MDIO	I/O	1	leave unconnected		
MDC	O	1	leave unconnected		
TSEC2_GTX_CLK	O	1	leave unconnected		
TSEC2_TX_CLK	I	1	not used		
TSEC2_TxD[0]	O	1	FIFO1_TxD[8]	O	1
TSEC2_TxD[1]	O	1	FIFO1_TxD[9]	O	1
TSEC2_TxD[2]	O	1	FIFO1_TxD[10]	O	1
TSEC2_TxD[3]	O	1	FIFO1_TxD[11]	O	1
TSEC2_TxD[4]	O	1	FIFO1_TxD[12]	O	1
TSEC2_TxD[5]	O	1	FIFO1_TxD[13]	O	1
TSEC2_TxD[6]	O	1	FIFO1_TxD[14]	O	1
TSEC2_TxD[7]	O	1	FIFO1_TxD[15]	O	1
TSEC2_TX_EN	O	1	FIFO1_TXC[2]	O	1
TSEC2_TX_ER	O	1	leave unconnected		
TSEC2_RX_CLK	I	1	not used		
TSEC2_RxD[0]	I	1	FIFO1_RxD[8]	I	1
TSEC2_RxD[1]	I	1	FIFO1_RxD[9]	I	1
TSEC2_RxD[2]	I	1	FIFO1_RxD[10]	I	1
TSEC2_RxD[3]	I	1	FIFO1_RxD[11]	I	1
TSEC2_RxD[4]	I	1	FIFO1_RxD[12]	I	1
TSEC2_RxD[5]	I	1	FIFO1_RxD[13]	I	1
TSEC2_RxD[6]	I	1	FIFO1_RxD[14]	I	1
TSEC2_RxD[7]	I	1	FIFO1_RxD[15]	I	1
TSEC2_RX_DV	I	1	FIFO1_RXC[2]	I	1
TSEC2_RX_ER	I	1	not used		
TSEC2_COL	I	1	not used		
TSEC2_CRS	I	1	not used		
Sum		52	**Sum**		43

Likewise, Table 13-171 describes the signal configurations required when eTSECs 3 and 4 are placed into 16-bit FIFO interface mode.

Table 13-171. 16-Bit FIFO Interface Mode Signal Configuration (eTSECs 3 and 4)

eTSEC Signals			16-Bit FIFO Interface		
			Frequency [MHz] 155		
			Voltage [V] 2.5		
Signals	**I/O**	**No. of Signals**	**Signals**	**I/O**	**No. of Signals**
TSEC3_GTX_CLK	O	1	FIFO2_GTX_CLK	O	1
TSEC3_TX_CLK	I	1	FIFO2_TX_CLK	I	1
TSEC3_TxD[0]	O	1	FIFO2_TxD[0]	O	1
TSEC3_TxD[1]	O	1	FIFO2_TxD[1]	O	1
TSEC3_TxD[2]	O	1	FIFO2_TxD[2]	O	1
TSEC3_TxD[3]	O	1	FIFO2_TxD[3]	O	1
TSEC3_TxD[4]	O	1	FIFO2_TxD[4]	O	1
TSEC3_TxD[5]	O	1	FIFO2_TxD[5]	O	1
TSEC3_TxD[6]	O	1	FIFO2_TxD[6]	O	1
TSEC3_TxD[7]	O	1	FIFO2_TxD[7]	O	1
TSEC3_TX_EN	O	1	FIFO2_TXC[0]	O	1
TSEC3_TX_ER	O	1	FIFO2_TXC[1]	O	1
TSEC3_RX_CLK	I	1	FIFO2_RX_CLK	I	1
TSEC3_RxD[0]	I	1	FIFO2_RxD[0]	I	1
TSEC3_RxD[1]	I	1	FIFO2_RxD[1]	I	1
TSEC3_RxD[2]	I	1	FIFO2_RxD[2]	I	1
TSEC3_RxD[3]	I	1	FIFO2_RxD[3]	I	1
TSEC3_RxD[4]	I	1	FIFO2_RxD[4]	I	1
TSEC3_RxD[5]	I	1	FIFO2_RxD[5]	I	1
TSEC3_RxD[6]	I	1	FIFO2_RxD[6]	I	1
TSEC3_RxD[7]	I	1	FIFO2_RxD[7]	I	1
TSEC3_RX_DV	I	1	FIFO2_RXC[0]	I	1
TSEC3_RX_ER	I	1	FIFO2_RXC[1]	I	1
TSEC3_COL	I	1	FIFO2_TX_FC	I	1
TSEC3_CRS	I/O	1	FIFO2_RX_FC	O	1
TSEC3_MDIO	I/O	1	leave unconnected		
TSEC3_MDC	O	1	leave unconnected		

Table 13-171. 16-Bit FIFO Interface Mode Signal Configuration (eTSECs 3 and 4) (continued)

eTSEC Signals			16-Bit FIFO Interface		
			Frequency [MHz] 155		
			Voltage [V] 2.5		
TSEC4_GTX_CLK	O	1	leave unconnected		
TSEC4_TX_CLK	I	1	not used		
TSEC4_TxD[0]	O	1	FIFO2_TxD[8]	O	1
TSEC4_TxD[1]	O	1	FIFO2_TxD[9]	O	1
TSEC4_TxD[2]	O	1	FIFO2_TxD[10]	O	1
TSEC4_TxD[3]	O	1	FIFO2_TxD[11]	O	1
TSEC4_TxD[4]	O	1	FIFO2_TxD[12]	O	1
TSEC4_TxD[5]	O	1	FIFO2_TxD[13]	O	1
TSEC4_TxD[6]	O	1	FIFO2_TxD[14]	O	1
TSEC4_TxD[7]	O	1	FIFO2_TxD[15]	O	1
TSEC4_TX_EN	O	1	FIFO2_TXC[2]	O	1
TSEC4_TX_ER	O	1	leave unconnected		
TSEC4_RX_CLK	I	1	not used		
TSEC4_RxD[0]	I	1	FIFO2_RxD[8]	I	1
TSEC4_RxD[1]	I	1	FIFO2_RxD[9]	I	1
TSEC4_RxD[2]	I	1	FIFO2_RxD[10]	I	1
TSEC4_RxD[3]	I	1	FIFO2_RxD[11]	I	1
TSEC4_RxD[4]	I	1	FIFO2_RxD[12]	I	1
TSEC4_RxD[5]	I	1	FIFO2_RxD[13]	I	1
TSEC4_RxD[6]	I	1	FIFO2_RxD[14]	I	1
TSEC4_RxD[7]	I	1	FIFO2_RxD[15]	I	1
TSEC4_RX_DV	I	1	FIFO2_RXC[2]	I	1
TSEC4_RX_ER	I	1	not used		
TSEC4_COL	I	1	not used		
TSEC4_CRS	I	1	not used		
Sum		52	**Sum**		43

Table 13-172 describes the register initializations required to configure the controlling the eTSEC (eTSEC1 or 3) in 16-bit FIFO mode. The disabled eTSEC (eTSEC2 or 4) must be held in soft reset, and have its transmit and receive functions disabled.

Table 13-172. 16-Bit FIFO Mode Register Initialization Steps

Set FIFO Soft_Reset, FIFOCFG[0000_0000_0000_0000_1100_0000_0000_0000] (Reset RX = 1, reset Tx = 1, Rx enable = 0, Tx enable = 0)
Clear FIFO Soft_Reset, FIFOCFG[0000_0000_0000_0000_0000_0000_0000_1000] (Reset RX = 0, reset Tx = 0, Rx enable = 0, Tx enable = 0)
Ensure MACCFG2 is set to default values. MACCFG2[0000_0000_0000_0000_0111_0000_0000_0000]
Initialize ECNTRL, ECNTRL[0000_0000_0000_0000_1000_0000_0000_0000] (Used to setup FIFO mode = 1, and statistics enable = 0)
Clear IEVENT register, IEVENT[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize IMASK (Optional) IMASK[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize RCTRL (Optional) RCTRL[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize DMACTRL (Optional) DMACTRL[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize (Empty) Transmit Descriptor ring and fill buffers with Data Initialize TBASE0–TBASE7, TBASE0–TBASE7[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]
Initialize (Empty) Receive Descriptor ring and fill with empty buffers Initialize RBASE0–RBASE7, RBASE0–RBASE7[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]
Enable Transmit Queues Initialize TQUEUE
Enable Receive Queues Initialize RQUEUE
Enable Rx and Tx over FIFO, FIFOCFG[0000_0000_0000_0000_0011_0000_1101_1100] (Rx enable = 1, Tx enable = 1, enable flow control and CRC, 16-bit mode)

Chapter 14
DMA Controller

This chapter describes the DMA controller offered on this device.

14.1 Introduction

The DMA controller transfers blocks of data between the many interface and functional blocks of this device, independent of the e600 cores or external hosts.

14.1.1 Block Diagram

Figure 14-1 shows the block diagram of the DMA controller.

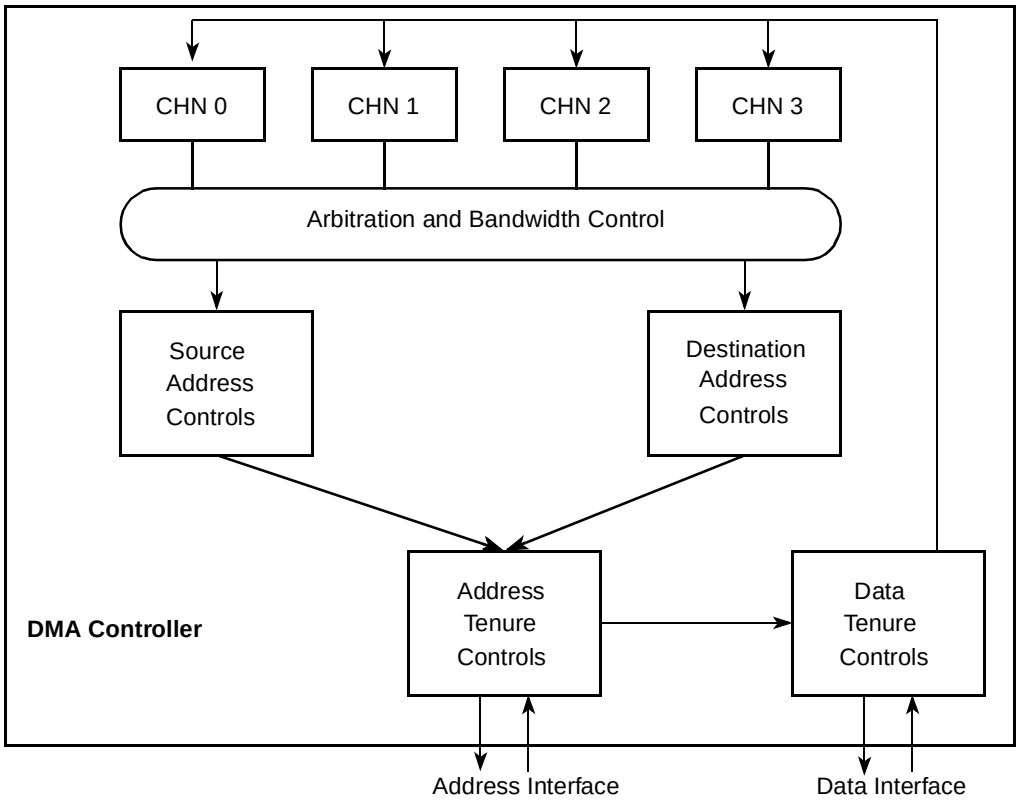


Figure 14-1. DMA Block Diagram

14.1.2 Overview

The DMA controller has four high-speed DMA channels. Both the e600 cores and external devices can initiate DMA transfers. All channels are capable of complex data movement and advanced transaction chaining. Figure 14-1 is a high-level block diagram of the DMA controller. Operations such as descriptor fetches and block transfers are initiated by each channel. A channel is selected by the arbitration logic and information is passed to the source and destination control blocks for processing. The source and destination blocks generate read and write requests to the address tenure engine, which manages the DMA master port address interface. After a transaction is accepted by the master port, control is transferred to the data tenure engine that manages the read and write data transfers. A channel remains active in the shared resources for the duration of the data transfer unless the allotted bandwidth per channel is reached.

14.1.3 Features

The DMA controller offers the following features:

- Four high-speed/high-bandwidth channels accessible by local and remote masters
- Basic DMA operation modes (direct, simple chaining)
- Extended DMA operation modes (advanced chaining and stride capability)
- Cascading descriptor chains
- Misaligned transfers
- Programmable bandwidth control between channels
- Up to 256 bytes for DMA sub-block transfers to maximize performance
- Three priority levels supported for source and destination transactions
- Interrupt on error and completed segment, list, or link
- Externally-controlled transfer using $\overline{\text{DMA\_DREQ}}$, $\overline{\text{DMA\_DACK}}$, and $\overline{\text{DMA\_DDONE}}$

14.1.4 Modes of Operation

The DMA block has two modes of operation: basic and extended. Basic mode is the DMA legacy mode, which does not support advanced features. Extended mode supports advanced features such as striding and flexible descriptor structures.

These two basic modes allow users to initiate and end DMA transfers in various ways. Table 14-1 summarizes the relationship between the modes and the following features:

- Direct mode. No descriptors are involved. Software must initialize the required fields as described in Table 14-1 before starting a transfer.
- Chaining mode. Software must initialize descriptors in memory and the required fields as described in Table 14-1 before starting a transfer.
- Single-write start mode. The DMA process can be started using a single-write command to either the descriptor address register in one of the chaining modes or the source/destination address registers in one of the direct modes.
- External control capability. This allows an external agent to start, pause, and check the status of a DMA transfer which has already been initialized.

- Channel continue capability. The channel continue capability allows software the flexibility of having the DMA controller start with descriptors that have already been programmed while software continues to build more descriptors in memory.
- Channel abort capability. The software can abort a previously initiated transfer by setting the bit MR*n*[CA]. The DMA controller terminates all outstanding transfers initiated by the channel without generating any errors before entering an idle state.

Table 14-1. Relationship of Modes and Features

Mode	Mode with One Additional Feature	Mode with Two Additional Features
B (Basic)	BD (basic direct)	BDS (BD single-write start)
		BDE (BD external control)
	BC (basic chaining)	BCE (BC external control)
		BCS (BC single-write start)
Ext (Extended)	ExtD (extended direct)	ExtDS (ExtD single-write start)
		ExtDE (ExtD external control)
	ExtC (extended chaining)	ExtCE (ExtC external control)
		ExtCS (ExtC single-write start)

Table 14-2 describes bit settings required for each DMA mode of operation.

Table 14-2. DMA Mode Bit Settings

Modes with Features	MR*n*[XFE]	MR*n*[CTM]	MR*n*[SRW]	MR*n*[CDSM/SWSM]	MR*n*[EMS_EN]
Basic Direct Modes					
Basic direct	0	1	0	0	0
Basic direct external control	0	1	0	0	1
Basic direct single-write start	0	1	1	1 or 0	0
Basic Chaining Modes					
Basic chaining	0	0	Reserved	0	0
Basic chaining external control	0	0	Reserved	0	1
Basic chaining single-write start	0	0	Reserved	1	0
Extended Direct Modes					
Extended direct	1	1	0	0	0
Extended direct external control	1	1	0	0	1
Extended direct single-write start	1	1	1	1 or 0	0

Table 14-2. DMA Mode Bit Settings (continued)

Modes with Features	MR*n*[XFE]	MR*n*[CTM]	MR*n*[SRW]	MR*n*[CDSM/SWSM]	MR*n*[EMS_EN]
Extended Chaining Modes					
Extended chaining	1	0	Reserved	0	0
Extended chaining external control	1	0	Reserved	0	1
Extended chaining single-write start	1	0	Reserved	1	0

Refer to Section 14.4, “Functional Description,” for details on these modes.

Figure 14-2 shows the general DMA operational flow chart.

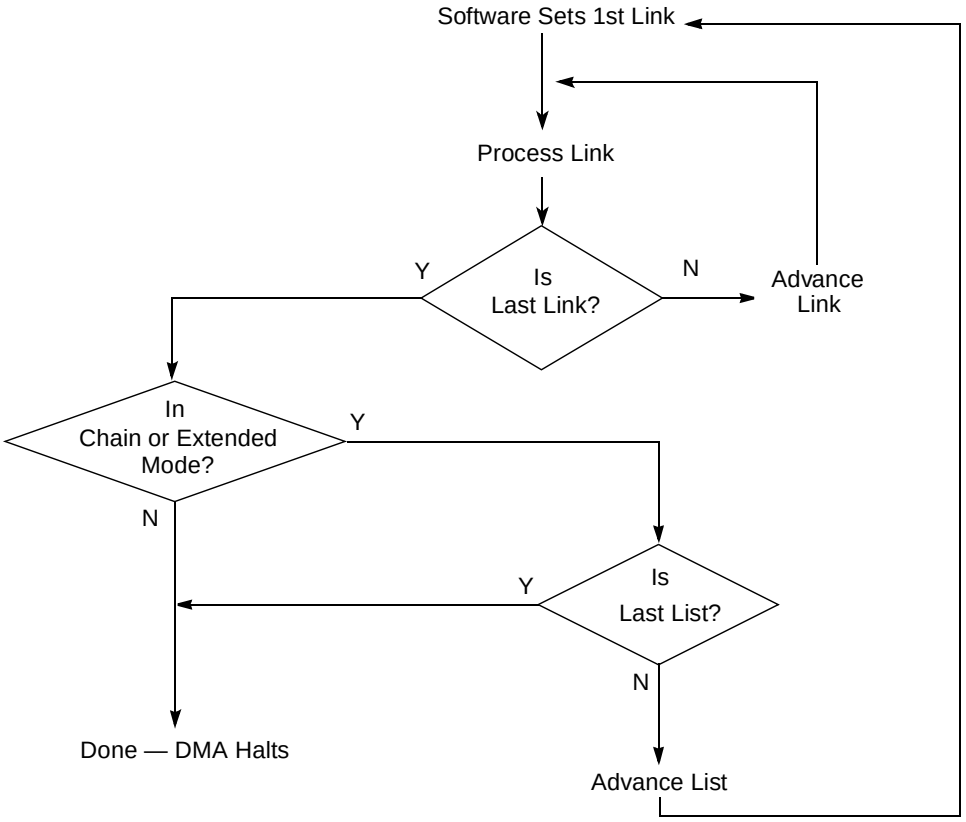


Figure 14-2. DMA Operational Flow Chart

14.2 External Signal Description

This section describes the DMA signals.

14.2.1 Signal Overview

Figure 14-3 summarizes the DMA controller signals.

Note that DMA channels 2 and 3 are multiplexed with local bus and interrupt signals as follows:

$\overline{\text{DMA\_DREQ[2]}}$ / $\overline{\text{LCS[5]}}$

$\overline{\text{DMA\_DACK[2]}}$ / $\overline{\text{LCS[6]}}$

$\overline{\text{DMA\_DDONE[2]}}$ / $\overline{\text{LCS[7]}}$

$\overline{\text{DMA\_DREQ[3]}}$ / IRQ[9]

$\overline{\text{DMA\_DACK[3]}}$ / IRQ[10]

$\overline{\text{DMA\_DDONE[3]}}$ / IRQ[11]

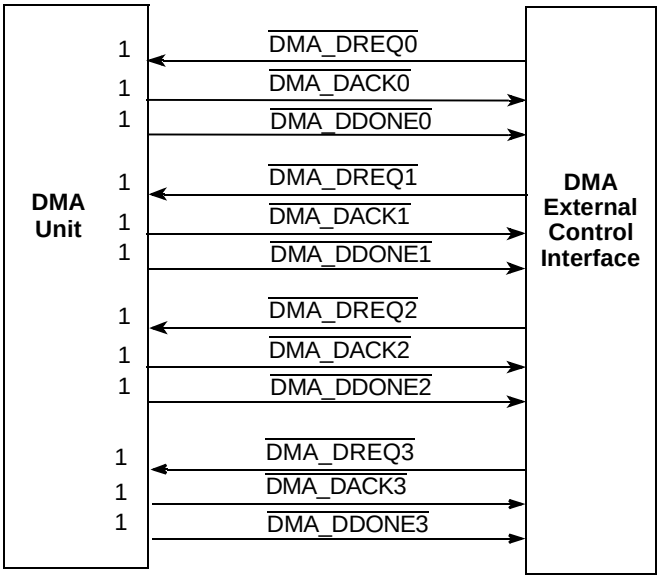


Figure 14-3. DMA Signal Summary

Note that the three DMA signals for DMA channel 3 are multiplexed with the IRQ9–11 signals on the device. These functions are mutually exclusive and the active function is specified in the PMUXCR register of the global utilities block as described in Section 17.4.1.8, “Alternate Function Signal Multiplex Control Register (PMUXCR).”

14.2.2 Detailed Signal Descriptions

Table 14-3 describes the DMA signals.

Table 14-3. DMA Signals—Detailed Signal Descriptions

Signal	I/O	Description	
$\overline{\text{DMA\_DREQ}}n$ DMA request	I	DMA request. Indicates the start of a DMA transfer or a restart from a paused request. Assertion of $\overline{\text{DMA\_DREQ}}n$ causes MR*n*[CS] to be set, thereby activating the corresponding DMA channel.	
		State Meaning	Asserted—Assertion of $\overline{\text{DMA\_DREQ}}n$ while $\overline{\text{DMA\_DACK}}n$ is negated causes a new transfer to start OR resumes a paused transfer if the EMP_EN bit is set. Assertion while $\overline{\text{DMA\_DACK}}n$ is asserted results in an illegal condition. Negated—Negation while $\overline{\text{DMA\_DACK}}n$ is asserted has no effect. Negation before the assertion of $\overline{\text{DMA\_DACK}}n$ results in an illegal condition.
		Timing	Assertion—Can be asserted asynchronously Negation— Must remain asserted at least until the assertion of the corresponding $\overline{\text{DMA\_DACK}}n$
$\overline{\text{DMA\_DACK}}n$	O	DMA acknowledge. Indicates that a DMA transfer is currently in progress	
		State Meaning	Asserted—Indicates that a DMA transfer is currently in progress. Asserted after the assertion of $\overline{\text{DMA\_DREQ}}n$ to indicate the start of a transfer Negated—Negated after finishing a complete transfer or after entering a paused state if MR*n*[EMP_EN] is set
		Timing	Assertion—Asynchronous assertion; asserted for more than three system clocks Negation—Asynchronous negation; negated for more than three system clocks
$\overline{\text{DMA\_DDONE}}n$	O	DMA done. Indicates that a DMA transfer is complete	
		State Meaning	Asserted—Indicates transfer completion. SRn[CB] is clear. Note, however, that write data may still be queued at the target interface or in the process of transfer on an external interface. Negated—Indicates that the current transfer is in process
		Timing	Assertion—Always asserts asynchronously after the negation of the final $\overline{\text{DMA\_DACK}}n$ to indicate completion of a transfer. For a paused transfer, $\overline{\text{DMA\_DDONE}}n$ is asserted asynchronously after the negation of the final $\overline{\text{DMA\_DACK}}n$. Negation—Negated asynchronously after the assertion of $\overline{\text{DMA\_DREQ}}n$ for the next transfer

14.3 Memory Map/Register Definition

This section provides a detailed description of all accessible DMA memory and registers. The descriptions include individual bit level descriptions and reset states of each register. Undefined 4-byte address spaces within offset 0x000–0xFFF are reserved.

Table 14-4 lists the DMA registers and their offsets. Note that the full register address is comprised of the programmable CCSRBAR together with the fixed DMA block base address and offset listed in Table 14-4.

Table 14-4. DMA Register Summary

DMA Controller —Block Base Address 0x2_1000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0x100	MR0—DMA 0 mode register	R/W	0x0000_0000	14.3.1.1/14-10
0x104	SR0—DMA 0 status register	Mixed	0x0000_0000	14.3.1.2/14-12
0x108	ECLNDAR0—DMA 0 current link descriptor extended address register	R/W	0x0000_0000	14.3.1.3/14-14
0x10C	CLNDAR0—DMA 0 current link descriptor address register	R/W	0x0000_0000	14.3.1.3/14-14
0x110	SATR0—DMA 0 source attributes register	R/W	0x0000_0000	14.3.1.4/14-16
0x114	SAR0—DMA 0 source address register	R/W	0x0000_0000	14.3.1.5/14-17
0x118	DATR0—DMA 0 destination attributes register	R/W	0x0000_0000	14.3.1.6/14-19
0x11C	DAR0—DMA 0 destination address register	R/W	0x0000_0000	14.3.1.7/14-20
0x120	BCR0—DMA 0 byte count register	R/W	0x0000_0000	14.3.1.8/14-22
0x124	ENLNDAR0—DMA 0 next link descriptor extended address register	R/W	0x0000_0000	14.3.1.9/14-22
0x128	NLNDAR0—DMA 0 next link descriptor address register	R/W	0x0000_0000	14.3.1.9/14-22
0x130	ECLSDAR0—DMA 0 current list descriptor extended address register	R/W	0x0000_0000	14.3.1.10/14-23
0x134	CLSDAR0—DMA 0 current list descriptor address register	R/W	0x0000_0000	14.3.1.10/14-23
0x138	ENLSDAR0—DMA 0 next list descriptor extended address register	R/W	0x0000_0000	14.3.1.11/14-25
0x13C	NLSDAR0—DMA 0 next list descriptor address register	Mixed	0x0000_0000	14.3.1.11/14-25
0x140	SSR0—DMA 0 source stride register	R/W	0x0000_0000	14.3.1.12/14-26
0x144	DSR0—DMA 0 destination stride register	R/W	0x0000_0000	14.3.1.13/14-26
0x148–0x17C	Reserved	—	—	—
0x180	MR1—DMA 1 mode register	R/W	0x0000_0000	14.3.1.1/14-10
0x184	SR1—DMA 1 status register	Mixed	0x0000_0000	14.3.1.2/14-12
0x188	ECLNDAR1—DMA 1 current link descriptor extended address register	R/W	0x0000_0000	14.3.1.3/14-14
0x18C	CLNDAR1—DMA 1 current link descriptor address register	R/W	0x0000_0000	14.3.1.3/14-14
0x190	SATR1—DMA 1 source attributes register	R/W	0x0000_0000	14.3.1.4/14-16
0x194	SAR1—DMA 1 source address register	R/W	0x0000_0000	14.3.1.5/14-17
0x198	DATR1—DMA 1 destination attributes register	R/W	0x0000_0000	14.3.1.6/14-19
0x19C	DAR1—DMA 1 destination address register	R/W	0x0000_0000	14.3.1.7/14-20
0x1A0	BCR1—DMA 1 byte count register	R/W	0x0000_0000	14.3.1.8/14-22
0x1A4	ENLNDAR1—DMA 1 next link descriptor extended address register	R/W	0x0000_0000	14.3.1.9/14-22

Table 14-4. DMA Register Summary (continued)

DMA Controller —Block Base Address 0x2_1000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0x1A8	NLNDAR1—DMA 1 next link descriptor address register	R/W	0x0000_0000	14.3.1.9/14-22
0x1B0	ECLSDAR1—DMA 1 current list descriptor extended address register	R/W	0x0000_0000	14.3.1.10/14-23
0x1B4	CLSDAR1—DMA 1 current list descriptor address register	R/W	0x0000_0000	14.3.1.10/14-23
0x1B8	ENLSDAR1—DMA 1 next list descriptor extended address register	R/W	0x0000_0000	14.3.1.11/14-25
0x1BC	NLSDAR1—DMA 1 next list descriptor address register	R/W	0x0000_0000	14.3.1.11/14-25
0x1C0	SSR1—DMA 1 source stride register	R/W	0x0000_0000	14.3.1.12/14-26
0x1C4	DSR1—DMA 1 destination stride register	R/W	0x0000_0000	14.3.1.13/14-26
0x1C8–0x1FC	Reserved	—	—	—
0x200	MR2—DMA 2 mode register	R/W	0x0000_0000	14.3.1.1/14-10
0x204	SR2—DMA 2 status register	Mixed	0x0000_0000	14.3.1.2/14-12
0x208	ECLNDAR2—DMA 2 current link descriptor extended address register	R/W	0x0000_0000	14.3.1.3/14-14
0x20C	CLNDAR2—DMA 2 current link descriptor address register	R/W	0x0000_0000	14.3.1.3/14-14
0x210	SATR2—DMA 2 source attributes register	R/W	0x0000_0000	14.3.1.4/14-16
0x214	SAR2—DMA 2 source address register	R/W	0x0000_0000	14.3.1.5/14-17
0x218	DATR2—DMA 2 destination attributes register	R/W	0x0000_0000	14.3.1.6/14-19
0x21C	DAR2—DMA 2 destination address register	R/W	0x0000_0000	14.3.1.7/14-20
0x220	BCR2—DMA 2 byte count register	R/W	0x0000_0000	14.3.1.8/14-22
0x224	ENLNDAR2—DMA 2 next link descriptor extended address register	R/W	0x0000_0000	14.3.1.9/14-22
0x228	NLNDAR2—DMA 2 next link descriptor address register	R/W	0x0000_0000	14.3.1.9/14-22
0x230	ECLSDAR2—DMA 2 current list descriptor extended address register	R/W	0x0000_0000	14.3.1.10/14-23
0x234	CLSDAR2—DMA 2 current list descriptor address register	R/W	0x0000_0000	14.3.1.10/14-23
0x238	ENLSDAR2—DMA 2 next list descriptor extended address register	R/W	0x0000_0000	14.3.1.11/14-25
0x23C	NLSDAR2—DMA 2 next list descriptor address register	R/W	0x0000_0000	14.3.1.11/14-25
0x240	SSR2—DMA 2 source stride register	R/W	0x0000_0000	14.3.1.12/14-26
0x244	DSR2—DMA 2 destination stride register	R/W	0x0000_0000	14.3.1.13/14-26
0x248–0x27C	Reserved	—	—	—
0x280	MR3—DMA 3 mode register	R/W	0x0000_0000	14.3.1.1/14-10
0x284	SR3—DMA 3 status register	Mixed	0x0000_0000	14.3.1.2/14-12

Table 14-4. DMA Register Summary (continued)

DMA Controller —Block Base Address 0x2_1000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0x288	ECLNDAR3—DMA 3 current link descriptor extended address register	R/W	0x0000_0000	14.3.1.3/14-14
0x28C	CLNDAR3—DMA 3 current link descriptor address register	R/W	0x0000_0000	14.3.1.3/14-14
0x290	SATR3—DMA 3 source attributes register	R/W	0x0000_0000	14.3.1.4/14-16
0x294	SAR3—DMA 3 source address register	R/W	0x0000_0000	14.3.1.5/14-17
0x298	DATR3—DMA 3 destination attributes register	R/W	0x0000_0000	14.3.1.6/14-19
0x29C	DAR3—DMA 3 destination address register	R/W	0x0000_0000	14.3.1.7/14-20
0x2A0	BCR3—DMA 3 byte count register	R/W	0x0000_0000	14.3.1.8/14-22
0x2A4	ENLNDAR3—DMA 3 next link descriptor extended address register	R/W	0x0000_0000	14.3.1.9/14-22
0x2A8	NLNDAR3—DMA 3 next link descriptor address register	R/W	0x0000_0000	14.3.1.9/14-22
0x2B0	ECLSDAR3—DMA 3 current list descriptor extended address register	R/W	0x0000_0000	14.3.1.10/14-23
0x2B4	CLSDAR3—DMA 3 current list descriptor address register	R/W	0x0000_0000	14.3.1.10/14-23
0x2B8	ENLSDAR3—DMA 3 next list descriptor extended address register	R/W	0x0000_0000	14.3.1.11/14-25
0x2BC	NLSDAR3—DMA 3 next list descriptor address register	R/W	0x0000_0000	14.3.1.11/14-25
0x2C0	SSR3—DMA 3 source stride register	R/W	0x0000_0000	14.3.1.12/14-26
0x2C4	DSR3—DMA 3 destination stride register	R/W	0x0000_0000	14.3.1.13/14-26
0x2C8–0x2FC	Reserved	—	—	—
0x300	DGSR—DMA general status register	R	0x0000_0000	14.3.1.14/14-27

14.3.1 DMA Register Descriptions

The following sections describe the DMA registers. The majority of these registers are channel-specific and can be identified by one of the four offsets that describe the register.

14.3.1.1 Mode Registers (MR*n*)

The mode register allows software to start a DMA transfer and to control various DMA transfer characteristics. Figure 14-4 describes the MR*n*.

Offset 0x100, 0x180, 0x200, 0x280 — Access: Read/Write

	0–3	4–7	8	9	10	11–12	13	14	15
R W	—	BWC	—		EMP_EN	—	EMS_EN	DAHTS	
Reset	All zeros								

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	SAHTS		DAHE	SAHE	—	SRW	EOSIE	EOLNIE	EOLSIE	EIE	XFE	CDSM/SWSM	CA	CTM	CC	CS
Reset	All zeros															

Figure 14-4. DMA Mode Registers (MR*n*)

Table 14-5 describes the MR*n* fields.

Table 14-5. MR*n* Field Descriptions

Bits	Name	Description
0–3	—	Reserved
4–7	BWC	Bandwidth/pause control. If multiple channels are executing transfers concurrently the value of MR*n*[BWC] determines how many bytes a given channel is allowed to transfer before the DMA engine pauses the current channel and switches to the next channel. If only one channel is executing transfers the value of MR*n*[BWC] dictates how many bytes are allowed to transfer before pausing the channel, after which a new assertion of $\overline{\text{DREQ}}$ resumes channel operation. 0000 1 byte 0001 2 bytes 0010 4 bytes 0011 8 bytes 0100 16 bytes 0101 32 bytes 0110 64 bytes 0111 128 bytes 1000 256 bytes 1001 512 bytes 1010 1024 bytes 1011–1110 Reserved 1111 Disable bandwidth sharing to allow uninterrupted transfers from each channel.
8–9	—	Reserved
10	EMP_EN	External master pause enable. Valid only if MR*n*[EMS_EN] is set. 0 Disable the external master pause feature. 1 Enable the external master pause feature. Channel is paused as described by MR*n*[BWC].
11–12	—	Reserved
13	EMS_EN	External master start enable. This bit does not apply to single-write start modes (direct or chaining). 0 Disable the channel from being started by an external DMA start pin. 1 Enable the channel to be started by an external DMA start pin, which sets MR*n*[CS].

Table 14-5. MR*n* Field Descriptions (continued)

Bits	Name	Description
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.
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.

Table 14-5. MR*n* Field Descriptions (continued)

Bits	Name	Description
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.
30	CC	Channel continue. This bit applies only to chaining mode and is cleared by hardware after the first descriptor read when continuing a transfer. This bit is reserved for external master mode. 0 No effect 1 The DMA transfer restarts the transferring process starting at the current descriptor address.
31	CS	Channel start. This bit is also automatically set by hardware during single-write start mode and external master start enable mode. Note that in external control mode, deasserting $\overline{\text{DMA\_DREQ}}$ does NOT clear this bit. 0 Halt the DMA process if channel is busy (SR*n*[CB] is set). No effect if the channel is not busy. 1 Start the DMA process if channel is not busy (CB is cleared). If the channel was halted (CS = 0 and CB = 1), the transfer continues from the point at which it was halted.

14.3.1.2 Status Registers (SR*n*)

The status registers, shown in Figure 14-5, report various DMA conditions during and after a DMA transfer.

Offset 0x104 0x184 0x204 0x284 Access: Mixed

	0 – 23	24	25	26	27	28	29	30	31
R	—	TE	—	CH	PE	EOLNI	CB	EOSI	EOLSI
W		w1c			w1c	w1c		w1c	w1c
Reset	All zeros								

Figure 14-5. Status Registers (SR*n*)

Table 14-6 describes the bits of the SR*n*.

Table 14-6. SR*n* Field Descriptions

Bits	Name	Description
0–23	—	Reserved
24	TE	Transfer error (Bit reset, write 1 to clear) 0 No error condition during the DMA transfer 1 Error condition during the DMA transfer. See Section 14.4.3, “DMA Errors,” for additional information.
25	—	Reserved
26	CH	Channel halted. Cleared automatically by hardware if MR*n*[CS] is set again for resuming a halted transfer 0 Channel is not halted. If software attempts to halt an idle channel (SR*n*[CB] is cleared), this bit remains 0. 1 DMA transfer was successfully halted by software and can be resumed.
27	PE	Programming error (bit reset, write 1 to clear) 0 No programming error detected 1 A programming error is detected that prevents the DMA transfer from occurring.
28	EOLNI	End-of-links interrupt. After transferring the last block of data in the last link descriptor, if MR*n*[EOLSIE] is set, then this bit is set and an interrupt is generated. (Bit reset, write 1 to clear)
29	CB	Channel busy 0 DMA transfer is finished, an error occurred, or a channel abort occurred. 1 DMA transfer is currently in progress.
30	EOSI	End-of-segment interrupt. In chaining mode, after finishing a data transfer, if MR*n*[EOSIE] is set or if CLNDAR*n*[EOSIE] is set, this bit gets set and an interrupt is generated. In direct mode, if MRn[EOSIE] is set, this bit gets set and an interrupt is generated. (Bit reset, write 1 to clear)
31	EOLSI	End-of-list interrupt. After transferring the last block of data in the last list descriptor, if MR*n*[EOLSIE] is set, then this bit is set and an interrupt is generated. (Bit reset, write 1 to clear)

14.3.1.3 Current Link Descriptor Address Registers (CLNDAR*n* and ECLNDAR*n*)

Current link descriptor address registers contain the address of the current link descriptor. In basic chaining mode, shown in Figure 14-6, software must initialize these registers to point to the first link descriptors in memory.

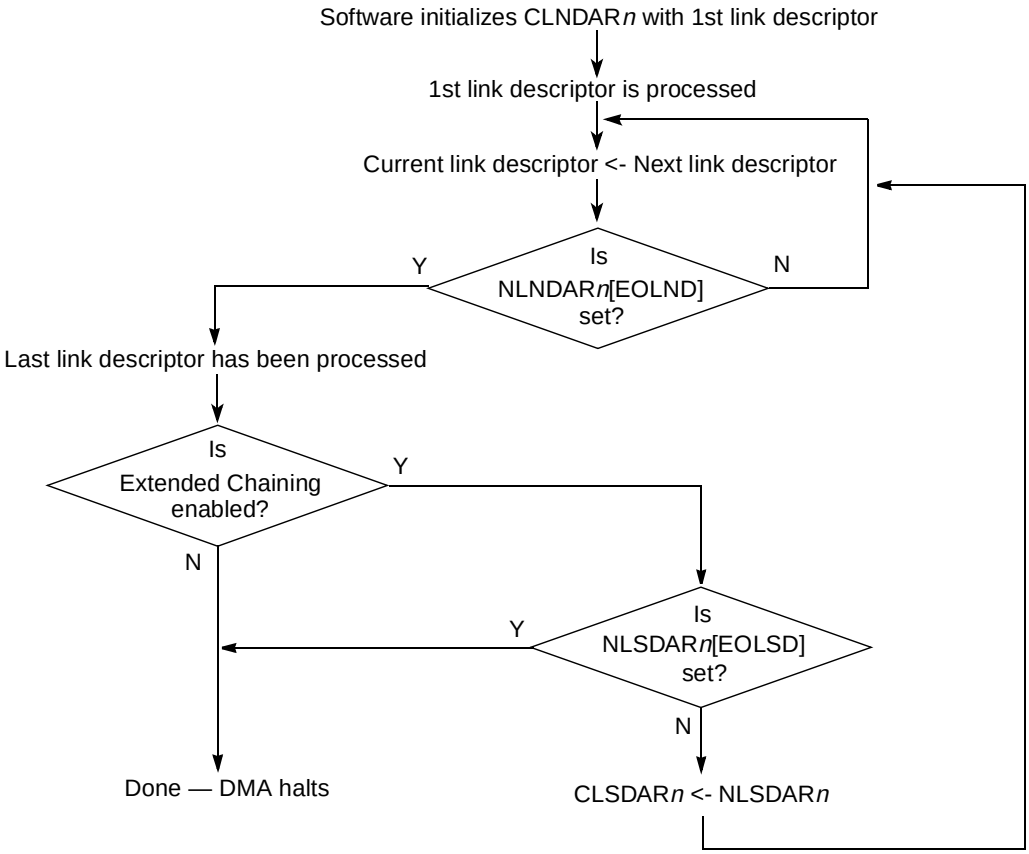


Figure 14-6. Basic Chaining Mode Flow Chart

After the current descriptor is processed, the current link descriptor address register is loaded from the next link descriptor address registers and NLNDAR*n*[EOLND] in the next link descriptor address register is examined. If EOLND is zero, the DMA controller reads in the new current link descriptor for processing. If EOLND is set, the last descriptor of the list was just completed. If extended chaining mode is not enabled, all DMA transfers are complete and the DMA controller halts.

If extended chaining mode is enabled, the DMA controller examines the state of NLSDAR*n*[EOLSD] in the next list descriptor address register. If EOLSD is clear, the controller loads the contents of the next list descriptor address register into the current list descriptor address register and reads the new list descriptor from memory. If EOLSD is set, all DMA transfers are complete and the DMA controller halts.

Figure 14-7 shows ECLNDAR*n*.

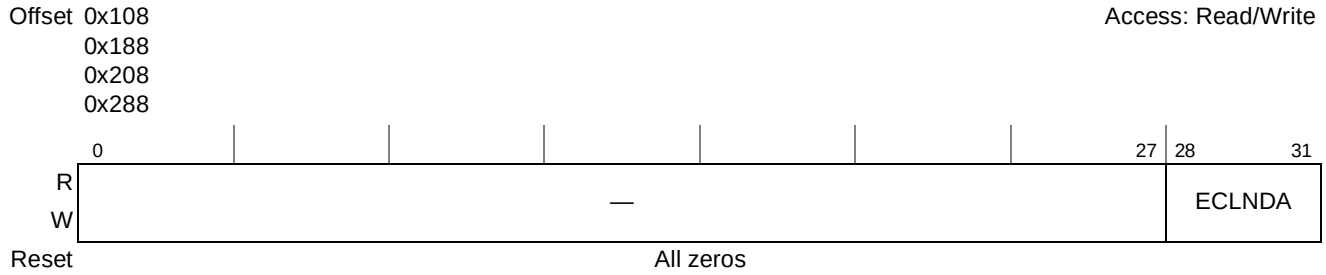


Figure 14-7. Extended Current Link Descriptor Address Registers (ECLNDAR*n*)

Table 14-7. ECLNDAR*n* Field Descriptions

Bit	Name	Description
0–27	—	Reserved
28–31	ECLNDA	Current link descriptor extended address (upper 4 bits of 36-bit address)

Figure 14-8 shows CLNDAR*n*.

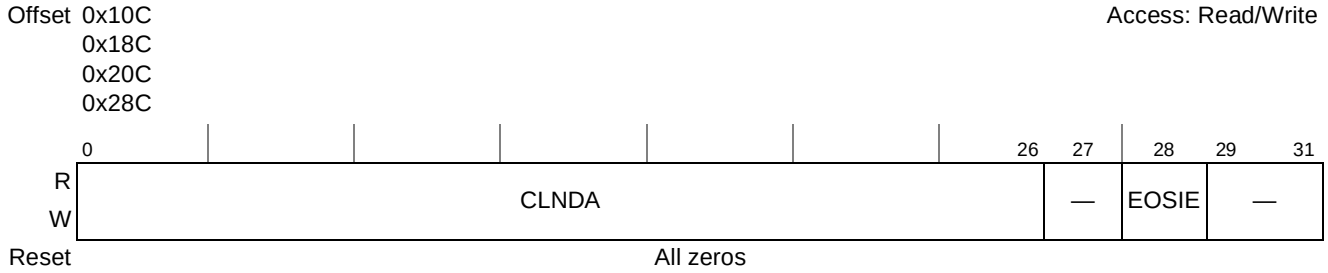


Figure 14-8. Current Link Descriptor Address Registers (CLNDAR*n*)

Table 14-8 describes the fields of the CLNDAR*n*.

Table 14-8. CLNDAR*n* Field Descriptions

Bits	Name	Description
0–26	CLNDA	Current link descriptor address. Contains the current descriptor address of the buffer descriptor in memory. The descriptor must be aligned to a 32-byte boundary. (This is the lower portion of the 36-bit address formed by CLNDAR*n*[CLNDA] and ECLNDAR*n*[ECLNDA].)
27	—	Reserved
28	EOSIE	End-of-segment interrupt enable 0 Do not generate an interrupt upon completion of the current DMA transfer for the current link descriptor. 1 Generate an interrupt upon completion of the current DMA transfer for the current link descriptor.
29–31	—	Reserved

14.3.1.4 Source Attributes Registers (SATR*n*)

The source attributes registers, shown in Figure 14-9, contain the transaction attributes to be used for the DMA operation. Stride mode is enabled by setting SATR*n*[SSME].

If SATR*n*[SBPATMU] is cleared, the target interface is derived from the local access ATMU mapping and the transaction is obtained from the value specified in SATR*n*[SREADTTYPE] using the local address space category.

ATMU bypass mode is enabled by setting SATR*n*[SBPATMU] and is only applicable for accesses to RapidIO. If SATR*n*[SBPATMU] is set, SATR*n*[STRANSINT] must be set to RapidIO and attributes that would otherwise come from the ATMU must be specified in this register. If SATR*n*[SBPATMU] is cleared, attributes are derived from local access windows and outbound ATMU registers according to the transaction address. Note that ATMU bypass mode is not supported in RapidIO large transport system size.

Offset 0x110, 0x190, 0x210, 0x290 — Access: Read/Write

	0–1	2	3	4–5	6	7	8–11	12–15	16–21	22–31
R W	—	SPBATMU	—	STFLOWLVL	SPCIORDER	SSME	STRANSINT	SREADTTYPE	—	ESAD
Reset	All zeros									

Figure 14-9. Source Attributes Registers (SATR*n*)

Table 14-9 describes the fields of the SATR*n*.

Table 14-9. 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.

Table 14-9. SATR*n* Field Descriptions (continued)

Bits	Name	Description
7	SSME	Source stride mode enable 0 Stride mode disabled 1 Stride mode enabled Ignored in basic mode (MR*n*[XFE] is cleared). Striding on the source address can be accomplished by enabling SATR*n*[SSME] and setting the desired stride size and distance in the SSR*n*.
8–11	STRANSINT	DMA source transaction interface. 0000 PCI Express interface 1 0001 PCI Express interface 2 0010 Reserved 0011–1011Reserved 1100 Serial RapidIO interface 1101–1110Reserved Applicable only to RapidIO interface, while SATR*n*[SBPATMU] is set.
12–15	SREADTTYPE	DMA source transaction type. Reserved values 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 0111–1111 Reserved
16–21	—	Reserved
22–31	ESAD	Extended source address. ESAD[6–9] represents the four high-order bits of the 36-bit source address. When used for RapidIO interface, ESAD[0–7] represents the target ID and ESAD[8–9] represent the two high-order bits of the local device offset over the RapidIO interface.

14.3.1.5 Source Address Registers (SAR*n*)

The source address registers, shown in Figure 14-10, contain the address from which the DMA controller reads data. In direct mode, if MR*n*[CDSM/SWSM] and MR*n*[SRW] are set, a write to this register simultaneously sets MR*n*[CS], starting a DMA transfer. Software must ensure that this is a valid address.

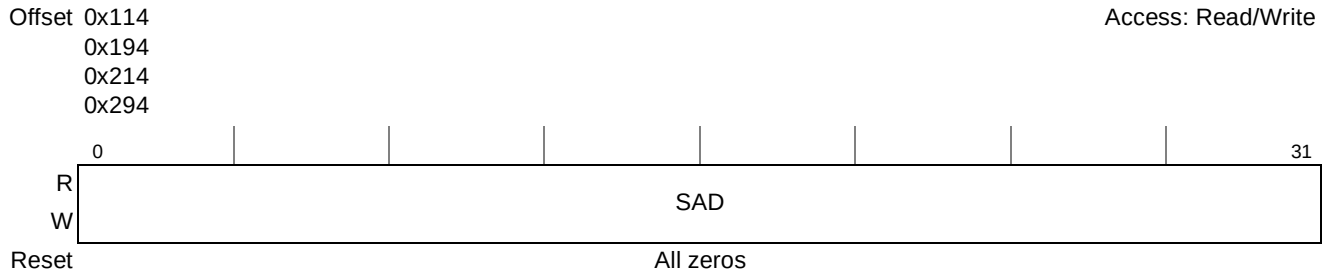


Figure 14-10. Source Address Registers (SAR*n*)

Table 14-10 describes the field of the SAR*n*.

Table 14-10. SAR*n* Field Descriptions

Bits	Name	Description
0–31	SAD	Source address. This register contains the low-order bits of the 36-bit source address of the DMA transfer. The contents are updated after every DMA write operation unless the final stride of a striding operation is less than the stride size, in which case it remains equal to the address from which the last stride began.

14.3.1.5.1 Source Address Registers for RapidIO Maintenance Reads (SAR*n*)

If RapidIO is the source of a transaction, the SAR*n* registers are redefined as shown below. Several options exist for the packet type that can be specified. A maintenance read is one of many possible reads. Maintenance packets have an offset instead of an address. Figure 14-11 describes the SAR*n*.

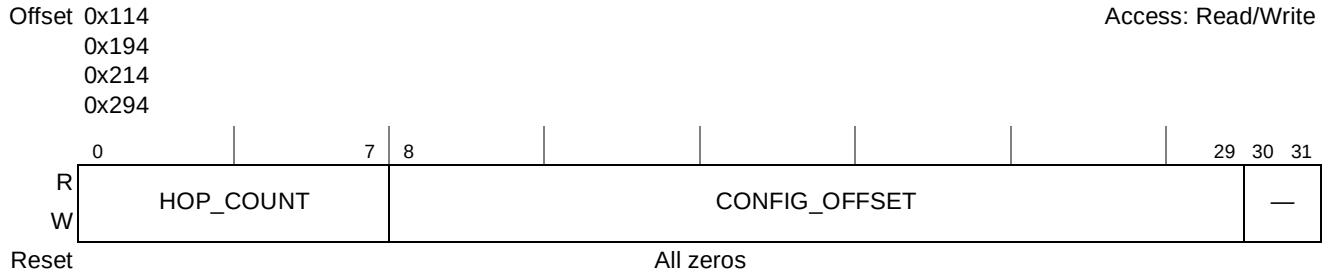


Figure 14-11. Source Address Registers for RapidIO Maintenance Reads (SAR*n*)

Table 14-11 describes the fields of the SAR*n*.

Table 14-11. SAR*n* Field Descriptions

Bits	Name	Description
0–7	HOP_COUNT	Maintenance packet hop_count as defined by the *RapidIO Interconnect Specification 1.2*
8–29	CONFIG_OFFSET	Maintenance packet word offset as defined by the *RapidIO Interconnect Specification 1.2*
30–31	—	Reserved

14.3.1.6 Destination Attributes Registers (DATR*n*)

The destination attributes registers, shown in Figure 14-12, contain the transaction attributes for the DMA operation. Stride mode is enabled by setting DATR*n*[DSME].

ATMU bypass mode, which is applicable only for accesses to RapidIO, is enabled by setting DATR*n*[DBPATMU]. If DATR*n*[DBPATMU] is set, DATR*n*[DTRANSINT] must be set to RapidIO and attributes that would otherwise come from the ATMU must be specified in this register. Note that ATMU bypass mode is not supported in RapidIO large transport system size.

If DATR*n*[DBPATMU] is cleared, the target interface is derived from the local access ATMU mapping and the transaction is obtained from the value specified in DATR*n*[DWRITETTYPE] using the local address space category.

Offset 0x118, 0x198, 0x218, 0x298 — Access: Read/Write

	0	1	2	3	4	5	6	7	8	11	12	15	16	21	22		31
R W	—		DPBATMU	—	DTFLOWLVL		DPCIORDER	DSME	DTRANSINT		DWRITETTYPE		—		EDAD		
Reset	All zeros																

Figure 14-12. Destination Attributes Registers (DATR*n*)

Table 14-12 describes the fields of the DATR*n*.

Table 14-12. 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	PCI ordering rules enable. Applicable only while DATR*n*[DBPATMU] is set. 0 Retain original transaction ordering 1 Follow PCI transaction ordering rules on RapidIO (elevate write priority one level over reads).

Table 14-12. DATR*n* Field Descriptions (continued)

Bits	Name	Description
7	DSME	Destination stride mode enable 0 Stride mode disabled 1 Stride mode enabled Ignored in basic mode (MR*n*[XFE] is cleared). Striding on the destination address can be accomplished by setting DSME and setting the desired stride size and distance in DSR*n*.
8–11	DTRANSINT	DMA destination transaction interface. Applicable only while DATR*n*[DBPATMU] is set. 0000 PCI Express interface 1 0001 PCI Express interface 2 0010 Reserved 0011–1011Reserved 1100 Serial RapidIO interface 1101–1110Reserved
12–15	DWRITETTYPE	DMA destination transaction type. Reserved values result in a programming error being detected and logged in SR[PE]. Transaction type to run on RapidIO interface in ATMU bypass mode 0000 Reserved 0001 Flush 0010 Reserved 0011 SWRITE for all but last, NWRITE_R for last transaction 0100 NWRITE for all but last, NWRITE_R for last transaction 0101 NWRITE_R 0110 Message of size 8, 16, 32, 64, 128 or 256 bytes. (Other message sizes produce boundedly undefined behavior.) 0111 MAINTENANCE write 1000–1111 Reserved Transaction type to run on local address space—Used even in non-ATMU bypass mode 0000–0011 Reserved 0100 Write, don’t snoop local processor 0101 Write, snoop local processor 0110 Reserved 0111 Reserved 1000–1111 Reserved
16–21	—	Reserved
22–31	EDAD	Extended destination address. EDAD[6–9] represents the four high-order bits of the 36-bit destination address. When used for RapidIO interface, EDAD[0–7] represents the target ID and EDAD[8–9] is defined as follows, subject to the transaction type: Message: EDAD[8–9] represents the value for the mbox field in the message packet. Other: EDAD[8–9] represents the two high-order bits of the local device offset.

14.3.1.7 Destination Address Registers (DAR*n*)

The destination address registers, shown in Figure 14-13, contain the addresses to which the DMA controller writes data.

In direct mode, if MR*n*[SRW] is set and MR*n*[CDSM/SWSM] is cleared, a write to this register simultaneously sets MR*n*[CS], starting a DMA transfer. Software must ensure that this is a valid address.

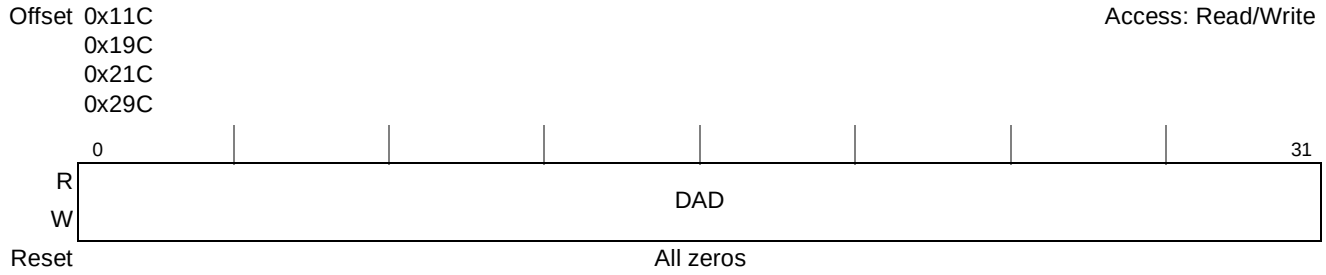


Figure 14-13. Destination Address Registers (DAR*n*)

Table 14-13 describes the field of the DAR*n*.

Table 14-13. DAR*n* Field Descriptions

Bits	Name	Description
0–31	DAD	Destination address. This register contains the low-order bits of the 36-bit destination address of the DMA transfer. The contents are updated after every DMA write operation unless the final stride of a striding operation is less than the stride size, in which case it remains equal to the address from which the last stride began.

14.3.1.7.1 Destination Address Registers for RapidIO Maintenance Writes (DAR*n*)

If RapidIO is the destination of a transaction, the DAR*n* registers are redefined as shown below. Several options exist for the transaction type that can be specified. There are a number of noncoherent write and flush types for address-based write transactions, and message types for port-based write transactions.

Maintenance packets have an offset instead of an address. Figure 14-14 shows the DAR*n*.

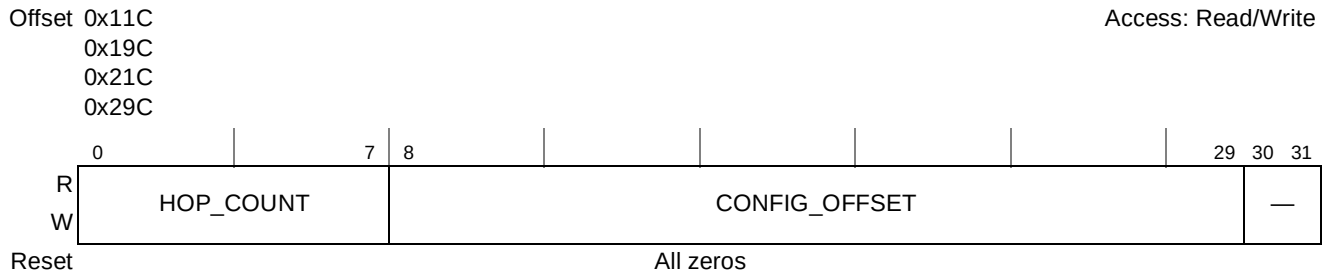


Figure 14-14. Destination Address Registers for RapidIO Maintenance Writes (DAR*n*)

Table 14-14 describes the fields of the DAR*n*.

Table 14-14. DAR*n* Field Descriptions

Bits	Name	Description
0–7	HOP_COUNT	Maintenance packet hop count as defined by the *RapidIO Interconnect Specification 1.2*
8–29	CONFIG_OFFSET	Maintenance packet word offset as defined by the *RapidIO Interconnect Specification 1.2*
30–31	—	Reserved

14.3.1.8 Byte Count Registers (BCR*n*)

The byte count register, shown in Figure 14-15, contains the number of bytes to transfer.

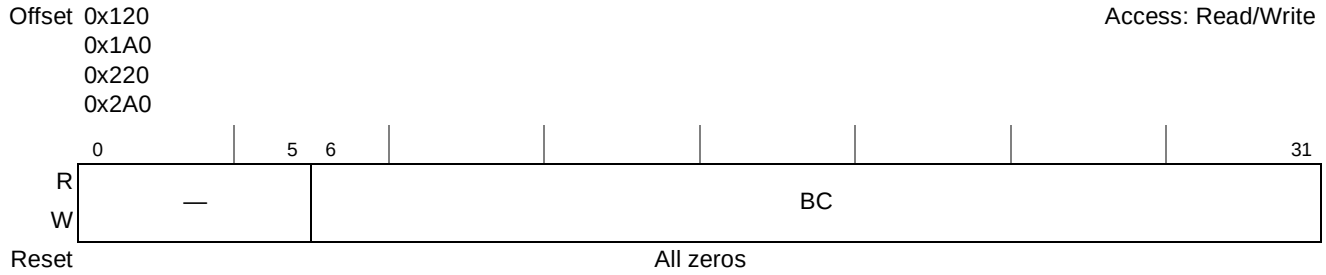


Figure 14-15. Byte Count Registers (BCR*n*)

Table 14-15 describes the fields of the BCR*n*.

Table 14-15. BCR*n* Field Descriptions

Bits	Name	Description
0–5	—	Reserved
6–31	BC	Byte count. Contains the number of bytes to transfer. The value in this register is decremented after each DMA read operation. The maximum transfer size is (2^{26}) – 1 bytes.

14.3.1.9 Next Link Descriptor Address Registers (NLNDAR*n* and ENLNDAR*n*)

The next link descriptor address registers, shown in Figure 14-16 and Figure 14-17, contain the address for the next link descriptor in memory. Contents transferred to the current descriptor address registers become effective for the current transfer in basic and extended chaining modes.

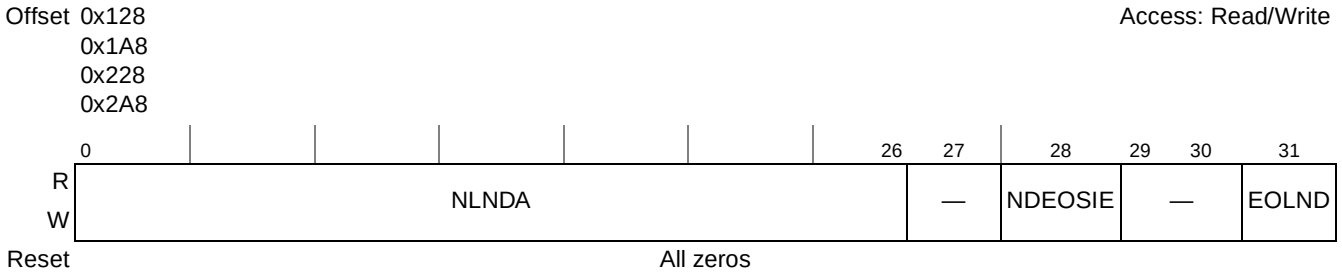


Figure 14-16. Next Link Descriptor Address Registers (NLNDAR*n*)

Table 14-16 describes the fields of the NLNDAR*n* registers.

Table 14-16. NLNDAR*n* Field Descriptions

Bits	Name	Description
0–26	NLNDA	Next link descriptor address. Contains the next link descriptor address in memory. The descriptor must be aligned to a 32-byte boundary.
27	—	Reserved

Table 14-16. 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.

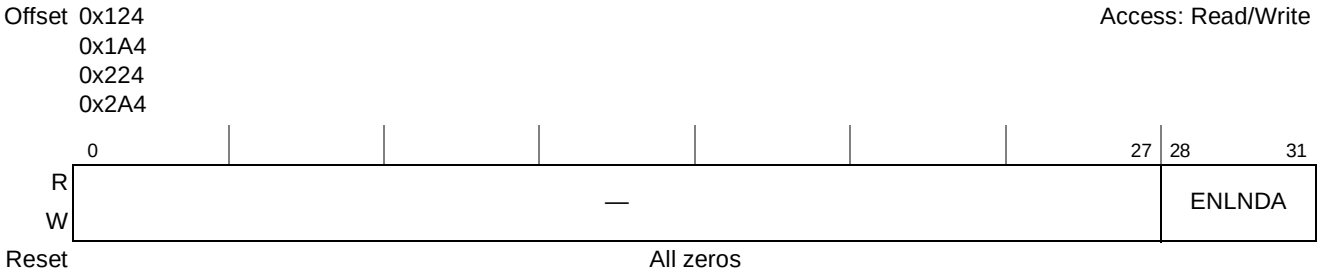


Figure 14-17. Extended Next Link Descriptor Address Registers (ENLNDAR*n*)

Table 14-17 describes the fields of the ENLNDAR*n* registers.

Table 14-17. ENLNDAR*n* Field Descriptions

Bit	Name	Description
0–27	—	Reserved
28–31	ENLNDA	Next link descriptor extended address bits (upper 4 bits of 36-bit address)

14.3.1.10 Current List Descriptor Address Registers (CLSDAR*n* and ECLSDAR*n*)

The current list descriptor address registers, shown in Figure 14-19 and Table 14-19, contain the current address of the list descriptor in memory in extended chaining mode.

In extended chaining mode, software must initialize CLSDAR*n* and ECLSDAR*n* to point to the first list descriptor in memory. After finishing the last link descriptor in the current list, the DMA controller loads the contents of the next list descriptor address register into the current list descriptor address register. If NLSDAR*n*[EOLSD] in the next list descriptor address register is clear, the DMA controller reads the new current list descriptor from memory to process that list. If EOLSD in the next list descriptor address register is set and the last link in the current list is finished all DMA transfers are complete.

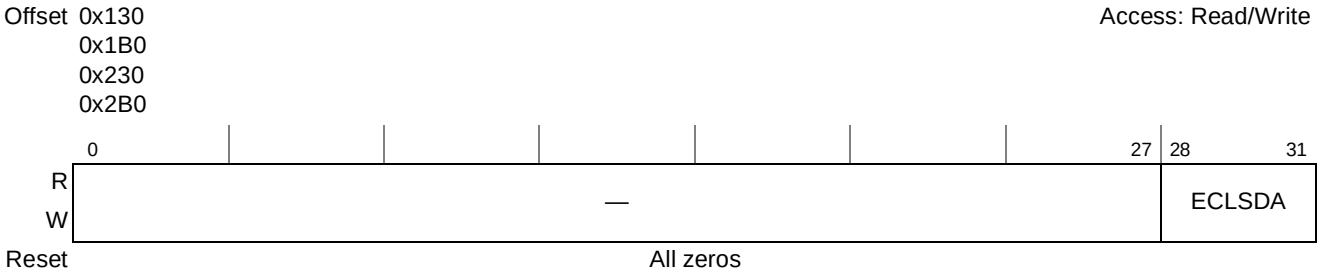


Figure 14-18. Extended Current List Descriptor Address Registers (ECLSDAR*n*)

Table 14-18. ECLSDAR*n* Field Descriptions

Bits	Name	Description
0–27	—	Reserved
28–31	ECLSDA	Current list descriptor extended address bits (upper 4 bits of 36-bit address)

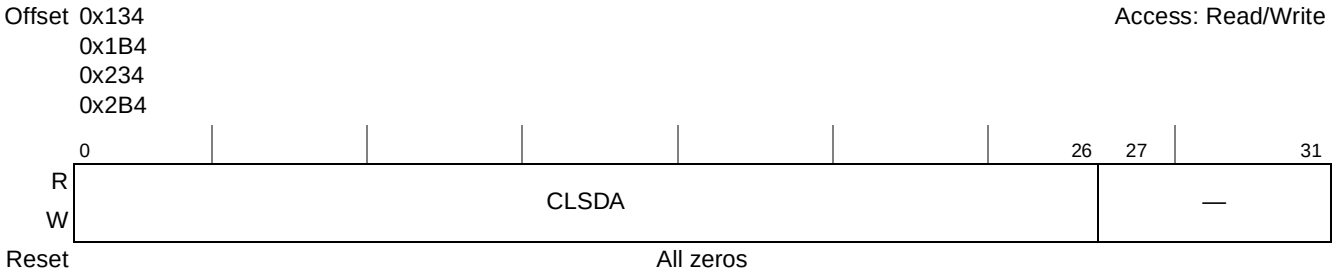


Figure 14-19. Current List Descriptor Address Registers (CLSDAR*n*)

Table 14-19 describes the fields of the CLSDAR*n*.

Table 14-19. CLSDAR*n* Field Descriptions

Bits	Name	Description
0–26	CLSDA	Current list descriptor address. Contains the low-order bits of the 36-bit current list descriptor address of the buffer descriptor in memory in extended chaining mode. The descriptor must be aligned to a 32-byte boundary.
27–31	—	Reserved

14.3.1.11 Next List Descriptor Address Registers (NLSDAR*n* and ENLSDAR*n*)

The next list descriptor address registers, shown in Figure 14-20 and Figure 14-21, contain the address for the next list descriptor in memory. If the contents are transferred to the current list descriptor address register they become effective for the current transfer in extended chaining mode.

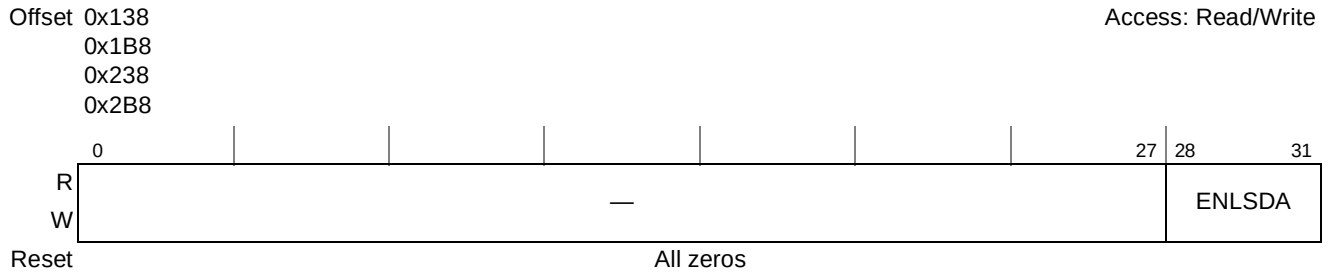


Figure 14-20. Extended Next List Descriptor Address Registers (ENLSDAR*n*)

Table 14-20. ENLSDAR*n* Field Descriptions

Bits	Name	Description
0–27	—	Reserved
28–31	ENLSDA	Next list descriptor extended address bits (upper 4 bits of 36-bit address)

Table 14-21 describes the fields of the NLSDAR*n*.

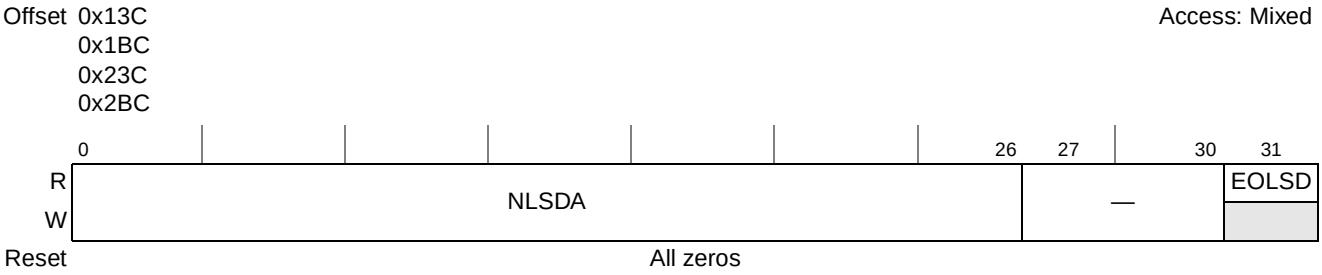


Figure 14-21. Next List Descriptor Address Registers (NLSDAR*n*)

Table 14-21. NLSDAR*n* Field Descriptions

Bits	Name	Description
0–26	NLSDA	Next list descriptor address. Contains the low-order bits of the 36-bit next descriptor address of the buffer descriptor in memory. The descriptor must be aligned on a 32-byte boundary.
27–30	—	Reserved
31	EOLSD	End-of-lists descriptor. This bit is ignored in direct mode. 0 This list descriptor is not the last list descriptor in memory. 1 This list descriptor is the last list descriptor in memory. If this bit is set, then the DMA controller halts after the last link descriptor transaction is finished.

14.3.1.12 Source Stride Registers (SSR*n*)

The source stride register, shown in Figure 14-22, contains the stride size and distance. Note that the source stride information is loaded when a new list descriptor is read from memory. Therefore, the source stride register is applicable for all link descriptors in the new list. Changing the source stride information for a link requires that a new list be generated.

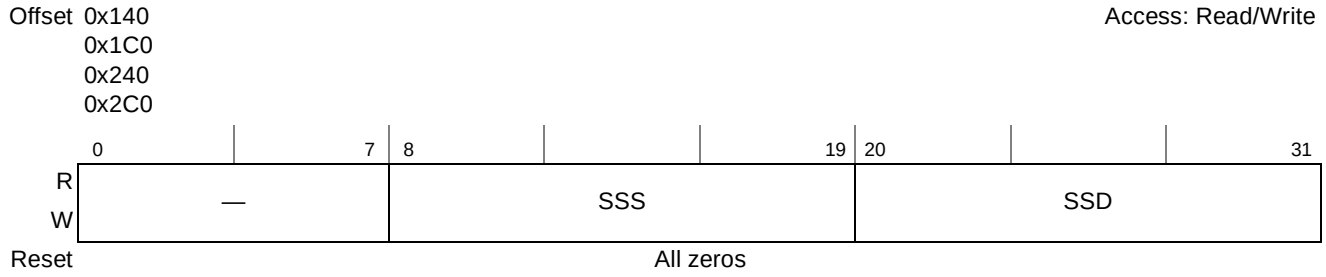


Figure 14-22. Source Stride Registers (SSR*n*)

Table 14-22 describes the fields of the SSR*n*.

Table 14-22. SSR*n* Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–19	SSS	Source stride size. Number of bytes to transfer before jumping to the next address as specified in the source stride distance field.
20–31	SSD	Source stride distance. The source stride distance in bytes from start byte to start byte.

14.3.1.13 Destination Stride Registers (DSR*n*)

The destination stride register contains the stride size, and distance. Note that the destination stride information is loaded when a new list descriptor is read from memory. Therefore, the destination stride register is applicable for all link descriptors in the new list. Changing the destination stride information for a link requires that a new list be generated. Figure 14-23 describes the DSR*n*.

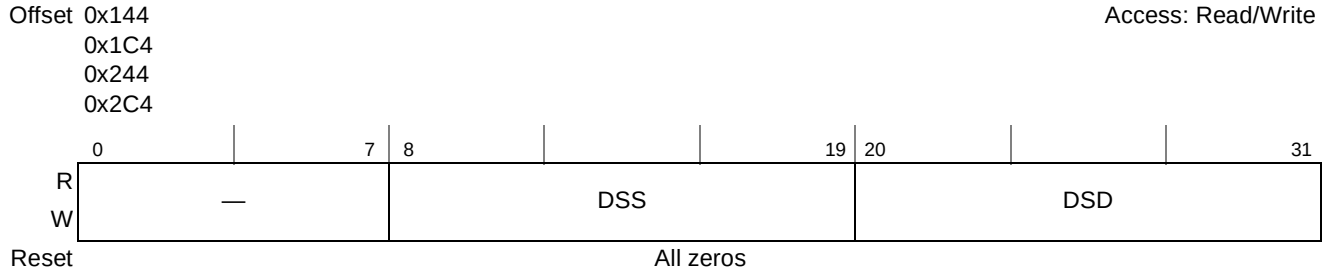


Figure 14-23. Destination Stride Registers (DSR*n*)

Table 14-23 describes the fields of the DSR*n*.

Table 14-23. DSR*n* Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–19	DSS	Destination stride size. Number of bytes to transfer before jumping to the next address as specified in the destination stride distance field.
20–31	DSD	Destination stride distance. The destination stride distance in bytes from start byte to start byte.

14.3.1.14 DMA General Status Register (DGSR)

The DMA general status register combines all of the status bits from each channel into one register. This register is read-only. Figure 14-24 describes the DGSR.

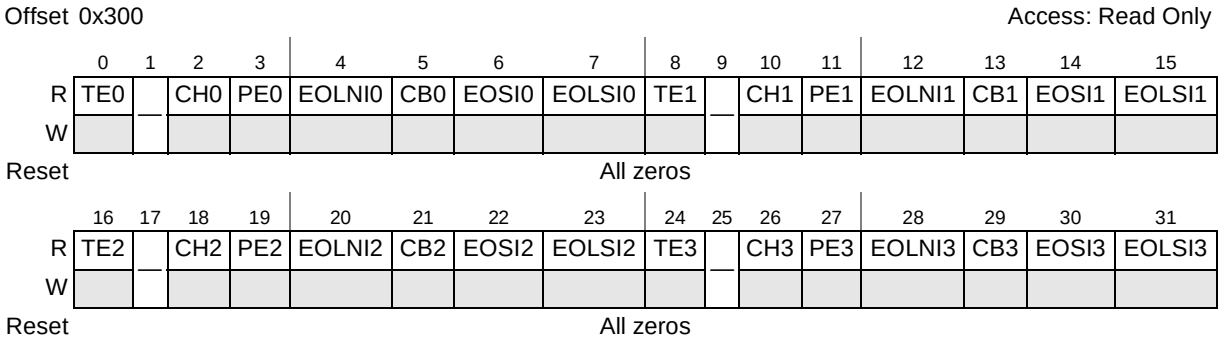


Figure 14-24. DMA General Status Register (DGSR)

Table 14-24 describes the fields of the DGSR.

Table 14-24. DGSR Field Descriptions

Bits	Name	Description
0	TE0	Transfer error, channel 0 0 Normal operation 1 An error condition occurred during the DMA transfer.
1	—	Reserved
2	CH0	Channel halted, channel 0
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

Table 14-24. DGSR Field Descriptions (continued)

Bits	Name	Description
10	CH1	Channel halted, channel 1
11	PE1	Programming error, channel 1
12	EOLNI1	End-of-links interrupt, channel 1
13	CB1	Channel busy, channel 1
14	EOSI1	End-of-segment interrupt, channel 1
15	EOLSI1	End-of-lists/direct interrupt, channel 1
16	TE2	Transfer error, channel 2 0 Normal operation 1 An error condition occurred during the DMA transfer.
17	—	Reserved
18	CH2	Channel halted, channel 2
19	PE2	Programming error, channel 2
20	EOLNI2	End-of-links interrupt, channel 2
21	CB2	Channel busy, channel 2
22	EOSI2	End-of-segment interrupt, channel 2
23	EOLSI2	End-of-lists/direct interrupt, channel 2
24	TE3	Transfer error, channel 3 0 Normal operation 1 An error condition occurred during the DMA transfer.
25	—	Reserved
26	CH3	Channel halted, channel 3
27	PE3	Programming error, channel 3
28	EOLNI3	End-of-links interrupt, channel 3
29	CB3	Channel busy, channel 3
30	EOSI3	End-of-segment interrupt, channel 3
31	EOLSI3	End-of-lists/direct interrupt, channel 3

14.4 Functional Description

This section describes the function of the DMA controller.

14.4.1 DMA Channel Operation

All DMA channels support two different modes of operation: a basic mode (MR*n*[XFE] is cleared) and an extended mode (MR*n*[XFE] is set). In both modes, a channel can be activated by clearing and setting MR*n*[CS], or through the single-write start mode using MR*n*[CDSM/SWSM] and MR*n*[SRW], or through an external control mode using MR*n*[ECS_EN].

In basic mode, the channel can be programmed in basic direct mode or basic chaining mode. In extended mode, the channel can be programmed in extended direct mode or extended chaining mode. Extended mode provides more capabilities, such as extended descriptor chaining, striding capabilities, and a more flexible descriptor structure.

The DMA controller supports misaligned transfers for both the source and destination addresses. In order to maximize performance, the source and destination engines align the source and destination addresses to a 64-byte boundary. The DMA always reads/writes the maximum number of bytes for a given transfer as described by the capability inputs of the DMA controller except for globally coherent transactions that use the size of the cache coherence granule as described by the mode select input. Using 256 bytes over the RapidIO interface reduces packet overhead which translates to increased bandwidth utilization through the interface.

The DMA controller supports bandwidth control, which prevents a channel from consuming all the data bandwidth in the controller. Each channel is allowed to consume the bandwidth of the shared resources as specified by the bandwidth control value. After the channel uses its allotted bandwidth, the arbiter grants the next channel access to the shared resources. The arbitration is round robin between the channels. This feature is also used to implement the external control pause feature. If the external control start and pause are enabled in the MR*n*, the channel enters a paused state after transferring the data described in the bandwidth control. External control can restart the channel from a paused state.

The DMA controller is designed to support RapidIO transaction types, including various priority level support. The DMA controller offers additional features from previous generations in which the reads can be mapped to globally coherent (IO_READ), non-coherent (NREAD), or maintenance reads. In addition, the writes can be mapped to coherent (flush with data) and non-coherent (NWRITE, NWRITE_R) writes, messages, and maintenance writes.

The DMA programming model permits software to program each DMA engine independently to interrupt on completed segment, chain, or error. It also provides the capability for software to resume the DMA engine from a hardware halted condition by setting the channel continue bit, MR*n*[CC]. See Table 14-25 for more complete descriptions of the channel states and state transitions.

14.4.1.1 Basic DMA Mode Transfer

This mode is primarily included for backward compatibility with existing DMA controllers which use a simple programming model. This is the default mode out of reset. The different modes of operation under the basic mode are explained in the following sections.

14.4.1.1.1 Basic Direct Mode

In basic direct mode, the DMA controller does not read descriptors from memory, but instead uses the current parameters programmed in the DMA registers to start the DMA transfer. Software is responsible for initializing SAR*n*, SATR*n*, DAR*n*, DATR*n*, and BCR*n* registers. The DMA transfer is started when MR*n*[CS] is set. Software is expected to program all the appropriate registers before setting MR*n*[CS] to a 1. The transfer is finished after all the bytes specified in the byte count register have been transferred or if an error condition occurs. The sequence of events to start and complete a transfer in basic direct mode is as follows:

1. Poll the channel state (see Table 14-25), to confirm that the specific DMA channel is idle.

2. Initialize SAR*n*, SATR*n*, DAR*n*, DATR*n* and BCR*n*.
3. Set the mode register channel transfer mode bit, MR*n*[CTM], to indicate direct mode. Other control parameters may also be initialized in the mode register.
4. Clear then set the mode register channel start bit, MR*n*[CS], to start the DMA transfer.
5. SR*n*[CB] is set by the DMA controller to indicate the DMA transfer is in progress.
6. SR*n*[CB] is automatically cleared by the DMA controller after the transfer is finished, or if the transfer is aborted (MR*n*[CA] transitions from a 0 to 1), or if a transfer error occurs.
7. End of segment interrupt is generated if MR*n*[EOSIE] is set.

14.4.1.1.2 Basic Direct Single-Write Start Mode

In basic direct single-write start mode, the DMA controller does not read descriptors from memory, but instead uses the current parameters programmed in the DMA registers to start the DMA transfer. Software is responsible for initializing the SATR*n*, DATR*n*, and BCR*n* registers. Setting MR*n*[SRW] configures the DMA controller to begin the DMA transfer either when SAR*n* is written or when DAR*n* is written, determined by the state of MR*n*[CDSM/SWSM]. Writing to SAR*n* initiates the DMA transfer if MR*n*[CDSM/SWSM] is set. Writing to DAR*n* initiates the DMA transfer if MR*n*[CDSM/SWSM] is cleared. The DMA controller automatically sets the channel start bit, MR*n*[CS]. Software is expected to program all the appropriate registers before writing the source or destination address registers. The transfer is finished after all the bytes specified in the byte count register have been transferred or if an error condition occurs. The sequence of events to start and complete a transfer in single-write start basic direct mode is as follows:

1. Poll the channel state (see Table 14-25), to confirm that the specific DMA channel is idle.
2. Initialize the source attributes (SATR*n*), DATR*n*, and BCR*n* registers.
3. Set the mode register channel transfer mode bit, MR*n*[CTM], and the single-write start direct mode bit, MR*n*[SRW]. Other control parameters may also be initialized in the mode register. Set MR*n*[CDSM/SWSM] for transfers started using SAR*n*. Clear MR*n*[CDSM/SWSM] for transfers started using the DAR*n*.
4. A write to the source or destination address register starts the DMA transfer and automatically sets MR*n*[CS].
5. SR*n*[CB] is set by the DMA controller to indicate the DMA transfer is in progress.
6. SR*n*[CB] is automatically cleared by the DMA controller after the transfer is finished, or if the transfer is aborted (MR*n*[CA] transitions from a 0 to 1), or if a transfer error occurs.
7. End of segment interrupt is generated if MR*n*[EOSIE] is set.

14.4.1.1.3 Basic Chaining Mode

In basic chaining mode, software must first build link descriptor segments in memory. Then the current link descriptor address register must be initialized to point to the first descriptor in memory. The DMA controller loads descriptors from memory prior to a DMA transfer. The DMA controller begins the transfer according to the link descriptor information loaded for the segment. After the current segment is finished, the DMA controller reads the next link descriptor from memory and begins another DMA transfer. The

transfer is finished if the current link descriptor is the last one in memory or if an error condition occurs. The sequence of events to start and complete a transfer in chaining mode is as follows:

1. Build link descriptor segments in memory.
2. Poll the channel state (see Table 14-25), to confirm that the specific DMA channel is idle.
3. Initialize CLNDAR*n* and ECLNDAR*n* to point to the first link descriptor in memory.
4. Clear the mode register channel transfer mode bit, MR*n*[CTM], as well as MR*n*[XFE], to indicate basic chaining mode. Other control parameters may also be initialized in the mode register.
5. Clear, then set the mode register channel start bit, MR*n*[CS], to start the DMA transfer.
6. SR*n*[CB] is set by the DMA controller to indicate the DMA transfer is in progress.
7. SR*n*[CB] is automatically cleared by the DMA controller after finishing the transfer of the last descriptor segment, or if the transfer is aborted (MR*n*[CA] transitions from a 0 to 1), or if an error occurs during any of the transfers.

14.4.1.1.4 Basic Chaining Single-Write Start Mode

Basic chaining single-write start mode allows a chain to be started by writing the current link descriptor address register (CLNDAR*n*). (Note that ECLNDAR*n* must be written *first* so that the full 36-bit descriptor address is present when the chain starts.) Setting MR*n*[CDSM/SWSM] in the mode register causes MR*n*[CS] to be automatically set when the current link descriptor address register is written. The sequence of events to start and complete a chain using single-write start mode is as follows:

1. Set the mode register current descriptor start mode bit, MR*n*[CDSM/SWSM], and the extended features enable bit MR*n*[XFE]. Also, clear the channel transfer mode bit, MR*n*[CTM]. This initialization indicates basic chaining and single-write start mode. Also other control parameters may be initialized in the mode register.
2. Build link descriptor segments in memory.
3. Poll the channel state (see Table 14-25), to confirm that the specific DMA channel is idle.
4. Initialize CLNDAR*n* and ECLNDAR*n* to point to the first descriptor segment in memory. This write automatically causes the DMA controller to begin the link descriptor fetch and set MR*n*[CS].
5. SR*n*[CB] is set by the DMA controller to indicate the DMA transfer is in progress.
6. SR*n*[CB] is automatically cleared by the DMA controller after finishing the transfer of the last descriptor segment, or if the transfer is aborted (MR*n*[CA] transitions from a 0 to 1), or if an error occurs during any of the transfers.

14.4.1.2 Extended DMA Mode Transfer

The extended DMA mode also operates in chaining and direct mode. It offers additional capability over the basic mode by supporting striding and a more flexible descriptor structure. This additional functionality also requires a new and more complex programming model. The extended DMA mode is activated by setting MR*n*[XFE].

14.4.1.2.1 Extended Direct Mode

Extended direct mode has the same functionality as basic direct mode with the addition of stride capabilities. The bit settings are the same as in direct mode with the exception of the MR*n*[XFE] being set. Striding on the source address can be accomplished by setting SATR*n*[SSME] and setting the desired stride size and distance in SSR*n*. Striding on the destination address can be accomplished by setting DATR*n*[DSME] and setting the desired stride size and distance in DSR*n*.

14.4.1.2.2 Extended Direct Single-Write Start Mode

Extended direct single-write start mode has the same functionality as the basic direct single-write start mode with the addition of stride capabilities. The bit settings are also the same with the exception of MR*n*[XFE] being set. Striding on the source address can be accomplished by setting SATR*n*[SSME] and setting the desired stride size and distance in SSR*n*. Striding on the destination address can be accomplished by setting DATR*n*[DSME] and setting the desired stride size and distance in DSR*n*.

14.4.1.2.3 Extended Chaining Mode

In extended chaining mode, the software must first build list and link descriptor segments in memory. Then CLSDAR*n* and ECLSDAR*n* must be initialized to point to the first list descriptor in memory. The DMA controller loads list descriptors and link descriptors from memory prior to a DMA transfer. The DMA controller begins the transfer according to the link descriptor information loaded. Once the current link descriptor is finished, the DMA controller reads the next link descriptor from memory and begins another DMA transfer. If the current link descriptor is the last in the list, the DMA controller reads the next list descriptor in memory. The transfer is finished if the current link descriptor is the last one in the last list in memory or if an error condition occurs. The sequence of events to start and complete a transfer in extended chaining mode is as follows:

1. Build link and list descriptor segments in memory.
2. Poll the channel state (see Table 14-25), to confirm that the specific DMA channel is idle.
3. Initialize CLSDAR*n* and ECLSDAR*n* to point to the first list descriptor in memory.
4. Clear the mode register channel transfer mode bit, MR*n*[CTM], to indicate chaining mode. MR*n*[XFE] must be set to indicate extended DMA mode. Other control parameters may also be initialized in the mode register.
5. Clear, then set the mode register channel start bit, MR*n*[CS], to start the DMA transfer.
6. SR*n*[CB] is set by the DMA controller to indicate the DMA transfer is in progress.
7. SR*n*[CB] is automatically cleared by the DMA controller after finishing the transfer of the last descriptor segment, or if the transfer is aborted (MR*n*[CA] transitions from a 0 to 1), or if an error occurs during any of the transfers.

14.4.1.2.4 Extended Chaining Single-Write Start Mode

In the extended mode, the single-write start feature allows a chain to be started by writing the current list descriptor pointer. Setting MR*n*[CDSM/SWSM] causes MR*n*[CS] to be set automatically when CLSDAR*n* is written. (Note that ECLSDAR*n* must be written *first* so that the full 36-bit descriptor address

is present when the chain starts.) The sequence of events to start and complete an extended chain using single-write start mode is as follows:

1. Set MR*n*[CDSM/SWSM], MR*n*[CTM], and MR*n*[XFE] to indicate extended chaining and single-write start mode. Also other control parameters may be initialized in the mode register.
2. Build list and link descriptor segments in local memory.
3. Poll the channel state (see Table 14-25), to confirm that the specific DMA channel is idle.
4. Initialize the current list descriptor address register to point to the first list descriptor segment in memory. This write automatically causes the DMA controller to begin the list descriptor fetch and set MR*n*[CS].
5. SR*n*[CB] is set by the DMA controller to indicate the DMA transfer is in progress.
6. SR*n*[CB] is automatically cleared by the DMA controller after finishing the transfer of the last descriptor segment, or if the transfer is aborted (MR*n*[CA] transitions from a 0 to 1), or if an error occurs during any of the transfers.

14.4.1.3 External Control Mode Transfer

An external control can be used to control all DMA channels by setting MR*n*[EMS_EN]. The external control can control the DMA channel in the following transfer modes:

- Basic direct
- Basic chaining
- Extended direct
- Extended chaining

Note that when operating the DMA in chaining mode the register byte count field, BCR[BC], must be initialized to zero before enabling the pause feature. In chaining modes, the channel does not pause for descriptor fetch transfer.

The external control and the DMA controller use a well defined protocol to communicate. The external control can start or restart a paused DMA transfer. The DMA controller acknowledges a DMA transfer in progress and also indicates a transfer completion. Note that external control cannot cause a channel to enter a paused state.

The pause feature can be enabled by setting MR*n*[EMP_EN]. MR*n*[BWC] specifies how much data to allow a specific channel to transfer before entering a paused state by clearing MR*n*[CS]. Note, however, that write data for a paused transfer may not have reached the target interface when so indicated. The channel can be restarted from a paused state by the asserted edge of $\overline{\text{DREQ}}$ as driven by an external master. In chaining modes, the channel does not pause for descriptor fetch transfer. It only pauses during the actual data transfer.

The following signals are defined for the external control interface:

- $\overline{\text{DMA\_DREQ}}$ - Asserting edge triggers a DMA transfer start or restart from a pause request. Sets MR*n*[CS]. (Note that negating $\overline{\text{DMA\_DREQ}}$ does NOT clear MR*n*[CS].)
- $\overline{\text{DMA\_DACK}}$ - Indicates a DMA transfer currently in progress. SR*n*[CB] is set.

- $\overline{\text{DMA\_DDONE}}$ - Indicates the completion of the DMA controller's involvement in the transfer and the readiness to accept a new DMA command. SR*n*[CB] is clear. Note, however, that write data may still be queued at the target interface or in the process of transfer on an external interface.

Detailed descriptions of the external control interface are in Table 14-3. The timing diagram of the external control interface is shown in Figure 14-25.

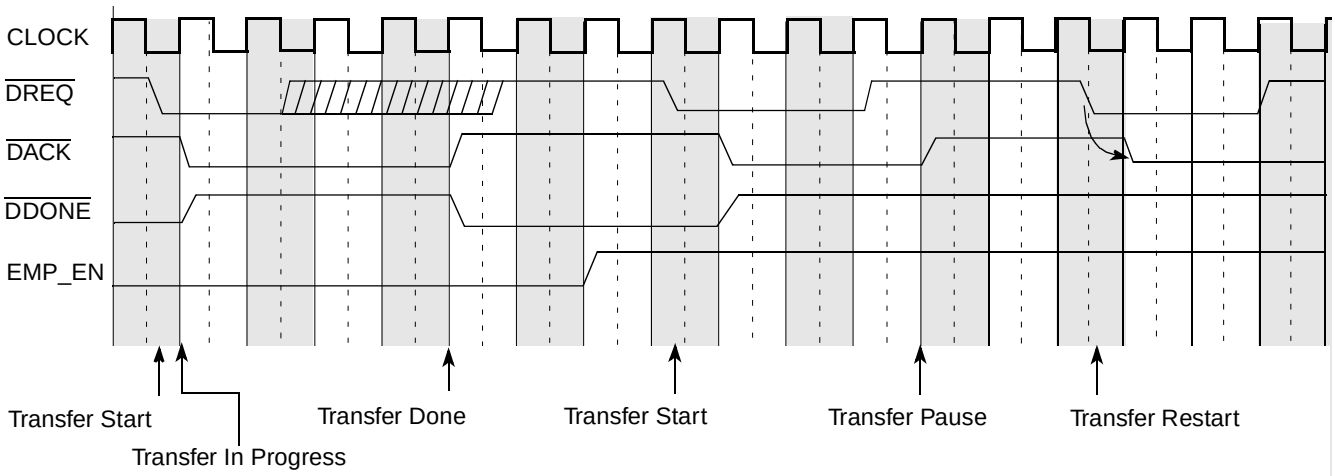


Figure 14-25. External Control Interface Timing

14.4.1.4 Channel Continue Mode for Cascading Transfer Chains

The channel continue mode (enabled when MR*n*[CC] is set) offers software the flexibility of having the DMA controller get started on descriptors that have already been programmed while software continues to build more descriptors in memory. Software can set the end-of-links descriptor (EOLND) in basic mode, or end-of-lists descriptor (EOLSD) in extended mode, to cause the channel to go into a halted state while software continues to build other descriptors in memory. Software can then set CC to force hardware to continue where it left off. channel continue is only meaningful for chaining modes, not direct mode.

If CC is set by software while the channel is busy with a transfer, the DMA controller finishes all transfers until it reaches the EOLND in basic mode or EOLSD in extended mode. The DMA controller then refetches the last link descriptor in basic mode, or the last list descriptor in extended mode and clears the channel continue bit. If EOLND or EOLSD is still set for their respective modes, the DMA controller remains in the idle state. If EOLND or EOLSD is not set, the DMA controller continues the transfer by refetching the new descriptor. The channel busy (SR*n*[CB]) bit is cleared when the DMA controller reaches EOLND/EOLSD and is set again when it initiates the refetch of the link or list descriptor.

If CC is set by software while the channel is not busy with a transfer, the DMA controller refetches the last link descriptor in basic mode, or the last list descriptor in extended mode and clears the channel continue bit. If EOLND or EOLSD is still set for their respective modes, the DMA controller remains in the idle state. If the EOLND or EOLSD bits are not set, the DMA controller continues the transfer by refetching the new descriptor.

14.4.1.4.1 Basic Mode

On a channel continue, the descriptor at the current link descriptor address registers (CLNDAR*n* and ECLNDAR*n*) is refetched to get the next link descriptor address field as updated by software. The channel halts if NLNDAR*n*[EOLND] is still set. If EOLND is zero, the next link descriptor address is copied into CLNDAR*n* and ECLNDAR*n* and the channel continues with another descriptor fetch of the current link descriptor address. As a result, two link descriptor fetches always exist after channel continue before starting the first transfer.

14.4.1.4.2 Extended Mode

On a channel continue, the descriptor at the current list descriptor (CLSDAR*n* and ECLSDAR*n*) address register is refetched to get the next list descriptor address field as updated by software. The channel halts if NLSDAR*n*[EOLSD] is still set. If not, the next list descriptor address is copied into the CLSDAR*n* and ECLSDAR*n* registers and the channel continues with another descriptor fetch of the current list descriptor address. As a result, two list descriptor fetches always exist after channel continue before the first link descriptor fetch and the first transfer.

14.4.1.5 Channel Abort

Software can abort a previously initiated transfer by setting MR*n*[CA]. Once the DMA channel controller detects a zero-to-one transition of MR*n*[CA], it finishes the current sub-block transfer and halts all further activity. The controller then waits for all previously initiated transfers from the specified channel to drain and clears SR*n*[CB]. Successful completion of a software initiated abort request can be recognized by MR*n*[CA] being set and SR*n*[CB] being cleared. Obviously, if the controller was already halted because of an error condition (SR*n*[TE] is set), or the channel has completed all transfers, then SR*n*[CB] being cleared may not signify that the controller entered a halt state due to the abort request.

14.4.1.6 Bandwidth Control

MR*n*[BWC] specifies how much data to allow a specific channel to transfer before allowing the next channel to use the shared data transfer hardware. This promotes equitable bandwidth allocation between channels. However, if only one channel is busy, hardware overrides the specified bandwidth control size value. The DMA controller allows a channel to transfer up to 1 Kbyte at a time when no other channel is active.

14.4.1.7 Channel State

Table 14-25 defines the state of a channel based on the values of the channel start (MR*n*[CS]), channel busy (SR*n*[CB]), transfer error (SR*n*[TE]), and channel continue (MR*n*[CC]) bits.

Table 14-25. Channel State Table

MR*n*[CS]	SR*n*[CB]	SR*n*[TE]	MR*n*[CC]	Channel State
0	0	0	0	Idle state. This is the state of the bits out of reset.
0	0	0	1	Channel continue unexpected. Channel remains idle
0	0	1	0	Error occurred after software halted the channel.

Table 14-25. Channel State Table (continued)

MR*n*[CS]	SR*n*[CB]	SR*n*[TE]	MR*n*[CC]	Channel State
0	0	1	1	Channel Continue unexpected. Channel remains in error halt state
0	1	0	0	Software halted channel. The channel was busy and software cleared MR*n*[CS].
0	1	0	1	Channel remains in halt state.
—	1	1	—	The channel has encountered an error condition and it is trying to halt.
1	0	0	0	Ready to start a transfer, or transfer completed
1	0	0	1	Continue transfer (only meaningful in chaining mode, not direct mode). In direct mode, the channel continue has no effect.
1	0	1	0	Error occurred during transfer
1	0	1	1	Channel remains in error halt state
1	1	0	0	Transfer in progress
1	1	0	1	Continue after reaching the end of list/link, or the first descriptor fetch after channel continue

14.4.1.8 Illustration of Stride Size and Stride Distance

If operating in stride mode, the stride size defines the amount of data to transfer before jumping to the next quantity of data as specified by the stride distance. The stride distance is added to the current base address to point to the next quantity of data to be transferred. Figure 14-26 illustrates the stride size and distance parameters. As shown, each time the stride distance is added to the base address, the resulting address becomes the new base address. This sequence repeats until the amount of data transferred equals the transfer size.

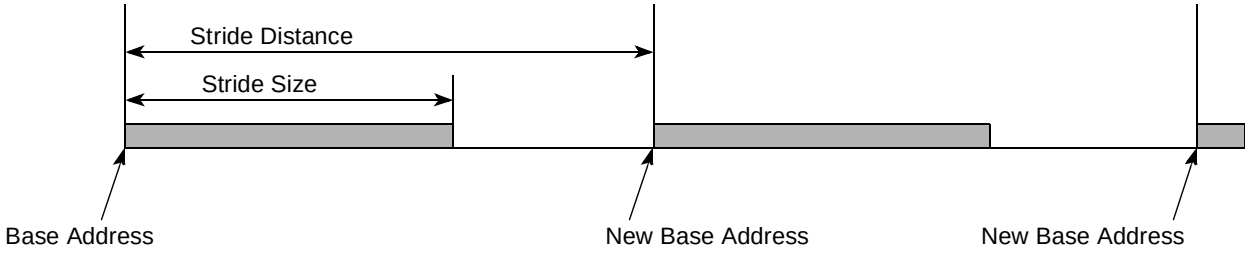


Figure 14-26. Stride Size and Stride Distance

14.4.2 DMA Transfer Interfaces

The DMA can be used to achieve data transfers across the entire memory map. Note that a single DMA transfer in any of the direct or chaining modes must not cross a 16GB (34-bit) address boundary.

14.4.3 DMA Errors

On a transfer error (uncorrectable ECC errors on memory accesses, parity errors on local bus or PCI, address mapping errors, for example), the DMA halts by setting SR*n*[TE] and generates an interrupt if

MR*n*[EIE] is set. On a programming error, the DMA sets SR*n*[PE] and generates an interrupt if MR*n*[EIE] is set. The DMA controller detects the following programming errors:

- Transfer started with a byte count of zero
- Stride transfer started with a stride size of zero
- Transfer started with a priority of three
- Illegal type, defined by SATR*n*[SREADTTYPE] and DATR*n*[DWRITETTYPE], used for the transfer.
- Invalid interface—Defined by SATR*n*[STRANSINT] and DATR*n*[DTRANSINT]. Used for the transfer when in ATMU bypass mode.

14.4.4 DMA Descriptors

The DMA engine recognizes list descriptors and link descriptors. List descriptors connect lists of link descriptors. Link descriptors describe the DMA activity that is to take place. DMA descriptors are built in either local or remote memory and are connected by the next descriptor fields. Only link descriptors contain information for the DMA controller to transfer data. Software must ensure that each descriptor is 32-byte aligned. The last link descriptor in the last list in memory sets the NLNDAR*n*[EOLND] bit in the next link descriptor and NLSDAR*n*[EOLSD] in the next list descriptor fields indicating that these are the last descriptors in memory. Software initializes the current list descriptor address register to point to the first list descriptor in memory. The DMA controller traverses through the descriptor lists until the last link descriptor is met. For each link descriptor in the chain, the DMA controller starts a new DMA transfer with the control parameters specified by that descriptor. Link and list descriptor fetches always snoop the local memory space.

NOTE

Software must ensure that each descriptor is aligned on a 32-byte boundary.

Table 14-26 summarizes the DMA list descriptors.

Table 14-26. List DMA Descriptor Summary

Descriptor Field	Description
Next list descriptor extended address	Points to the next list descriptor in memory. After the DMA controller reads the descriptor from memory, this field is loaded into the next list descriptor extended address registers.
Next list descriptor address	Points to the next list descriptor in memory. After the DMA controller reads the descriptor from memory, this field is loaded into the next list descriptor address registers.
First link descriptor extended address	Points to the first link descriptor in memory for this list. After the DMA controller reads the descriptor from memory, this field is loaded into the current link descriptor extended address registers.
First link descriptor address	Points to the first link descriptor in memory for this list. After the DMA controller reads the descriptor from memory, this field is loaded into the current link descriptor address registers.
Source stride	Contains the stride information used for the data source if striding is enabled for a link in the list
Destination stride	Contains the stride information used for the data destination if striding is enabled for a link in the list

Table 14-27 summarizes the DMA link descriptors.

Table 14-27. Link DMA Descriptor Summary

Descriptor Field	Description
Source attributes register	Contains source transaction attributes
Source address	Contains the source address of the DMA transfer. After the DMA controller reads the descriptor from memory, this field is loaded into the Source address register.
Destination attributes register	Contains destination transaction attributes
Destination address	Contains the destination address of the DMA transfer. After the DMA controller reads the descriptor from memory, this field is loaded into the destination address register.
Next link descriptor extended address	Points to the next link descriptor in memory. After the DMA controller reads the link descriptor from memory, this field is loaded into the extended next link descriptor address registers
Next link descriptor address	Points to the next link descriptor in memory. After the DMA controller reads the link descriptor from memory, this field is loaded into the next link descriptor address registers.
Byte count	Contains the number of bytes to transfer. After the DMA controller reads the descriptor from memory, this field is loaded into the byte count register.

Figure 14-27 describes the DMA transaction flow.

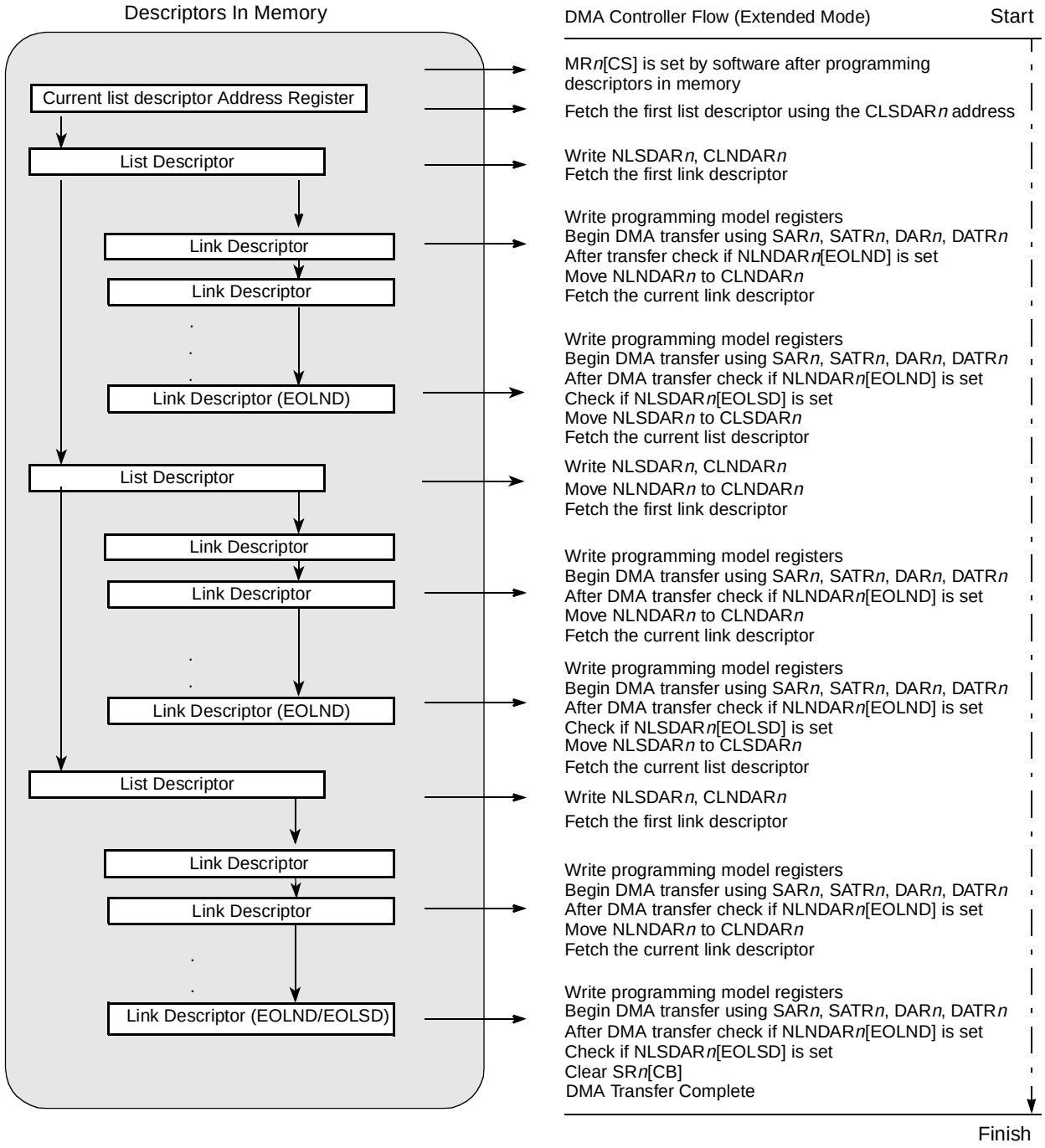


Figure 14-27. DMA Transaction Flow with DMA Descriptors

Figure 14-28 describes the format of the list descriptors.

Offset	
0x00	Next List Descriptor Extended Address
0x04	Next List Descriptor Address
0x08	First Link Descriptor Extended Address
0x0C	First Link Descriptor Address
0x10	Source Stride
0x14	Destination Stride
0x18	Reserved
0x1C	Reserved

Figure 14-28. List Descriptor Format

Figure 14-29 describes the format of the link descriptors.

Offset	
0x00	Source Attributes
0x04	Source Address
0x08	Destination Attributes
0x0C	Destination Address
0x10	Next Link Descriptor Extended Address
0x14	Next Link Descriptor Address
0x18	Byte Count
0x1C	Reserved

Figure 14-29. Link Descriptor Format

14.4.5 Limitations and Restrictions

This section addresses some of the limitations and restrictions of the DMA controller and is intended to help software maximize the DMA performance and avoid DMA programming errors.

The limitations of the DMA controller are the following:

- Due to the limited number of buffers that the DMA controller can use, stride sizes less than 64 bytes should be avoided. Maximum utilization is obtained from strides greater than or equal to 256 bytes. However, small stride sizes can be used for scatter-gather functions.
- Coherent reads or writes are broken up into cache line accesses in the DMA.

The DMA controller restrictions are as follows:

- Setting the source or destination priority level (STFLOWLVL or DTFLOWLVL) to a value of three (0b11) is considered a programming error.
- All interface capabilities from where descriptors are being fetched must support read sizes of 32 bytes or greater.

- If MR*n*[SAHE] is set, the source interface transfer size capability must be greater than or equal to MR*n*[SAHTS]. The source address must be aligned to a size specified by SAHTS.
- If MR*n*[DAHE] is set, the destination interface transfer size capability must be greater than or equal to MR*n*[DAHTS]. The destination address must be aligned to the size specified by DAHTS.
- Destination striding is not supported if MR*n*[DAHE] is set and source striding is not supported if MR*n*[SAHE] is set.
- If the DMA is programmed to send SWRITEs over RapidIO, the programmer must ensure that the destination address is double-word aligned and that the byte count is a double-word multiple.
- If the DMA is programmed to send messages over RapidIO, the programmer must ensure that the message length (BCR*n*[BC]) is 8, 16, 32, 64, 128 or 256 bytes. This can be achieved by setting the byte count register (BCR) to a power of 2 value equal to 8 or greater.
- Striding does not work if the destination transaction type is MESSAGE (DATR[DWRITETTYPE] = 0x0110) because messages have no memory addresses. As well, destination address hold should be disabled (MR*n*[DAHE] is cleared) unless the destination address hold transfer size indicates an 8-byte message (MR*n*[DAHTS] = 0x11). Software is responsible for disabling striding and DAHE, in this case, and for ensuring that the bandwidth control is large enough to support the desired message size. Failure to adhere to these restrictions results in boundedly undefined behavior.
- When DMA is used to issue maintenance reads and writes in bypass mode, the sum of the starting offset field in DAR and the byte count must not cause the offset to rollover. Failure to adhere to this restriction results in boundedly undefined behavior.

14.5 DMA System Considerations

This section provides information about how to make most effective use of the DMA channels.

Figure 14-30 shows the most important data paths within the device. Many of these paths are served by captive DMA controllers and virtually all can be served by the general purpose four-channel DMA controller.

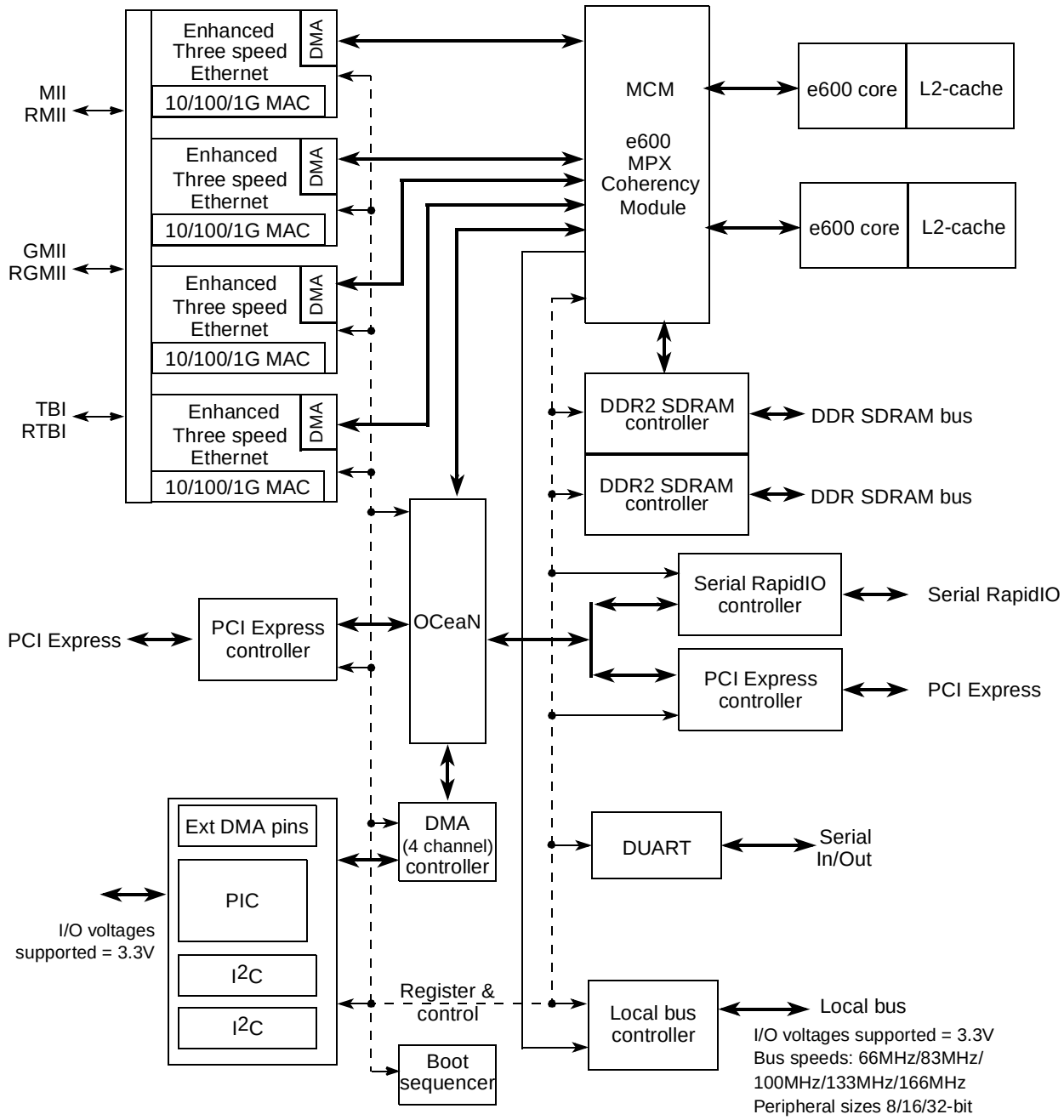


Figure 14-30. DMA Data Paths

Note: On-chip target configuration registers include I^2C data register.

Note: On-chip Ethernet is captive resource. Not available to external masters.

Note: On-chip 4-channel controller can serve external masters.

14.5.1 Unusual DMA Scenarios

The following is a description of unusual DMA paths including explanations of why some functional blocks cannot serve as DMA targets. The following topics are addressed:

- DMA transaction initiators (masters)
- DMA targets, that is, data sources or destinations
- Transparency of the bus controllers to DMA transactions
- What is useful as opposed to what is possible. For example, any register can be addressed through an internal control bus, which means configuration and control registers can be DMA targets.

14.5.1.1 DMA to e600 Core

The L1 cache cannot be a direct DMA target because it cannot be directly addressed by software. However, DMA access into the L1 cache occurs indirectly if a block of memory that is cached in the L1 is specified as the DMA target. This effect is deterministic if the target memory block was locked into the L1 with cache locking instructions.

14.5.1.2 DMA to 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 using a general-purpose DMA controller to access the transmit and receive buffers defined by the Ethernet buffer descriptors. Because Ethernet data buffers are located in RAM outside of the Ethernet controllers, general-purpose DMA engines can move data to or from these memory regions. Also, because Ethernet controllers automatically read buffer descriptors and send (or load) data buffers, a DMA transfer into (or out of) these buffers is effectively a transfer into (or out of) the Ethernet ports.

14.5.1.3 DMA to Configuration, Control, and Status Registers

Because any internal register can be addressed with the four-channel DMA controller, configuration, control, and status registers throughout the device are valid DMA targets. However, the primary purpose of DMA—to reduce processor load by moving large blocks of data— is not served by DMA transfers of configuration data. For example, while it is possible to DMA into the I^2C controller or programmable interrupt controller (PIC), doing so is extremely inefficient and is seldom beneficial in normal operation. The overhead of creating DMA descriptors far exceeds any savings in CPU cycles.

14.5.1.4 DMA to I²C

The I²C controller is not transparent to DMA transfers. Observe the caveats listed in Section 14.5.1.3, "DMA to Configuration, Control, and Status Registers," when accessing any I²C register, including the data register (I2CDR).

14.5.1.5 DMA to DUART

The DUART provides complete and sophisticated DMA support which is described in Chapter 11, "DUART," specifically, Section 11.4.5, "FIFO Mode."

Chapter 15
Serial RapidIO Interface

The serial RapidIO controller consists of a RapidIO endpoint and the RapidIO messaging unit (RMU). Both portions are compliant with the *RapidIO Interconnect Specification, Revision 1.2*.

15.1 Overview

The serial RapidIO interface provides a RapidIO port and message unit to communicate with other RapidIO devices. Figure 15-1 shows the RapidIO port and message unit as they interface with OCeaN and with each other.

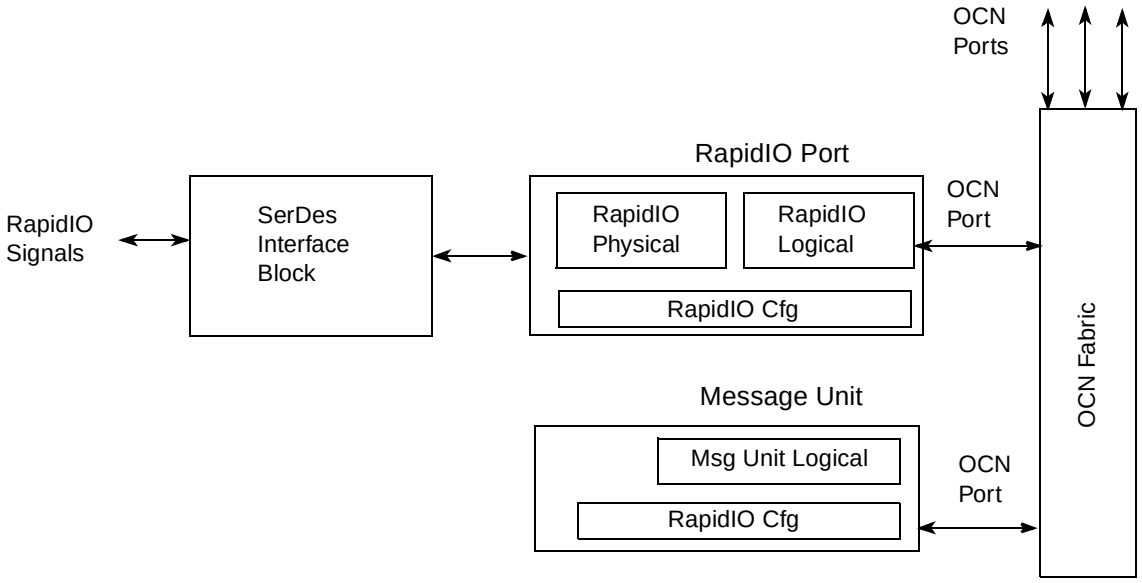


Figure 15-1. RapidIO Endpoint and RMU

15.2 Features

The RapidIO port supports the following features of the *RapidIO Interconnect Specification, Revision 1.2*:

- Small or large size transport information field
- 34-bit addressing
- Up to 256-byte data payload
- Up to eight outstanding unacknowledged RapidIO transactions
- Hardware recovery only
- All transaction flows and all priorities

- Register and register bit extensions as described in the *RapidIO Interconnect Specification, Revision 1.2, Part VIII: Error Management Extensions Specification*.
- Hot swap
- ATOMIC set/clr/inc/dec for read-modify-write operations
- IO_READ_HOME and FLUSH w/data for accessing cache-coherent data from a remote memory system
- Only supports receiver-controlled flow control
- Inbound transactions to the configuration registers are limited to 32-bit accesses only.
- Outbound maintenance transactions can be any valid size.

The RMU supports the following features of the *RapidIO Interconnect Specification, Revision 1.2*:

- Two outbound message controllers
- Two inbound message controllers
- One outbound doorbell controller
- One inbound doorbell controllers
- One inbound port-write controller

RapidIO endpoint supports the following user-defined features:

- Nine outbound ATMU windows with each window having up to 32 subwindows except the default window
- Five inbound ATMU windows
- Logical outbound packet time-to-live counter to prevent local processor from hanging when the RIO interface fails
- Accept-all mode of operation for failover support
- RapidIO random bit error injection
- Performance monitor interface

The RMU supports the following user-defined features:

- Performance monitor interface

RapidIO endpoint does not support or has limited support of the following features of the *RapidIO Interconnect Specification, Revision 1.2*:

- No support for 50- and 66-bit addressing
- No support for software assisted error recovery
- No support for ATOMIC test-and-swap transaction
- No support for coherent (CC-NUMA) transactions with the exception of IO_READ_HOME and FLUSH w/data transactions
- No support for transmitter-controlled flow control
- No decrementing of a maintenance packet hop count (pass-through support does not imply switch functionality)
- No support for multicast event control symbols

RapidIO endpoint supports the following features of RapidIO 1x/4x LP-Serial:

- Both 1x and 4x LP-Serial link interfaces
- Transmission rates of 1.25, 2.5, and 3.125 Gbaud (data rates of 1.0, 2.0, and 2.5 Gbps) per lane
- Auto detection of 1x and 4x mode operation during port initialization
- Error detection for packets and control symbols
- Support for link initialization, synchronization, error recovery, and time-out

RapidIO endpoint does not support the following features of RapidIO 1x/4x LP-Serial:

- RapidIO endpoint cannot be configured as four 1x ports

15.3 Modes of Operation

15.3.1 RapidIO Port

The RapidIO port's primary operating modes are the following:

- 1x or 4x LP-Serial link interfaces
- Transmission rates of 1.25, 2.5, or 3.125 Gbaud (data rates of 1.0, 2.0, and 2.5 Gbps) per lane
- Small or large size transport information field
- Accept-all mode of operation—all packets are accepted regardless of the target ID

15.3.2 Message Unit

The message unit's primary operating modes are the following:

- Outbound message controller
  - — Direct mode. No descriptors are involved. Software must initialize the required fields before starting a transfer.
  - — Chaining mode. Software must initialize descriptors in memory and the required fields before starting a transfer
  - — Multicast mode. Single segment messages can be transferred to up to 32 destinations.

15.4 1x/4x LP-Serial Signal Descriptions

The high-speed interface signals used for the serial RapidIO interface are shared with other I/O ports and must be configured at power-on reset for use with the serial RapidIO controller. See Section 4.4.3.8, "SerDes Port Selection," for specific configuration details.

The following sections describe the serial RapidIO signal functionality. Refer to the *RapidIO Interconnect Specification, Revision 1.2, Part VI: Physical Layer 1x/4x LP-Serial Specifications, Chapter 8, Electrical Specifications,* for electrical characteristic details.

15.4.1 Serial Rapid I/O Interface Overview

The serial Rapid I/O (SRIO) interface is compatible with the *RapidIO Interconnect Specification, Revision 1.2, Part VI: Physical Layer 1x/4x LP-Serial Specifications*.

15.4.2 Serial Rapid I/O Interface Detailed Signal Descriptions

15.4.2.1 SD2_TX[4:7]/$\overline{\text{SD2\_TX}}$[4:7]—Outputs

These are the serial data outputs. They are differential pairs, one for each lane in 4x mode of operation. In the case of 1x mode of operation on a 1x/4x-compatible serial RapidIO interface, only the first and third lanes may be used.

Note that for this controller, serial RapidIO lane 0 corresponds to SD2 lane 4; serial RapidIO lane 1 corresponds to SD2 lane 5, serial RapidIO lane 2 corresponds to SD2 lane 6, and serial RapidIO lane 3 corresponds to SD2 lane 7. See *Part VI: Physical Layer 1x/4x LP-Serial Specification, RapidIO Interconnect Specification, Revision 1.2,* for complete details.

These outputs are asynchronous, as described in Part VI of the *RapidIO Interconnect Specification, Revision 1.2*. This implementation supports data rates of 1.25, 2.5, and 3.125 Gbaud.

15.4.2.2 SD2_RX[4:7]/$\overline{\text{SD2\_RX}}$[4:7]—Inputs

These are the serial data input pads. They are differential pairs, one for each lane in 4x mode of operation. In the case of 1x mode of operation on a 1x/4x-compatible serial RapidIO interface, only the first lane is used.

Note that for this controller, serial RapidIO lane 0 corresponds to SD2 lane 4; serial RapidIO lane 1 corresponds to SD2 lane 5, serial RapidIO lane 2 corresponds to SD2 lane 6, and serial RapidIO lane 3 corresponds to SD2 lane 7. See *Part VI: Physical Layer 1x/4x LP-Serial Specification, RapidIO Interconnect Specification, Revision 1.2.* for complete details.

These inputs are asynchronous, as described in Part VI of the *RapidIO Interconnect Specification, Revision 1.2*. This implementation supports data rates of 1.25, 2.5, and 3.125 Gbaud.

15.5 Memory Map/Register Definition

Table 15-1 is the memory map of the RapidIO configuration registers.

Table 15-1. RapidIO Memory Map

Offset	Register	Access	Reset[1]	Section/Page
Architectural				
0xC_0000	Device identity capability register (DIDCAR)	R	0x701*n*_0002 = MPC8641	15.6.1.1/15-10
0xC_0004	Device information capability register (DICAR)	R	0x*nnnn_nnnn*	15.6.1.2/15-11
0xC_0008	Assembly identity capability register (AIDCAR)	R/W	0x*nnnn_nnnn*	15.6.1.3/15-12

Table 15-1. RapidIO Memory Map (continued)

Offset	Register	Access	Reset[1]	Section/Page
0xC_000C	Assembly information capability register (AICAR)	R/W	0x0000_0000	15.6.1.4/15-12
0xC_0010	Processing element features capability register (PEFCAR)	R	0xE0F8_00*n*9	15.6.1.5/15-13
0xC_0018	Source operations capability register (SOCAR)	R	0x0600_FCF0	15.6.1.6/15-14
0xC_001C	Destination operations capability register (DOCAR)	R	0x0000_FCF4	15.6.1.7/15-15
0xC_0040	Mailbox command and status register (MCSR)	R	0x2020_0000	15.6.1.8/15-16
0xC_0044	Port -Write and doorbell command and status register (PWDCSR)	R	0x2000_0020	15.6.1.9/15-17
0xC_004C	Processing element logical layer control command and status register (PELLCCSR)	R	0x0000_0001	15.6.1.10/15-19
0xC_005C	Local configuration space base address 1 command and status register (LCSBA1CSR)	R/W	0x0000_0000	15.6.1.11/15-19
0xC_0060	Base device ID command and status register (BDIDCSR)	R/W	0x00*nn_nnnn*	15.6.1.12/15-20
0xC_0068	Host base device ID lock command and status register (HBDIDLCSR)	Special	0x0000_FFFF	15.6.1.13/15-21
0xC_006C	Component tag command and status register (CTCSR)	R/W	0x0000_0000	15.6.1.14/15-21
Extended Features Space				
1x/4x LP-Serial				
0xC_0100	Port maintenance block header 0 (PMBH0)	R	0x0600_0001	15.6.2.1/15-22
0xC_0120	Port link time-out control command and status register (PLTOCCSR)	R/W	0xFFFF_FF00	15.6.2.2/15-22
0xC_0124	Port response time-out control command and status register (PRTOCCSR)	R/W	0xFFFF_FF00	15.6.2.3/15-23
0xC_013C	General control command and status register (GCCSR)	R/W	0x*n*000_0000	15.6.2.4/15-23
0xC_0140	Link maintenance request command and status register (LMREQCSR)	R/W	0x0000_0000	15.6.2.5/15-24
0xC_0144	Link maintenance response command and status register (LMRESPCSR)	R	0x0000_0000	15.6.2.6/15-25
0xC_0148	Local ackID status command and status register (LASCSR)	Mixed	0x0000_0000	15.6.2.7/15-25
0xC_0158	Error and status command and status register (ESCSR)	Mixed	0x0000_0001	15.6.2.8/15-26
0xC_015C	Control command and status register (CCSR)	R/W	0x5060_0001	15.6.2.9/15-28
Error Reporting, Logical				
0xC_0600	Error reporting block header (ERBH)	R	0x0000_0007	15.6.3.1/15-30
0xC_0608	Logical/Transport layer error detect command and status register (LTLEDCSR)	R/W	0x0000_0000	15.6.3.2/15-30
0xC_060C	Logical/Transport layer error enable command and status register (LTLEECSR)	R/W	0x0000_0000	15.6.3.3/15-32
0xC_0614	Logical/Transport layer address capture command and status register (LTLACCSR)	R/W	0x0000_0000	15.6.3.4/15-33
0xC_0618	Logical/Transport layer device ID capture command and status register (LTLDIDCCSR)	R/W	0x0000_0000	15.6.3.5/15-34

Table 15-1. RapidIO Memory Map (continued)

Offset	Register	Access	Reset[1]	Section/Page
0xC_061C	Logical/Transport layer control capture command and status register (LTLCCCSR)	R/W	0x0000_0000	15.6.3.6/15-35
Error Reporting, Physical				
0xC_0640	Error detect command and status register (EDCSR)	R/W	0x0000_0000	15.6.4.1/15-36
0xC_0644	Error rate enable command and status register (ERECSR)	R/W	0x0000_0000	15.6.4.2/15-36
0xC_0648	Error capture attributes command and status register (ECACSR)	R/W	0x0000_0000	15.6.4.3/15-37
0xC_064C	Packet/control symbol error capture command and status register 0 (PCSECCSR0)	R/W	0x0000_0000	15.6.4.4/15-38
0xC_0650	Packet error capture command and status register 1 (PECCSR1)	R/W	0x0000_0000	15.6.4.5/15-39
0xC_0654	Packet error capture command and status register 2 (PECCSR2)	R/W	0x0000_0000	15.6.4.6/15-39
0xC_0658	Packet error capture command and status register 3 (PECCSR3)	R/W	0x0000_0000	15.6.4.7/15-40
0xC_0668	Error rate command and status register (ERCSR)	R/W	0x8000_0000	15.6.4.8/15-40
0xC_066C	Error rate threshold command and status register (ERTCSR)	R/W	0xFFFF_0000	15.6.4.9/15-41
0xC_0670 –0xC_067C	Reserved			
Implementation Space				
General				
0xD_0004	Logical layer configuration register (LLCR)	R/W	0x0000_0000	15.6.5.1/15-42
0xD_0010	Error / port-write interrupt status register (EPWISR)	R	0x0000_0000	15.6.5.2/15-43
0xD_0020	Logical retry error threshold configuration register (LRETCR)	R/W	0x0000_00FF	15.6.5.3/15-43
0xD_0080	Physical retry error threshold configuration register (PRETCR)	R/W	0x0000_00FF	15.6.5.4/15-44
0xD_0100	Alternate device ID command and status register (ADIDCSR)	R/W	0x0000_0000	15.6.5.5/15-44
0xD_0120	Accept-all configuration register (AACR)	R/W	0x*n*000_000*n*	15.6.5.6/15-45
0xD_0124	Logical Outbound Packet time-to-live configuration register (LOPTTLCR)	R/W	0x0000_0000	15.6.5.7/15-45
0xD_0130	Implementation error command and status register (IECSR)	w1c	0x0000_0000	15.6.5.8/15-46
0xD_0140	Physical configuration register (PCR)	R/W	0x0000_8010	15.6.5.9/15-47
0xD_0158	Serial link command and status register (SLCSR)	w1c	0x0000_0000	15.6.5.10/15-47
0xD_0160	Serial link error injection configuration register (SLEICR)	R/W	0x0000_0000	15.6.5.11/15-48
Revision Control Register				
0xD_0BF8	IP Block Revision Register 1 (IPBRR1)	R	0x01C0_0000	15.6.6.1/15-49
0xD_0BFC	IP Block Revision Register 2 (IPBRR2)	R	0x0000_0000	15.6.6.2/15-49

Table 15-1. RapidIO Memory Map (continued)

Offset	Register	Access	Reset[1]	Section/Page
	ATMU			
0xD_0C00	RapidIO outbound window translation address register 0 (ROWTAR0)	R/W	0xFF80_0000	15.6.7.2/15-52
0xD_0C04	RapidIO outbound window translation extended address register 0 (ROWTEAR0)	R/W	0x0000_003F	15.6.7.3/15-53
0xD_0C10	RapidIO outbound window attributes register 0 (ROWAR0)	R/W	0x8004_4023	15.6.7.5/15-54
0xD_0C20	RapidIO outbound window translation address register 1 (ROWTAR1)	R/W	0x0000_0000	15.6.7.2/15-52
0xD_0C24	RapidIO outbound window translation extended address register 1 (ROWTEAR1)	R/W	0x0000_0000	15.6.7.3/15-53
0xD_0C28	RapidIO outbound window base address register 1 (ROWBAR1)	R/W	0x0000_0000	15.6.7.4/15-54
0xD_0C30	RapidIO outbound window attributes register 1 (ROWAR1)	R/W	0x0004_4023	15.6.7.5/15-54
0xD_0C34	RapidIO outbound window segment 1 register 1 (ROWS1R1)	R/W	0x0044_0000	15.6.7.6/15-56
0xD_0C38	RapidIO outbound window segment 2 register 1 (ROWS2R1)	R/W	0x0044_0000	15.6.7.6/15-56
0xD_0C3C	RapidIO outbound window segment 3 register 1 (ROWS3R1)	R/W	0x0044_0000	15.6.7.6/15-56
0xD_0C40 – 0xD_0CFC	RapidIO outbound window 2 through outbound window 7	—	—	—
0xD_0D00	RapidIO outbound window translation address register 8 (ROWTAR8)	R/W	0x0000_0000	15.6.7.2/15-52
0xD_0D04	RapidIO outbound window translation extended address register 8 (ROWTEAR8)	R/W	0x0000_0000	15.6.7.3/15-53
0xD_0D08	RapidIO outbound window base address register 8 (ROWBAR8)	R/W	0x0000_0000	15.6.7.4/15-54
0xD_0D10	RapidIO outbound window attributes register 8 (ROWAR8)	R/W	0x0004_4023	15.6.7.5/15-54
0xD_0D14	RapidIO outbound window segment 1 register 8 (ROWS1R8)	R/W	0x0044_0000	15.6.7.6/15-56
0xD_0D18	RapidIO outbound window segment 2 register 8 (ROWS2R8)	R/W	0x0044_0000	15.6.7.6/15-56
0xD_0D1C	RapidIO outbound window segment 3 register 8 (ROWS3R8)	R/W	0x0044_0000	15.6.7.6/15-56
0xD_0D60	RapidIO Inbound window translation address register 4 (RIWTAR4)	R/W	0x0000_0000	15.6.7.7/15-57
0xD_0D68	RapidIO Inbound window base address register 4 (RIWBAR4)	R/W	0x0000_0000	15.6.7.8/15-58
0xD_0D70	RapidIO inbound window attributes register 4 (RIWAR4)	R/W	0x0004_4021	15.6.7.9/15-59
0xD_0D80– 0xD_0DBC	RapidIO inbound window 3 through inbound window 2			
0xD_0DC0	RapidIO inbound window translation address register 1 (RIWTAR1)	R/W	0x0000_0000	15.6.7.7/15-57
0xD_0DC8	RapidIO inbound window base address register 1 (RIWBAR1)	R/W	0x0000_0000	15.6.7.8/15-58
0xD_0DD0	RapidIO inbound window attributes register 1 (RIWAR1)	R/W	0x0004_4021	15.6.7.9/15-59

Table 15-1. RapidIO Memory Map (continued)

Offset	Register	Access	Reset[1]	Section/Page
0xD_0DE0	RapidIO inbound window translation address register 0 (RIWTAR0)	R/W	0x0000_0000	15.6.7.7/15-57
0xD_0DF0	RapidIO inbound window attributes register 0 (RIWAR0)	Mixed	0x8004_4021	15.6.7.9/15-59
RapidIO Message Unit				
RapidIO Outbound Message Unit 0 Registers				
0xD_3000	Outbound message 0 mode register (OM0MR)	R/W	0x0000_0000	15.7.1.1/15-61
0xD_3004	Outbound message 0 status register (OM0SR)	Mixed	0x0000_0000	15.7.1.2/15-63
0xD_3008	Extended outbound message 0 descriptor queue dequeue pointer address register (EOM0DQDPAR)	R/W	0x0000_0000	15.7.1.3/15-64
0xD_300C	Outbound message 0 descriptor queue dequeue pointer address register (OM0DQDPAR)	R/W	0x0000_0000	15.7.1.3/15-64
0xD_3010	Extended outbound message 0 source address register (EOM0SAR)	R/W	0x0000_0000	15.7.1.5/15-67
0xD_3014	Outbound message 0 source address register (OM0SAR)	R/W	0x0000_0000	15.7.1.5/15-67
0xD_3018	Outbound message 0 destination port register (OM0DPR)	R/W	0x0000_0000	15.7.1.6/15-68
0xD_301C	Outbound message 0 destination attributes Register (OM0DATR)	R/W	0x0000_0000	15.7.1.7/15-69
0xD_3020	Outbound message 0 double-word count register (OM0DCR)	R/W	0x0000_0000	15.7.1.8/15-69
0xD_3024	Extended outbound message 0 descriptor queue enqueue pointer address register (EOM0DQEPAR)	R/W	0x0000_0000	15.7.1.4/15-66
0xD_3028	Outbound message 0 descriptor queue enqueue pointer address register (OM0DQEPAR)	R/W	0x0000_0000	15.7.1.4/15-66
0xD_302C	Outbound message 0 retry error threshold configuration register (OM0RETCR)	R/W	0x0000_0000	15.7.1.9/15-70
0xD_3030	Outbound message 0 multicast group register (OM0MGR)	R/W	0x0000_0000	15.7.1.10/15-71
0xD_3034	Outbound message 0 multicast list register (OM0MLR)	R/W	0x0000_0000	15.7.1.11/15-71
RapidIO Inbound Message Unit 0 Registers				
0xD_3060	Inbound message 0 mode register (IM0MR)	R/W	0x0000_0000	15.7.2.1/15-72
0xD_3064	Inbound message 0 status register (IM0SR)	Mixed	0x0000_0002	15.7.2.2/15-74
0xD_3068	Extended inbound message 0 frame queue dequeue pointer address register (EIM0FQDPAR)	R/W	0x0000_0000	15.7.2.3/15-75
0xD_306C	Inbound message 0 frame queue dequeue pointer address register (IM0FQDPAR)	R/W	0x0000_0000	15.7.2.3/15-75
0xD_3070	Extended inbound message 0 frame queue enqueue pointer address register (EIM0FQEPAR)	R/W	0x0000_0000	15.7.2.4/15-76
0xD_3074	Inbound message 0 frame queue enqueue pointer address register (IM0FQEPAR)	R/W	0x0000_0000	15.7.2.4/15-76
0xD_3078	Inbound message 0 maximum interrupt report interval register (IM0MIRIR)	R/W	0xFFFF_FF00	15.7.2.5/15-77

Table 15-1. RapidIO Memory Map (continued)

Offset	Register	Access	Reset[1]	Section/Page
RapidIO Outbound Message Unit 1 Registers				
0xD_3100	Outbound message 1 mode register (OM1MR)	R/W	0x0000_0000	15.7.1.1/15-61
0xD_3104	Outbound message 1 status register (OM1SR)	Mixed	0x0000_0000	15.7.1.2/15-63
0xD_3108	Extended outbound message 1 descriptor queue dequeue pointer address register (EOM1DQDPAR)	R/W	0x0000_0000	15.7.1.3/15-64
0xD_310C	Outbound message 1 descriptor queue dequeue pointer address register (OM1DQDPAR)	R/W	0x0000_0000	15.7.1.3/15-64
0xD_3110	Extended outbound message 1 source address register (EOM1SAR)	R/W	0x0000_0000	15.7.1.5/15-67
0xD_3114	Outbound message 1 source address register (OM1SAR)	R/W	0x0000_0000	15.7.1.5/15-67
0xD_3118	Outbound message 1 destination port register (OM1DPR)	R/W	0x0000_0000	15.7.1.6/15-68
0xD_311C	Outbound message 1 destination attributes register (OM1DATR)	R/W	0x0006_0000	15.7.1.7/15-69
0xD_3120	Outbound message 1 double-word count register (OM1DCR)	R/W	0x0000_0000	15.7.1.8/15-69
0xD_3124	Extended outbound message 1 descriptor queue enqueue pointer address register (EOM1DQEPAR)	R/W	0x0000_0000	15.7.1.4/15-66
0xD_3128	Outbound message 1 descriptor queue enqueue pointer address register (OM1DQEPAR)	R/W	0x0000_0000	15.7.1.4/15-66
0xD_312C	Outbound message 1 retry error threshold configuration register (OM1RETCR)	R/W	0x0000_0000	15.7.1.9/15-70
0xD_3130	Outbound message 1 multicast group register (OM1MGR)	R/W	0x0000_0000	15.7.1.10/15-71
0xD_3134	Outbound message 1 multicast list register (OM1MLR)	R/W	0x0000_0000	15.7.1.11/15-71
RapidIO Inbound Message Unit 1 Registers				
0xD_3160	Inbound message 1 mode register (IM1MR)	R/W	0x0000_0000	15.7.2.1/15-72
0xD_3164	Inbound message 1 status register (IM1SR)	Mixed	0x0000_0002	15.7.2.2/15-74
0xD_3168	Extended inbound message 1 frame queue dequeue pointer address register (EIM1FQDPAR)	R/W	0x0000_0000	15.7.2.3/15-75
0xD_316C	Inbound message 1 frame queue dequeue pointer address register (IM1FQDPAR)	R/W	0x0000_0000	15.7.2.3/15-75
0xD_3170	Extended inbound message 1 frame queue enqueue pointer address register (EIM1FQEPAR)	R/W	0x0000_0000	15.7.2.4/15-76
0xD_3174	Inbound message 1 frame queue enqueue pointer address register (IM1FQEPAR)	R/W	0x0000_0000	15.7.2.4/15-76
0xD_3178	Inbound message 1 maximum interrupt report interval register (IM1MIRIR)	R/W	0xFFFF_FF00	15.7.2.5/15-77
RapidIO Doorbell Registers				
0xD_3400	Outbound doorbell mode register (ODMR)	R/W	0x0000_0000	15.7.3.1/15-78
0xD_3404	Outbound doorbell status register (ODSR)	Mixed	0x0000_0000	15.7.3.2/15-79
0xD_3408–0xD_3414	Reserved			

Table 15-1. RapidIO Memory Map (continued)

Offset	Register	Access	Reset[1]	Section/Page
0xD_3418	Outbound doorbell destination port register (ODDPR)	R/W	0x0000_0000	15.7.3.3/15-79
0xD_341C	Outbound doorbell destination attributes register (ODDATR)	R/W	0x0000_0000	15.7.3.4/15-80
0xD_3420–0xD_3428	Reserved			
0xD_342C	Outbound doorbell retry error threshold configuration register (ODRETCR)	R/W	0x0000_0000	15.7.3.5/15-81
0xD_3430–0xD_345C	Reserved			
0xD_3460	Inbound doorbell mode register (IDMR)	R/W	0x0000_0000	15.7.4.1/15-82
0xD_3464	Inbound doorbell status register (IDSR)	Mixed	0x0000_0002	15.7.4.2/15-83
0xD_3468	Extended inbound doorbell queue dequeue pointer address register (EIDQDPAR)	R/W	0x0000_0000	15.7.4.3/15-84
0xD_346C	Inbound doorbell queue dequeue Pointer address register (IDQDPAR)	R/W	0x0000_0000	15.7.4.3/15-84
0xD_3470	Extended inbound doorbell queue enqueue pointer address register (EIDQEPAR)	R/W	0x0000_0000	15.7.4.4/15-85
0xD_3474	Inbound doorbell Queue enqueue pointer address register (IDQEPAR)	R/W	0x0000_0000	15.7.4.4/15-85
0xD_3478	Inbound doorbell maximum interrupt report interval register (IDMIRIR)	R/W	0xFFFF_FF00	15.7.4.5/15-86
0xD_347C	Reserved			
RapidIO Port-Write Registers				
0xD_34E0	Inbound port-write mode register (IPWMR)	R/W	0x0000_0000	15.7.5.1/15-87
0xD_34E4	Inbound port-write status register (IPWSR)	Mixed	0x0000_0000	15.7.5.2/15-88
0xD_34E8	Extended inbound port-write queue base address register (EIPWQBAR)	R/W	0x0000_0000	15.7.5.3/15-88
0xD_34EC	Inbound port-write queue base address register (IPWQBAR)	R/W	0x0000_0000	15.7.5.3/15-88

1 Values indicated with *n* are read from configuration signals at reset; see register description for details.

15.6 RapidIO Endpoint Configuration Register Definitions

The RapidIO endpoint registers are described in detail below.

15.6.1 RapidIO Architectural Registers

15.6.1.1 Device Identity Capability Register (DIDCAR)

Figure 15-2 shows the fields of the device identity capability register (DIDCAR). The device vendor identity field (DVI) identifies the vendor that manufactured the device containing the processing element. A value for DVI is uniquely assigned to a device vendor by the registration authority of the RapidIO Trade Association.

The device identity field (DI) is intended to uniquely identify the type of device from the vendor specified by DVI. The values for DI are assigned and managed by the respective vendor.

DIDCAR is a read-only register.

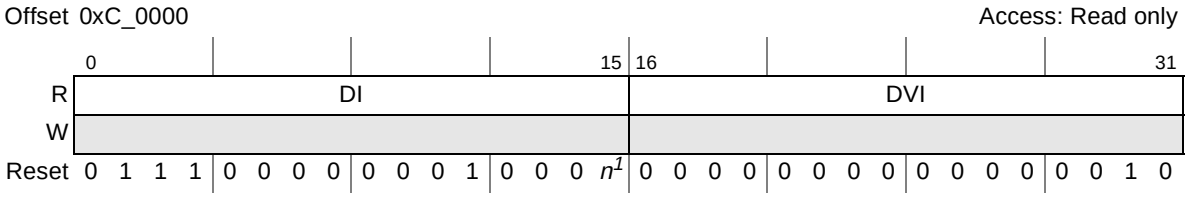


Figure 15-2. Device Identity Capability Register (DIDCAR)

1 This bit has a reset value of 0 for single core device and a value of 1 for a dual core device.

Table 15-2 lists DIDCAR fields.

Table 15-2. DIDCAR Field Descriptions

Bits	Name	Description
0–15	DI	Device identity 0x7010 = MPC8641(single core); 0x7011 = MPC8641D(dual core)
16–31	DVI	Device vendor identity (Freescale = 0x0002)

15.6.1.2 Device Information Capability Register (DICAR)

Figure 15-3 shows the fields of the read-only device information capability register (DICAR). The device revision field (DR) is intended to identify the revision level of the device. The value for DR is assigned and managed by the vendor specified by DIDCAR[DVI]. DICAR represents a copy of the device's system version register (SVR). See Section 17.4.1.14, "System Version Register (SVR)," for more information.

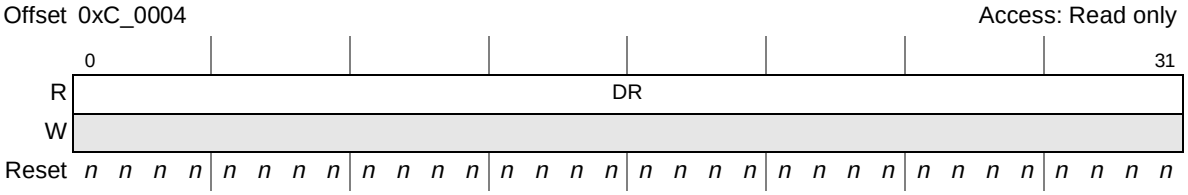


Figure 15-3. Device Information Capability Register (DICAR)

Table 15-3 lists DICAR fields.

Table 15-3. DICAR Field Descriptions

Bits	Name	Description
0–31	DR	Device revision. This is a copy of the device's system version register (SVR). See Section 17.4.1.14, "System Version Register (SVR)," for additional information. 0x8090_00*nn* MPC8641 single core device 0x8090_01*nn* MPC8641D dual core device

15.6.1.3 Assembly Identity Capability Register (AIDCAR)

Figure 15-4 shows the fields of the assembly identity capability register (AIDCAR). The assembly vendor identity field (AVI) identifies the vendor that manufactured the assembly or subsystem containing the device. A value for AVI is uniquely assigned to an assembly vendor by the registration authority of the RapidIO Trade Association.

The assembly identity field (AI) is intended to uniquely identify the type of assembly from the vendor specified by AVI. The values for AI are assigned and managed by the respective vendor.

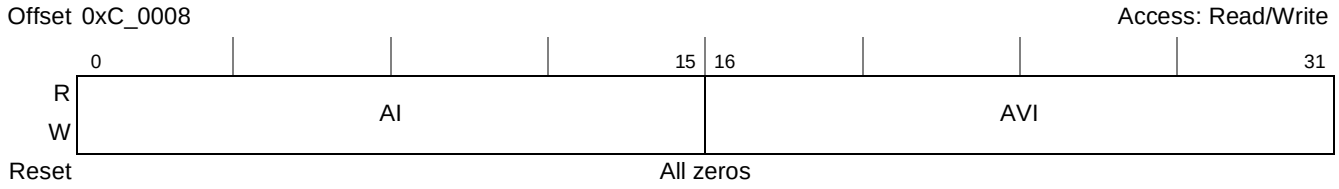


Figure 15-4. Assembly Identity Capability Register (AIDCAR)

Table 15-4 lists the fields of the AIDCAR.

Table 15-4. AIDCAR Field Descriptions

Bits	Name	Description
0–15	AI	Assembly identity (all zeros)
16–31	AVI	Assembly vendor identity (all zeros)

15.6.1.4 Assembly Information Capability Register (AICAR)

Figure 15-5 shows the fields of the assembly information capability register (AICAR). AICAR contains additional information about the assembly and the pointer to the first entry in the extended features list.

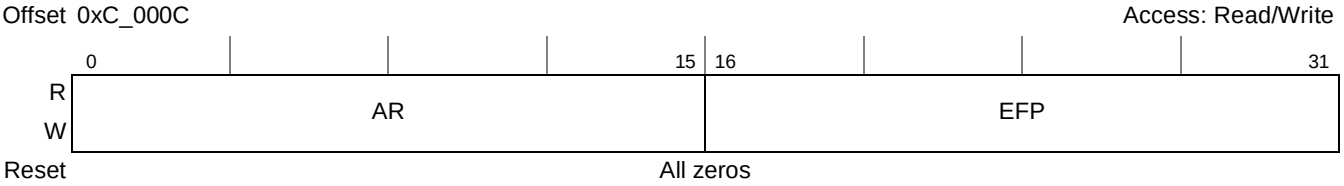


Figure 15-5. Assembly Information Capability Register (AICAR)

Table 15-5 lists AICAR fields.

Table 15-5. AICAR Field Descriptions

Bits	Name	Description
0–15	AR	AssyRev field (all zeros)
16–31	EFP	ExtendedFeaturesPtr field (all zeros)

15.6.1.5 Processing Element Features Capability Register (PEFCAR)

The processing element features capability register (PEFCAR), shown in Figure 15-6, identifies the major functionality provided by the processing element. PEFCAR is a read-only register.

Offset 0xC_0010 Access: Read only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BR	MEM	PROC	SW	—				MB				DB	—														CTLS	EF	EAS		
W																																
Reset	1	1	1	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	*n*	1	0	0	1

Figure 15-6. Processing Element Features Capability Register (PEFCAR)

Table 15-6 lists fields of the PEFCAR.

Table 15-6. PEFCAR Field Descriptions

Bits	Name	Description
0	BR	Bridge. PE can bridge to another interface. (BR = 1)
1	MEM	Memory. PE has physically addressable local address space and can be accessed as an endpoint through non-maintenance operations. (MEM = 1)
2	PROC	Processor. PE physically contains a local processor that executes code. (PROC = 1)
3	SW	Switch. PE does not bridge to another external RapidIO interface. (SW = 0)
4–7	—	Reserved
8–11	MB	Mailbox 0–3. Bit 0 indicates PE supports inbound mailbox 0. Bit 1 indicates PE supports inbound mailbox 1. Bit 2 indicates PE supports inbound mailbox 2. Bit 3 indicates PE supports inbound mailbox 3. (MB = 1111)
12	DB	Doorbell. The RapidIO controller supports inbound doorbells. (DB = 1)
13–26	—	Reserved
27	CTLS	This bit reflects the selected RapidIO common transport system size. The RapidIO common transport system size is determined at power-on reset. See Section 4.4.3.16, “Serial RapidIO System Size,” for POR configuration details. 0 PE only supports common transport small systems (up to 256 devices). 1 PE supports common transport large systems (up to 65,536 devices).
28	EF	Extended features pointer is valid. (EF = 1)
29–31	EAS	Indicates the number of address bits supported by the PE both as a source and target of an operation. EAS = 001(34-bit addresses)

15.6.1.6 Source Operations Capability Register (SOCAR)

The source operations capability register (SOCAR), shown in Figure 15-7, defines the set of RapidIO I/O logical operations that can be issued by this processing element. SOCAR is a read-only register.

Offset 0xC_0018 Access: Read only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	R	IR	RO	DCI	C	F	IOR	ICI	TIE	TIES	—					
W																
Reset	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	NR	NW	SW	NWR	M	D	—	ATS	AI	AD	AS	AC	—	PW	—	
W																
Reset	1	1	1	1	1	1	0	0	1	1	1	1	0	0	0	0

Figure 15-7. Source Operations Capability Register (SOCAR)

Table 15-7 lists SOCAR fields.

Table 15-7. SOCAR Field Descriptions

Bits	Name	Description
0	R	PE does not support a Read operation. (R = 0)
1	IR	PE does not support an IRead operation. (IR = 0)
2	RO	PE does not support a Read_To_Own operation. (RO = 0)
3	DCI	PE does not support a Data Cache Invalidate operation. (DCI = 0)
4	C	PE does not support a Castout operation. (C = 0)
5	F	PE supports a Flush operation. (F = 1)
6	IOR	PE supports an I/O-Read operation. (IOR = 1)
7	ICI	PE does not support an Instruction Cache Invalidate operation. (ICI = 0)
8	TIE	PE does not support a TLBIE operation. (TIE = 0)
9	TIES	PE does not support a TLBSYNC operation. (TIES = 0)
10–15	—	Reserved
16	NR	PE supports a Nread operation. (NR = 1)
17	NW	PE supports a Nwrite operation. (NW = 1)
18	SW	PE supports an Swrite operation. (SW = 1)
19	NWR	PE supports a Nwrite_R operation. (NWR = 1)
20	M	PE supports a Message operation. (M = 1)
21	D	PE supports a Doorbell operation. (D = 1)
22	—	Reserved
23	ATS	PE does not support an Atomic-Test-and-Swap operation. (ATS = 0)
24	AI	PE supports an Atomic-Inc operation. (AI = 1)
25	AD	PE supports an Atomic-Dec operation. (AD = 1)

Table 15-7. SOCAR Field Descriptions (continued)

Bits	Name	Description
26	AS	PE supports an Atomic-Set operation. (AS = 1)
27	AC	PE supports an Atomic-Clr operation. (AC = 1)
28	—	Reserved
29	PW	PE does not support a Port-Write operation. (PW = 0)
30–31	—	Reserved

15.6.1.7 Destination Operations Capability Register (DOCAR)

The destination operations capability register (DOCAR), shown in Figure 15-8, defines the set of RapidIO I/O operations that can be supported by this processing element. DOCAR is a read-only register.

Offset 0xC_001C Access: Read only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	R	IR	RO	DCI	C	F	IOR	ICI	TIE	TIES	—					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	NR	NW	SW	NWR	M	D	—	ATS	AI	AD	AS	AC	—	PW	—	
Reset	1	1	1	1	1	1	0	0	1	1	1	1	0	1	0	0

Figure 15-8. Destination Operations Capability Register (DOCAR)

Table 15-8 lists DOCAR fields.

Table 15-8. DOCAR Field Descriptions

Bits	Name	Description
0	R	PE does not support a Read operation. (R = 0)
1	IR	PE does not support an IRead operation. (IR = 0)
2	RO	PE does not support a Read_To_Own operation. (RO = 0)
3	DCI	PE does not support a Data Cache Invalidate operation. (DCI = 0)
4	C	PE does not support a Castout operation. (C = 0)
5	F	PE does not support a Flush operation. (F = 0)
6	IOR	PE does not support an I/O-Read operation. (IOR = 0)
7	ICI	PE does not support an Instruction Cache Invalidate operation. (ICI = 0)
8	TIE	PE does not support a TLBIE operation. (TIE = 0)
9	TIES	PE does not support a TLBSYNC operation. (TIES = 0)
10–15	—	Reserved
16	NR	PE supports a Nread operation. (NR = 1)

Table 15-8. DOCAR Field Descriptions (continued)

Bits	Name	Description
17	NW	PE supports a Nwrite operation. (NW = 1)
18	SW	PE supports an Swrite operation. (SW = 1)
19	NWR	PE supports a Nwrite_R operation. (NWR = 1)
20	M	PE supports a Message operation. (M = 1)
21	D	PE supports a Doorbell operation. (D = 1)
22	—	Reserved
23	ATS	PE does not support an Atomic-Test-and-Swap operation. (ATS = 0)
24	AI	PE supports an Atomic-Inc operation. (AI = 1)
25	AD	PE supports an Atomic-Dec operation. (AD = 1)
26	AS	PE supports an Atomic-Set operation. (AS = 1)
27	AC	PE supports an Atomic-Clr operation. (AC = 1)
28	—	Reserved
29	PW	PE supports a Port-Write operation. (PW = 1)
30–31	—	Reserved

15.6.1.8 Mailbox Command and Status Register (MCSR)

The MCSR, shown in Figure 15-9, reflects the status of the RapidIO message controllers on this device.

Offset 0xC_0040 Access: Read only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15–31
R	A0	FU0	EM0	B0	FA0	ERR0	—		A1	FU1	EM1	B1	FA1	ERR1	—	
W																
Reset	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0

Figure 15-9. Mailbox Command and Status Register (MCSR)

Table 15-9 describes the fields of the MCSR.

Table 15-9. MCSR Field Definitions

Bits	Name	Description
0	A0	Available 0 Message Controller 0 is not ready to accept messages. All incoming message transactions return error responses. 1 Message Controller 0 is initialized and ready to accept messages.
1	FU0	Full 0 Message Controller 0 is not full. 1 Message Controller 0 is full. New messages are retried.

Table 15-9. MCSR Field Definitions (continued)

Bits	Name	Description
2	EM0	Empty 0 Message Controller 0 contains outstanding messages. 1 Message Controller 0 contains no outstanding messages.
3	B0	Busy 0 Message Controller 0 is not busy processing a message. 1 Message Controller 0 is busy processing a message. New message operations return retry responses.
4	FA0	Failure 0 Message Controller 0 has not encountered an internal error. 1 Message Controller 0 had an internal fault or error condition and is waiting for assistance. All incoming message transactions return error responses.
5	ERR0	Error This field always returns a 0.
6–7	—	Reserved
8	A1	Available 0 Message Controller 1 is not ready to accept messages. All incoming message transactions return error responses. 1 Message Controller 1 is initialized and ready to accept messages.
9	FU1	Full 0 Message Controller 1 is not full. 1 Message Controller 1 is full. New messages are retried.
10	EM1	Empty 0 Message Controller 1 contains outstanding messages. 1 Message Controller 1 contains no outstanding messages.
11	B1	Busy 0 Message Controller 1 is not busy processing a message. 1 Message Controller 1 is busy processing a message. New message operations return retry responses.
12	FA1	Failure 0 Message Controller 1 has not encountered an internal error. 1 Message Controller 1 had an internal fault or error condition and is waiting for assistance. All incoming message transactions return error responses.
13	ERR1	Error This field always returns a 0.
14–31	—	Reserved

15.6.1.9 Port-Write and Doorbell Command and Status Register (PWDCSR)

The PWDCSR, shown in Figure 15-10, reflects the status of the RapidIO doorbell and port-write hardware on this device. Additional details can be found in the *RapidIO Interconnect Specification, Revision 1.2*, in sections “Doorbell CSR” and “Write Port CSR.”

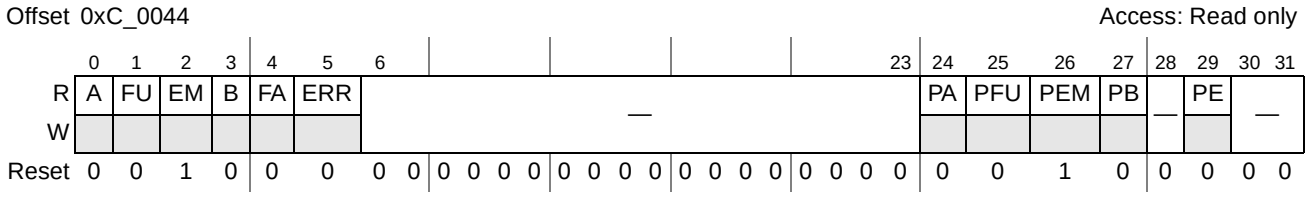


Figure 15-10. Port-Write and Doorbell Command and Status Register (PWDCSR)

Table 15-10 describes the fields of the PWDCSR.

Table 15-10. PWDCSR Field Descriptions

Bits	Name	Description
0	A	Available 0 Doorbell unit is not ready to accept doorbell messages. All incoming doorbell transactions return error responses. 1 Doorbell unit is initialized and ready to accept doorbell messages.
1	FU	Full 0 Doorbell unit is not full. 1 Doorbell unit is full. New doorbell messages are retried.
2	EM	Empty 0 Doorbell unit has outstanding doorbell messages. 1 Doorbell unit has no outstanding doorbell messages.
3	B	Busy 0 Doorbell unit is not busy processing a doorbell message. 1 Doorbell unit is busy processing a doorbell message. Incoming transactions are not retried.
4	FA	Failure 0 Doorbell unit has not encountered an internal error. 1 Doorbell unit had an internal fault or error condition and is waiting for assistance. All incoming doorbell transactions return error responses.
5	ERR	Error This field always returns a 0.
6–23	—	Reserved
24	PA	Port-write unit available. 0 Port-write unit is not available. All incoming port-write packets are discarded. 1 Port-write unit is initialized and ready to accept a port-write packet.
25	PFU	Port-write unit full 0 Port-write unit is not full. 1 Port-write unit is full. All incoming port-write packets are discarded.
26	PEM	Port-write unit empty This field always returns a 1.
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 15-10. PWDCSR Field Descriptions (continued)

Bits	Name	Description
29	PE	Error This field always returns a 0.
30–31	—	Reserved

15.6.1.10 Processing Element Logical Layer Control Command and Status Register (PELLCCSR)

The processing element logical layer control command and status register (PELLCCSR), shown in Figure 15-11, controls the extended addressing abilities. The RapidIO endpoint only supports 34-bit addressing. PELLCCSR is a read-only register.

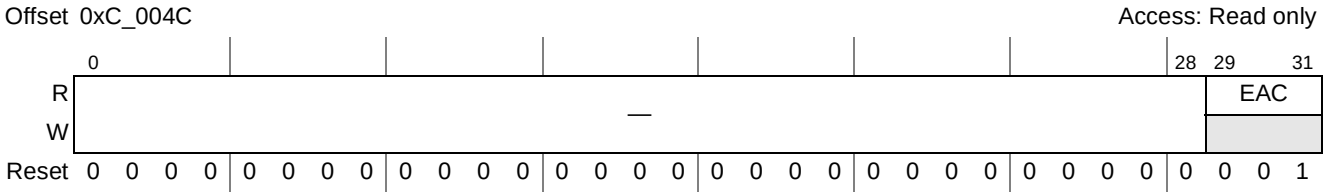


Figure 15-11. Processing Element Logic Layer Control Command and Status Register (PELLCCSR)

Table 15-11 lists PELLCCSR fields.

Table 15-11. PELLCCSR Field Descriptions

Bits	Name	Description
0–28	—	Reserved
29–31	EAC	Extended addressing control. Read-only value is 0b001.

15.6.1.11 Local Configuration Space Base Address 1 Command and Status Register (LCSBA1CSR)

The local configuration space base address 1 command and status register (LCSBA1CSR), shown in Figure 15-12, specifies the least-significant bits of the local physical address double-word offset for the processing element's configuration register space, allowing the configuration register space to be physically mapped in the processing element. This register allows configuration and maintenance of a processing element through regular read and write operations rather than maintenance operations. The double-word offset is right-justified in the register. This window has priority over all ATMU windows. As is the case with all registers, an external processor writing to LCSBA1CSR should not assume it has been written until a response has been received.

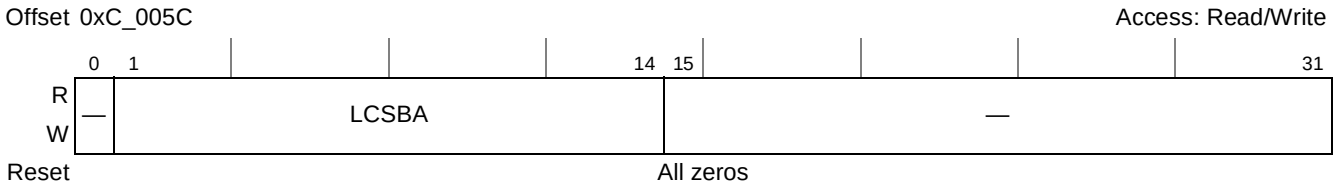


Figure 15-12. Local Configuration Space Base Address 1 Command and Status Register (LCSBA1CSR)

Table 15-12 lists LCSBA1CSR fields.

Table 15-12. LCSBA1CSR Field Descriptions

Bits	Name	Description
0	—	Reserved
1–14	LCSBA	Local configuration space base address. These bits correspond to the highest 14 bits of the 34-bit RapidIO address space.
15–31	—	Reserved

15.6.1.12 Base Device ID Command and Status Register (BDIDCSR)

BDIDCSR, shown in Figure 15-13, contains the base device ID values for the processing element.

Figure 15-13. Base Device ID Command and Status Register (BDIDCSR)

1 Values indicated with *n* are determined by configuration signals at reset.

Table 15-13 lists BDIDCSR fields.

Table 15-13. BDIDCSR Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–15	BDID	Base device ID of the device in a small common transport system (RapidIO device ID). If RapidIO is configured as a host, BDID = {0b0000_0, cfg_device_ID[5:7]}. If RapidIO is configured as an agent, BDID = {0b1111_111, cfg_device_ID[7]}.
16–31	LBDID	Large base device ID of the device in a large common transport system. This field is valid only if PEFCAR[CTLS] is set. If RapidIO is configured as a host, LBDID = 0b0000_0000. If RapidIO is configured as an agent, LBDID = 0b1111_1111.

15.6.1.13 Host Base Device ID Lock Command and Status Register (HBDIDLCSR)

The host base device ID lock command and status register (HBDIDLCSR) contains the base device ID value for the processing element in the system that is responsible for initializing this processing element. The HBDID field is a write-once/resettable field which provides a lock function. Once HBDID is written, all subsequent writes to the field are ignored, except in the case that the value written matches the value contained in the field. In this case, the register is re-initialized to 0xFFFF. After writing HBDID, a processing element must then read the host base device ID lock CSR to verify that it owns the lock before attempting to initialize this processing element. HBDIDLCSR is shown in Figure 15-14.

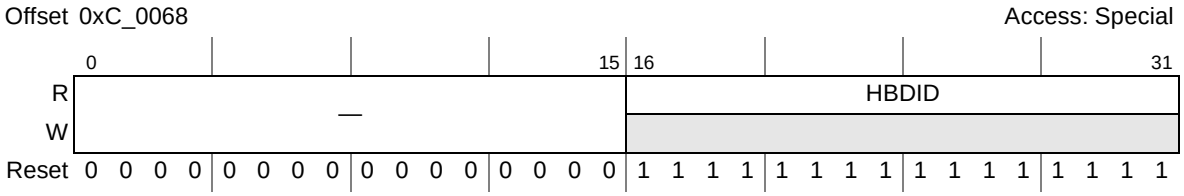


Figure 15-14. Host Base Device ID Lock Command and Status Register (HBDIDLCSR)

Table 15-14 lists HBDIDLCSR fields.

Table 15-14. HBDIDLCSR Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	HBDID	This is the host base device ID for the processing element that is responsible for initializing this device. Only the first write to this field is accepted; all other writes are ignored, except in the case that the value written matches the value contained in the field. In this case, the register is re-initialized to 0xFFFF.

15.6.1.14 Component Tag Command and Status Register (CTCSR)

The component tag command and status register (CTCSR), shown in Figure 15-15, contains a component tag value for the processing element and can be assigned by software when the device is initialized. It is unused internally in the RapidIO endpoint.

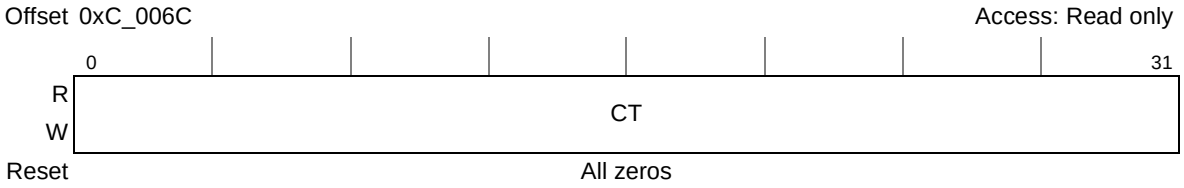


Figure 15-15. Component Tag Command and Status Register (CTCSR)

Table 15-15 lists CTCSR fields.

Table 15-15. CTCSR Field Descriptions

Bits	Name	Description
0–31	CT	Component tag

15.6.2 RapidIO Extended Features Space, 1x/4x LP-Serial Registers

15.6.2.1 Port Maintenance Block Header 0 Register (PMBH0)

The port maintenance block header 0 register (PMBH0), shown in Figure 15-16, contains a pointer to the next EF_BLK (Extended Features Space, Error Management) and the EF_ID that identifies this as the generic end point port maintenance block header. Note that while registers defined by software-assisted error recovery are supported, software-assisted error recovery is not (these registers are included for hot insertion only); therefore, the RapidIO endpoint is defined here as not supporting software-assisted error recovery. PMBH0 is a read-only register.

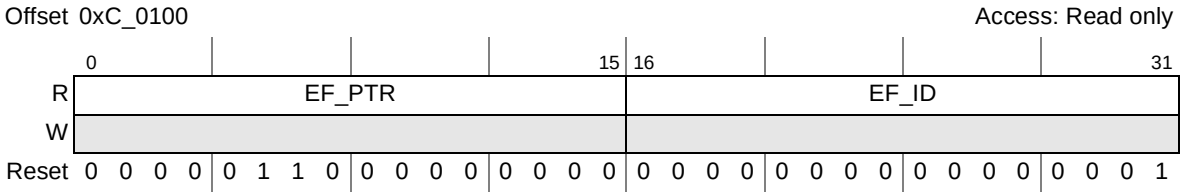


Figure 15-16. Port Maintenance Block Header 0 (PMBH0)

Table 15-16 lists PMBH0 fields.

Table 15-16. PMBH0 Field Descriptions

Bits	Name	Description
0–15	EF_PTR	Extended features pointer
16–31	EF_ID	Extended features ID

15.6.2.2 Port Link Time-Out Control Command and Status Register (PLTOCCSR)

The port link time-out control command and status register (PLTOCCSR), shown in Figure 15-17, contains the link time-out value for all ports on a device. This time-out is for link events such as sending a packet to receiving the corresponding acknowledge and sending a link-request to receiving the corresponding link-response. The reset value is the maximum time-out interval. The timer decrements at one-half the platform clock rate; thus at a platform clock rate of 400 MHz, for example, the maximum time-out value is approximately 83.9 ms.

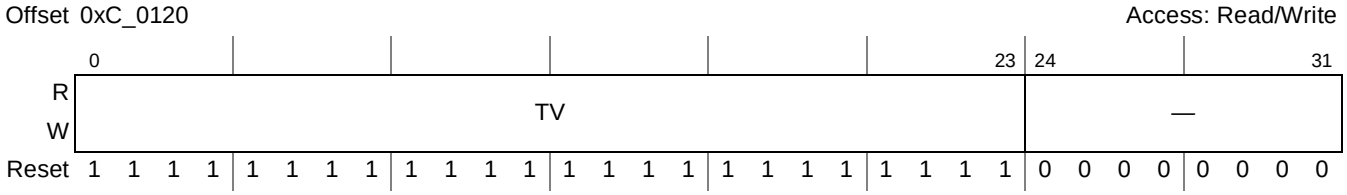


Figure 15-17. Port Link Time-Out Control Command and Status Register (PLTOCCSR)

Table 15-17 lists PLTOCCSR fields.

Table 15-17. PLTOCCSR Field Descriptions

Bits	Name	Description
0–23	TV	Time-out value. Setting to all zeros disables the link time-out timer. This value is loaded each time the link time-out timer starts.
24–31	—	Reserved

15.6.2.3 Port Response Time-Out Control Command and Status Register (PRTOCCSR)

The port response time-out control command and status register (PRTOCCSR), shown in Figure 15-18, contains the time-out timer count for all ports on a device. This time-out is for sending a request packet to receiving the corresponding response packet.The reset value is the maximum time-out interval. The timer decrements at one-half the platform clock rate; thus at a platform clock rate of 400 MHz, for example, the maximum time-out value is approximately 83.9 ms. Note that this applies to the RapidIO endpoint and the messaging unit.

Offset 0xC_0124 Access: Read/Write

	0–23	24–31
R/W	TV	—
Reset	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	0 0 0 0 0 0 0 0

Figure 15-18. Port Response Time-Out Control Command and Status Register (PRTOCCSR)

Table 15-18 lists PRTOCCSR fields.

Table 15-18. PRTOCCSR Field Descriptions

Bits	Name	Description
0–23	TV	Time-out value. Setting to all zeros disables the response time-out timer. This value is loaded each time the response time-out timer starts.
24–31	—	Reserved

15.6.2.4 General Control Command and Status Register (GCCSR)

The general control command and status register (GCCSR), shown in Figure 15-19, contains control register bits applicable to the RapidIO interface.

Offset 0xC_013C Access: Read/Write

	0	1	2	3–31
R/W	H	M	D	—
Reset	*n*	*n*	*n*	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0[1]

Figure 15-19. General Control Command and Status Register (GCCSR)

1 Values indicated with *n* are read from configuration signals at reset.

Table 15-19 lists GCCSR fields.

Table 15-19. GCCSR Field Descriptions

Bits	Name	Description
0	H	Host. The value of this bit is assigned by power-on reset configuration signals, $\overline{\text{LWE}}n/\overline{\text{LBS}}n$, which determine host/agent configuration for the device. See Section 4.4.3.9, "SerDes Host/Agent Configuration," for more information. (Note that although this status bit is R/W, manually changing its value does not affect logical operation.) 0 Agent device 1 Host device
1	M	Master. The value of this bit is identical to GCCSR[H] which is assigned by power-on reset configuration signals, $\overline{\text{LWE}}n/\overline{\text{LBS}}n$, which determine host/agent configuration for the device. See Section 4.4.3.9, "SerDes Host/Agent Configuration," for more information. (Note that although this status bit is R/W, manually changing its value does not affect logical operation.) 0 Device is not enabled to issue requests into the system. 1 Device is enabled to issue requests into the system. M is ignored by the RapidIO endpoint. It is expected that if M = 0, software will not send packets out of outbound. If packets are sent by OCN to outbound, they are sent out of the RapidIO endpoint, regardless of the value of M.
2	D	Discovered. The value of this bit is identical to GCCSR[H] which is assigned by power-on reset configuration signals, $\overline{\text{LWE}}n/\overline{\text{LBS}}n$, which determine host/agent configuration for the device. See Section 4.4.3.9, "SerDes Host/Agent Configuration," for more information. (Note that although this status bit is R/W, manually changing its value does not affect logical operation.) 0 Device has not been discovered by system host. 1 Device has been discovered by system host.
3–31	—	Reserved

15.6.2.5 Link Maintenance Request Command and Status Register (LMREQCSR)

The link maintenance request command and status register (LMREQCSR), shown in Figure 15-20, is accessible both by the local processor and an external device. A write to this register generates a link-request control symbol on the RapidIO endpoint port interface. Care should be taken when writing this register that it is only used for hot swap and not for software-assisted error recovery (which is not supported).

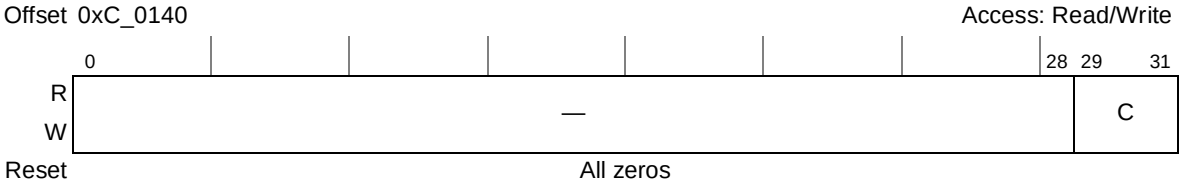


Figure 15-20. Link Maintenance Request Command and Status Register (LMREQCSR)

Table 15-20 lists LMREQCSR fields.

Table 15-20. LMREQCSR Field Descriptions

Bits	Name	Description
0–28	—	Reserved
29–31	C	LINK_REQUEST command to send. If read, this field returns the last written value. If written with a value other than 0b011 (reset-device) or 0b100 (input-status), the resulting operation is undefined, as all other values are reserved in the RapidIO specification.

15.6.2.6 Link Maintenance Response Command and Status Register (LMRESPCSR)

The link maintenance response command and status register (LMRESPCSR), shown in Figure 15-21, is accessible both by the local processor and an external device. A read to this register returns the status received in a link-response control symbol. This register is read only.

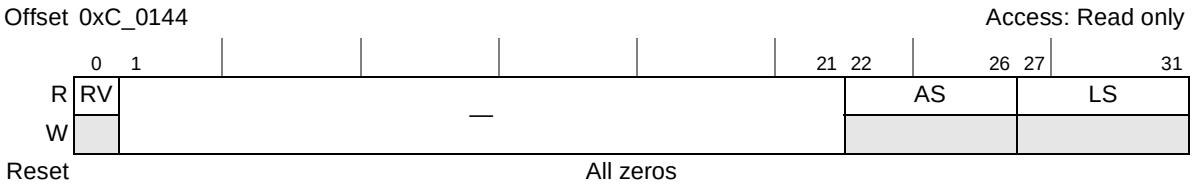


Figure 15-21. Link Maintenance Response Command and Status Register (LMRESPCSR)

Table 15-21 lists the fields of the LMRESPCSR.

Table 15-21. LMRESPCSR Field Descriptions

Bits	Name	Description
0	RV	Response valid. If the link-request causes a link-response, this bit indicates that the link-response has been received and the status fields are valid. If the link-request does not cause a link-response, this bit indicates that the link-request has been transmitted (clears on read)
1–21	—	Reserved
22–26	AS	AckID_status field from LINK_RESPONSE
27–31	LS	Link_status field from LINK_RESPONSE

15.6.2.7 Local ackID Status Command and Status Register (LASCSR)

The local ackID status command and status register (LASCSR), shown in Figure 15-22, is accessible both by the local processor and an external device. A read to this register returns the local ackID status for both the output and input ports of the device. Care should be taken to use this register only for hot swap and not software error management.

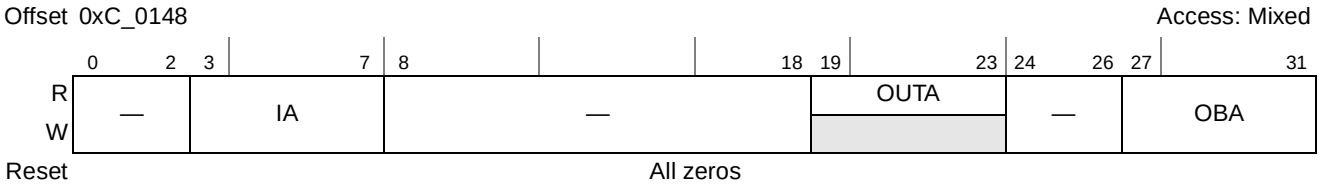


Figure 15-22. Local ackID Status Command and Status Register (LASCSR)

Table 15-22 lists LASCSR fields.

Table 15-22. LASCSR Field Descriptions

Bits	Name	Description
0–2	—	Reserved
3–7	IA	Input port next expected ackID value.
8–18	—	Reserved
19–23	OUTA	Outstanding port unacknowledged ackID status. Next expected acknowledge control symbol ackID field that indicates the ackID value expected in the next received acknowledge control symbol. Note that this value is read only even though RapidIO specification allows for it to be writable.
24–26	—	Reserved
27–31	OBA	Outbound ackID output port next transmitted ackID value. This can be written by software but only if there are no outstanding unacknowledged packets. If there are, the newly-written value is ignored.

15.6.2.8 Error and Status Command and Status Register (ESCSR)

The error and status command and status register (ESCSR), shown in Figure 15-23, is accessed when the local processor or an external device wishes to examine the port error and status information.

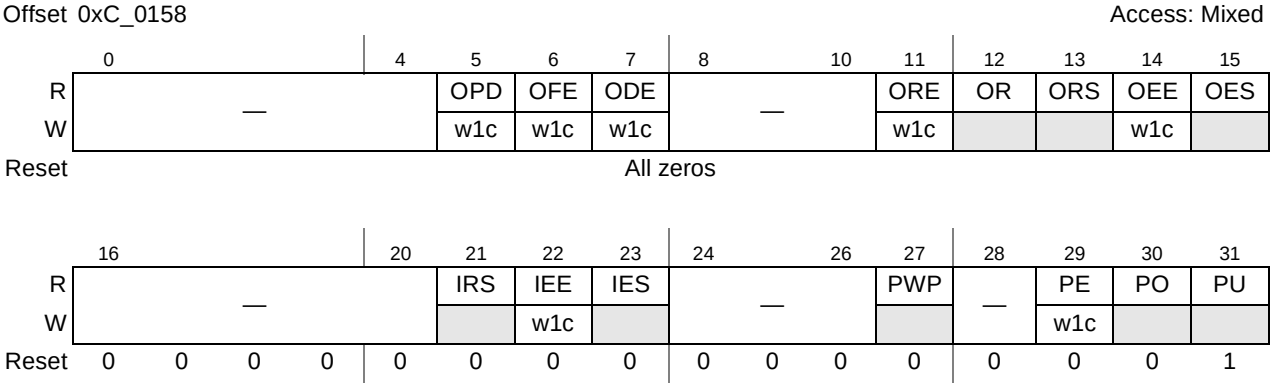


Figure 15-23. Error and Status Command and Status Register (ESCSR)

Table 15-23 lists the fields of the ESCSR.

Table 15-23. ESCSR Field Descriptions

Bits	Name	Description
0–4	—	Reserved
5	OPD	Output Packet-dropped. Output port has discarded a packet. A packet is discarded if: • It is received while OFE is set and CCSR[DPE] (drop packet enable) is set and CCSR[SPF] (stop on port failed) is set. • It is received while PCR[OBDEN] (output buffer drain enable) is set. • It is not-accepted by the link-partner while ERCSR[ERFTT] (error rate failed threshold trigger) is met or exceeded and CCSR[DPE] is set and CCSR[SPF] is not set (and link-response returns expected ackID). Once OPD is set, it remains set until written with a logic 1 to clear.
6	OFE	Output Failed-encountered. Output port has encountered a failed condition, meaning that the Error Rate Counter has met or exceeded the port's failed error threshold (ERFTT) Once set, remains set until written with a logic 1 to clear. Once cleared, does not assert again unless the Error Rate Counter dips below the port's failed error threshold and then meets or exceeds it again.
7	ODE	Output port has encountered a degraded condition, meaning that the Error Rate Counter has met or exceeded the port's degraded error threshold. Once set, remains set until written with a logic 1 to clear. Once cleared, does not assert again unless the Error Rate Counter dips below the port's degraded error threshold and then meets or exceeds it again.
8–10	—	Reserved
11	ORE	Output port has encountered a retry condition. This bit is set when bit 13 is set. Once set, remains set until written with a logic 1 to clear.
12	OR	Output port has received a packet retry control symbol and cannot make forward progress. This bit is set when bit 13 is set and cleared when a packet-accepted or packet-not-accepted control symbol is received. (read only)
13	ORS	Output port is stopped due to a retry (read only)
14	OEE	Output port has encountered (and possibly recovered from) a transmission error. This bit is set when bit 15 is set. Once set, remains set until written with a logic 1 to clear.
15	OES	Output port is stopped due to a transmission error (read only)
16–20	—	Reserved
21	IRS	Input port is stopped due to a retry (read only)
22	IEE	Input port has encountered (and possibly recovered from) a transmission error. This bit is set when bit 23 is set. Once set, remains set until written with a logic 1 to clear.
23	IES	Input port is stopped due to a transmission error (read-only)
24–26	—	Reserved
27	PWP	Port has encountered a condition which required it to initiate a maintenance port-write operation. This bit is only valid if the device is capable of issuing a maintenance port-write transaction. The RapidIO endpoint is not capable of issuing port-writes. This bit is hardwired to 0.
28	—	Reserved
29	PE	Input or output port has encountered an error from which hardware was unable to recover. Once set, remains set until written with a logic 1 to clear. This bit indicates that OFE is set while CCSR[SPF] is set; in other words, the failed threshold has been reached which has caused the output port to stop transmitting packets.

Table 15-23. ESCSR Field Descriptions (continued)

Bits	Name	Description
30	PO	The input and output ports are initialized and the port is exchanging error-free control symbols with the attached device. (read-only).
31	PU	Input and output ports are not initialized. This bit and bit 30 are mutually exclusive (read-only).

15.6.2.9 Control Command and Status Register (CCSR)

The control command and status register (CCSR), shown in Figure 15-24, contains control register bits for the RapidIO port.

Offset 0xC_015C Access: Read/Write

	0	1	2	4	5	7	8	9	10	11	12	13	27	28	29	30	31
R	PW		IPW		PWO		PD	OPE	IPE	ECD	MEP		—	SPF	DPE	PL	PT
W																	
Reset	0	1	0	1	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 15-24. Control Command and Status Register (CCSR)

Table 15-24 lists CCSR fields.

Table 15-24. CCSR Field Descriptions

Bits	Name	Description
0–1	PW	Hardware width of the port (read-only). 00 Single-lane port 01 Four-lane port 10–11 Reserved
2–4	IPW	Width of the ports after initialized (read-only). 000 Single-lane port, lane 0 001 Single-lane port, lane 2 010 Four-lane port 011–111 Reserved
5–7	PWO	Soft port configuration to override the hardware size. 000 No override 001 Reserved 010 Force single lane, lane 0 011 Force single lane, lane 2 100–111 Reserved, causes undefined operation This field should be changed only when the port is uninitialized. To achieve this, first disable the RapidIO port. Then change PWO to any valid value. Finally, re-enable the RapidIO port. To achieve this, first set CCSR[PD] (port disabled). Then change PWO to any legal value. Finally, clear CCSR[PD] (enabled).
8	PD	Port disable. 0 Input error state machine operates normally 1 Input error state machine is forced to normal state

Table 15-24. CCSR Field Descriptions (continued)

Bits	Name	Description
9	OPE	Output port transmit enable. 0 Port is stopped and not enabled to issue any packets except to route or respond to I/O logical MAINTENANCE packets. Control symbols are not affected and are sent normally. 1 Port is enabled to issue any packets. OPE is ignored by RapidIO endpoints. It is expected that if OPE = 0, software will not send packets out of outbound. If packets are sent by OCN to outbound, they are sent out of RapidIO endpoints, regardless of the value of OPE. Initial value read from configuration pins.
10	IPE	Input port receive enable. 0 Port is stopped and only enabled to route or respond to I/O logical MAINTENANCE packets. Other packets generate packet-not-accepted control symbols to force an error condition to be signaled by the sending device. Control symbols are not affected and are received and handled normally. 1 Port is enabled to respond to any packet. This bit value must equal the value of the output port enable (OPE) bit in order for the RapidIO controller to function properly. Initial value read from configuration pins.
11	ECD	Error checking disable. This bit is hardwired to 0. This bit disables all RapidIO transmission error checking 0 Error checking and recovery is enabled. 1 Error checking and recovery is disabled.
12	MEP	Multicast-event participant. This bit is hard-wired to 0.
13–27	—	Reserved
28	SPF	Stop on port failed-encountered enable. This bit is used with the drop packet enable bit to force certain behavior when the error rate failed threshold has been met or exceeded.
29	DPE	Drop packet enable. This bit is used with the stop on port failed-encountered enable bit to force certain behavior when the error rate failed threshold has been met or exceeded.
30	PL	Port lockout. 0 The packets that may be received and issued are controlled by the state of the OPE and IPE bits. 1 This port is stopped and is not enabled to issue or receive any packets; the input port can still follow the training procedure and can still send and respond to link-requests; all received packets return packet-not-accepted control symbols to force an error condition to be signaled by the sending device.
31	PT	Port type (read-only). 0 Reserved 1 Serial port.

15.6.3 RapidIO Extended Features Space—Error Reporting Logical Registers

15.6.3.1 Error Reporting Block Header Register (ERBH)

The error reporting block header register contains the EF_PTR to the next EF_BLK (the next EF_PTR is 0x0000 since this is the last set of registers in the extended features space) and the EF_ID that identifies this as the error reporting block header. ERBH is a read-only register.

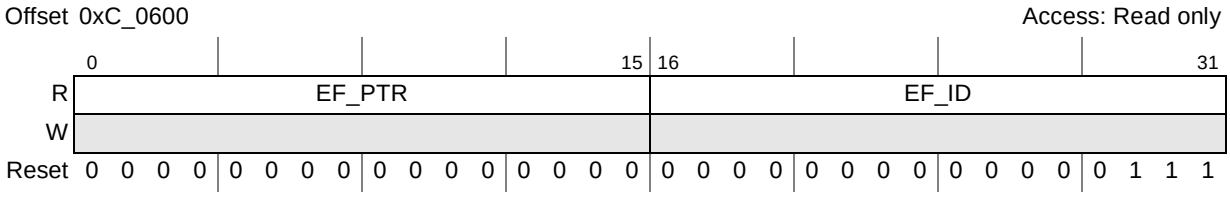


Figure 15-25. Error Reporting Block Header (ERBH)

Table 15-25 lists the fields of ERBH.

Table 15-25. ERBH Field Descriptions

Bits	Name	Description
0–15	EF_PTR	Extended features pointer
16–31	EF_ID	Extended features ID

15.6.3.2 Logical/Transport Layer Error Detect Command and Status Register (LTLEDCSR)

This register indicates the error that was detected by the Logical or Transport logic layer. Software should write this register with all 0s to clear the detected error and unlock the capture registers.

Error information that corresponds to two or more different error events are not captured on the same clock cycle. However, one error event (such as a corrupted inbound packet) can cause multiple bits to be set. The priority of errors is PRT and all other errors. OACB error results in PRT. When PRT bit is set with OACB bit, error capture is for an OCN transaction for which an Outbound ATMU window boundary was crossed or a segment or subsegment boundary was crossed.

An error that is not enabled will set the detect bit in this register as long as a capture has not yet occurred.

Note that fields in this register can be set by writing to this register. This can be used to emulate a hardware error during software development. However, undefined results occur if fields in this register are set while an actual Logical/Transport Layer error is being detected. Also, note that this register can be written with an invalid combination of bits set and care should be taken to avoid this.

LTLEDCSR is shown in Figure 15-26.

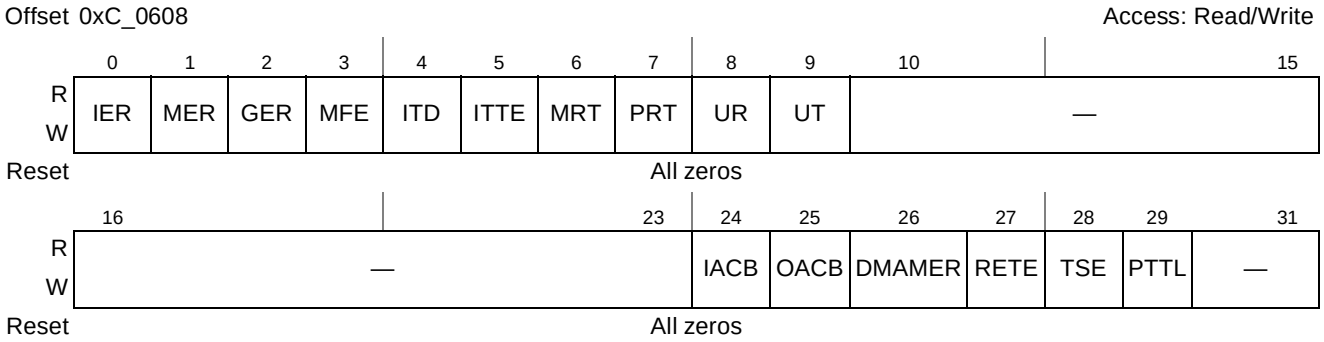


Figure 15-26. Logical/Transport Layer Error Detect Command and Status Register (LTLEDCSR)

Table 15-26 lists the LTLEDCSR fields.

Table 15-26. LTLEDCSR Field Descriptions

Bits	Name	Description
0	IER	IO error response. Received a response of ERROR for an IO logical layer request.
1	MER	Reserved for message error response. Received a response of ERROR for an MSG logical layer request. Error detected and captured in the message unit, if one exists.
2	GER	GSM error response. Received a response of ERROR for a GSM logical layer request.
3	MFE	Reserved for message format error. Received MESSAGE packet data payload with an invalid size or segment. Error detected and captured in the message unit, if one exists.
4	ITD	Illegal transaction decode. Received illegal fields in the request/response packet for a supported transaction (IO/MSG/GSM logical)
5	ITTE	Illegal transaction target error. Received a packet that contained a destination ID that is not defined for this end point when pass-through and accept-all are not enabled. Endpoints with multiple ports and a built-in switch function may not report this as an error (transport)
6	MRT	Reserved for message request time-out. A required message request has not been received within the specified time-out interval. Error detected and captured in the message unit, if one exists.
7	PRT	Packet response time-out. A required response has not been received within the specified time out interval (IO/MSG/GSM logical)
8	UR	Unsolicited response. An unsolicited/unexpected response packet was received (IO/MSG/GSM logical; only maintenance response for switches)
9	UT	Unsupported transaction. A transaction is received that is not supported in the destination operations CAR (IO/MSG/GSM logical; only maintenance port-write for switches)
10–23	—	Reserved
24	IACB	Inbound ATMU crossed boundary. A transaction is received that crosses an inbound ATMU boundary.

Table 15-26. LTLEDCSR Field Descriptions (continued)

Bits	Name	Description
25	OACB	Outbound ATMU crossed boundary. A transaction is being sent that crosses an outbound ATMU boundary, a segment boundary, or a subsegment boundary.
26	DMAMER	DMA message error response. An error response was received for a DMA message (detected in the RapidIO endpoint, not the message unit).
27	RETE	Retry error threshold exceeded. The allowed number of logical retries (given by LRETCR[RET] has been exceeded. This bit is also driven by the Message Unit when the allowed number of message retries has been exceeded.
28	TSE	Transport size error. The tt field is not consistent with bit 27 of the processing element features CAR (that is, the tt value is reserved or indicates a common transport system that is unsupported by this device).
29	PTTL	Packet time-to-live error. A packet time-to-live error occurred (a packet could not be successfully transmitted before the packet time-to-live counter expired).
30–31	—	Reserved

15.6.3.3 Logical/Transport Layer Error Enable Command and Status Register (LTLEECSR)

This register contains the bits that control whether an error condition locks the logical/transport layer error detect and capture registers and is reported to the system host. LTLEEDCSR, shown in Figure 15-27, is stored in all ports and the message unit.

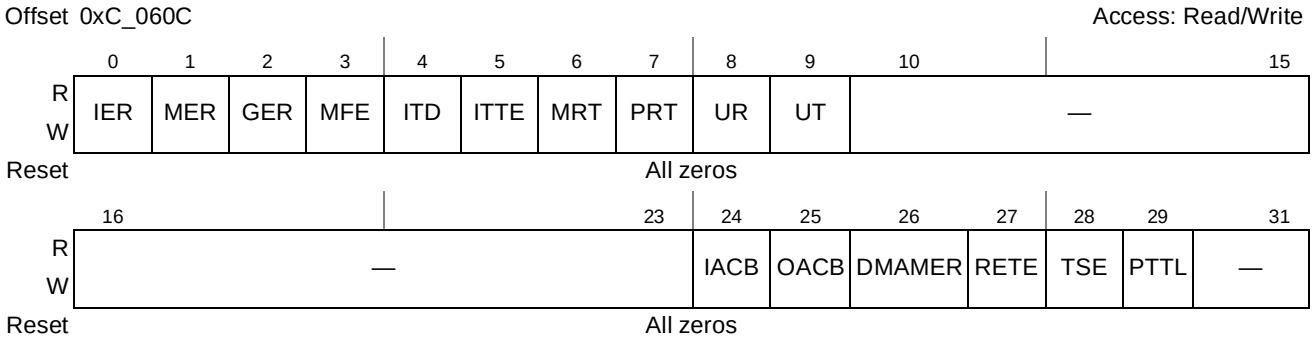


Figure 15-27. Logical/Transport Layer Error Enable Command and Status Register (LTLEECSR)

Table 15-27 lists the LTLEEDCSR fields.

Table 15-27. LTLEECSR Field Descriptions

Bits	Name	Description
0	IER	IO error response enable.Enable reporting of an IO error response. Capture and lock the error.
1	MER	Message error response enable. Enable reporting of a Message error response. Capture and lock the error (capture done in Message Unit, if one exists).

Table 15-27. LTLEECSR Field Descriptions (continued)

Bits	Name	Description
2	GER	GSM error response enable. Enable reporting of a GSM error response. Capture and lock the error.
3	MFE	Message format error enable. Enable reporting of a message format error. Capture and lock the error.(capture done in Message Unit, if one exists).
4	ITD	Illegal transaction decode enable. Enable reporting of an illegal transaction decode error. Capture and lock the error.
5	ITTE	Illegal transaction target error enable. Enable reporting of an illegal transaction target error. Capture and lock the error.
6	MRT	Message request time-out enable. Enable reporting of a Message Request time-out error. Capture and lock the error. (capture done in Message Unit, if one exists)
7	PRT	Packet response time-out error enable. Enable reporting of a packet response time-out error. Capture and lock the error.
8	UR	Unsolicited response error enable. Enable reporting of an unsolicited response error. Capture and lock the error.
9	UT	Unsupported transaction error enable. Enable reporting of an unsupported transaction error. Capture and lock the error.
10–23	—	Reserved
24	IACB	Inbound ATMU crossed boundary error enable. Enable reporting of a received transaction that crosses an inbound ATMU boundary. Capture and lock the error.
25	OACB	Outbound ATMU crossed boundary error enable. Enable reporting of a transaction that crosses an outbound ATMU boundary, a segment boundary, or a subsegment boundary. Capture and lock the error.
26	DMAMER	DMA message error response. Enable error reporting of an error response for a DMA message. Capture and lock the error.
27	RETE	Retry error threshold exceeded. Enable error reporting when the allowed number of logical retries has been exceeded.
28	TSE	Transport size error. Enable error reporting when the tt field is not consistent with bit 27 of the Processing Element Features CAR (that is, the tt value is reserved or indicates a common transport system that is unsupported by this device).
29	PTTL	Packet time-to-live error. Enable reporting of a packet time-to-live time-out error. Capture and lock the error.
30–31	—	Reserved

15.6.3.4 Logical/Transport Layer Address Capture Command and Status Register (LTLACCSR)

The LTLACCSR provides error information. It is locked when a logical/transport error is detected, and the corresponding enable bit is set. LTLACCSR is stored in each port and in the message unit, although the values in this register can differ between each port and message unit. The message unit LTLACCSR cannot lock if any port has locked; no port LTLACCSR can lock if the message unit or any other port has locked.

Undefined results occur if this register is written while actual logical/transport layer errors are being detected by the port.

Note that fields in this register can be set by to writing this register. This can be used to emulate a hardware error during software development. However, undefined results occur if fields in this register are set while an actual logical/transport layer error is being detected.

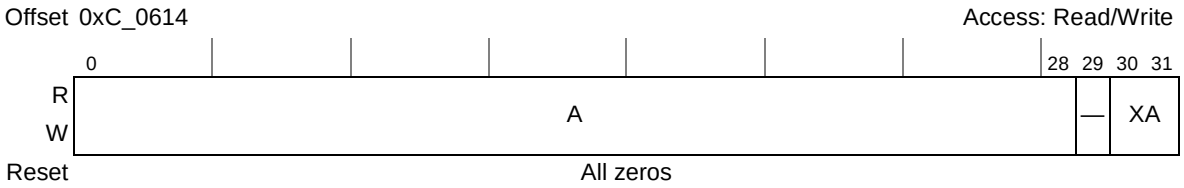


Figure 15-28. Logical/Transport Layer Address Capture Command and Status Register (LTLACCSR)

Table 15-28. LTLACCSR Field Descriptions

Bits	Name	Description
0–28	A	Normally the least significant 29 bits of the address associated with the error (for requests, for responses if available). Please see Section 15.8.12.3, “Logical Layer Errors and Error Handling,” for details.
29	—	Reserved
30–31	XA	xamsbs. Normally the extended address bits of the address associated with the error (for requests, responses, if available). Please see Section 15.8.12.3, “Logical Layer Errors and Error Handling,” for details.

15.6.3.5 Logical/Transport Layer Device ID Capture Command and Status Register (LTLDIDCCSR)

LTLDIDCCSR contains error information. It is locked when a logical/transport error is detected, and the corresponding enable bit is set. LTLDIDCCSR is stored in each port and in the message unit, although the values in this register can differ between each port and message unit. The message unit LTLDIDCCSR cannot lock if any port has locked; no port LTLDIDCCSR can lock if the message unit or any other port has locked. Undefined results occur if this register is written while actual logical/transport layer errors are being detected by the port.

Note that fields in this register can be set by writing to this register, in order to emulate a hardware error during software development. However, undefined results occur if fields in this register are set while an actual logical/transport layer error is being detected.

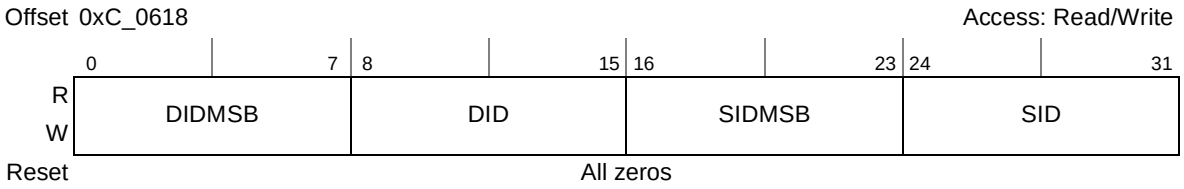


Figure 15-29. Logical/Transport Layer Device ID Capture Command and Status Register (LTLDIDCCSR)

Table 15-29. LTLDIDCCSR Field Descriptions

Bits	Name	Description
0–7	DIDMSB	Normally the most significant byte of the destinationID associated with the error. This field is valid only if bit 27 of the Processing Element Features CAR is set (large transport systems only). Please see Section 15.8.12.3, “Logical Layer Errors and Error Handling,” for details.
8–15	DID	Normally the destinationID (or least significant byte of the destination ID if large transport system) associated with the error. Please see Section 15.8.12.3, “Logical Layer Errors and Error Handling,” for details.
16–23	SIDMSB	Normally the most significant byte of the sourceID associated with the error. This field is valid only if bit 27 of the Processing Element Features CAR is set (large transport systems only). Please see Section 15.8.12.3, “Logical Layer Errors and Error Handling,” for details.
24–31	SID	Normally the sourceID (or least significant byte of the source ID if large transport system) associated with the error. Please see Section 15.8.12.3, “Logical Layer Errors and Error Handling,” for details.

15.6.3.6 Logical/Transport Layer Control Capture Command and Status Register (LTLCCCSR)

LTLCCCSR contains error information. It is stored in each port and in the message unit, although the values in this register can differ between each port and message unit. The message unit LTLCCCSR cannot lock if any port has locked; no port LTLCCCSR can lock if the message unit or any other port has locked. Undefined results occur if this register is written while actual logical/transport layer errors are being detected by the port.

Note that fields in this register can be set by writing to this register, in order to emulate a hardware error during software development. However, undefined results occur if fields in this register are set while an actual logical/transport layer error is being detected.

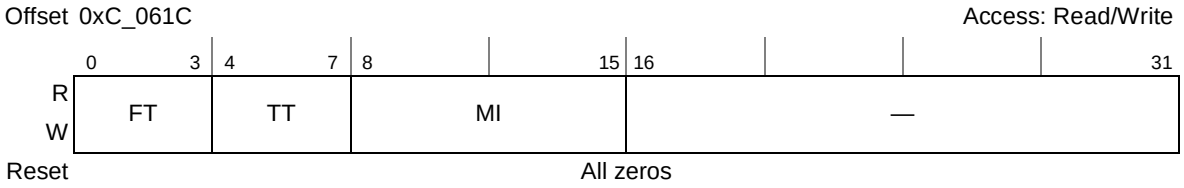


Figure 15-30. Logical/Transport Layer Control Capture Command and Status Register (LTLCCCSR)

Table 15-30. LTLCCCSR Field Descriptions

Bits	Name	Description
0–3	FT	Normally the format type associated with the error. Please see Section 15.8.12.3, “Logical Layer Errors and Error Handling,” for details.
4–7	TT	Normally the transaction type associated with the error. Please see Section 15.8.12.3, “Logical Layer Errors and Error Handling,” for details.
8–15	MI	Normally the message Information: letter, mbox, and msgseg for the last message request received for the mailbox that had an error (message errors only). Please see Section 15.8.12.3, “Logical Layer Errors and Error Handling,” for details.
16–31	—	Reserved

15.6.4 RapidIO Extended Features Space—Error Reporting Physical Registers

15.6.4.1 Error Detect Command and Status Register (EDCSR)

The error detect command and status register (EDCSR), shown in Figure 15-31, indicates transmission errors that are detected by the hardware. Software can write bits in this register with 1 to cause the Error Rate Counter to increment. Undefined results occur if this register is written while actual physical layer errors are being detected by the port.

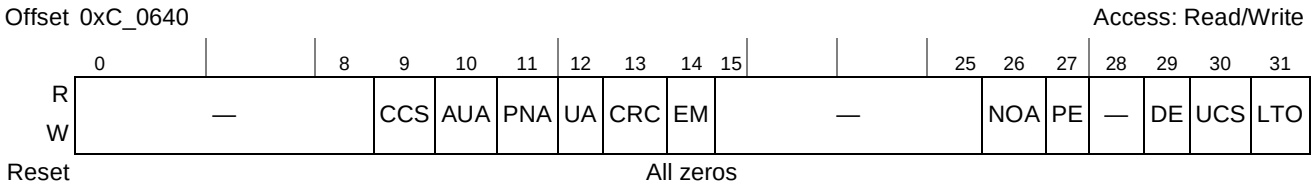


Figure 15-31. Port *n* Error Detect Command and Status Register (EDCSR)

Table 15-31. EDCSR Field Descriptions

Bits	Name	Description
0–8	—	Reserved
9	CCS	Received a control symbol with a bad CRC value.
10	AUA	Received acknowledge control symbol with unexpected ackID (packet-accepted or packet-retry).
11	PNA	Received packet-not-accepted acknowledge control symbol
12	UA	Received packet with unexpected ackID value.
13	CRC	Received a packet with a bad CRC value
14	EM	Received packet which exceed the maximum allowed size (276 bytes).
15–25	—	Reserved
26	NOA	Link-response received with an ackID that is not outstanding.
27	PE	Protocol Error: An unexpected packet or control symbol was received.
28	—	Reserved
29	DE	Received unaligned /SC/ or /PD/ or undefined code-group.
30	UCS	An unexpected acknowledge control symbol was received.
31	LTO	An acknowledge or link-response control symbol is not received within the specified time-out interval.

15.6.4.2 Error Rate Enable Command and Status Register (ERECSR)

This register contains the bits that control when an error condition is allowed to increment the error rate counter in the error rate threshold register and lock the error capture registers.

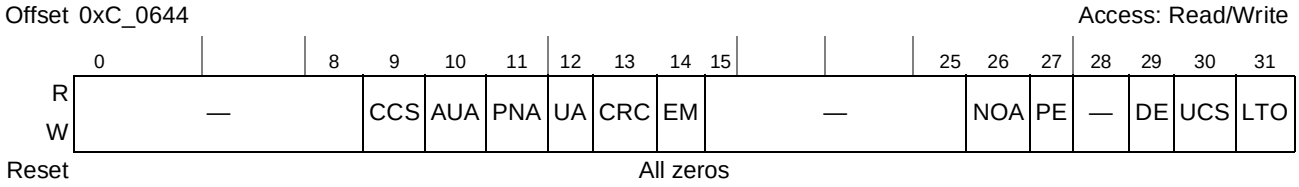


Figure 15-32. Error Rate Enable Command and Status Register (ERECSR)

Table 15-32. ERECSR Field Descriptions

Bits	Name	Description
0–8	—	Reserved
9	CCS	Enable error rate counting of a corrupt control symbol
10	AUA	Enable error rate counting of an acknowledge control symbol with an unexpected ackID
11	PNA	Enable error rate counting of received packet-not-accepted control symbols.
12	UA	Enable error rate counting of packet with unexpected ackID value.
13	CRC	Enable error rate counting of packet with a bad CRC value.
14	EM	Enable error rate counting of packet which exceeds the maximum allowed size
15–25	—	Reserved
26	NOA	Enable error rate counting of link-responses received with an ackID that is not outstanding.
27	PE	Enable error rate counting of protocol errors
28	—	Reserved
29	DE	Enable error rate counting of delineation errors.
30	UCS	Enable error rate counting of unsolicited acknowledge control symbol errors.
31	LTO	Enable error rate counting of link time-out errors.

15.6.4.3 Error Capture Attributes Command and Status Register (ECACSR)

The error capture attribute register indicates the type of information contained in the error capture registers. In the case of multiple detected errors during the same clock cycle one of the errors must be reflected in the Error type field. Undefined results occur if this register is written while actual physical layer errors are being detected by the port. Software should check that the ECACSR[CVI] bit is set before reading the capture registers to ensure that the error has been properly captured.

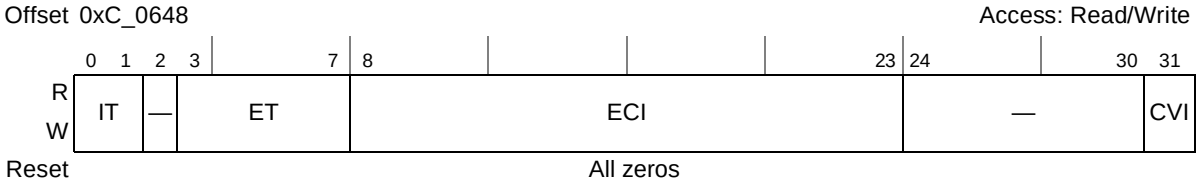


Figure 15-33. Error Capture Attributes Command and Status Register (ECACSR)

Table 15-33. ECACSR Field Descriptions

Bits	Name	Description
0–1	IT	Type of information logged: 00 Packet (error capture registers hold the first 4 words of the packet, or the entire packet if it is less than 4 words long). 01 Control symbol (only error capture register 0 is valid) 10 Reserved 11 Undefined (not clearly a control symbol or packet error. Error capture registers hold the symbol that caused the error and the next 3 symbols.)
2	—	Reserved
3–7	ET	The encoded value of the bit in the error detect CSR that describes the error captured in the error capture CSRs
8–23	ECI	Extended capture information [0:15]. ECI contains the control/data character signal corresponding to each byte of captured data. Each ECI bit reflects the validity of captured data. If a bit is set, then the designated byte of captured data is valid. If a bit is cleared, then the designated byte of the specified register does not contain valid data and should be disregarded until the bit is set. ECI[0] reflects validity of PCSECCSR0[0:7] ECI[1] reflects validity of PCSECCSR0[8:15] ECI[2] reflects validity of PCSECCSR0[16:23] ECI[3] reflects validity of PCSECCSR0[24:31] ECI[4] reflects validity of PECCSR1[0:7] ECI[5] reflects validity of PECCSR1[8:15] ... ECI[14] reflects validity of PECCSR3[16:23] ECI[15] reflects validity of PECCSR3[24:31]
24–30	—	Reserved
31	CVI	This bit is set by hardware to indicate that the Packet/control symbol capture registers contain valid information. For control symbols, only capture register 0 contains meaningful information.

15.6.4.4 Packet/Control Symbol Error Capture Command and Status Register 0 (PCSECCSR0)

ECACSR contains the first four bytes of captured packet symbol information or a control character and control symbol. Undefined results occur if this register is written while actual physical layer errors are being detected by the port. Software should check that the ECACSR[CVI] bit is set before reading the capture registers to ensure that the error has been properly captured.

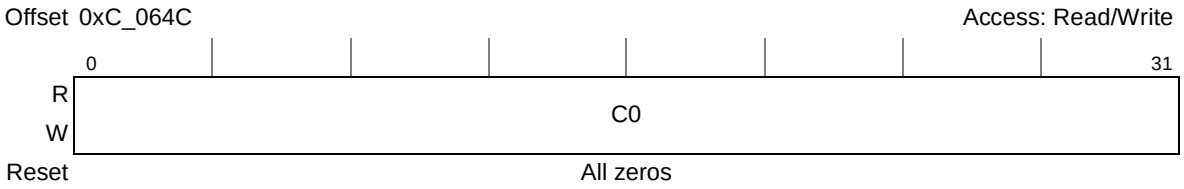


Figure 15-34. Packet/Control Symbol Error Capture Command and Status Register 0 (PCSECCSR0)

Table 15-34. PCSECCSR0 Field Descriptions

Bits	Name	Description
0–31	C0	Capture 0: Control Character and control symbol or bytes 0 to 3 of packet header.

15.6.4.5 Packet Error Capture Command and Status Register 1 (PECCSR1)

Error capture register 1 contains bytes 4 through 7 of the packet header. Undefined results occur if this register is written while actual physical layer errors are being detected by the port. Software should check that the ECACSR[CVI] bit is set before reading the capture registers to ensure that the error has been properly captured.

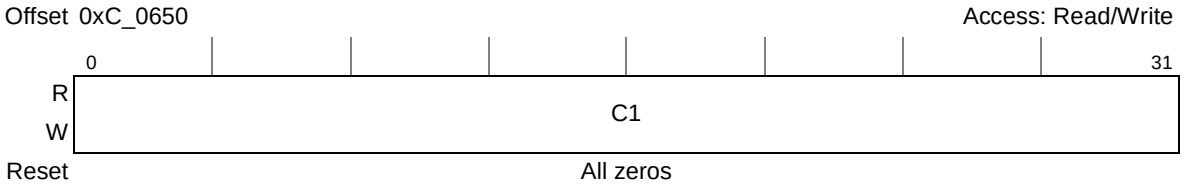


Figure 15-35. Packet Error Capture Command and Status Register 1 (PECCSR1)

Table 15-35. PECCSR1 Field Descriptions

Bits	Name	Description
0–31	C1	Capture 1. Bytes 4 to 7 of the packet header

15.6.4.6 Packet Error Capture Command and Status Register 2 (PECCSR2)

Error capture register 2 contains bytes 8 through 11 of the packet header. Undefined results occur if this register is written while actual physical layer errors are being detected by the port. Software should check that the ECACSR[CVI] bit is set before reading the capture registers to ensure that the error has been properly captured.

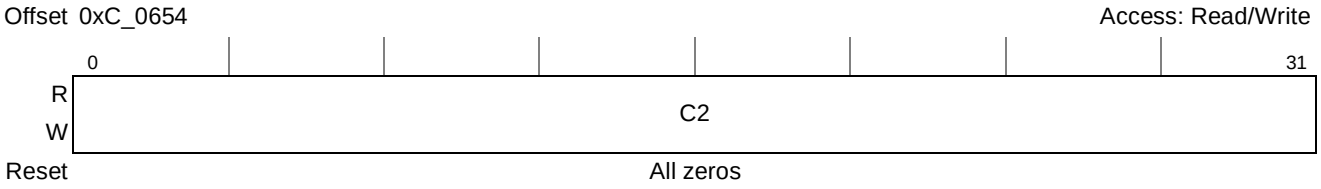


Figure 15-36. Packet Error Capture Command and Status Register 2 (PECCSR2)

Table 15-36. PECCSR2 Field Descriptions

Bits	Name	Description
0–31	C2	Capture 2. Bytes 8 to 11 of the packet header

15.6.4.7 Packet Error Capture Command and Status Register 3 (PECCSR3)

Error capture register 3 contains bytes 12 through 15 of the packet header. Undefined results occur if this register is written while actual physical layer errors are being detected by the port. Software should check that the ECACSR[CVI] bit is set before reading the capture registers to ensure that the error has been properly captured.

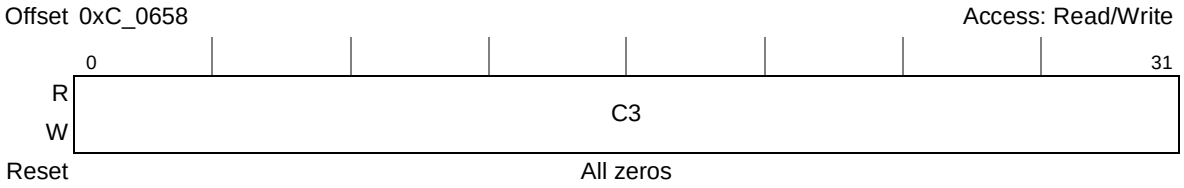


Figure 15-37. Packet Error Capture Command and Status Register 3 (PECCSR3)

Table 15-37. PECCSR3 Field Descriptions

Bits	Name	Description
0–31	C3	Capture 3. Bytes 12 to 15 of the packet header

15.6.4.8 Error Rate Command and Status Register (ERCSR)

The error rate register is a 32-bit register used with the error rate threshold register to monitor and control the reporting of transmission errors.

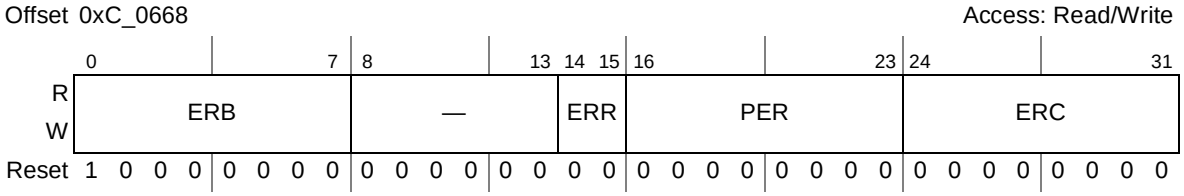


Figure 15-38. Error Rate Command and Status Register (ERCSR)

Table 15-38. ERCSR Field Descriptions

Bits	Name	Description
0–7	ERB	These bits provide the error rate bias value. 0x00 Do not decrement the error rate counter 0x01 Decrement every 1 ms (±34%) 0x02 Decrement every 10 ms (±34%) 0x04 Decrement every 100 ms (±34%) 0x08 Decrement every 1 s (±34%) 0x10 Decrement every 10 s (±34%) 0x20 Decrement every 100 s (±34%) 0x40 Decrement every 1000 s (±34%) 0x80 Decrement every 10000 s (±34%) Other values are reserved and cause undefined operation.
8–13	—	Reserved

Table 15-38. ERCSR Field Descriptions (continued)

Bits	Name	Description
14–15	ERR	These bits limit the incrementing of the error rate counter above the failed threshold trigger. 0b00 Only count 2 errors above 0b01 Only count 4 errors above 0b10 Only count 16 error above 0b11 Do not limit incrementing the error rate count Note that the Error Rate Counter never increments above 0xFF, even if the combination of the settings of ERR and the failed threshold trigger might imply that it might.
16–23	PER	Peak error rate. Contains the peak value attained by the error rate counter
24–31	ERC	Error rate counter. These bits maintain a count of the number of transmission errors that have been detected by the port, decremented by the Error Rate Bias mechanism, to create an indication of the link error rate. Software should not attempt to write this field to a value higher than failed threshold trigger plus the number of errors specified in the ERR field (the maximum ERC value).

15.6.4.9 Error Rate Threshold Command and Status Register (ERTCSR)

The error rate threshold register is a 32-bit register used to control the reporting of the link status to the system host.

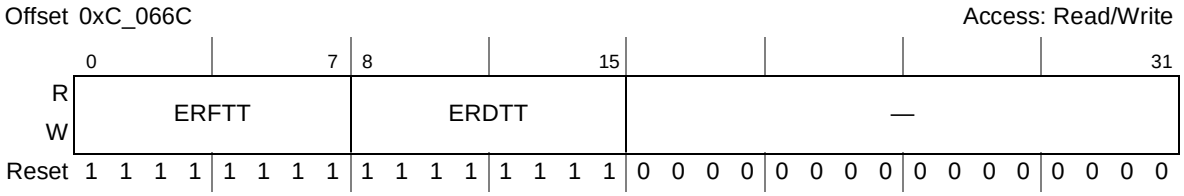


Figure 15-39. Error Rate Threshold Command and Status Register (ERTCSR)

Table 15-39. ERTCSR Field Descriptions

Bits	Name	Description
0–7	ERFTT	Error rate failed threshold trigger. These bits provide the threshold value for reporting an error condition due to a possibly broken link. 0x00 Disable the Error Rate Failed Threshold Trigger. 0x01 Set the error reporting threshold to 1. 0x02 Set the error reporting threshold to 2. ... 0xFF Set the error reporting threshold to 255. The ESCSR[OFE] bit is not set if ERFTT is written to a value lower than or equal to the ERCSR[ERC].
8–15	ERDTT	Error rate degraded threshold trigger. These bits provide the threshold value for reporting an error condition due to a degrading link. 0x00 Disable the Error Rate Degraded Threshold Trigger. 0x01 Set the error reporting threshold to 1. 0x02 Set the error reporting threshold to 2. ... 0xFF Set the error reporting threshold to 255. The ESCSR[ODE] bit is not set if ERDTT is written to a value lower than or equal to the ERCSR[ERC].
16–31	—	Reserved

15.6.5 RapidIO Implementation Space Registers

15.6.5.1 Logical Layer Configuration Register (LLCR)

The logical layer configuration register contains general port-common logical layer mode enables.

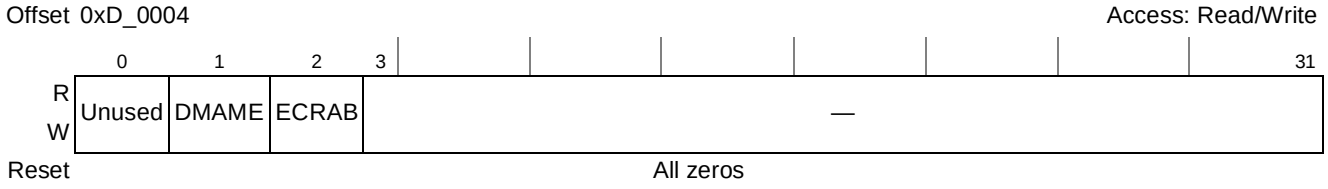


Figure 15-40. Logical Layer Configuration Register (LLCR)

Table 15-40. LLCR Field Descriptions

Bits	Name	Description
0	—	This bit is unused. It is readable and writable.
1	DMAME	DMA message enable (for outbound DMA messages). 0 Message Unit messages can be assigned to letter 0,1,2,3. No DMA Messages. 1 DMA messages are assigned letter 3, Message Unit messages 0,1,2
2	ECRAB	External configuration register access block. When set, all maintenance requests and accesses that hit LCSBA1CSR are blocked; reads return all 0's, and writes are ignored (both return done response). When clear, any external RapidIO device can access registers.
3–31	—	Reserved

15.6.5.2 Error/Port-Write Interrupt Status Register (EPWISR)

The EPWISR register contains status bits of the interrupts that have been generated by any port or the message unit for physical or logical/transport layer errors or inbound port-writes. Because errors from all ports are reported to the core with one interrupt signal, this register provides the core with quick access to where the error occurred. This register is read only and is stored in each port and the message unit as a logically equivalent copy.

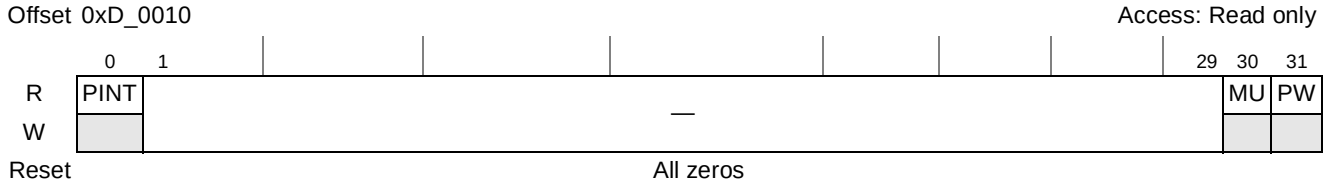


Figure 15-41. Error/Port-Write Interrupt Status Register (EPWISR)

Table 15-41. EPWISR Field Descriptions

Bits	Name	Description
0	PINT	A physical or logical/transport error interrupt was generated.
1–29	—	Reserved (can be used to indicate error on more ports if they exist).
30	MU	A logical/transport layer error interrupt was generated in the message unit.
31	PW	An inbound port-write was received.

15.6.5.3 Logical Retry Error Threshold Configuration Register (LRETCR)

The LRETCR register contains the retry error threshold for the logical layer. When the number of consecutive logical retries for a given packet is greater than to this value, an error interrupt is generated. Note that the number of retries must be greater than this value unlike other registers that define a retry threshold.

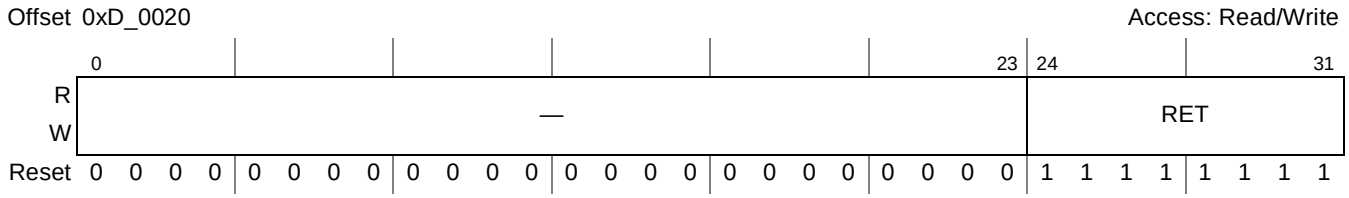


Figure 15-42. Logical Retry Error Threshold Configuration Register (LRETCR)

Table 15-42. LRETCR Field Descriptions

Bits	Name	Description
0–23	—	Reserved
24–31	RET	Retry error threshold. These bits provide the threshold value for the number of consecutive logical retries (for GSM responses) received for a given packet that causes RAPIDIO ENDPOINT to report an error condition. 0x00 Disable the RET 0x01 Set the error reporting threshold to 1 ... 0xFF Set the error reporting threshold to 255

15.6.5.4 Physical Retry Error Threshold Configuration Register (PRETCR)

The PRETCR register contains the retry error threshold for the physical layer. When the number of consecutive ACK-retries is greater than or equal to this value, an error interrupt is generated.

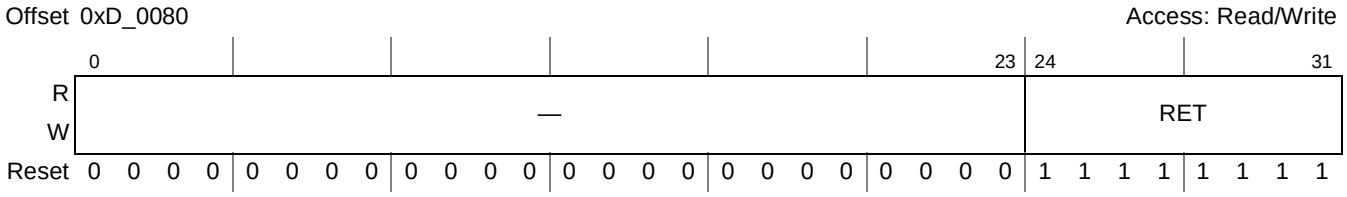


Figure 15-43. Physical Retry Error Threshold Configuration Register (PRETCR)

Table 15-43. PRETCR Field Descriptions

Bits	Name	Description
0–23	—	Reserved
24–31	RET	Retry error threshold. These bits provide the threshold value for the number of consecutive ACK retries received that causes RAPIDIO ENDPOINT to report an error condition. 0x00 Disable the RET 0x01 Set the error reporting threshold to 1 ... 0xFF Set the error reporting threshold to 255

15.6.5.5 Alternate Device ID Command and Status Register (ADIDCSR)

The alternate device id CSR contains an alternate deviceID. It is intended that this register should be enabled before the master enabled bit of the GCCSR is set, such that when it is enabled, all other devices in the RapidIO system (including switches) send packets to and receive packets from the deviceID contained in this register, instead of the deviceID contained in BDIDCSR.

When the alternate deviceID is enabled, the inbound RapidIO endpoint only accepts packets sent with the deviceID contained in ADIDCSR or with the deviceID contained in BDIDCSR (except during Accept All mode, during which the inbound RapidIO endpoint accepts packets using the same common transport system). In addition, the outbound RapidIO endpoint only generates requests using the deviceID contained in ADIDCSR; it generates responses with the deviceID given in the original request packet (either from ADIDCSR or BDIDCSR).

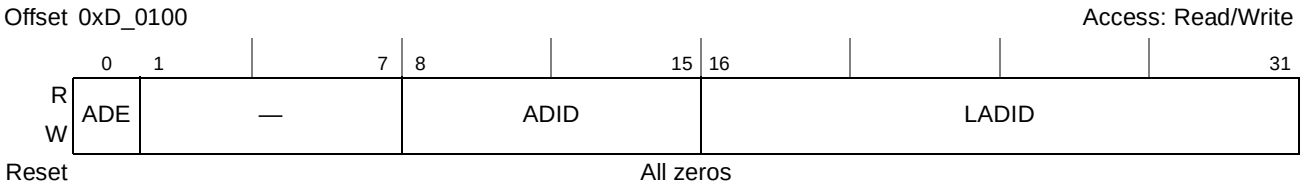


Figure 15-44. Alternate Device ID Command and Status Register (ADIDCSR)

Table 15-44. ADIDCSR Field Descriptions

Bits	Name	Description
0	ADE	Alternate device ID enable When set, this bit causes the port to use the deviceID specified in this register instead of the deviceid specified in BDIDCSR
1–7	—	Reserved
8–15	ADID	Alternate device ID for the device in a small transport system
16–31	LADID	Alternate device ID for the device in a large transport system

15.6.5.6 Accept-All Configuration Register (AACR)

The accept-all configuration register contains information on accept-all mode.

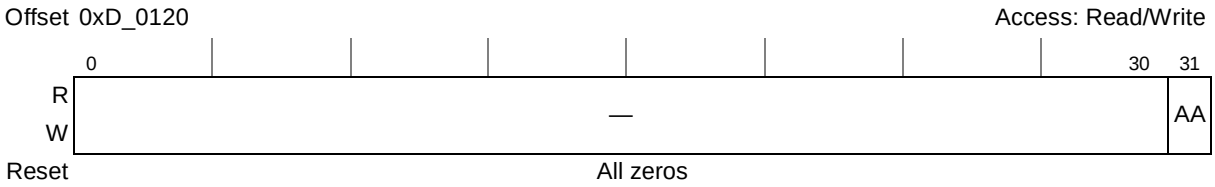


Figure 15-45. Accept-All Configuration Register (AACR)

Table 15-45. AACR Field Descriptions

Bits	Name	Description
0–30	—	Reserved
31	AA	Accept all. 1 All packets are accepted without checking the target ID. However, the tt field must be consistent with the common transport system specified by bit 27 of the processing element features CAR. 0 Normal RapidIO acceptance based on target ID.

15.6.5.7 Logical Outbound Packet Time-to-Live Configuration Register (LOPTTLCR)

The logical outbound packet time-to-live configuration register, shown in Figure 15-46, contains the time-to-live count for all ports on a device. This packet time-to-live counter starts when a packet is ready to be transmitted. If the packet is not successfully transmitted before the timer expires, the packet is discarded. Successfully transmitted means that a packet accept was received for the packet on the RIO interface. If the packet requires a response, an internal error response is returned after the response

time-out occurs (PRTOCCSR). The packet time-to-live counter prevents the local processor from being stalled when packets cannot be successfully transmitted (acknowledged with an accept by the link partner at the physical level). The value of this register should always be larger than the link time-out value (PLTOCCSR). The reset value is the maximum time-out interval. The timer decrements at one-half the platform clock rate; thus at a platform clock rate of 400 MHz, for example, the maximum time-out value is approximately 83.9 ms.

When the packet time-to-live counter expires, PCR[OBDEN] is automatically set. PCR[OBDEN] must be cleared by software.

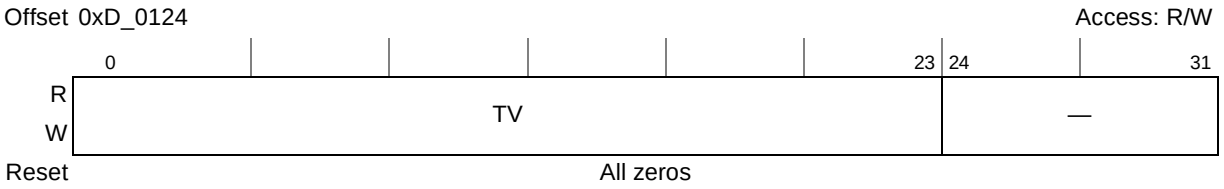


Figure 15-46. Logical Outbound Packet Time-to-Live Configuration Register (LOPTTLCR)

Table 15-46 lists LOPTTLCR fields.

Table 15-46. LOPTTLCR Field Descriptions

Bits	Name	Description
0–23	TV	Time-out value. Setting to all zeros disables the time-to-live time-out timer. This value is loaded each time the time-to-live time-out timer starts.
24–31	—	Reserved

15.6.5.8 Implementation Error Command and Status Register (IECSR)

The IECSR register contains status bits that are asserted whenever an implementation-defined error occurs.

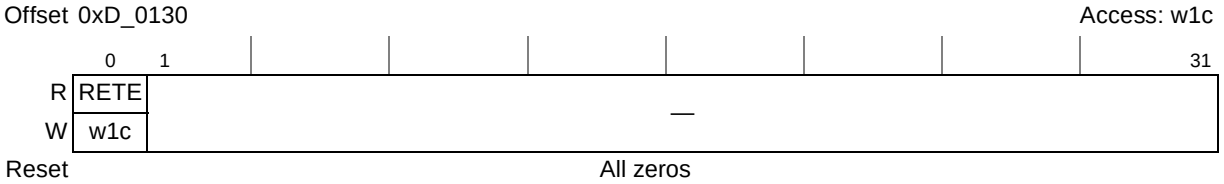


Figure 15-47. Implementation Error Command and Status Register (IECSR)

Table 15-47. IECSR Field Descriptions

Bits	Name	Description
0	RETE	Retry error threshold exceeded. This bit is asserted when the number of consecutive retries has reached retry error threshold in the retry error threshold register. This bit is cleared by writing a 1 to it. This bit sets again if another retry is received and the number of consecutive retries continues to exceed the retry error threshold.
1–31	—	Reserved

15.6.5.9 Physical Configuration Register (PCR)

The PCR contains general physical layer protocol and link mode enables.

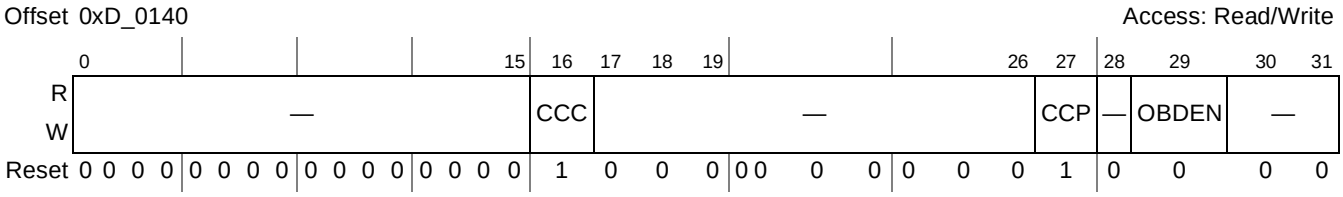


Figure 15-48. Physical Configuration Register (P*n*PCR)

Table 15-48. PCR Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16	CCC	CRC checking enable - control symbol. When set, CRC is checked on received control symbols. When cleared, no CRC is checked on received control symbols.
17-18	—	Reserved
19–26	—	Reserved
27	CCP	CRC checking enable - packet. When set, CRC is checked on received packets. When cleared, no CRC is checked on received packets
28	—	Reserved
29	OBDEN	Output buffer drain enable. When set, the output drains packets from the outbound buffer and does not send them out. This intentionally causes the inbound to time-out (when a response on a drained request was expected) and send an error response to OCN. A packet time-to-live time-out causes this bit to be set. (See Section 15.6.5.7, “Logical Outbound Packet Time-to-Live Configuration Register (LOPTTLCR).”)
30–31	—	Reserved

15.6.5.10 Serial Link Command and Status Register (SLCSR)

The SLCSR contains status of the of the serial physical link.

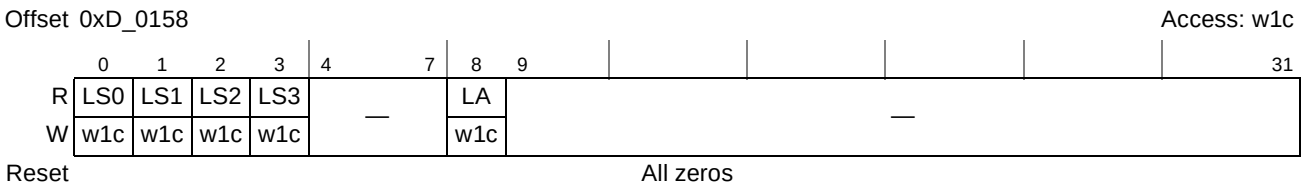


Figure 15-49. Serial Link Command and Status Register (SLCSR)

Table 15-49. SLCSR Field Descriptions

Bits	Name	Description
0	LS0	Lane sync achieved for lane 0. Write with 1 to clear
1	LS1	Lane sync achieved for lane 1. Write with 1 to clear
2	LS2	Lane sync achieved for lane 2. Write with 1 to clear
3	LS3	Lane sync achieved for lane 3. Write with 1 to clear.
4–7	—	Reserved
8	LA	Lane alignment achieved. Write with 1 to clear.
9–31	—	Reserved

15.6.5.11 Serial Link Error Injection Configuration Register (SLEICR)

The SLEICR is used to control the injection of bit errors into the transmit bit stream.

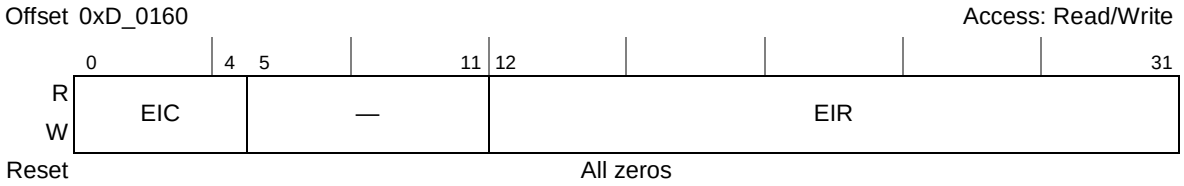


Figure 15-50. Serial Link Error Injection Configuration Register (SLEICR)

Table 15-50. SLEICR Field Descriptions

Bits	Name	Description
0–4	EIC	Error injection control. Enables and controls serial link error injection as follows: 00000 Error injection is disabled. 10000 Error injection, lane 0 only 01000 Error injection, lane 1 only 00100 Error injection, lane 2 only 00010 Error injection, lane 3 only 11110 Error injection, all 4 lanes simultaneously 11111 Error injection, randomly distributed over all 4 lanes All other values are reserved.
5–11	—	Reserved
12–31	EIR	Error injection range. The value of EIR × 32 determines the maximum value of the pseudo-random delay between errors. For example, a value of 0x1 would indicate a maximum delay of 32 character times. Value within this register should be right-justified.

The SLEICR register is used to generate pseudo-random errors into the outbound serial RapidIO data stream. If the EIC field is any of the allowable non-zero values (as shown in the table above), then error injection is enabled for one lane or all four lanes, as selected by the EIC value. When enabled, at

pseudo-random intervals, an error is injected by inverting a single bit in the outgoing data stream. This occurs only in the lane(s) that have error injection enabled. The range of the pseudo-random value (delay between injected errors) is controlled by the EIR field, as described above. That is, the value of EIR, multiplied by 32, determines the maximum number of character times between injected errors.

15.6.6 Revision Control Registers

15.6.6.1 IP Block Revision Register 1 (IPBRR1)

IP block revision register 1 is used to track changes and revisions of the RapidIO endpoint.

IP block revision register 1 is a read-only register.

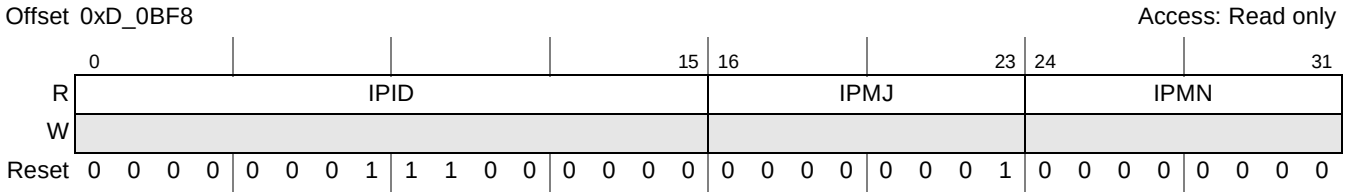


Figure 15-51. IP Block Revision Register 1 (IPBRR1)

Table 15-51. IPBRR1 Field Descriptions

Bits	Name	Description
0–15	IPID	IP block ID = 0x01C0
16–23	IPMJ	Major revision of the IP block = 0x01
24–31	IPMN	Minor revision of the IP block = 0x0

15.6.6.2 IP Block Revision Register 2 (IPBRR2)

IP block revision register 2 is used to track changes and revisions of the RapidIO endpoint.

IP block revision register 2 is a read-only register.

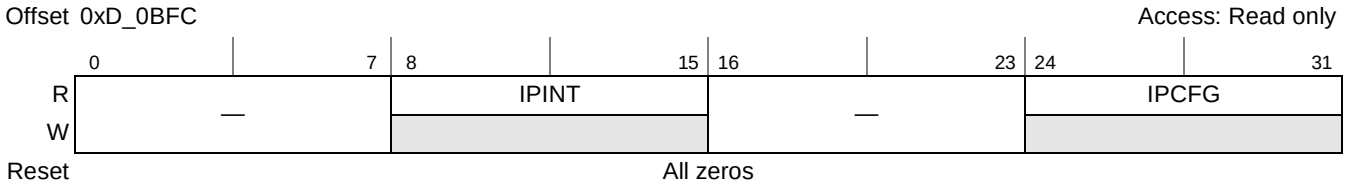


Figure 15-52. IP Block Revision Register 2 (IPBRR2)

Table 15-52. IPBRR2 Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–15	IPBID	IP block Integration options = 0x0
16–23	—	Reserved
24–31	IPCFG	IP block Configuration options = 0x0

15.6.7 RapidIO Implementation Space—ATMU Registers

ATMU registers are used for outbound and inbound transactions. Their purpose is to translate RapidIO packets to OCN packets on inbound and to translate OCN packets to RapidIO packets on outbound. ATMU window misses use the window 0 register set by default, and overlapping window hits results in the use of the lowest number window register set hit. For both inbound and outbound translation, the smallest window size is 4K and the largest window size is 16G for inbound translation and 64G for outbound translation. The default window register set causes no translation of the transaction address for inbound transactions since the RapidIO address space has 34 bits and the OCN address space has 36 bits. For outbound transactions, the default window maps each of the four 16G chunks to the RapidIO 16G address space. The inbound and outbound translation windows must be aligned based on the granularity selected by the size fields. The packet device ID fields are not used in the inbound translation process, only the address field.

The RapidIO endpoint implementation allows up to a 34-bit (0:33) RapidIO address and a 36-bit (0:35) OCN address. In a device confined to 32-bit OCN addresses, the top 4 bits (0:3) of the Inbound translation address and the Outbound base address should be set to all 0’s; setting them otherwise results in undefined behavior.

As is the case with all registers, an external processor writing the ATMU registers should not assume that the write has completed until a response is received.

Note that when booting from serial RapidIO, outbound ATMU window 0 must be used.

15.6.7.1 Segmented Outbound Window Description

All outbound windows have the capability to have 2 or 4 segments, all of which are equal in size, numbered 0 and 1, or 0 through 3, respectively (the standard 8540 unsegmented window definition becomes segment 0). Each segment assigns attributes and the target deviceID for an outbound transaction. All segments of a window translate to the same translation address in the target.

Additionally, each segment can be set up with 2, 4, or 8 subsegments, all of which are equal in size. These subsegments allow a single segment to target a number of numerically adjacent target device IDs, and, again, they all translate to the same translation address in the targets. For example, a segment with 8 subsegments can be configured to generate a transaction with the same set of attributes to target deviceIDs 0, 1, 2, 3, 4, 5, 6, or 7, depending on which subsegment is addressed.

Note that subsegments are only supported when multiple segments are chosen.

This allows a window to be configured so that aliases can be created to the same offset within the target device so that a single window can be used to generate different transaction types. Without segmented windows, achieving the equivalent behavior would require multiple windows. Figure 15-53 shows an example of this capability. A window is defined to be 4kB in size, and is defined to have 4 segments and no subsegments. Each segment is assigned to target deviceID 0x05, and each segment is given a different write transaction type attribute - segment 0 is assigned NWRITE, segment 1 is assigned SWRITE, segment 2 is assigned NWRITE_R, and segment 3 is assigned FLUSH. Since all of the segments are assigned to target the same device, by writing to the same offset in each segment, a different write transaction can be generated to the target to the same offset in the target.

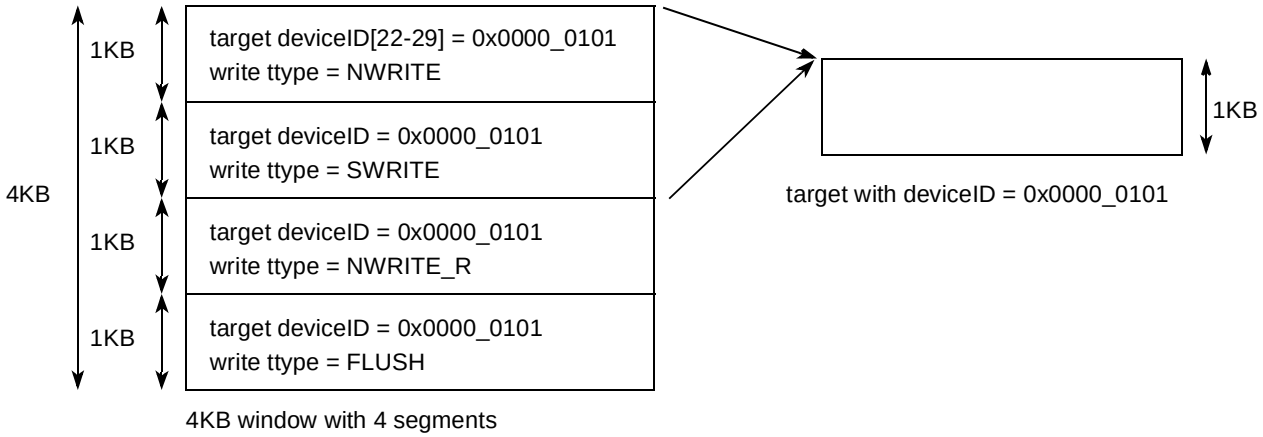


Figure 15-53. Example of Attribute Aliasing

So, writing to offset 0x0 in segment 0 is translated (as defined in the translation address registers) and generates a NWRITE transaction to offset 0x0 in a 1KB window in the target with deviceID = 0x05. A write to offset 0x0 in segment 2 is also translated, to the same offset in the target device as the write to segment 0, but this time a NWRITE_R transaction is generated.

Another use is that the same window can be used to target multiple devices with the same translation offset. Without segmented, (and subsegmented) windows, achieving the equivalent behavior would require multiple windows. Figure 15-54 shows an example of this multi-targeting. For example, a 4kB window is set up with 2 segments of 2 subsegments. Each segment is assigned a write ttype of NWRITE, but each segment and subsegment has a different target deviceID. Segments 0 and 1 are assigned target deviceIDs 4 and 5, and 8 and 9, respectively.

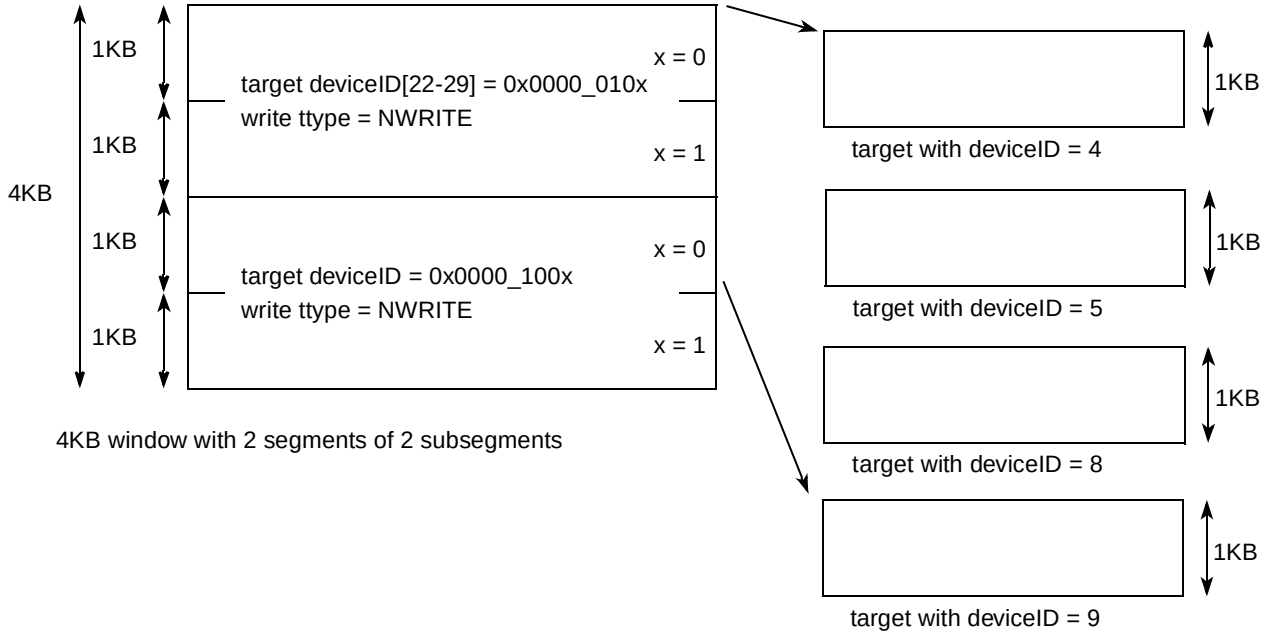


Figure 15-54. Example of Multi-Targeting

In this example, a write to offset 0x0 in segment 0 is translated as defined, and a NWRITE transaction is generated targeted to deviceID 4. A corresponding write to segment 1 to offset 0x400 is also translated but also using the assigned deviceID instead of the translation address bits [22–29]. The generated NWRITE transaction has the same target device offset as the write to segment 0, but is instead targeted to deviceID 9.

Combinations of aliasing and multi-targeting are also possible for a window.

15.6.7.2 RapidIO Outbound Window Translation Address Registers 0–8 (ROWTAR*n*)

The RapidIO outbound window translation address registers select the starting addresses in the external address space for window hits within the outbound translation windows. The new translated address is created by concatenating the transaction offset to this translation address.

Figure 15-55. RapidIO Outbound Window Translation Address Registers 0–8 (ROWTAR*n*)

Table 15-53. ROWTAR*n* Field Descriptions

Bits	Name	Description
0–1	LTGTID	LTGTID correspond to bits 6–7 of the target ID for a large transport system. This field is valid only if bit 27 of PEFCAR is set. Bits 0–5 of the target ID are specified in the window's RapidIO outbound window translation extended address register.
2–11	TREXAD	Translation extended address. TREXAD[0–7] correspond to the target ID for a small transport system or the least significant byte (bits 8–15) of the target ID for a large transport system. TREXAD[8–9] corresponds to bits [0–1] of a 34-bit RapidIO translation address. For maintenance transactions and default window 0, TREXAD[8–9] is reserved.
12–31	TRAD	Translation address. System address which represents the starting point of the outbound translated address. The translation address must be aligned based on the size field. This corresponds to bits [2–21] of the 34-bit RapidIO translation address. For maintenance transactions, the hop count is formed from TRAD[0–7], and the upper 12 bits of the maintenance offset is formed from TRAD[8–19]; the rest of the maintenance offset is formed from the untranslated address. This field is reserved for default window 0.

15.6.7.3 RapidIO Outbound Window Translation Extended Address Registers 0–8 (ROWTEAR*n*)

The RapidIO outbound window translation extended address registers contain bits 0–5 of the target ID for a common transport large system.

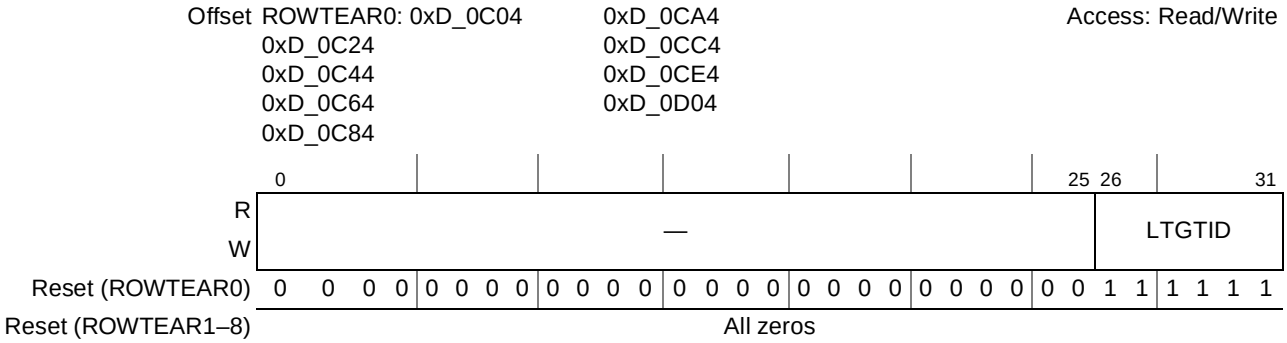


Figure 15-56. RapidIO Outbound Window Translation Extended Address Registers 0–8

Table 15-54. ROWTEAR*n* Field Descriptions

Bits	Name	Description
0–25	—	Reserved
26–31	LTGTID	LTGTID correspond to bits 0–5 of the target ID for a large transport system. This field is valid only if bit 27 of PEFCAR is set. Bits 6–7 of the target ID are specified in the window's RapidIO outbound window translation address register.

15.6.7.4 RapidIO Outbound Window Base Address Registers 1–8 (ROWBAR*n*)

The RapidIO outbound window base address registers select the base address for the windows which are translated to an alternate system address space. Addresses for outbound transactions are compared to these windows. If such a transaction does not fall within one of these spaces the transaction is forwarded through the default register set. For transactions that cross more than one window, please see Section 15.8.5.2, "Window Boundary Crossing Errors."

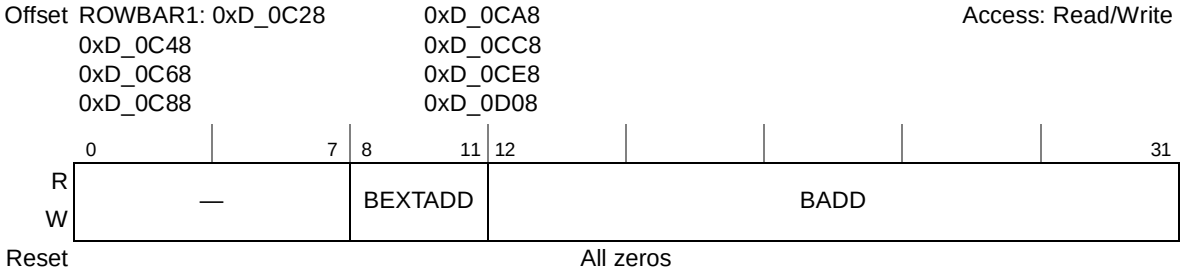


Figure 15-57. RapidIO Outbound Window Base Address Registers 1–8

Table 15-55. ROWBAR*n* Descriptions

Bits	Name	Description
0–7	—	Reserved
8–11	BEXTADD	Window base extended address. Corresponds to bits [0–3] of the 36-bit OCN base address.
12–31	BADD	Window base address. Source address which is the starting point for the outbound translation window. The window must be aligned based on the size selected in the window size bits. This corresponds to bits [4–23] of the 36-bit OCN base address.

15.6.7.5 RapidIO Outbound Window Attributes Registers 0–8 (ROWAR*n*)

The RapidIO outbound window attributes registers, shown in Figure 15-58, define the window sizes to translate and other attributes for the translations. 64G is the largest window size allowed. For a segmented window, these attributes are used for segment 0. The PCI_Window bit applies for all segments.

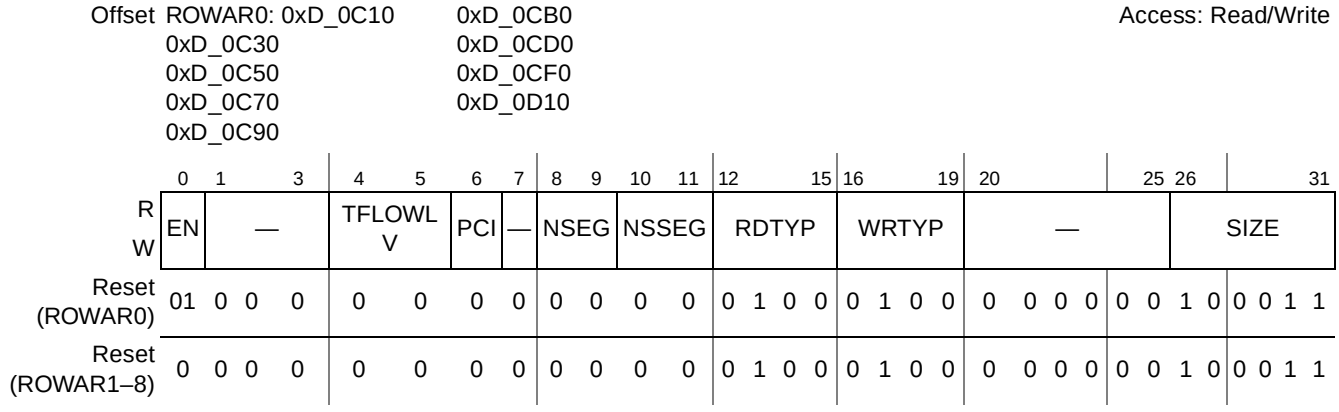


Figure 15-58. RapidIO Outbound Window Attributes Registers 0–8

Table 15-56. ROWAR*n* Field Descriptions

Bits	Name	Description
0	EN	This field enables this address translation window. It is set to 1 and is read only for default window 0.
1–3	—	Reserved
4–5	TFLOWLV	Transaction flow level. 00 Lowest priority transaction request flow 01 Next highest priority transaction request flow 10 Highest priority level transaction request flow 11 Reserved This field must be set to 00 if the PCI bit is set. Also, the RapidIO priority given by this field must always be greater than or equal to the OCN priority or a deadlock can occur. Normally, the OCN priority of all packets is 0 except for packets coming from a PCI type interface. Read type packets coming from a PCI type interface are priority 0 and write type packets are priority 1. Packets coming from a PCI type interface should set this field to 0 and set the PCI field to 1.
6	PCI	PCI_Window. This window follows PCI ordering rules as defined in the RapidIO Inter-operability specification The TFLOWLV field must be set to 00 if this bit is set causing reads to have RapidIO priority 0 and writes to have RapidIO priority 1.
7	—	Reserved
8–9	NSEG	Number of segments for this window. 00 One segment (normal window) 01 Two segments (half size aliasing window) 10 Four segments (quarter size aliasing window) 11 Reserved This field is reserved for default window 0.
10–11	NSSEG	Number of subsegments for this segment. 00 One target deviceID for this segment 01 Two target deviceIDs for this segment 10 Four target deviceIDs for this segment 11 Eight target deviceIDs for this segment This field is reserved for default window 0. Note that this field is valid only when ROWARn[NSEG] contains a non-zero value (1 or 2).
12–15	RDTYP	Transaction type to run on RapidIO interface if access is a read. 0000 Reserved 0001 Reserved 0010 IO_READ_HOME 0011 Reserved 0100 NREAD 0101 Reserved 0110 Reserved 0111 MAINTENANCE read 1000 Reserved ... 1100 ATOMIC increment 1101 ATOMIC decrement 1110 ATOMIC set 1111 ATOMIC clear

Table 15-56. ROWAR*n* Field Descriptions (continued)

Bits	Name	Description
16–19	WRTYP	Transaction type to run on RapidIO interface if access was a write. 0000 Reserved 0001 FLUSH 0010 Reserved 0011 SWRITE 0100 NWRITE 0101 NWRITE_R 0110 Reserved 0111 MAINTENANCE write 1000 Reserved ... 1111 Reserved Write-requiring-response sent from OCN must generate a write-requiring-response to RapidIO. Therefore, if an OCN write-requiring-response request hits a window with WRTYP = SWRITE or NWRITE, the RapidIO endpoint generates a NWRITE_R instead.
20–25	—	Reserved
26–31	SIZE	Outbound translation window size N which is the encoded $2^{(N+1)}$ bytes window size. The smallest window size is 4 Kbytes. 000000 Reserved ... 001011 4K window size 001100 8K window size ... 011111 4G window size 100000 8G window size 100001 16G window size 100010 32G window size 100011 64G window size 100100 Reserved ... 111111 Reserved This field is read only for default window 0.

15.6.7.6 RapidIO Outbound Window Segment 1–3 Registers 1–8 (ROWS*n*R*m*)

The RapidIO outbound window segment registers define the attributes and target device ID that are to be used for a transaction that hits in that segment rather than the primary attributes and target deviceID. There is one of these registers for each segment (except segment 0).

Offset		Segment 1:	Segment 2:	Segment 3:	Access: Read/Write
	Reg1:	0xD_0C34	0xD_0C38	0xD_0C3C	
	Reg2:	0xD_0C54	0xD_0C58	0xD_0C5C	
	Reg3:	0xD_0C74	0xD_0C78	0xD_0C7C	
	Reg4:	0xD_0C94	0xD_0C98	0xD_0C9C	
	Reg5:	0xD_0CB4	0xD_0CB8	0xD_0CBC	
	Reg6:	0xD_0CD4	0xD_0CD8	0xD_0CDC	
	Reg7:	0xD_0CF4	0xD_0CF8	0xD_0CFC	
	Reg8:	0xD_0D14	0xD_0D18	0xD_0D1C	

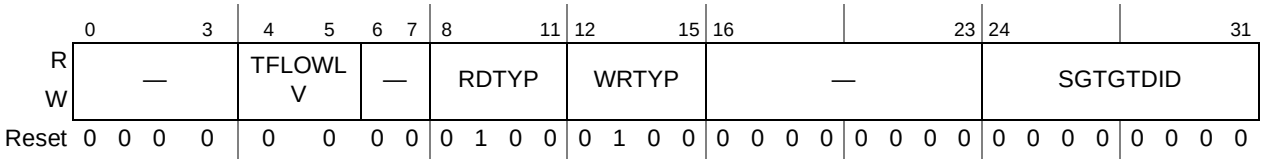


Figure 15-59. RapidIO Outbound Window Segment 1−3 Registers 1−8 (ROWS*n*R*m*)

Table 15-57. ROWS*n*R*n* Field Descriptions

Bits	Name	Description
0–3	—	Reserved
4–5	TFLOWLV	Transaction flow level. 00 Lowest priority transaction request flow 01 Next highest priority transaction request flow 10 Highest priority level transaction request flow 11 Reserved This field must be set to 00 if the PCI_Window bit is set
6–7	—	Reserved
8–11	RDTYP	Transaction type to run on RapidIO interface if access was a read. 0000 Reserved 0001 Reserved 0010 IO_READ_HOME 0011 Reserved 0100 NREAD 0101 Reserved 0110 Reserved 0111 MAINTENANCE read 1000 Reserved ... 1100 ATOMIC increment 1101 ATOMIC decrement 1110 ATOMIC set 1111 ATOMIC clear
12–15	WRTYP	Transaction type to run on RapidIO interface if access was a write. 0000 Reserved 0001 FLUSH 0010 Reserved 0011 SWRITE 0100 NWRITE 0101 NWRITE_R 0110 Reserved 0111 MAINTENANCE write 1000 Reserved ... 1110 Reserved 1111 Reserved Writes-requiring-response sent from OCN must generate a write-requiring-response to RapidIO. Therefore, if an OCN write-requiring-response request hits a window with WRTYP = SWRITE or NWRITE, the RapidIO endpoint generates a NWRITE_R instead.
16–23	—	Reserved
24–28	SGTGTDID	Bits 0-4 (or bits 8-12 if large transport system) of the target device ID for this segment.
29	SGTGTDID	Bit 5 (or bit 13 if large transport system) of the target deviceID; this bit is reserved if 8 target subsegments are selected.
30	SGTGTDID	Bit 6 (or bit 14 if large transport system) of the target deviceID; this bit is reserved if 8 or 4 target subsegments are selected.
31	SGTGTDID	Bit 7 (or bit 15 if large transport system) of the target deviceID; this bit is reserved if 8, 4, or 2 target subsegments are selected.

15.6.7.7 RapidIO Inbound Window Translation Address Registers 0–4 (RIWTAR*n*)

The RapidIO inbound window translation address registers point to the starting addresses in local address space for window hits within the inbound translation windows. The new translated address is created by concatenating the transaction offset to this translation address.

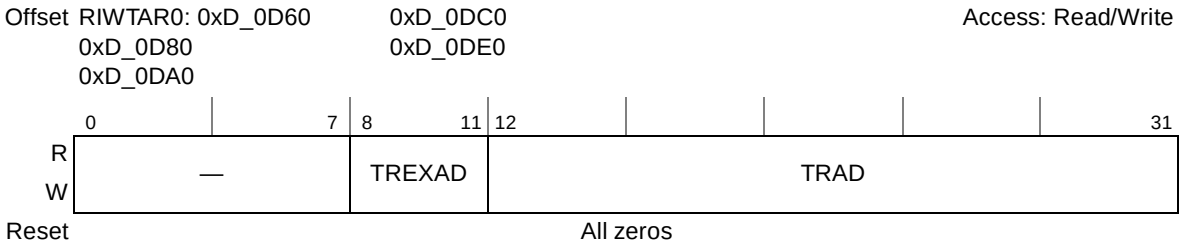


Figure 15-60. RapidIO Inbound Window Translation Address Registers 0–4 (RIWTAR*n*)

Table 15-58. RIWTAR*n* Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–11	TREXAD	Translation extended address. Corresponds to bits 0–3 of the 36-bit OCN translation address. TREXAD[2–3] are reserved for default window 0.
12–31	TRAD	Translation address. Target address which indicates the starting point of the inbound translated address. The translation address must be aligned based on the size field. This corresponds to bits 4–23 of the 36-bit OCN translation address. TRAD is reserved for default window 0.

15.6.7.8 RapidIO Inbound Window Base Address Registers 1–4 (RIWBAR*n*)

The RapidIO inbound window *n* base address registers select the base address for the windows which are translated to an alternate target address space. Addresses for inbound transactions are compared to these windows. If such a transaction does not fall within one of these spaces, then the transaction is forwarded to the interior of the chip using the default window. For transactions that cross more than one window, please see Section 15.8.6.2, “Window Boundary Crossing Errors.”

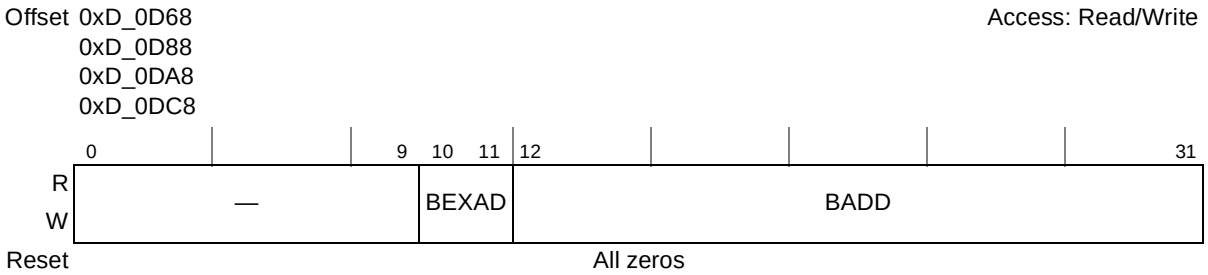


Figure 15-61. RapidIO Inbound Window Base Address Registers1–4

Table 15-59. RIWBAR*n* Field Descriptions

Bits	Name	Description
0–9	—	Reserved

Table 15-59. RIWBAR*n* Field Descriptions

Bits	Name	Description
10–11	BEXAD	Base extended address. BEXAD represents bits 0–1 of the 34-bit RapidIO address.
12–31	BADD	Base address. System address which is the starting point for the inbound translation window. The window must be aligned based on the size selected in the window size bits. This corresponds to bits 2–21 of the 34-bit RapidIO base address.

15.6.7.9 RapidIO Inbound Window Attributes Registers 0–4 (RIWAR*n*)

The RapidIO inbound window attributes registers define the window sizes to translate and other attributes for the translations. 16 Gbytes is the largest window size allowed. The RDTYP and WRTYP fields are used for attributes on the request, but do not actually change the transaction type of the transaction. In other words, RIWAR*n* does not modify whether or not the request requires a response or if the request was atomic; this type of attribute remains unchanged on the request as it is translated through the ATMU.

Offset 0xD_0D70
0xD_0D90
0xD_0DB0
0xD_0DD0

Access: Read/Write

	0	1	2	7	8	11	12	15	16	19	20	25	26	31
R W	EN	PW	—		TGINT		RDTYP		WRTYP		—		SIZE	
Reset	0	0	0 0 0 0 0		0 0 0 0		0 1 0 0		0 1 0 0		0 0 0 0 0		0 0 1 0	0 0 0 1

Table 15-60. RapidIO Inbound Window Attributes Register 1–4

Offset 0xD_0DF0

Access: Mixed

	0	1	2	7	8	11	12	15	16	19	20	25	26	31
R	EN	PW	—		TGINT		RDTYP		WRTYP		—		SIZE	
W														
Reset	1	0	0 0 0 0 0		0 0 0 0		0 1 0 0		0 1 0 0		0 0 0 0 0		0 0 1 0	0 0 0 1

Table 15-61. RapidIO Inbound Window Attributes Register 0

Table 15-62. RIWAR*n* Field Descriptions

Bits	Name	Description
0	EN	This bit enables this address translation. This field is set to 1 and read-only for default window 0.
1	PW	Protected Window. This bit indicates that this window is protected. Writes requiring response and reads to this window generate an error response. Writes not requiring response are silently discarded.
2–7	—	Reserved

Table 15-62. RIWAR*n* Field Descriptions (continued)

Bits	Name	Description
8–11	TGINT	Target interface. If this field is set to anything other than local address space, the attributes for the transaction must be assigned in a corresponding outbound window at the target. This is the field definition for the MPC8641: 0000 PCI Express interface 1 0001 PCI Express interface 2 0010 Reserved 0011 Reserved ... 1110 Reserved 1111 Local memory (DDRs, local bus controller, memory-mapped registers)
12–15	RDTYP	Transaction type to run on the I/O interface if access is a read. 0000 Reserved ... 0100 Read 0101 Reserved ... 1111 Reserved Transaction type to run on local memory if access is a read. 0000 Reserved ... 0100 Read, don't snoop local processor 0101 Read, snoop local processor 0110 Reserved 0111 Reserved ... 1111 Reserved
16–19	WRTYP	Transaction type to run on I/O interface if access is a write. 0000 Reserved ... 0100 write 0101 Reserved ... 1111 Reserved Transaction type to run on local memory if access is a write. 0000 Reserved ... 0011 Reserved 0100 Write, don't snoop local processor 0101 Write, snoop local processor 0110 Reserved ... 1111 Reserved
20–25	—	Reserved

Table 15-62. RIWAR*n* Field Descriptions (continued)

Bits	Name	Description
26–31	SIZE	Inbound translation window size N which is the encoded 2^(N+1) bytes window size. The smallest window size is 4K bytes. 000000 Reserved ... 001011 4K window size 001100 8K window size ... 011111 4G window size 100000 8G window size 100001 16G window size 100010 Reserved ... 111111 Reserved This field is read only for default window 0.

15.7 RapidIO Message Unit Registers

15.7.1 RapidIO Outbound Message 0 Registers

The registers in this section control RapidIO outbound messages. The following provide partial descriptions of these registers.

15.7.1.1 Outbound Message *n* Mode Registers (OM*n*MR)

The outbound message mode register allows software to start a message operation and to control various message operation characteristics.

Offset Msg unit 0: 0xD_3000
Msg unit 1: 0xD_3100

Access: Read/Write

	0–3	4–6	7–10	11	12–15	16–19	20–21	22	23	24	25	26	27	28–29	30	31
R W	—	SCTL	—	DES_SEN	—	CIRQ_SIZ	—	QOIE	QFIE	—	QEIE	EIE	—	MUTM	MUI	MUS
Reset	All zeros															

Figure 15-62. Outbound Message *n* Mode Registers (OM*n*MR)

Table 15-63. OM*n*MR Field Descriptions

Bits	Name	Description
0–3	—	Reserved
4–6	SCTL	Service control. Determines the number of descriptors to process before servicing the next queue. 0b000 Fixed priority based on outbound message unit number 0b001 1 descriptors 0b010 2 descriptors 0b011 4 descriptors 0b100 8 descriptors 0b101 16 descriptors 0b110 32 descriptors 0b111 64 descriptors Note that if one queue has SCTL set to fixed priority, all SCTL values in other queues are ignored. For example, of the two outbound message units, if unit 1's service control value is set to 0b000, fixed priority results with unit 0 the highest priority results. If a queue is in direct mode only, one message operation is serviced before servicing the next queue. For proper operation, this field should only be modified when the outbound message controller is not enabled. The value of this field cannot be changed unless both units are disabled.
7–10	—	Reserved
11	DES_SEN	Descriptor snoop enable. When set enables snooping of the local processor when reading descriptors from memory. For proper operation, this field should only be modified when the outbound message controller is not enabled
12–14	—	Reserved
15	—	Reserved
16–19	CIRQ_SIZ	Circular descriptor queue size. Determines the number of descriptors that can be placed on the circular queue without overflow. 0b0000 2 — 0b1000 512 0b0001 4 — 0b1001 1024 0b0010 8 — 0b1010 2048 0b0011 16 — 0b1011 Reserved 0b0100 32 — 0b1100 Reserved 0b0101 64 — 0b1101 Reserved 0b0110 128 — 0b1110 Reserved 0b0111 256 — 0b1111 Reserved For proper operation, this field should only be modified when the outbound message controller is not enabled
20–21	—	Reserved
22	QOIE	Queue overflow interrupt enable. When set, bit generates an interrupt on detection of a queue overflow (that is, the enqueue and dequeue pointers are no longer equal after being incremented by the processor and the queue was full). No queue overflow interrupt is generated if this bit is cleared. (only applicable to chaining mode). If this bit is not set and the queue overflows, the result is undefined.
23	QFIE	Queue full interrupt enable. When set will generate an interrupt when the queue transitions to full (that is, the enqueue and dequeue pointers are equal after being incremented by the processor). No QF interrupt is generated if this bit is cleared. If this bit is set and OM*n*SR[QF] is a 1, OM*n*SR[QFI] will assert.
24	—	Reserved

Table 15-63. OM*n*MR Field Descriptions (continued)

Bits	Name	Description
25	QEIE	Queue empty interrupt enable. When set, bit 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). No QE interrupt is generated if this bit is cleared. For proper operation, this field should only be modified when the outbound message controller is not enabled
26	EIE	Error interrupt enable. When set, bit generates the port-write/error interrupt when a transfer error (OM*n*SR[TE]), a message error response (OM*n*SR[MER]), a packet response time-out (OM*n*SR[PRT]), or a retry threshold event exceeded (OM*n*SR[RETE]) event occurs. No port-write/error interrupt is generated if this bit is cleared.
27–28	—	Reserved
29	MUTM	Message unit transfer mode. Setting this bit 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. Clearing this bit configures the message unit in chaining mode.
30	MUI	Message unit increment. Software sets this bit after writing a descriptor to memory. Hardware then increments the OM*n*DQEPAR and clear this bit. Always reads as 0 when MUS is set.
31	MUS	Message unit start. Direct mode—A 0 to 1 transition when the message unit is not busy (MUB bit is 0) starts the message unit. A 1 to 0 transition will have no effect. Chaining mode—If this bit is set, the message unit starts whenever the enqueue and dequeue pointers are not equal.

15.7.1.2 Outbound Message *n* Status Registers (OM*n*SR)

The outbound message status register reports various message unit conditions during and after a message operation. Writing a 1 to the corresponding set bit clears the bit.

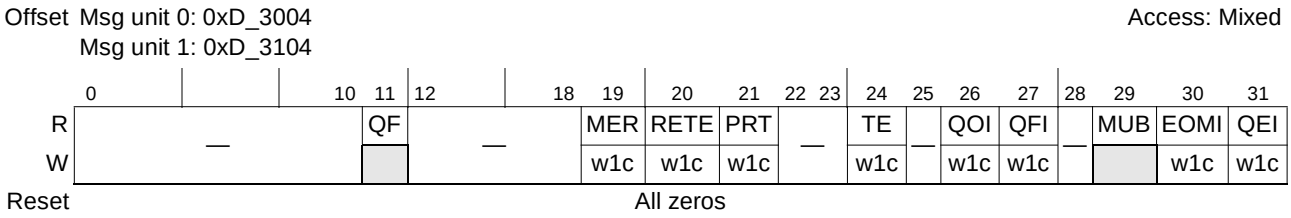


Figure 15-63. Outbound Message *n* Status Registers (OM*n*SR)

Table 15-64. OM*n*SR Field Descriptions

Bits	Name	Description
0–10	—	Reserved
11	QF	Queue full - If the queue becomes full, then this bit is set. (Read-only)
12–18	—	Reserved
19	MER	Message error response. This bit is set when an ERROR response is received from the message target. The Error response received field indicates value of the error response status bits when an error response is received. This bit is cleared by writing a 1.

Table 15-64. OM*n*SR Field Descriptions (continued)

Bits	Name	Description
20	RETE	Retry error threshold exceeded. This bit is set when the message unit has been unable to complete a message operation because the retry error threshold value has been exceeded due to RapidIO retry response.-This bit is cleared by writing a 1.
21	PRT	Packet response time-out. This bit is set when the message unit has been unable to complete a message operation and a packet response time-out occurred. This bit is cleared by writing a 1.
22–23	—	Reserved
24	TE	Transaction error. This bit is set when an internal error condition occurs during the message operation. This bit is cleared by writing a 1. For proper operation, this field should only be modified when the outbound message controller is not enabled
25	—	Reserved
26	QOI	Queue overflow interrupt. This bit is set when a queue overflow condition is detected. This bit is cleared by writing a 1. (only applicable to chaining mode)
27	QFI	Queue full interrupt. If the queue becomes full, if the QFIE bit in the Mode Register is set, then this bit is set and an interrupt generated. This bit is cleared by writing a 1.
28	—	Reserved
29	MUB	Message unit busy. When set indicates that a message operation is currently in progress. This bit is cleared as a result of an error or the message operation is finished. (Read-only)
30	EOMI	End-Of-Message interrupt. After finishing this message operation, if the EOMIE bit in the Destination Attributes Register is set, then this bit is set and an interrupt generated. This bit is cleared by writing a 1.
31	QEI	Queue Empty Interrupt. When the last message operation in the outbound descriptor queue is finished, if the QEIE bit in the Mode Register is set, then this bit is set and an interrupt is generated. Otherwise, no interrupt is generated. This bit is cleared by writing a 1.

15.7.1.3 Extended Outbound Message *n* Descriptor Queue Dequeue Pointer Address Registers (EOM*n*DQDPAR) and Outbound Descriptor Queue Dequeue Pointer Address Registers (OM*n*DQDPAR)

The outbound message descriptor queue dequeue pointer address registers contain the address of the first descriptor in memory to be processed. Software must initialize these registers to point to the first descriptor in memory. After processing this descriptor, the message unit controller increments the outbound message descriptor queue dequeue pointer address in OM*n*DQDPAR and EOM*n*DQDPAR to point to the next descriptor. If the outbound message descriptor queue enqueue pointer and the outbound message descriptor queue dequeue pointer are not equal (indicating that the queue is not empty), the message unit controller reads the next descriptor from memory for processing. If the enqueue and dequeue pointers are equal after the dequeue pointer has been incremented by the message unit controller, the queue is empty and the message unit halts until the enqueue pointer is incremented by the processor. Incrementing the pointer indicates that a new descriptor has been added to the queue and is ready for processing. If the queue becomes empty and OM*n*MR[QEIE] is set, OM*n*SR[QEI] is set and an interrupt is generated.

When software initializes these registers, the queue to which they point must be aligned on a boundary equal to number of queue entries × 32 bytes (size of each queue descriptor). For example, if there are eight entries in the queue, the queue must be 256-byte aligned.

The number of queue entries is set in OM*n*MR[CIRQ_SIZ]; see Section 15.7.1.1, “Outbound Message n Mode Registers (OMnMR)”).

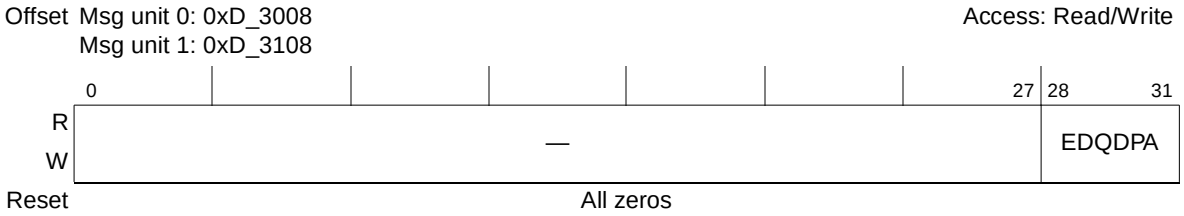


Figure 15-64. Extended Outbound Message *n* Descriptor Queue Dequeue Pointer Address Registers (EOM*n*DQDPAR)

Table 15-65. EOM*n*DQDPAR Field Descriptions

Bits	Name	Description
0–27	—	Reserved
28–31	EDQDPA	Extended descriptor queue dequeue pointer address bits. These are the highest order address bits. For proper operation, this field should only be modified when the outbound message controller is not enabled.

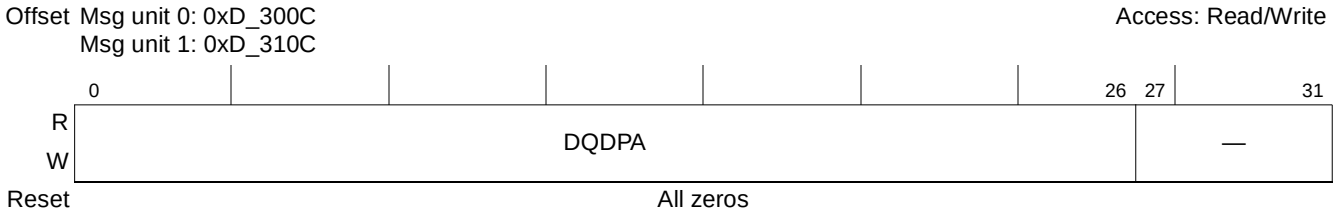


Figure 15-65. Outbound Message *n* Descriptor Queue Dequeue Pointer Address Registers (OM*n*DQDPAR)

Table 15-66. OM*n*DQDPAR Field Descriptions

Bits	Name	Description
0–26	DQDPA	Descriptor queue dequeue pointer address. Contains the address of the first descriptor in memory to process. The descriptor must be aligned to a 32-byte boundary. For proper operation, this field should only be modified when the outbound message controller is not enabled.
27–31	—	Reserved

15.7.1.4 Extended Outbound Message *n* Descriptor Queue Enqueue Pointer Address Registers (EOM*n*DQEPAR) and Outbound Message *n* Descriptor Queue Enqueue Pointer Address Registers (OM*n*DQEPAR)

The outbound message descriptor queue enqueue pointer address registers contain the address for the next descriptor in memory to be added to the queue. Software must initialize these registers to match the outbound message descriptor queue dequeue pointer address. When a message is ready to be sent, the processor writes a descriptor to the next location in the queue (indicated by the address in OM*n*DQEPAR and EOM*n*DQEPAR), and then either writes the OM*n*DQEPAR to point to the next descriptor location in memory or sets OM*n*MR[MUI]. This can result in a number of actions:

- If the enqueue and dequeue pointers match, the queue is now full. If the OM*n*MR[QFIE] bit is set, then the OM*n*SR[QFI] bit is set, and an interrupt is generated.
- If the enqueue and dequeue pointer no longer match after the enqueue pointer has been incremented and the queue is full, then the queue overflows, and the message unit stops. If OM*n*MR[QOIE] is set, OM*n*SR[QOI] is set and an interrupt is generated. OM*n*MR[MUS] must transition to a 0 to clear this error condition. If the enqueue pointer is directly written, the queue overflow condition is not detected.
- If the enqueue and dequeue pointers were the same before reading the register, the message unit controller will start (if enabled).

When software initializes these registers, the queue to which they point must be aligned on a boundary equal to number of queue entries × 32 bytes (size of each queue descriptor). For example, if there are eight entries in the queue, the queue must be 256-byte aligned.

The number of queue entries is set in OM*n*MR[CIRQ_SIZ]; see Section 15.7.1.1, “Outbound Message n Mode Registers (OMnMR)”).

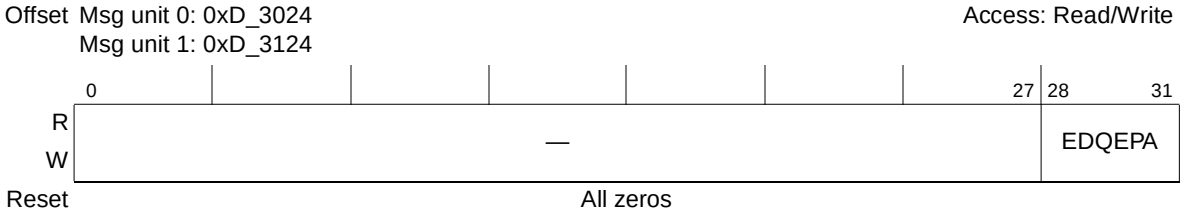


Figure 15-66. Extended Outbound Message *n* Descriptor Queue Enqueue Pointer Registers (EOM*n*DQEPAR)

Table 15-67. EOM*n*DQEPAR Field Descriptions

Bits	Name	Description
0–27	—	Reserved
28–31	EDQEPA	Extended descriptor queue enqueue pointer address. These are the highest-order address bits.

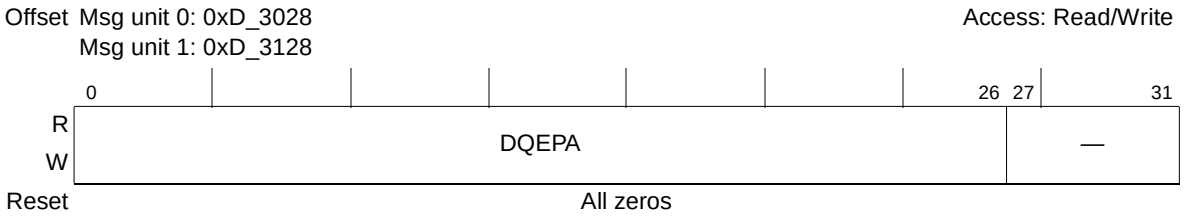


Figure 15-67. Outbound Message *n* Descriptor Queue Enqueue Pointer Registers (OM*n*DQEPAR)

Table 15-68. OM*n*DQEPAR Field Descriptions

Bits	Name	Description
0–26	DQEPA	Descriptor queue enqueue pointer address. Contains the address of the last descriptor in memory to process. The descriptor must be aligned to a 32 byte boundary and a descriptor queue boundary.
27–31	—	Reserved

15.7.1.5 Extended Outbound Message *n* Source Address Registers (EOM*n*SAR) and Outbound Message *n* Source Address Registers (OM*n*SAR)

The outbound message unit source address registers indicate the address from which the message unit controller is to read data. Software must ensure that this is a valid local memory address. The source address must be aligned to a double-word boundary, so the least significant three bits are reserved.

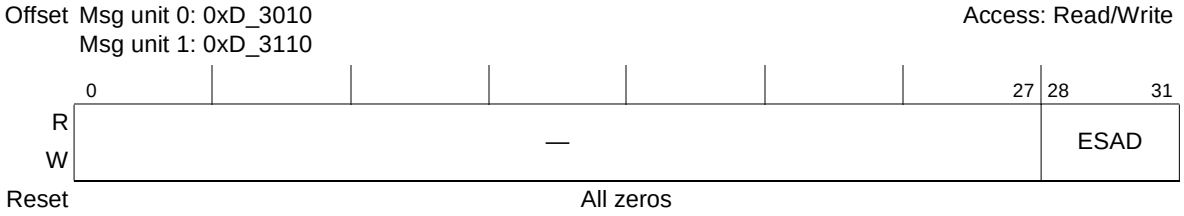


Figure 15-68. Extended Outbound Message *n* Source Address Registers (EOM*n*SAR)

Table 15-69. EOM*n*SAR Field Descriptions

Bits	Name	Description
0–27	—	Reserved
28–31	ESAD	Extended source address bits. These are the highest order address bits. For proper operation, this field should only be modified when the outbound message controller is not enabled

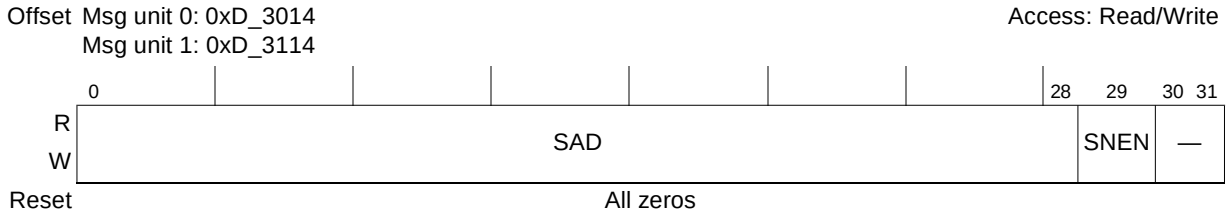


Figure 15-69. Outbound Message *n* Source Address Registers (OM*n*SAR)

Table 15-70. OM*n*SAR Field Descriptions

Bits	Name	Description
0–28	SAD	Source address. This is the source address of the message operation. The contents are updated after every memory read operation. For proper operation, this field should only be modified when the outbound message controller is not enabled
29	SNEN	Snoop enable. This bit enables snooping of the local processor caches for the data reads from local memory. For proper operation, this field should only be modified when the outbound message controller is not enabled
30–31	—	Reserved

15.7.1.6 Outbound Message *n* Destination Port Registers (OM*n*DPR)

The destination port register indicates the RapidIO destination ID and mailbox to which the message unit controller is to send data. The software must ensure that this is a valid port in the receiving device.

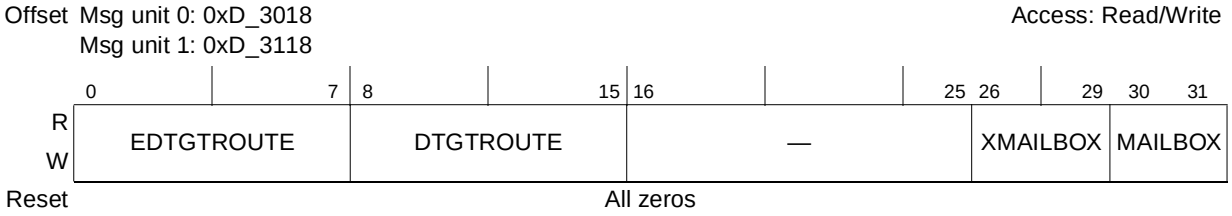


Figure 15-70. Outbound Message *n* Destination Port Registers (OM*n*DPR)

Table 15-71. OM*n*DPR Field Descriptions

Bits	Name	Description
0–7	EDTGTROUTE	Extended destination target route. Most significant byte of a 16-bit target route when operating in large transport mode. Reserved when operating in small transport mode. For proper operation, this field should only be modified when the outbound message controller is not enabled
8–15	DTGTROUTE	Destination target route. Contains the target route field of the transaction (Device ID of the target). This value is overridden by the multicast group and list if multicast mode is enabled. On error, if multicast mode is enabled, this field is loaded with the destination of the failed operation. For proper operation, this field should only be modified when the outbound message controller is not enabled
16–25	—	Reserved
26–29	XMAILBOX	Value for ‘xmbox’ field in MESSAGE packet. This field is only used when OM*n*DATR[MM] is set. For proper operation, this field should only be modified when the outbound message controller is not enabled
30–31	MAILBOX	Value for ‘mbox’ field in MESSAGE packet. For proper operation, this field should only be modified when the outbound message controller is not enabled

15.7.1.7 Outbound Message *n* Destination Attributes Registers (OM*n*DATR)

The outbound message destination attributes register contains the transaction attributes to be used for the message operation.

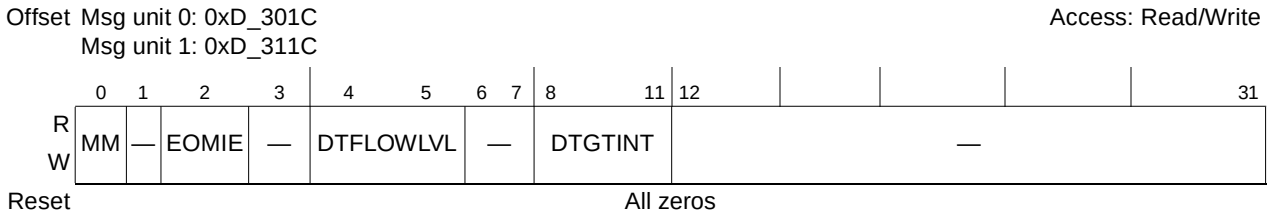


Figure 15-71. Outbound Message *n* Destination Attributes Registers (OM*n*DATR)

Table 15-72. OM*n*DATR Field Descriptions

Bits	Name	Description
0	MM	Multicast mode. When set, the message operation is sent to all of the targets indicated by the multicast group and list. Messages are limited to one segment and 256 bytes or less when this mode is enabled.
1	—	Reserved
2	EOMIE	End-of-Message interrupt enable. When set, generates an interrupt when the current message operation is finished. For proper operation, this field should only be modified when the outbound message controller is not enabled.
3	—	Reserved
4–5	DTFLOWLVL	Transaction flow level. 00 Lowest priority transaction request flow 01 Next highest priority transaction request flow 10 Highest priority transaction request flow 11 Reserved For proper operation, this field should only be modified when the outbound message controller is not enabled
6–7	—	Reserved
8–11	DTGTINT	Target interface. The value of this field should always be set to RapidIO (0x0).
12–31	—	Reserved

15.7.1.8 Outbound Message *n* Double-word Count Registers (OM*n*DCR)

The outbound message double-word count register contains the number of double-words for the message operation. The maximum message operation size is 4 Kbytes and the minimum is 8 bytes.

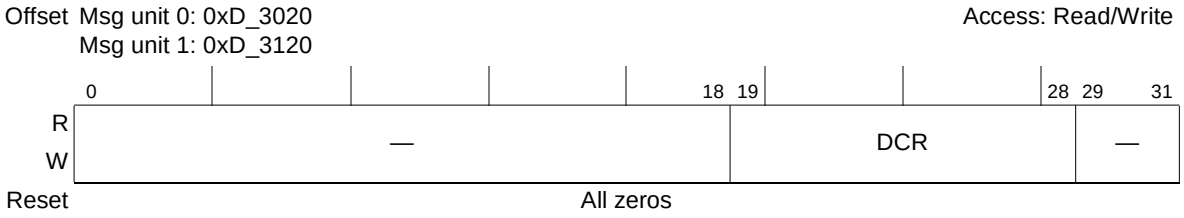


Figure 15-72. Outbound Message *n* Double Word Count Registers (OM*n*DCR)

Table 15-73. OM*n*DCR Field Descriptions

Bits	Name	Description
0–18	—	Reserved
19–28	DCR	Transfer count register. Contains the number of bytes for the message operation. 00 0000 0000 Reserved 00 0000 0001 8 bytes 00 0000 0010 16 bytes 00 0000 0100 32 bytes 00 0000 1000 64 bytes 00 0001 0000 128 bytes 00 0010 0000 256 bytes 00 0100 0000 512 bytes (multi segment mode only) 00 1000 0000 1024 bytes (multi segment mode only) 01 0000 0000 2048 bytes (multi segment mode only) 10 0000 0000 4096 bytes (multi segment mode only) All other values yield undefined behavior. For proper operation, this field should only be modified when the outbound message controller is not enabled.
29–31	—	Reserved

15.7.1.9 Outbound Message *n* Retry Error Threshold Configuration Registers (OM*n*RETCR)

The outbound message retry error threshold configuration register controls the number of times that the message unit attempts to transfer a message segment to a particular destination before reporting an error. A message segment is retransmitted if a RETRY response is received from the target.

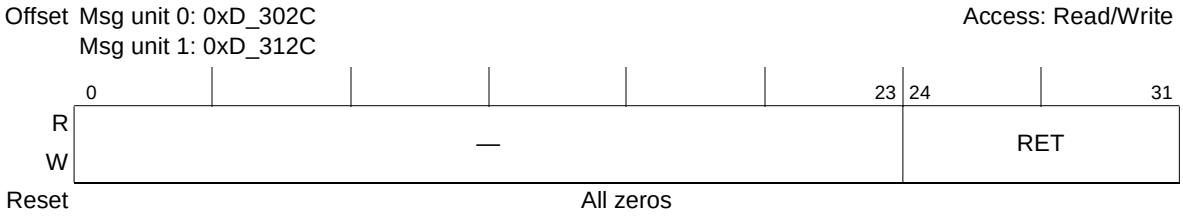


Figure 15-73. Outbound Message *n* Retry Error Threshold Configuration Registers (OM*n*RETCR)

Table 15-74. OM*n*RETCR Field Descriptions

Bits	Name	Description
0–23	—	Reserved
24–31	RET	Retry error threshold. This value is the number of times that the message unit attempts to transmit a message segment to a particular target. 0x00 Disabled 0x01 Message segment transmitted only 1 time 0x02 Message segment transmitted up to 2 times ... 0xFF Message segment transmitted up to 255 times For proper operation, this field should only be modified when the outbound message controller is not enabled

15.7.1.10 Outbound Message *n* Multicast Group Registers (OM*n*MGR)

The multicast group register contains a multicast group (MG) value and extended multicast group number (EMG) which, in combination with the multicast list register and the multicast enable (OM*n*DATR[MM]), indicates which deviceIDs are targets of a multicast operation. The multicast group represents the most significant three bits (bits[0-2]) of the RapidIO deviceIDs that are destinations of the message operation, whereas the multicast list indicates a list of targets within that group. Each individual set bit, when encoded, determines the least significant five bits of the RapidIO deviceIDs (bits[3–7]) that are targets of the message operation. Therefore, multicast group 0 (MG = 0) contains target deviceIDs (0,1,...,31), multicast group 1 (MG = 1) contains target deviceIDs (32,33,...63), and so on.

If in large transport mode, the extended multicast group represents the eight most significant bits (bits[0–7]), the multicast group represents the next three bits (bits[8–10]) and the multicast list indicates a list of targets within that group.

If multicast is enabled, this information in the multicast group and mask register is used to determine the target of the message operation instead of the DTGTROUTE and EDTGTROUTE fields in the outbound message *n* destination attributes register.

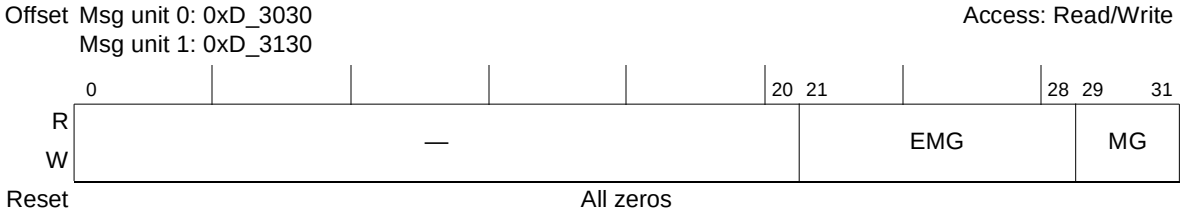


Figure 15-74. Outbound Message *n* Multicast Group Registers (OM*n*MGR)

Table 15-75. OM*n*MGR Field Descriptions

Bits	Name	Description
0–20	—	Reserved
21–28	EMG	Extended multicast group. This is the most significant eight bits of the target deviceIDs for the multicast operation when operating in large transport mode. For proper operation, this field should only be modified when the outbound message controller is not enabled
29–31	MG	Multicast group. This is the most significant three bits of the target deviceIDs for the multicast operation. For proper operation, this field should only be modified when the outbound message controller is not enabled

15.7.1.11 Outbound Message *n* Multicast List Registers (OM*n*MLR)

See the multicast group register description for more information.

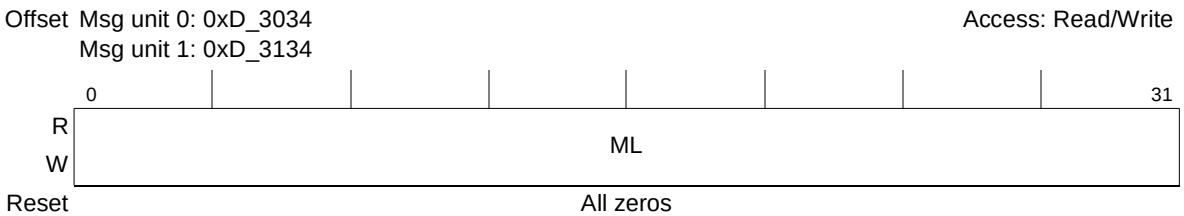


Figure 15-75. Outbound Message *n* Multicast List Registers (OM*n*MLR)

Table 15-76. OM*n*MLR Field Descriptions

Bits	Name	Description
0–31	ML	Multicast list. This is the group target list for the message operation. Depending on the value of the multicast group value, bit 0 corresponds to deviceID 0, 32, 64, 96, etc., bit 1 corresponds to deviceID 1, 33, 65, 97, etc. and so on. If none of the bits are set, bit 0 is assumed to be set. For proper operation, this field should only be modified when the outbound message controller is not enabled.

15.7.2 RapidIO Inbound Message Registers

Registers in this section describe the RapidIO inbound message registers.

15.7.2.1 Inbound Message *n* Mode Registers (IM*n*MR)

The inbound message mode register allows software to enable the mailbox controller and to control various message operation characteristics.

Offset Msg unit 0: 0xD_3060
Msg unit 1: 0xD_3160

Access: Read/Write

	0–3	4–10	11	12–15	16–19	20–22	23	24	25	26	27–29	30	31
R W	MIQ_THRESH	—	SEN	FRM_SIZ	CIRQ_SIZ	—	QFIE	—	MIQIE	EIE	—	MI	ME
Reset	All zeros												

Figure 15-76. Inbound Message *n* Mode Registers (IM*n*MR)

Table 15-77. IM*n*MR Field Descriptions

Bits	Name	Description
0–3	MIQ_THRESH	Message in queue threshold. Determines the number of message frames to be accumulated in the frame queue before Message In Queue is signaled. Undefined operation results if the message in queue threshold is set greater than or equal to the message queue size (IM*n*MR[CIRQ_SIZ]). 0000 1 0001 2 0010 4 0011 8 0100 16 0101 32 0110 64 0111 128 1000 256 1001 512 1010 1024 1011 Reserved ... 1111 Reserved For proper operation, this field should only be modified when the inbound message controller is not enabled.
4–10	—	Reserved
11	SEN	Snoop enable. When set, enables snooping the local processor when writing messages into memory. For proper operation, this field should only be modified when the inbound message controller is not enabled.

Table 15-77. IM*n*MR Field Descriptions (continued)

Bits	Name	Description
12–15	FRM_SIZ	Message frame size. Determines the maximum message size that can be accepted by this mailbox without error. This parameter combined with CIRQ_SIZ determine the maximum contiguous memory space allocated to the mailbox. 0000 Reserved ... 0010 8 bytes 0011 16 bytes 0100 32 bytes 0101 64 bytes 0110 128 bytes 0111 256 bytes 1000 512 bytes 1001 1024 bytes 1010 2048 bytes 1011 4096 bytes 1100 Reserved ... 1111 Reserved For proper operation, this field should only be modified when the inbound message controller is not enabled.
16–19	CIRQ_SIZ	Circular frame queue size. Determines the number of message frames that can be place on the circular queue without overflow. This parameter combined with FRM_SIZ determine the maximum contiguous memory space allocated to the mailbox. 0000 2 0001 4 0010 8 0011 16 0100 32 0101 64 0110 128 0111 256 1000 512 1001 1024 1010 2048 1011 Reserved ... 1111 Reserved For proper operation, this field should only be modified when the inbound message controller is not enabled.
20–22	—	Reserved
23	QFIE	Queue full interrupt enable. When set, 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). No QFI interrupt is generated if this if this bit is cleared. If this bit is set and IM*n*SR[QF] is a 1, IM*n*SR[QFI] asserts.
24	—	Reserved
25	MIQIE	Message in queue interrupt enable. When set, generates an interrupt when the queue has accumulated the number of messages specified by the IM*n*MR[MIQ_THRESH]. No MIQ interrupt is generated if this bit is cleared. If this bit is set and IM*n*SR[MIQ] is a 1, IM*n*SR[MIQI] asserts. If this bit is set and IM*n*MR[MI] is also set simultaneously, IM*n*SR[MIQI] indicates reflects the value of MIQ after the increment.
26	EIE	Error interrupt enable. When set, generates the port-write/error interrupt when a transfer error (IM*n*SR[TE]) or a message request time-out (IM*n*SR[MRT]) event occurs. No port-write/error interrupt is generated if this bit is cleared.
27–29	—	Reserved

Table 15-77. IM*n*MR Field Descriptions (continued)

Bits	Name	Description
30	MI	Mailbox increment. Software sets this bit after processing an inbound message. Hardware increments the IM*n*FQDPAR and clears this bit. Always reads as 0.
31	ME	Mailbox enable. If this bit is set, the mailbox is initialized and can service incoming message operations. If this bit is cleared after the first segment of a multi-segment message has arrived, a message request time-out results (IM*n*SR[MRT]) and the busy bit (IM*n*SR[MB]) clears as long as the port response timer value (PRTOCCSR[TV]) is not set to the disabled value. If the port response timer value is set to the disabled value, the busy bit will not clear.

15.7.2.2 Inbound Message *n* Status Registers (IM*n*SR)

The inbound message status register reports various mailbox conditions during and after a message operation. Writing a 1 to the corresponding set bit clears the bit.

Offset Msg unit 0: 0xD_3064
Msg unit 1: 0xD_3164

Access: Mixed

	0										10	11	12		14	15	16				20	21	22	23	24		25	26	27	28	29	30	31
R	—											QF	—			MIQ	—					MRT	—			TE	—			QFI	—		MB	QE	MIQI
W																						w1c				w1c				w1c					w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Figure 15-77. Inbound Message *n* Status Registers (IM*n*SR)

Table 15-78. IM*n*SR Field Descriptions

Bits	Name	Description
0–10	—	Reserved
11	QF	Queue full. If the queue becomes full, then this bit will be set. This bit is cleared if the message controller is disabled. (Read-only)
12–14	—	Reserved
15	MIQ	Message-In-Queue. If the queue has accumulated the number of messages specified by the IM*n*MR[MIQ_THRESH], then this bit will be set. This bit is cleared if the message controller is disabled. (Read-only)
16–20	—	Reserved
21	MRT	Message request time-out. This bit is set when the message unit has not received another message segment for a multi segment message and a time-out occurred. This bit is cleared by writing a 1.
22–23	—	Reserved
24	TE	Transaction error. This bit is set when there is an internal error condition occurs during the message operation. This bit is cleared by writing a 1. For proper operation, this field should only be modified when the inbound message controller is not enabled.
25–26	—	Reserved
27	QFI	Queue full interrupt. If the queue is full and if the QFIE bit in the mode register is set, then this bit will be set and an interrupt generated.This bit is cleared by writing a 1.
28	—	Reserved
29	MB	Mailbox busy. When set, indicates that a message operation is currently in progress. This bit is cleared as a result of an error or the message operation is finished. (Read-only)

Table 15-78. IM*n*SR Field Descriptions (continued)

Bits	Name	Description
30	QE	Queue empty. If the queue is empty, then this bit is set. This bit is also set when the message controller is disabled. (Read-only)
31	MIQI	Message-In-Queue interrupt. If the queue has accumulated the number of messages specified by the IM*n*MR[MIQ_THRESH] and if the MIQIE bit in the Mode Register is set, then this bit is set and an interrupt generated. This bit is cleared by writing a 1.

15.7.2.3 Extended Inbound Message Frame Queue Dequeue Pointer Address Registers (EIM*n*FQDPAR) and Inbound Message Frame Queue Dequeue Pointer Address Registers (IM*n*FQDPAR)

The inbound message frame queue dequeue pointer address registers contain the address for the first message in memory to be processed. Software must initialize these registers to the first frame location in memory. When a message has been processed, the processor sets IM*n*MR[MI]. The mailbox hardware then increments IM*n*FQDPAR and EIM*n*FQDPAR to point to the next frame in memory and clears the IM*n*MR[MI] bit. If the inbound message frame queue enqueue pointer and the inbound message frame queue dequeue pointer are not equal (indicating that the queue is not empty), the processor can read the next message frame from memory for processing. If the enqueue and dequeue pointers are equal after being incremented by the processor, the queue is empty and all outstanding messages have been processed.

When software initializes these registers, the queue to which they point must be aligned on a boundary equal to number of queue entries × frame size. For example, if there are eight entries in the queue and the frame size is 128 bytes, the queue must be 1024-byte aligned.

The number of queue entries is set in IM*n*MR[CIRQ_SIZ], and the frame size is set in IM*n*MR[FRM_SIZ]. See Section 15.7.2.1, “Inbound Message n Mode Registers (IMnMR)”).

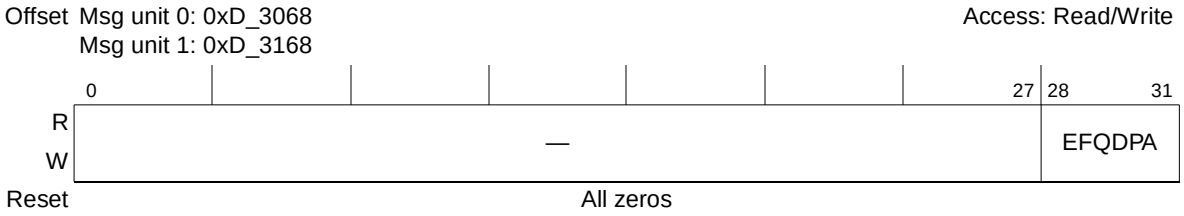


Figure 15-78. Extended Inbound Message *n* Frame Queue Dequeue Pointer Address Registers (EIM*n*FQDPAR)

Table 15-79. EIM*n*FQDPAR Field Descriptions

Bits	Name	Description
0–27	—	Reserved
28–31	EFQDPA	Extended frame queue dequeue pointer address bits. These are the highest order address bits.

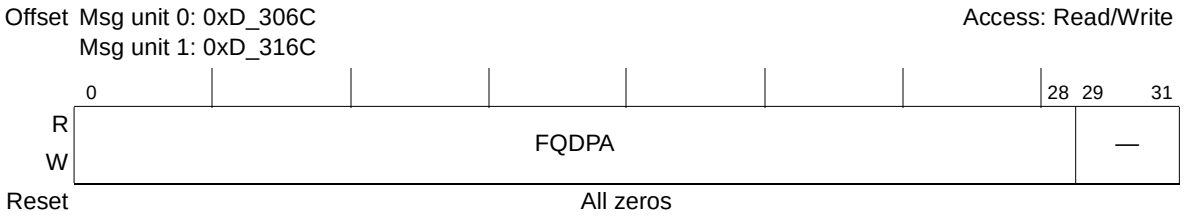


Figure 15-79. Inbound Message *n* Frame Queue Dequeue Pointer Address Registers (IM*n*FQDPAR)

Table 15-80. IM*n*FQDPAR Field Descriptions

Bits	Name	Description
0–28	FQDPA	Frame queue dequeue pointer address. Contains the address of the first message in memory to process.
29–31	—	Reserved

15.7.2.4 Extended Inbound Message Frame Queue Enqueue Pointer Address Registers (EIM*n*FQEPAR) and Inbound Message Frame Queue Enqueue Pointer Address Registers (IM*n*FQEPAR)

The inbound message frame queue enqueue pointer address registers contain the address for the next message frame in memory to be added to the queue. Software must initialize these registers to match the frame queue dequeue pointer address. When a message is received by the mailbox controller, it writes the message data to the next location in the queue (indicated by the address in IM*n*FQEPAR and EIM*n*FQEPAR) and then increments IM*n*FQEPAR and EIM*n*FQEPAR to point to the next frame location in memory. This can result in a number of actions:

- If the enqueue and dequeue pointers match, the queue is now full and the mailbox controller does not accept any more incoming messages, returning RETRY responses to the sending devices until the queue is no longer full. If the IM*n*MR[QFIE] bit is set, then the IM*n*MR[QFI] bit is set and an interrupt is generated.
- If the enqueue and dequeue pointers were the same before the register was read, the queue has transitioned from empty to not empty. If IM*n*MR[MIQIE] is set, IM*n*SR[MIQI] is set, and an interrupt is generated.

When software initializes these registers, the queue to which they point must be aligned on a boundary equal to number of queue entries × frame size. For example, if there are eight entries in the queue and the frame size is 128 bytes, the queue must be 1024-byte aligned.

The number of queue entries is set in IM*n*MR[CIRQ_SIZ], and the frame size is set in IM*n*MR[FRM_SIZ]. See Section 15.7.2.1, “Inbound Message n Mode Registers (IMnMR)”).

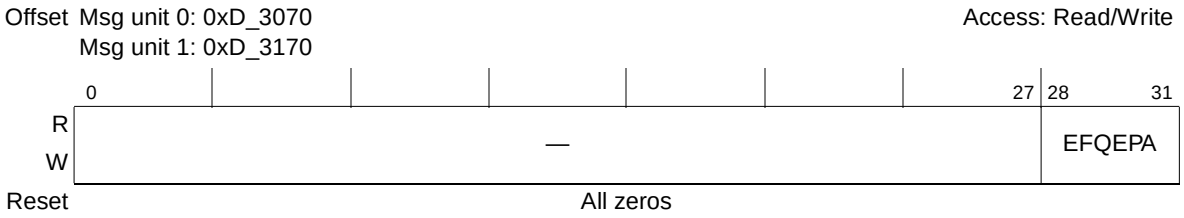


Figure 15-80. Extended Inbound Message *n* Frame Queue Enqueue Pointer Address Registers (EIM*n*FQEPAR)

Table 15-81. EIM*n*FQEPAR Field Descriptions

Bits	Name	Description
0–27	—	Reserved
28–31	EFQEPA	Extended frame queue enqueue pointer address bits. These are the highest order address bits. For proper operation, this field should only be modified when the inbound message controller is not enabled.

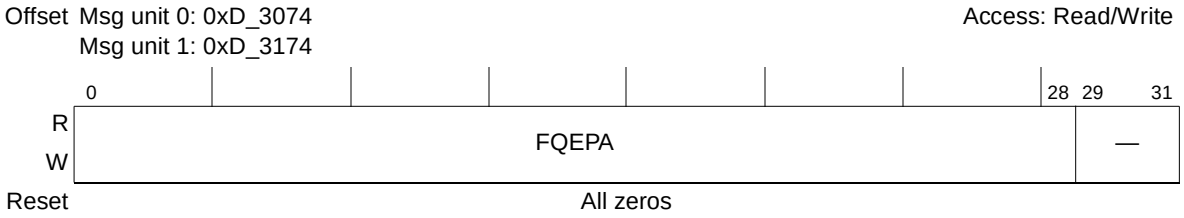


Figure 15-81. Inbound Message *n* Frame Queue Enqueue Pointer Address Registers (IM*n*FQEPAR)

Table 15-82. IM*n*FQEPAR Field Descriptions

Bits	Name	Description
0–28	FQEPA	Frame queue enqueue pointer address. Contains the address of the next message frame to be added to the queue. For proper operation, this field should only be modified when the inbound message controller is not enabled.
29–31	—	Reserved

15.7.2.5 Inbound Message *n* Maximum Interrupt Report Interval Registers (IM*n*MIRIR)

The maximum interrupt interval register contains a time-out timer value to define the maximum amount of time that the mailbox controller is to wait from transitioning from not empty to signaling an interrupt (if enabled) to the local processor if the IM*n*MR[MIQ_THRESH] limit is not reached. The reset value is the maximum time-out interval. The timer decrements at one-half the platform clock rate; thus at a platform clock rate of 400 MHz, for example, the maximum time-out value is approximately 83.9 ms.

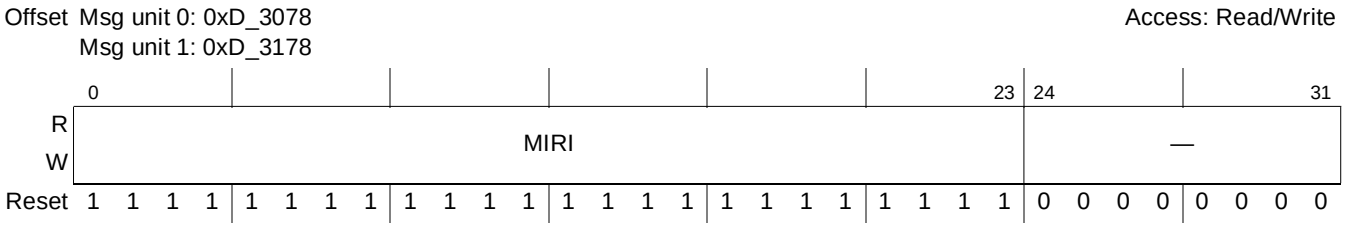


Figure 15-82. Inbound Message *n* Maximum Interrupt Report Interval Registers (IM*n*MIRIR)

Table 15-83. IM*n*MIRIR Field Descriptions

Bits	Name	Description
0–23	MIRI	Maximum interrupt report interval. Maximum message-in-queue to interrupt generation time. A value of 0 disables the time-out timer. For proper operation, this field should only be modified when the inbound message controller is not enabled.
24–31	—	Reserved

15.7.3 Outbound RapidIO Doorbell Controller Registers

These registers control the outbound RapidIO doorbell controller. The following sections provide descriptions of these registers.

15.7.3.1 Outbound Doorbell Mode Register (ODMR)

The outbound mode register allows software to start a doorbell operation and to control various doorbell operation characteristics.

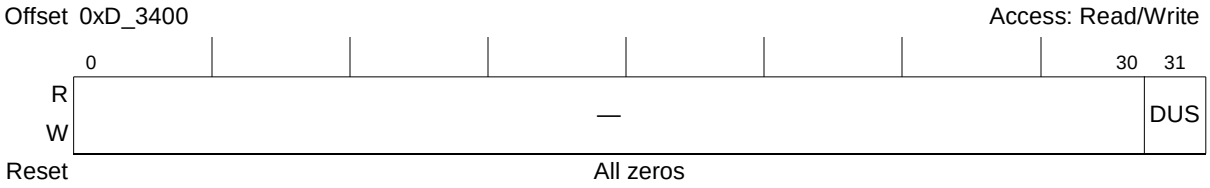


Figure 15-83. Outbound Mode Register (ODMR)

Table 15-84. ODMR Field Descriptions

Bits	Name	Description
0–30	—	Reserved
31	DUS	Doorbell unit start. Direct mode - A 0 to 1 transition when the doorbell unit is not busy (DUB bit is 0) starts the doorbell unit. A 1 to 0 transition has no effect.

15.7.3.2 Outbound Doorbell Status Register (ODSR)

The outbound status register reports various doorbell unit conditions during and after a doorbell operation. For some bits, writing a 1 clears the bit.

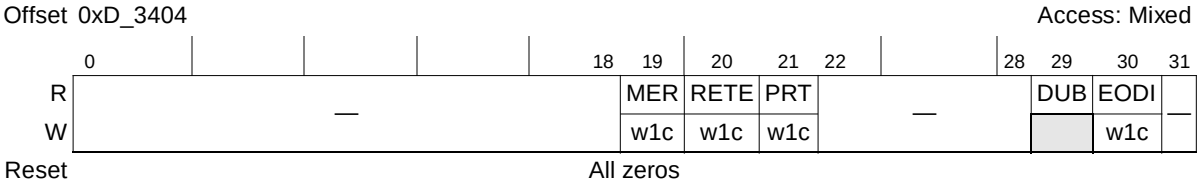


Figure 15-84. Outbound Doorbell Status Register (ODSR)

Table 15-85. ODSR Field Descriptions

Bits	Name	Description
0–18	—	Reserved
19	MER	Message error response. This bit is set when an ERROR response is received from the doorbell target. The Error response received field indicates value of the error response status bits when an error response is received.This bit is cleared by writing a 1. For proper operation, this bit should only be cleared when a doorbell operation is not in progress.
20	RETE	Retry error threshold exceeded. This bit is set when the doorbell unit has been unable to complete a doorbell operation because the retry error threshold value has been exceeded due to RapidIO retry response. This bit is cleared by writing a 1. For proper operation, this bit should only be cleared when a doorbell operation is not in progress.
21	PRT	Packet response time-out. This bit is set when the doorbell unit has been unable to complete a doorbell operation and a packet response time-out occurred. This bit is cleared by writing a 1. For proper operation, this bit should only be cleared when a doorbell operation is not in progress.
22–28	—	Reserved
29	DUB	Doorbell unit busy. When set, indicates that a doorbell operation is currently in progress. This bit is cleared as a result of an error or when the doorbell operation is finished. (Read-only)
30	EODI	End-of-doorbell interrupt. After finishing this doorbell operation, if the EODIE bit in the destination attributes register is set, then this bit will be set and an interrupt generated. This bit is cleared by writing a 1. For proper operation, this bit should only be cleared when a doorbell operation is not in progress.
31	—	Reserved

15.7.3.3 Outbound Doorbell Destination Port Register (ODDPR)

The destination port register indicates the RapidIO Destination ID and mailbox to which the doorbell unit controller is to send data. The software must ensure that this is a valid port in the receiving device.

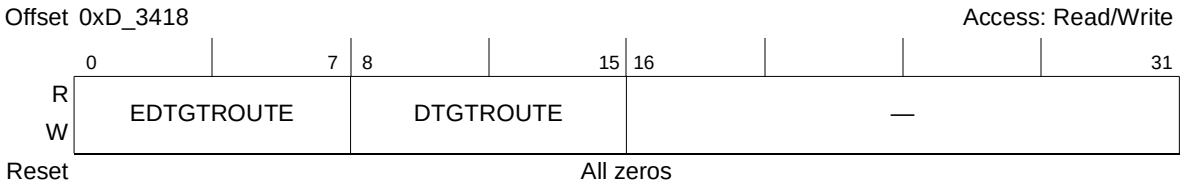


Figure 15-85. Outbound Doorbell Destination Port Registers (ODDPR)

Table 15-86. ODDPR Field Descriptions

Bits	Name	Description
0–7	EDTGTROUTE	Extended destination target route. Most significant byte of a 16-bit target route when operating in large transport mode. Reserved when operating in small transport mode. For proper operation, this field should only be modified when a doorbell operation is not in progress.
8–15	DTGTROUTE	Destination target route. Contains the target route field of the transaction (Device ID of the target). For proper operation, this field should only be modified when a doorbell operation is not in progress.
16–31	—	Reserved

15.7.3.4 Outbound Doorbell Destination Attributes Register (ODDATR)

The outbound destination attributes register contains the transaction attributes to be used for the doorbell operation.

Offset 0xD_341C Access: Read/Write

	0	1	2	3	4	5	6	7	8	11	12	15	16	23	24	31
R W	—		EODIE	—	DTFLOWLVL		—		DTGTINT		—		INFO_MSB		INFO_LSB	
Reset	All zeros															

Figure 15-86. Outbound Doorbell Destination Attributes Register (ODDATR)

Table 15-87. ODDATR Field Descriptions

Bits	Name	Description
0	—	Reserved
1	—	Reserved, hard wired to 0
2	EODIE	End-of-Doorbell interrupt enable. When set, generates an interrupt when the current doorbell operation is finished. For proper operation, this field should only be modified when a doorbell operation is not in progress.
3	—	Reserved
4–5	DTFLOWLVL	Transaction flow level. 00 Lowest priority transaction flow 01 Next highest priority transaction flow 10 Highest priority transaction flow 11 Reserved For proper operation, this field should only be modified when a doorbell operation is not in progress.
6–7	—	Reserved
8–11	DTGTINT	Target interface. The value of this field should always be set to RapidIO (0x0).
12–15	—	Reserved

Table 15-87. ODDATR Field Descriptions (continued)

Bits	Name	Description
16–23	INFO_MSB	Most significant byte of the doorbell “info” field. For proper operation, this field should only be modified when a doorbell operation is not in progress.
24–31	INFO_LSB	Least significant byte of the doorbell “info” field. For proper operation, this field should only be modified when a doorbell operation is not in progress.

15.7.3.5 Outbound Doorbell Retry Error Threshold Configuration Register (ODRETCR)

The retry error threshold configuration register controls the number of times that the doorbell unit attempts to complete a doorbell operation before reporting an error and moving on to the next task, if one is available. Failures to complete an operation are indicated by receiving a RapidIO logical layer RETRY response from the target. If the programmed count is exceeded, the ODSR[RETE] bit is set.

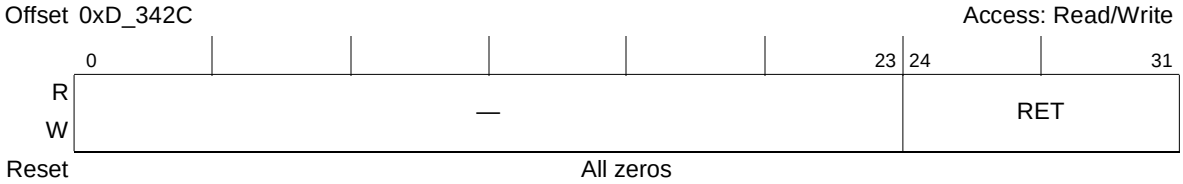


Figure 15-87. Outbound Doorbell Retry Error Threshold Configuration Register (ODRETCR)

Table 15-88. ODRETCR Field Descriptions

Bits	Name	Description
0–23	—	Reserved
24–31	RET	Retry error threshold. This value is the number of times that the doorbell unit attempts to transmit a doorbell. 0x00 Disabled 0x01 Doorbell transmitted only 1 time 0x02 Doorbell transmitted up to 2 times ... 0xFF Doorbell transmitted up to 255 times For proper operation, this field should only be modified when a doorbell operation is not in progress.

15.7.4 Inbound RapidIO Doorbell Controller

These registers control the inbound RapidIO doorbell controller. The following sections provide descriptions of these registers.

15.7.4.1 Inbound Doorbell Mode Register (IDMR)

The doorbell mode register allows software to enable the doorbell controller to control various doorbell operation characteristics. The IDMR is shown in Figure 15-88.

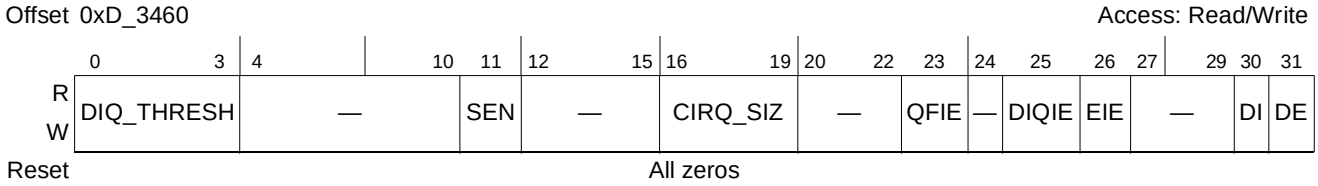


Figure 15-88. Inbound Doorbell Mode Register (IDMR)

Table 15-89. IDMR Field Descriptions

Bits	Name	Description
0–3	DIQ_THRESH	Doorbell-in-queue threshold. Determines the number of doorbells to be accumulated in the doorbell queue before the Doorbell-in-queue bit is set (IDSR[DIQ]). Undefined operation results if the actual number of entries defined by the doorbell-in-queue threshold is set greater than or equal to the actual size of the doorbell queue. 0000 1 0111 128 0001 2 1000 256 0010 4 1001 512 0011 8 1010 1024 0100 16 1011 reserved 0101 32 ... 0110 64 1111 reserved For proper operation, this field should only be modified when the doorbell controller is not enabled.
4–10	—	Reserved
11	SEN	Snoop enable. When set enables snooping the local processor when writing messages into memory. For proper operation, this field should only be modified when the doorbell controller is not enabled.
12–15	—	Reserved
16–19	CIRQ_SIZ	Circular doorbell queue size. Determines the number of doorbell entries that can be place on the circular queue. CIRQ_SIZ × 8 bytes determine the maximum contiguous memory space allocated to the doorbell unit. For proper operation, this field should only be modified when the doorbell controller is not enabled. 0000 2 0111 256 0001 4 1000 512 (4-kbyte page boundary) 0010 8 1001 1024 0011 16 1010 2048 0100 32 1011 reserved 0101 64 ... 0110 128 1111 reserved For proper operation, this field should only be modified when the doorbell controller is not enabled.
20–22	—	Reserved
23	QFIE	Queue full interrupt enable. 0 No QF interrupt is generated 1 Generates an interrupt when the queue is full (that is, the enqueue and dequeue pointers are equal after the dequeue pointer was incremented by the doorbell controller). If this bit is set and IDSR[QF] is a 1, IDSR[QFI] is asserted.
24	—	Reserved

Table 15-89. IDMR Field Descriptions (continued)

Bits	Name	Description
25	DIQIE	Doorbell in queue interrupt enable. When set, allows the doorbell in queue interrupt to assert. The doorbell in queue interrupt cannot be asserted if this bit is cleared. Doorbell in queue interrupt enable. 0 No DIQ interrupt is generated 1 Generates an interrupt when the DIQ bit is set (IDSR[DIQ]) If this bit is set and IDSR[DIQ] is a 1, IDSR[DIQI] will assert. If this bit is set and IDMR[DI] is also set simultaneously, IDSR[DIQI] indicates reflects the value of DIQ after the increment.
26	EIE	Error interrupt enable. When set, generates the port-write/error interrupt when a transfer error (IDSR[TE]) event occurs. No port-write/error interrupt is generated if this bit is cleared.
27–29	—	Reserved
30	DI	Doorbell increment. Software sets this bit after processing an inbound doorbell. Hardware then increments the ODQEPAR and clears this bit. Always reads as 0.
31	DE	Doorbell enable. 0 Doorbell disabled 1 The doorbell has been initialized and can service incoming doorbell operations.

15.7.4.2 Inbound Doorbell Status Register (IDSR)

The doorbell status register reports various doorbell conditions after a doorbell operation. Writing a 1 to the corresponding set bit will clear the bit. The IDSR is shown in Figure 15-89.

Offset 0xD_3464 Access: Mixed

	0										10	11	12			14	15	16							23	24	25	26	27	28	29	30	31
R	—											QF	—				DIQ	—								TE	—		QFI	—	DB	QE	DIQI
W																										w1c			w1c				w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Figure 15-89. Inbound Doorbell Status Register (IDSR)

Table 15-90. IDSR Field Descriptions

Bits	Name	Description
0–10	—	Reserved
11	QF	Queue full. If the queue is full, then this bit is set. This bit clears if the doorbell controller is disabled. (Read-only)
12–14	—	Reserved
15	DIQ	Doorbell-In-Queue. If the queue has accumulated the number of doorbells specified by DIQ_THRESH, then this bit is set. Also, if a valid entry pointed to by the dequeue address pointers has not been serviced within the configured maximum interval, this bit is set. This bit clears if the doorbell controller is disabled. (Read-only)
16–23	—	Reserved
24	TE	Transaction error. This bit is set when an internal error occurs during the doorbell operation. (Bit reset, write 1 clear). For proper operation, this field should only be modified when the doorbell controller is not enabled.

Table 15-90. IDSR Field Descriptions (continued)

Bits	Name	Description
25–26	—	Reserved
27	QFI	Queue full interrupt. If the queue becomes full and the QFIE bit in the Mode Register is set, then this bit is set and an interrupt generated. This bit is cleared by writing a 1.
28	—	Reserved
29	DB	Doorbell busy. When set, indicates that a doorbell has been received and the doorbell queue is currently being written. This bit clears when the write to memory has finished. Disabling the doorbell controller does not affect this bit. (Read-only)
30	QE	Queue empty. If the queue is empty, then this bit is set. This bit is also set if the doorbell controller is disabled.(Read-only)
31	DIQI	Doorbell-In-Queue interrupt. If DIQ is set, then this bit is set and an interrupt generated. This bit is cleared by writing a 1.

15.7.4.3 Extended Inbound Doorbell Queue Dequeue Pointer Address Register (EIDQDPAR) and Inbound Doorbell Queue Dequeue Pointer Address Register (IDQDPAR)

The inbound doorbell queue dequeue pointer address registers (EIDQDPAR and IDQDPAR) contain the double-word address for the first doorbell in memory to be processed. Software must initialize these registers to the first doorbell entry location in memory. When the current doorbell has been processed, the processor sets IDMR[DI]. The doorbell hardware then increments the doorbell queue dequeue pointer address register to point to the next doorbell entry in memory and clears IDMR[DI]. When processing multiple doorbells, instead of setting IDMR[DI] for each doorbell, the processor can write these registers directly.

If the doorbell queue enqueue pointer and the doorbell queue dequeue pointer are not equal (indicating that the queue is not empty), the processor can read the next doorbell from memory for processing. If the enqueue and dequeue pointers are equal after being incremented by the processor, the queue is empty and all outstanding doorbells have been processed.

The EIDQDPAR is shown in Figure 15-90 and the IDQDPAR is shown in Figure 15-91.

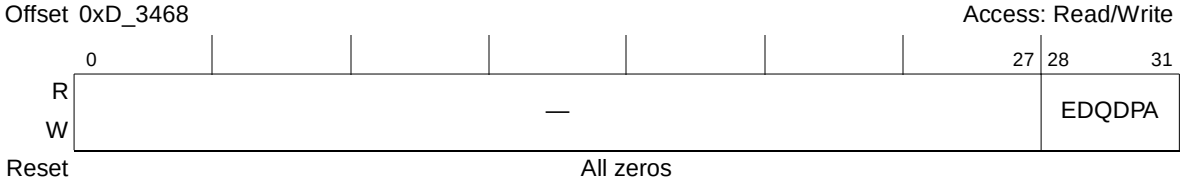


Figure 15-90. Extended Inbound Doorbell Queue Dequeue Pointer Address Registers (EIDQDPAR)

Table 15-91. EIDQDPAR Field Descriptions

Bits	Name	Description
0–27	—	Reserved
28–31	EDQDPA	Extended doorbell queue dequeue pointer address bits. These are the highest order address bits.

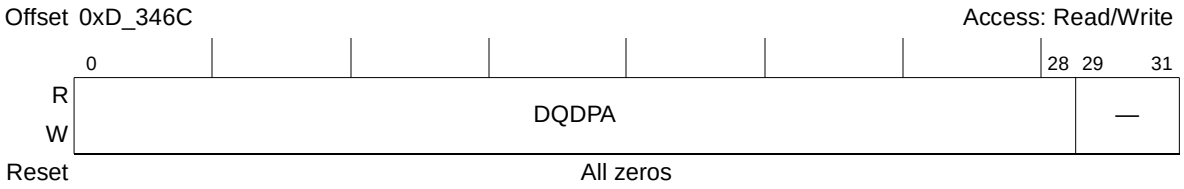


Figure 15-91. Inbound Doorbell Queue Dequeue Pointer Address Registers (IDQDPAR)

Table 15-92. IDQDPAR Field Descriptions

Bits	Name	Description
0–28	DQDPA	Doorbell queue dequeue pointer address. Contains the double-word address of the first doorbell in memory to process.
29–31	—	Reserved

15.7.4.4 Extended Inbound Doorbell Queue Enqueue Pointer Address Register (EIDQEPAR) and Inbound Doorbell Queue Enqueue Pointer Address Register (IDQEPAR)

The doorbell queue enqueue pointer address registers (EIDQEPAR and IDQEPAR) contain the double-word address for the next doorbell entry in memory to be added to the queue. Software must initialize IDQEPAR and EIDQEPAR to match the doorbell queue dequeue pointer address. When a doorbell packet is received by the doorbell controller, it writes the doorbell information to the next location in the queue (indicated by the address in IDQEPAR and EIDQEPAR) and then increments IDQEPAR and EIDQEPAR to point to the next doorbell location in memory. This can result in a number of actions:

- If the enqueue and dequeue pointers match, then the queue is now full and the doorbell controller does not accept any more incoming doorbell packets, returning RETRY responses to the sending devices until the queue is no longer full. If the IDMR[QFIE] bit is set, then the IDSR[QFI] is set and the interrupt is generated.
- If the enqueue and dequeue pointers were the same before receiving the doorbell, the queue has transitioned from empty to not empty. When the number of doorbells received matches the configured threshold, the IDSR[DIQ] bit is set. If the DRM[DIQIE] bit is set, then the IDSR[DIQI] bit is also set and the inbound doorbell interrupt is generated.

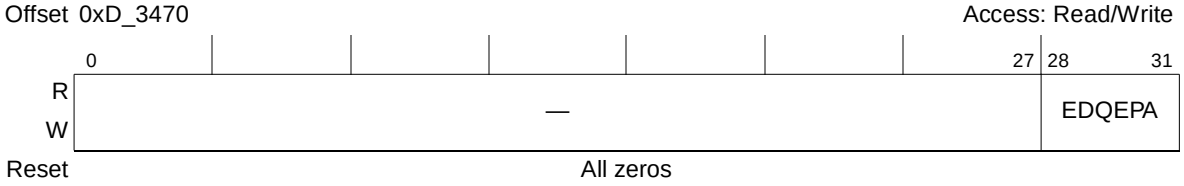


Figure 15-92. Extended Inbound Doorbell Queue Enqueue Pointer Address Register (EIDQEPAR)

Table 15-93. EIDQEPAR Field Descriptions

Bits	Name	Description
0–27	—	Reserved
28–31	EDQEPA	Doorbell queue enqueue pointer address bits. These are the highest order address bits. This field can only be written when the doorbell controller is disabled or undefined operation results.

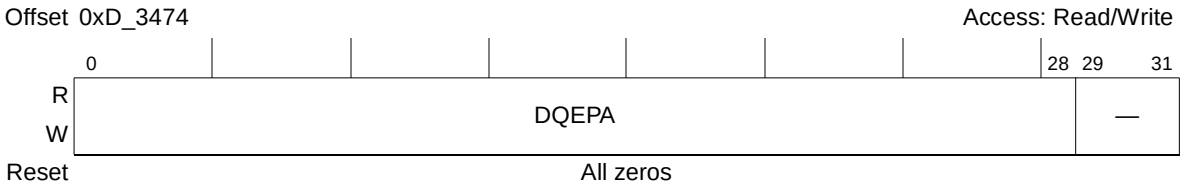


Figure 15-93. Inbound Doorbell Queue Enqueue Pointer Address Register (IDQEPAR)

Table 15-94. IDQEPAR Field Descriptions

Bits	Name	Description
0–28	DQEPA	Doorbell queue enqueue pointer address. Contains the double-word address of the next doorbell location to be added to the queue. This field can only be written when the doorbell controller is disabled or undefined operation results.
29–31	—	Reserved

15.7.4.5 Inbound Doorbell Maximum Interrupt Report Interval Register (IDMIRIR)

The maximum interrupt interval register contains a time-out timer value to define the maximum amount of time that a doorbell entry can be at the head of the doorbell queue before generating an interrupt (if enabled) if the DIQ_THRESH limit is not reached. The reset value is the maximum time-out interval, and represents between 3 and 6 seconds.

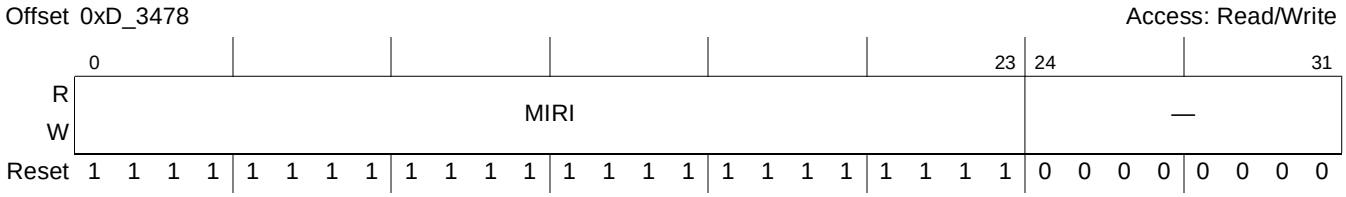


Figure 15-94. Inbound Doorbell Maximum Interrupt Report Interval Register (IDMIRIR)

Table 15-95. IDMIRIR Field Descriptions

Bits	Name	Description
0–23	MIRI	Maximum interrupt report interval - Maximum doorbell-in-queue to interrupt generation time. A value of 0 disables the timer. This field can only be written when the doorbell controller is disabled or undefined operation results.
24–31	—	Reserved

15.7.5 RapidIO Port-Write Registers

These registers control the RapidIO port-write unit. The following sections provide partial descriptions of these registers.

15.7.5.1 Inbound Port-Write Mode Register (IPWMR)

The port-write mode register allows software to enable the port-write controller and to control various port-write operation characteristics. The IPWMR is shown in Figure 15-95.

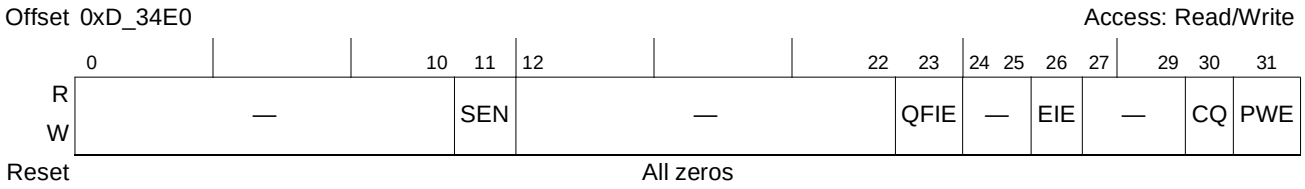


Figure 15-95. Inbound Port-Write Mode Register (IPWMR)

Table 15-96. IPWMR Field Descriptions

Bits	Name	Description
0–10	—	Reserved
11	SEN	Snoop enable. When set enables snooping the local processor when writing port-write data payload into memory. For proper operation, this field should only be modified when the port-write controller is not enabled.
12–22	—	Reserved
23	QFIE	Queue full interrupt enable. When seKilobytes generates the error/port-write interrupt if the queue is full (that is, the controller has written the port-write data payload into memory). No error/port-write interrupt is generated if this bit is cleared. For proper operation, this field should only be modified when the port-write controller is not enabled.
24–25	—	Reserved
26	EIE	Error Interrupt enable. When set, generates the port-write/error interrupt when a transfer error (IPW0SR[TE]) event occurs. No port-write/error interrupt is generated if this bit is cleared.
27–29	—	Reserved
30	CQ	Clear queue. Software sets this bit after processing an inbound port-write operation. Hardware clears the queue full bit (IPWSR[QF]), clears this bit, and allows another port-write to be received. This bit is always read as a 0.
31	PWE	Port write enable. If this bit is set the port-write controller has been initialized and can service an incoming port-write operation.

15.7.5.2 Inbound Port-Write Status Register (IPWSR)

The port-write status register reports various port-write conditions. The IPWSR is shown in Figure 15-96.

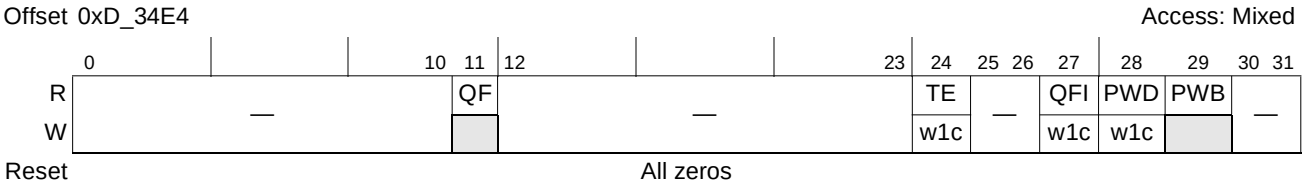


Figure 15-96. Inbound Port-Write Status Register (IPWSR)

Table 15-97. IPWSR Field Descriptions

Bits	Name	Description
0–10	—	Reserved
11	QF	Queue full. If the queue becomes full, then this bit is set. This bit is cleared when the clear queue bit is set (IPWMR[CQ]) and the queue is not full. This bit is also cleared if the port-write controller is disabled. (Read-only)
12–23	—	Reserved
24	TE	Transaction error. This bit is set when there an internal error condition occurs during the port-write operation. (Bit reset, write 1 clear). For proper operation, this field should only be modified when the port-write controller is not enabled.
25–26	—	Reserved
27	QFI	Queue full interrupt. If the queue becomes full and the QFIE bit in the Mode Register is set, then this bit is set and an interrupt generated. This bit is cleared by writing a 1.
28	PWD	Port-write discarded. This bit is set if a port-write is discarded while the port-write controller is enabled but busy. This bit is cleared by writing a 1.
29	PWB	Port-write busy (Read-only). 0 This bit is cleared after the port-write payload has been written to memory. 1 Indicates that a port-write has been received and the port-write payload is currently being written to memory. Disabling the port-write controller does not affect this bit.
30–31	—	Reserved

15.7.5.3 Extended Inbound Port-Write Queue Base Address Register (EIPWQBAR) and Inbound Port-Write Queue Base Address Register (IPWQBAR)

The port-write queue base address registers (EIPWQBAR and IPWQBAR) contain the 64-byte cache line address for the port-write data payload. Software must initialize these registers to the desired location in memory. The EIPWQBAR and IPWQBAR are shown in Figure 15-97 and Figure 15-98.

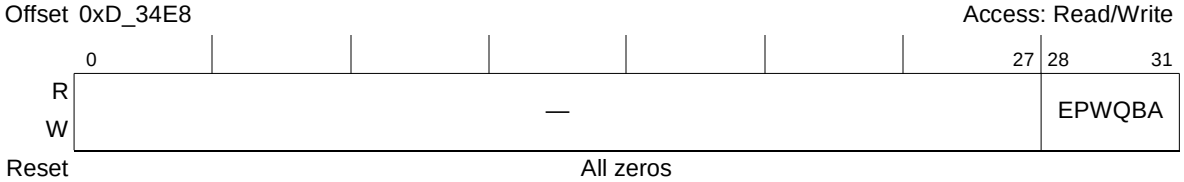


Figure 15-97. Extended Port-Write Queue Base Address Register (EIPWQBAR)

Table 15-98. EIPWQBAR Field Descriptions

Bits	Name	Description
0–27	—	Reserved
28–31	EPWQBA	Extended port-write queue base address bits. These are the highest order address bits.This field can only be written when the port-write controller is disabled or undefined operation results.

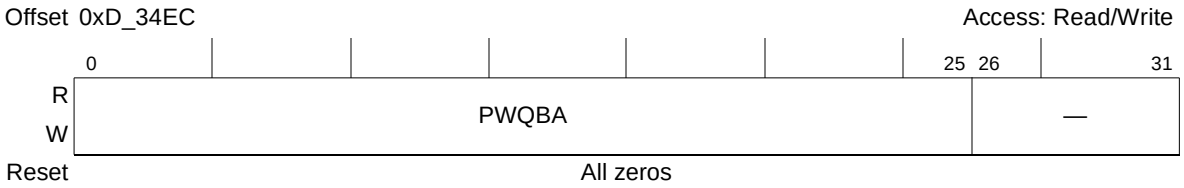


Figure 15-98. Inbound Port-Write Queue Base Address Register (IPWQBAR)

Table 15-99. IPWQBAR Field Descriptions

Bits	Name	Description
0–25	PWQBA	Port-write queue base address. Contains the cache line address of the port-write data payload. This field can only be written when the port-write controller is disabled or undefined operation results.
26–31	—	Reserved

15.8 Functional Description

15.8.1 RapidIO Transaction Summary

The RapidIO endpoint on this device supports all RapidIO I/O transactions, all RapidIO message passing transactions, and a few RapidIO GSM transactions.

Table 15-100 summarizes the RapidIO I/O transactions that are supported by this RapidIO endpoint.

Table 15-100. RapidIO I/O Transactions

IO Transaction	ftype	ttype	Status	Description
NREAD	0010	0100	NA	Read
ATOMIC inc		1100		Read and post-increment with response
ATOMIC dec		1101		Read and post-decrement with response
ATOMIC set		1110		Read and set to all-1s with response
ATOMIC clr		1111		Read and set to all-0s with response
NWRITE	0101	0100		Write with no response
NWRITE_R		0101		Write with response
SWRITE	0110	N/A		Streaming-Write
MAINT read	1000	0000		Maintenance read
MAINT write		0001		Maintenance write
MAINT read response		0010	0000	Done maintenance read response
			0111	Error response
MAINT write response		0011	0000	Done maintenance write response
			0111	Error response
MAINT port-write		0100	NA	Maintenance port-write[1]
RESPONSE without data	1101	0000	0000	I/O done response
			0111	I/O error response
RESPONSE with data		1000	0000	I/O done response with data

1). Limited to inbound RapidIO packets only

Table 15-101 summarizes the RapidIO message passing transactions that are supported by this RapidIO endpoint.

Table 15-101. RapidIO Message Passing Transactions

MSG Transaction	ftype	ttype	status	Description
DOORBELL	1010	NA	NA	Doorbell
MESSAGE	1011			Message
RESPONSE without data	1101	0000	0000	Doorbell done response
			0011	Doorbell retry response
			0111	Doorbell error response
		0001	0000	Message done response
			0011	Message retry response
			0111	Message error response

Table 15-102 summarizes the RapidIO GSM transactions that are supported by this RapidIO endpoint.

Table 15-102. RapidIO GSM Transactions

GSM Transaction	ftype	ttype	status	Description
IO_READ_HOME	0010	0010	NA	I/O Read Home[1]
FLUSH w/data	0101	0001		GSM Flush with data[1]
RESPONSE without data	1101	0000	0000	GSM done response
			0011	GSM retry response
			0101	GSM done intervention response
			0111	GSM error response
RESPONSE with data		1000	0000	GSM done response
			0001	GSM data only response

1). Limited to RapidIO packet generation only

15.8.2 RapidIO Packet Format Summary

Table 15-103 summarizes the small transport field packet formats for supported RapidIO transaction types for LP-Serial operation.

Note that this RapidIO endpoint limits configuration read and write requests to 32-bit data accesses.

Table 15-103. RapidIO Small Transport Field Packet Format

Transaction	Bits															
	32							32					16			64
	5	3	2	2	4	8	8	4	4	8	8	8	13	1	2	
NREAD, ATOMIC (inc/dec/inc/dec), IO_READ_HOME	ack ID	rsv = 0	prio	tt	ftype	dest ID	src ID	ttype	rdsize	src TID	addr			wd ptr	xam bs	NA
NWRITE_R, FLUSH w/data	ack ID	rsv = 0	prio	tt	ftype	dest ID	src ID	ttype	wrsize	src TID	addr			wd ptr	xam bs	dword 0 -> dword n
NWRITE	ack ID	rsv = 0	prio	tt	ftype	dest ID	src ID	ttype	wrsize	don't care	addr			wd ptr	xam bs	dword 0 -> dword n
SWRITE	ack ID	rsv = 0	prio	tt	ftype	dest ID	src ID	addr(29), rsv(1) = 0, xambs(2)					dword 0 -> dword n			
MAINT read	ack ID	rsv = 0	prio	tt	ftype	dest ID	src ID	ttype	rd/wrsize	src TID	hop cnt	cfg offset		wd ptr	rsv = 0	NA
MAINT write	ack ID	rsv = 0	prio	tt	ftype	dest ID	src ID	ttype	rd/wrsize	src TID	hop cnt	cfg offset		wd ptr	rsv = 0	dword 0 (32-bit)
MAINT port-write	ack ID	rsv = 0	prio	tt	ftype	dest ID	src ID	ttype	rd/wrsize	rsv = 0	hop cnt	rsv = 0		wd ptr	rsv = 0	dword 0 -> dword n
MAINT response without data	ack ID	rsv = 0	prio	tt	ftype	dest ID	src ID	ttype	status	tar TID	hop cnt	rsv = 0				NA

Table 15-103. RapidIO Small Transport Field Packet Format (continued)

Transaction	Bits															
	32							32					16			64
	5	3	2	2	4	8	8	4	4	8	8	8	13	1	2	
MAINT response with data	ack ID	rsv = 0	prio	tt	ftype	dest ID	src ID	ttype	status	tar TID	hop cnt	rsv = 0				dword 0 (32-bit)
RESPONSE without data	ack ID	rsv = 0	prio	tt	ftype	dest ID	src ID	ttype	status	tar TID	NA					
RESPONSE without data for message	ack ID	rsv = 0	prio	tt	ftype	dest ID	src ID	ttype	status	letter(2), mbox(2), msgseg(4)	NA					
RESPONSE with data	ack ID	rsv = 0	prio	tt	ftype	dest ID	src ID	ttype	status	tar TID	dword 0 -> dword n					
DOORBELL	ack ID	rsv = 0	prio	tt	ftype	des tID	src ID	rsv = 0		src TID	Info-msb	Info-lsb	NA			
MESSAGE	ack ID	rsv = 0	prio	tt	ftype	dest ID	src ID	msglen(4), ssize(4), letter(2), mbox(2), msgseg(4)			dword 0 -> dword n					

The large transport field packet formats extends the destination and source IDs to 16-bits each.

15.8.3 RapidIO Control Symbol Summary

Table 15-104 summarizes the 1x/4x LP-Serial control symbols and their format. Refer to the *RapidIO Interconnect Specification, Revision 1.2, Part IV: Physical Layer 1x/4x LP-Serial Specifications, Chapter 4, PCS and PMA Layers,* for 8B/10B data and special (/PD/, /SC/, idle, sync, skip, align) characters. The 32-bit LP-Serial control symbol is comprised of the eight bit special character and the 24-bit control symbol format.

Table 15-104. 1x/4x LP-Serial Control Symbol Format

Bits						Description
24						
stype0	param0	param1	stype1	cmd	CRC	
3	5	5	3	3	5	
000	pkt_ackID	buf_stat	-		crc	Packet accepted
001	pkt_ackID	buf_stat	-		crc	Packet retry
010	pkt_ackID	cause	-		crc	Packet not accepted cause: 00001: Received unexpected ackID on packet 00010: Received a control symbol with bad CRC 00011: Non-maintenance packet reception is stopped 00100: Received packet with bad CRC 00101: Received invalid character or a valid but illegal character 11111: General error

Table 15-104. 1x/4x LP-Serial Control Symbol Format (continued)

Bits						Description
24						
stype0	param0	param1	stype1	cmd	CRC	
3	5	5	3	3	5	
100	ackID_stat	buf_stat	-		crc	Status ackID_stat: 00000: Expecting ackID 0 00001: Expecting ackID 1 00010: Expecting ackID 2 00011: Expecting ackID 3 00100: Expecting ackID 4 00101: Expecting ackID 5 00110: Expecting ackID 6 00111: Expecting ackID 7
110	ackID_stat	port_stat	-		crc	Link-response ackID_stat: 00000: Expecting ackID 0 00001: Expecting ackID 1 00010: Expecting ackID 2 00011: Expecting ackID 3 00100: Expecting ackID 4 00101: Expecting ackID 5 00110: Expecting ackID 6 00111: Expecting ackID 7 port_stat: 00010: Error; unrecoverable 00100: Retry stopped 00101: Error stopped 10000: OK
-			000	000	crc	Start of packet
-			001	000	crc	Stomp
-			010	000	crc	End of packet
-			011	000	crc	Restart from retry
-			100	cmd	crc	Link request cmd: 011: Reset the receiving device 100: Return input port status; functions as a restart-from-error control symbol under error conditions
			101	000	crc	Multicast-event
-			111	000	crc	NOP (ignore)

15.8.4 Accessing Configuration Registers Using RapidIO Packets

15.8.4.1 Inbound Maintenance Accesses

There are two suggested methods by which RapidIO transactions can target RapidIO configuration register space in local memory.

The first method is based on RapidIO NREAD and NWRITE_R requests hitting a RapidIO address window defined by the local configuration space base address command and status register (LCSBACSR). See Section 15.6.1.11, “Local Configuration Space Base Address 1 Command and Status Register (LCSBA1CSR),” for details.

If external configuration accesses are disabled (LLCR[ECRAB] = 1), any configuration access through the LCSBACSR window is denied. A 32-bit data payload of all zeros is returned for a non-maintenance configuration read.

The second method is based on a RapidIO MAINT requests. This method allows an external device limited access to local RapidIO configuration register space only. Any maintenance access beyond the first 64 Kbytes (lower 64 Kbytes contain RapidIO architecture registers; upper 64 Kbytes contain RapidIO implementation registers) of RapidIO configuration register space, is denied. A 32-bit data payload of all zeros is returned if for a read response.

A third method, though not suggested, would use an inbound ATMU window to translate RapidIO NREAD and NWRITE_R requests to configuration accesses. This method does not support the configuration access protection features offered by the LCSBACSR window and RapidIO MAINT requests.

15.8.4.1.1 Guidelines

The RapidIO endpoint limits requests to configuration register space to 32-bit data accesses.

If the order of completion is important, inbound configuration accesses should be assumed incomplete until an appropriate response has been received. It is suggested that there be only one outstanding configuration request at a time to ensure that requests are completed in the order they were intended.

15.8.4.2 Outbound Maintenance Accesses

Outbound OCN NREAD_R or NWRITE requests can be translated to a RapidIO maintenance request if the OCN address falls within the bounds of an outbound ATMU window that is setup for generating a maintenance request. The ATMU window specifies the configuration offset, hop count, source and destination ID, and priority for the outbound RapidIO packet.

15.8.5 RapidIO Outbound ATMU

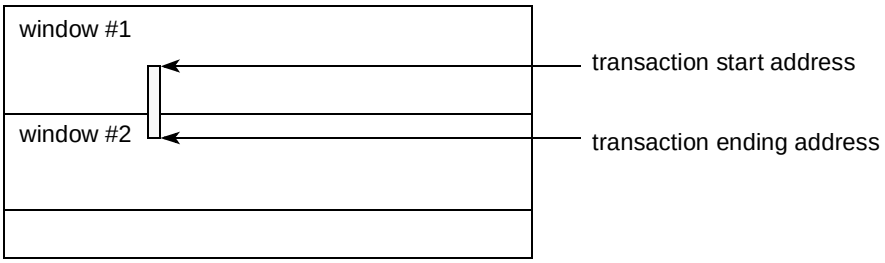
15.8.5.1 Valid Hits to Multiple ATMU Windows

If a request hits multiple ATMU windows, window 1 has the highest priority of the nine outbound ATMU windows (windows 1-8, default). Window 2 is given the next higher priority and is followed by windows 3 through 8. The default window has the lowest priority.

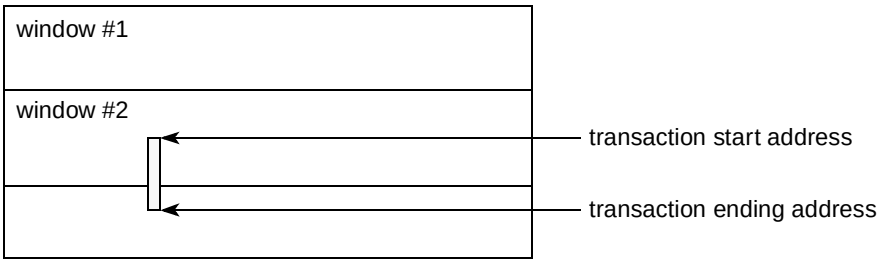
If a request hits (base address match) multiple ATMU windows and the transaction's end address is contained within the boundary of each hit window and does not extend into another ATMU window, the translation window is the highest priority window that is hit.

If a lower priority window is programmed to lie entirely within a higher priority window, then it is possible for a transaction to cross window boundaries. Although not a practical programming application, the RapidIO endpoint handles this as follows:

1. If a request hits (base address match) an ATMU window (1-8) and the transaction's end address extends into another ATMU window with lower priority but is still contained within the boundary of the hit window, the translation window is the hit window.



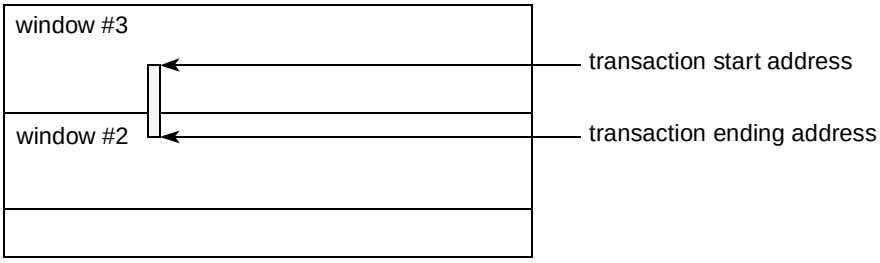
2. If a request hits (base address match) multiple ATMU windows (1–8) and the transaction's end address extends beyond the boundary of a lower priority hit window but is still contained within the boundary of a higher priority hit window, the translation window is the highest priority window that is hit.



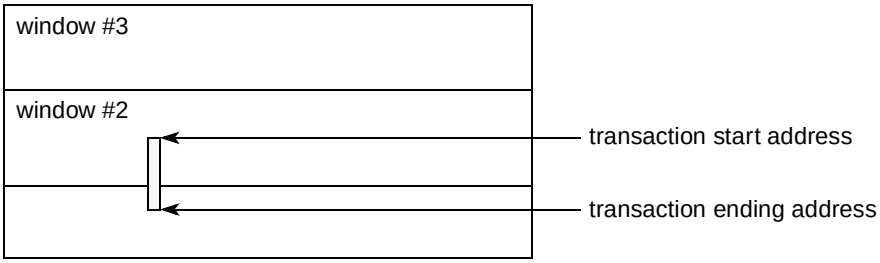
15.8.5.2 Window Boundary Crossing Errors

If a higher priority window is programmed to lie entirely within a lower priority window, then it is possible for a transaction to cross window boundaries. The RapidIO endpoint handles this as follows:

1. If a request hits (base address match) an ATMU window (1-8, default) and the transaction's end address extends into another ATMU window with higher priority, an ATMU crossed boundary error is generated and logged. The outbound request is discarded.

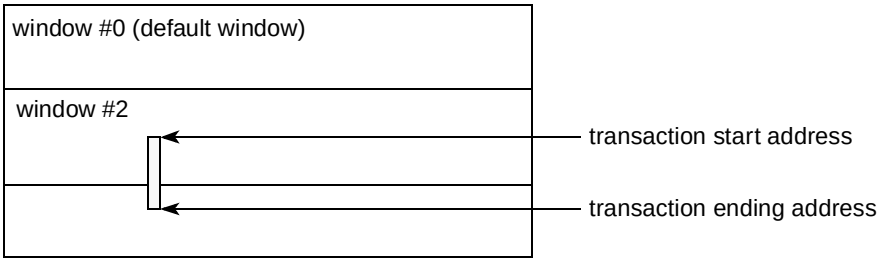


2. If a request hits multiple ATMU windows (1-8, default) and transaction's end address extends beyond the boundary of a higher priority hit window, an outbound ATMU crossed boundary error is generated and logged. The outbound request is discarded.



Other window boundary crossing errors are:

1. If a request hits (base address match) an ATMU window (1-8) and the transaction's end address exceeds the size of the window, an outbound ATMU crossed boundary error is generated and logged. The outbound request is discarded.



Boundary crossing errors (outbound ATMU boundary crossing, segment boundary crossing, and subsegment boundary crossing) are logged in the LTLEDCSR[OACB] configuration register field.

If a request misses all ATMU windows (1–8) and the transaction's end address exceeds the maximum size of the default window, an outbound ATMU crossed boundary error is not generated. The outbound request is forwarded to the RapidIO target device.

15.8.6 RapidIO Inbound ATMU

15.8.6.1 Hits to Multiple ATMU Windows

If a request hits multiple ATMU windows, window 1 has the highest priority of the five inbound ATMU windows (windows 1-4, default). Window 2 is given the next higher priority and is followed by windows 3 and 4. The default window has the lowest priority.

If a request hits (base address match) multiple ATMU windows and the transaction's end address is contained within the boundary of each hit window and does not extend into another ATMU window, the translation window is the highest priority window that is hit.

If a lower priority window is programmed to lie entirely within a higher priority window, then it is possible for a transaction to cross window boundaries. Although not a practical programming application, the RapidIO endpoint handles this as follows:

1. If a request hits (base address match) an ATMU window (1–4) and the transaction's end address extends into another ATMU window with lower priority but is still contained within the boundary of the hit window, the translation window is the hit window.
2. If a request hits (base address match) multiple ATMU windows (1–4) and transaction's end address extends beyond the boundary of a lower priority hit window but is still contained within the boundary of a higher priority hit window, the translation window is the highest priority window that is hit.

15.8.6.2 Window Boundary Crossing Errors

If a higher priority window is programmed to lie entirely within a lower priority window, then it is possible for a transaction to cross window boundaries. The RapidIO endpoint handles this as follows:

1. If a request hits (base address match) an ATMU window (1–4, default) and the transaction's end address extends into another ATMU window with higher priority, an ATMU crossed boundary error is generated and logged.
2. If a request hits multiple ATMU windows (1–4, default) and transaction's end address extends beyond the boundary of a higher priority hit window, an inbound ATMU crossed boundary error is generated and logged.

Other window boundary crossing errors are:

1. If a request hits (base address match) an ATMU window (1–4) and the transaction's end address exceeds the size of the window, an inbound ATMU crossed boundary error is generated and logged.
2. If a NREAD/NWRITE_R/NWRITE/SWRITE request hits (base address match) an ATMU window (1–4, default) and the transaction's end address extends into the region defined as the local configuration space window, an inbound ATMU crossed boundary error is generated and logged.

A RapidIO error response is generated for RapidIO requests that require a response. RapidIO requests that do not require a response are dropped.

Inbound ATMU boundary crossing errors are logged in the LTLEDCSR[IACB] configuration register.

If a request misses all ATMU windows (1–4) and the transaction's end address exceeds the maximum size of the default window, an inbound ATMU crossed boundary error is not generated.

15.8.7 Generating Link-Request/Reset-Device

In LP-Serial mode the link partner cannot be reliably reset using the link-request/reset-device control symbols since the input port receiver cannot be disabled independent of the output port driver. The input port driver needs to be disabled to prevent non idle control symbols from being transmitted between the four link-request/reset-device control symbols. For example, if a packet was received on the inbound side after one of the four link-request/reset-device control symbols was sent outbound, the required sequence of four link-request/reset-device symbols would be interrupted with the packet acknowledgement (packet accept, packet retry or packet not accept).

15.8.8 Outbound Drain Mode

- The RapidIO port is placed into Drain mode when one of the following occurs:
  - PCR[OBDEN] is set
  - the Failed Threshold has been encountered and the CCSR[SPF] and CCSR[DPE] are both set
  - the packet time-to-live counter expires causing PCR[OBDEN] to be set
- When the RapidIO port is placed into Drain mode, the RapidIO port discards all packets in the outgoing data stream. Since the data stream is invalid, the RapidIO port also puts its Outbound port back to normal state. Any received acknowledgements and link-responses are considered invalid during this period (since the RapidIO port has cleared out all acknowledgement history).
- The RapidIO port's outbound and outstanding ackID shows that all outstanding packets at the time Drain mode was entered were accepted, whether they truly were accepted or not. If the outbound ackID is not acceptable, then software should change it prior to taking the RapidIO port out of the Drain mode. Also, if the link-partner needs to be put back into inbound OK state, then software should send a link-request/input-status. The recommended sequence for recovering from Drain mode is given in Section 15.8.10, "Software Assisted Error Recovery Register Support."
- PCR[OBDEN] also causes any queued up packet acknowledgements to be discarded if the port is uninitialized (the RapidIO port lets them be transferred if the port is initialized). Drain mode due to failed threshold does not cause any packet acknowledgements to be dropped.
- If a discarded packet in the outgoing data stream requires a logical response, a packet response time-out occurs as long as the packet response timer is enabled (PRTOCCSR is non 0).

15.8.9 Input Port Disable Mode

- The RapidIO port is placed into Input Port Disable mode when CCSR[PD] is set.

- When the RapidIO port is placed in Input Port Disable mode, the RapidIO port discards all incoming data stream (obviously). Since the incoming stream is invalid, the RapidIO port also puts its inbound port back to normal state.
- When the RapidIO port is placed in input port disable mode, the RapidIO port also:
  — ends any packet capture that was in progress.
  — clears the link-request/reset-device count.
- The RapidIO port's inbound ackID shows that all packets successfully received by the RapidIO port at the time input port disable mode was entered were accepted. If output port disable mode was entered at the same time, some packet acknowledgements may be discarded by the RapidIO port. If the inbound ackID is not acceptable, then software should change it prior to taking the RapidIO port out of Input Port Disable mode.

15.8.10 Software Assisted Error Recovery Register Support

- LMREQCSR is only supported for recovery from drain mode, including hot-swap support. Consistent with this statement, software should only write this register when the port is in Drain mode.
- The proper sequence for recovering from Drain mode is:
  a) Software ensures that link activity is stopped. This should include:
     i. software polls for Port OK bit to be set
     ii. software waits longer than the link time-out value
  b) Software generates a link-request/input-status to obtain the link-partner's inbound ackID value.
  c) Software changes the RapidIO port's outbound ackID to this value (if necessary).
  d) If the link-partner was hot-inserted, software changes the RapidIO port's inbound ackID to zero. (Note that software needs to know when the link-partner was hot-inserted.)
  e) NOTE: If software can guarantee that the link-partner does not attempt to forward any packets to this RapidIO port, then software may write the outbound ackID of the link-partner to match the inbound ackID of this RapidIO port. However, if the link-partner attempts to forward another packet while this write is still outstanding, and the ackIDs already happened to line-up, then the write actually causes the ackIDs to not match, and the link can not be recovered.
  f) Software should cause the link-partner to send a link-request/input-status just to ensure that the RapidIO port's inbound port is in Normal operation.
  g) Software clears the Failed Encountered bit or the Output Buffer Drain Enable bit; whichever one caused the Drain mode (thus clearing Drain mode).
- Software is responsible for timing software generated link-requests. If the response valid bit is not set in some reasonable period of time, the software should write another request in the register.
- When software writes LMREQCSR, software should make sure to successfully read LMRESPCSR set, otherwise, software may read a "stale" ackID status/link status later.
- Note that when the RapidIO port's outbound ackID is written by software using LASCSR, the inbound ackID is also written. Care must be taken to ensure that the inbound ackID is not written

to an incorrect value. For example, CCSR[PL] could be set to prevent the inbound ackID from changing before LASCSR is written.

15.8.11 Hot-Swap Support

The two basic hot-insertion approaches described in the RapidIO Error Management Extensions are supported although the second approach is only partially supported. Method one has a host bringing a field replacement unit (FRU) into the system. Method two has the FRU bringing itself into the system. Note that this RapidIO port is most likely the device being hot-inserted/extracted since typically this device would be connected to a switch but it could be the link partner of a device being hot-inserted/extracted.

15.8.11.1 Method 1

One possible sequence for when the host brings the FRU into the system is given. This RapidIO port can be either the device being hot-inserted/extracted or the link partner of the device being hot-inserted/extracted. The goal of this sequence is to bring the FRU into the system cleanly without generating errors.

15.8.11.1.1 Extraction

- The host determines that the FRU has failed by getting a port response time-out when polling the port OK bit (PO bit in error and status CSR) of the FRU. Note that there are other ways to determine that the FRU has failed. This is one possible method to detect FRU failure.
- The host must perform the following steps to the FRU link partner before hot insertion of the FRU occurs.
  - set the port lockout bit (PL = 1 in the control CSR) preventing packet reception and transmission
  - prevent any new packets from arriving to the FRU link partner RapidIO port by all other RapidIO devices and discard all pending packets destined for the FRU so that
  - congestion to this RapidIO port does not occur and cause other system problems
  - the FRU is not immediately flooded with old packets after insertion causing other errors like unsolicited responses
  - note that this RapidIO endpoint has a specific bit to enable the discard of pending packets (OBDEN in PCR). Also note that setting OBDEN in PCR forces the output state from error or retry to normal. In a switch, the time-to-live feature may be used to discard packets.
  - force the input state to normal. In this RapidIO endpoint, this occurs by disabling the input port receiver.
  - leave the input port receivers and output port drivers enabled so that initialization can complete when the FRU is inserted
  - clear all errors pertaining to the extracted RapidIO port in the FRU link partner and any other RapidIO port that was affected by the FRU failing (port response time-outs)
  - if RapidIO endpoint, clear OBDEN in PCR for normal operation
  - set the outbound and inbound ackID to 0x00 in the local ackID status CSR so that the ackID is correct and packets can be transmitted and received when the FRU is inserted

- The host indicates that the FRU should be removed.
- The FRU is removed from the system.

15.8.11.1.2 Insertion

- The FRU is inserted into the system
- Link initialization occurs and initialization complete status is indicated (port OK bit in the error and status CSR) in both RapidIO ports
- The host determines that this link partner has been inserted by periodically polling the initialization complete status in the FRU link partner (PO bit = 1 in the error and status command and status register)
- The host clears the output and input port enable bits and clears the port lockout bit in the control CSR in the FRU link partner allowing only maintenance transactions
- The host sets the master enabled and discovered bits in the FRU (M = 1 and D = 1 in the general control CSR). Note that the master enable bit does not prevent the RapidIO endpoint from transmitting packets.
- The host completes configuration of the FRU
- The host sets the output and input port enable bits in the control CSR in the FRU link partner allowing transmission and reception of all packet types
- The host re-enables packets to be sent to this RapidIO port by other RapidIO devices
- System operation resumes

15.8.11.2 Method 2 with RapidIO Port Hot-Swapped

One possible sequence for when the FRU brings itself into the system is given. This RapidIO Port can be either the device being hot-inserted/extracted or the link partner of the device being hot-inserted/extracted. The goal of this sequence is to bring the FRU into the system cleanly without generating errors.

15.8.11.2.1 Extraction

- The FRU fails.
- New packets are prevented from arriving to the FRU link partner by all other RapidIO devices and all pending packets destined for the FRU are discarded so that the following occur:
  - Congestion to this RapidIO port does not occur and cause other system problems.
  - The FRU is not immediately flooded with old packets after insertion causing other errors such as unsolicited responses.
- This FRU link partner continues to have its drivers enabled.
- The FRU link partner continues to allow the transmission and reception of all packet types since its output and input port enable bits are set and the port lockout bit is cleared in the control CSR.
- The FRU is removed from the system.

15.8.11.2.2 Insertion

- The FRU is inserted.
- Link initialization occurs, initialization complete status is indicated in both RapidIO ports (PO bit = 1 in the error and status command and status register).
- After initialization is complete, both devices set the port OK bit = 1 in the control CSR.
- The FRU sets its port lockout bit in the control CSR.
- The FRU generates a link-request/input-status to its link partner using the link maintenance request register. The FRU determines the link partner's inbound ackID by reading the link maintenance response register. The FRU then sets its outbound ackID in the local ackID status CSR.
- The FRU only enables maintenance transactions by clearing its output and input port enable bits in the control CSR and by clearing its port lockout bit in the control CSR.
- The FRU generates a maintenance write to its link partner's local ackID status CSR to set the link partner's outbound ackID value to 0. Upon receipt of the maintenance write, the link partner sets its outbound ackID value and generates the maintenance response using the new value. Note that if all packets intended for the link partner have not been discarded, or if any new packets intended for the link partner arrive, this step may fail (the RapidIO endpoint has no way to protect against this).
- The FRU completes configuration.
- The FRU enables all packets to be transmitted and received by setting its output and input port enable bits in the control CSR.
- System operation resumes.

15.8.12 Errors and Error Handling

This section describes how the logical and physical layers detect and react to RapidIO errors. The action of the core on notification of any of these errors is described minimally here. See *RapidIO Interconnect Specification, Revision 1.2* Part VII (Error Management Extensions Specifications) for more details on specific errors described below.

15.8.12.1 RapidIO Error Description

RapidIO errors are classified under three categories: recoverable errors, notification errors, and fatal errors.

Recoverable errors are non-fatal transmission errors (such as corrupt packet or control symbols, and general protocol errors) that RapidIO supports hardware detection of and a recovery mechanism for, as described in the *RapidIO Interconnect Specification, Revision 1.2*. In these cases, the appropriate bit is set in the error detect CSR. Only the packet containing the first detected recoverable error that is enabled for error capture (by error enable CSR) is captured in the error capture CSRs. No interrupt is generated or actions required for a recoverable error. Recoverable errors are detected in the physical layer only.

Notification errors are non-recoverable non-fatal errors detected by RapidIO (such as degraded threshold, port-write received, and all logical/transport layer (LTL) errors captured). Because they are non-recoverable (and in some cases have caused a packet to be dropped), notification by interrupt is

available. However, because they are non-fatal, response to the interrupt is not crucial to port performance; that is, the port is still functional. When a notification error is detected, the appropriate bit is set in the error-specific register, an interrupt is generated, and in some cases, the error is captured. In all cases, the RapidIO port continues operating. Notification errors are detected in both the physical and logical layer.

The RapidIO controller detects two fatal errors: exceeded failed threshold and exceeded consecutive retry threshold. In these cases, the port has failed because its recoverable error rate has exceeded a predefined failed threshold or because it has received too many packet retries in a row. In the first case, the controller sets the output failed-encountered bit in the error and status CSR; the RapidIO output hardware may or may not stop (based on stop-on-port-failed-encounter-enable and drop-packet-enable bits). In the second case, the controller sets the retry counter threshold trigger exceeded bit in the implementation error CSR; the RapidIO hardware continues to operate. In both cases, an interrupt is generated, and while the port continues operating at least partially, a system-level fix (such as reset) is recommended to clean up the controller's internal queues and resume normal operation. Fatal errors are detected in the physical layer only.

15.8.12.2 Physical Layer RapidIO Errors Detected

Table 15-105 lists all the RapidIO link errors detected by the RapidIO endpoint physical layer and the actions taken by the RapidIO endpoint. The error enable column lists the control bits that may disable the error checking associated with a particular error (if blank, error checking cannot be disabled). The cause field column indicates what cause field is used with the associated packet-not-accept control symbol for input error recovery. The EME error enable/detect column indicates which bit of the ERECSR allows the error to increment the error rate counter and lock the error capture registers, and likewise which bit of the EDCSR is set when the error has been detected.

Table 15-106 lists the RapidIO endpoint behavior after exceeding certain preset limits (degraded threshold, failed threshold, retry threshold).

Table 15-105. Physical RapidIO Errors Detected

Level	Error	Error Enable	RapidIO Endpoint Action	Cause Field	EME Error Type	EME Error Enable / Detect
Recoverable Errors						
1a	Received character had a disparity error. (serial)	—	Enter Input Error Stopped. Enter Output Error Stopped.	5: Received invalid/illegal character	Delineation Error	DE
1a	Received an invalid character, or valid but illegal character (serial)	—	Enter Input Error Stopped. Enter Output Error Stopped.	5: Received invalid/illegal character		
1b	The four control character bits associated with the received symbol do not make sense (not 0000, 1000, 1111) (serial)	—	Enter Input Error Stopped. Enter Output Error Stopped.	5: Received invalid/illegal character		
1b	Control symbol does not begin with an /SC/ or /PD/ control character. (serial)	—	Enter Input Error Stopped. Enter Output Error Stopped.	5: Received invalid/illegal character		
1c	Received packet with embedded idles. (serial)	—	Enter Input Error Stopped.	5: Received invalid/illegal character		
1d	Received a control symbol with a bad CRC	PCR[CCC] enables detect.	Enter Input Error Stopped. Enter Output Error Stopped.	2: Received a control symbol with bad CRC	Received corrupt control symbol	CCS
1d	Missing start: Packet data received w/o previous SOP control symbol	—	Enter Input Error Stopped.	7/31: General error	Protocol Error (unexpected packet/control symbol received)	PE
1e	Received packet that is < 64 bits	—	Enter Input Error Stopped.	7/31: General error		
1e	Received an EOP control symbol when there is no packet being received.	—	Enter Input Error Stopped.	7/31: General error		
1e	Received a stomp control symbol when there is no packet being received.	—	Enter Input Error Stopped.	7/31: General error		
2a	Received a Restart-from-retry control symbol when in the “OK” state	—	Enter Input Error Stopped	7/31: General error	Protocol Error (unexpected packet/control symbol received)	PE

Table 15-105. Physical RapidIO Errors Detected (continued)

Level	Error	Error Enable	RapidIO Endpoint Action	Cause Field	EME Error Type	EME Error Enable / Detect
2a	Received packet with a bad CRC value.	PCR[CCP] enables detect.	Enter Input Error Stopped.	4: bad CRC on packet.	Received packet with bad CRC	CRC
2a	Received packet which exceeds the maximum allowed size by the RapidIO spec.	—	Enter Input Error Stopped.	7/31: General error	Received packet exceeds 276 Bytes	EM
2b	Received packet with unexpected ackID value (out-of-sequence ACKID)	—	Enter Input Error Stopped.	1: Received unexpected ACKID on packet	Received packet with unexpected ackID	UA
2c	Received a non-maintenance packet when non-maintenance packet reception is stopped	Non-maint. packet reception is stopped when ‘Input Port Enable’ = 0.	Enter Input Error Stopped.	3: Non-maintenance packet reception is stopped	Not Captured	—
2d	Any packet received while Port Lockout bit is set	All packet reception is stopped when Port Lockout bit is set.	Enter Input Error Stopped.	3: Non-maintenance packet reception is stopped	Not Captured	—
-	Received a Link request control symbol before servicing previous link request.	Not detected.				
2a	Received packet-not-accepted ACK control symbol.	—	Enter Output Error Stopped.	—	Received packet-not-accepted symbol	PNA
2b	Received an ACK (accepted, or retry) control symbol when there are no outstanding packets	—	Enter Output Error Stopped.	—	Unsolicited ACK symbol	UCS
2b	Received packet ACK (accepted) for a packet whose transmission has not finished	—	Enter Output Error Stopped.	—		
2b	Received a Link response control symbol when no outstanding request.	—	Enter Output Error Stopped.	—		

Table 15-105. Physical RapidIO Errors Detected (continued)

Level	Error	Error Enable	RapidIO Endpoint Action	Cause Field	EME Error Type	EME Error Enable / Detect
2c	Received an ACK (accepted or retry) control symbol with an unexpected ACKID.	—	Enter Output Error Stopped.	—	Received ack. control symbol with unexpected ackID	AUA
2c	Link_response received with an ackID that is not outstanding	—	Re-enter Output Error Stopped.	—	Non-outstanding ackID	NOA
2d	An ACK control symbol is not received within the specified time-out interval.	PLTOCCSR[TV] > 0 enables detect.	Enter Output Error Stopped.	—	Link time-out	LTO
2d	A Link response is not received within the specified time-out interval	PLTOCCSR[TV] > 0 enables detect.	(re-) Enter Output Error Stopped.	—		

Table 15-106. Physical RapidIO Threshold Response

Error	Error Enable	RapidIO Endpoint Action	EME Error Type	Error Detect	Interrupt clear[1]
Notification Errors					
Error Rate Counter has exceeded the Degraded Threshold.	ERTCSR[ERDTT] > 0 & any bit in EECSR enables detect and interrupt generation.	Generate Interrupt. Continue to operate normally.	Degraded Threshold	ESCSR[ODE]	Write 1 to ESCSR[ODE]
Fatal Errors					
Consecutive Retry Counter has exceeded the Retry Counter Threshold Trigger	PRETCR[RET] > 0 enables detect and interrupt generation	Generate Interrupt. Port is in priority order.	Consecutive Retry Threshold	IECSR[RETE]	Write 1 to IECSR[RETE]
Error Rate Counter has exceeded the Failed Threshold.	ERTCSR[ERFTT] > 0 & any bit in EECSR enables detect and interrupt generation.	Generate Interrupt. Port behavior depends on CCSR[SPF] and CCSR[DPE] -- port can continue transmitting packets or can stop sending output packets, keeping or dropping them.	Failed Threshold	ESCSR[OFE]	Write 1 to ESCSR[OFE].

1 Information given here is minimal for clearing the interrupt. More detailed steps should be taken to find the cause of the interrupt.

15.8.12.3 Logical Layer Errors and Error Handling

This section describes how the logical layer detects and reacts to RapidIO errors. The action of the core on notification of any of these errors is described minimally here. Reference *RapidIO Interconnect Specification, Revision 1.2* Part VII (Error Management Extensions Specifications).

15.8.12.3.1 Logical Layer RapidIO Errors Detected

Table 15-107 through Table 15-121 lists all the errors detected by the RapidIO endpoint logical layer and the actions taken by the RapidIO endpoint. Note that when the RapidIO endpoint action includes sending an error response to either OCN or RapidIO, an error response is only sent if the original transaction was a request that required a response. Otherwise, no error response is sent. When dealing with multiple errors, discard of packet has higher priority than error response.

For misaligned transactions, the error management extension registers are updated with each child.

All packet field positions are assumed to be in the mode (small or large transport) configured. For example, when configured for small transport mode with pass-through mode not enabled and a large transport mode NREAD packet is received, the transaction type field bit positions checked correspond to a small transport type NREAD packet.

Table 15-107. Hardware Errors For NRead Transaction

Error	Interrupt Generated	Status Bit Set	RapidIO Error Response Generated	Logical/Transport Layer Capture Register	Comments
Priority Priority of Read transaction is 3	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No	Using the incoming RapidIO packet, for Small Transport type packet, LTLACCSR[XA] gets packet bits 78–79, LTLACCSR[A] gets packet bits 48–76, LTLDIDCCSR[DIDMSB] gets 0's, LTLDIDCCSR[DID] gets packet bits 16–23, LTLDIDCCSR[SIDMSB] gets 0's, LTLDIDCCSR[SID] gets packet bits 24–31, LTLCCCSR[FT] gets packet bits 12–15, LTLCCCSR[TT] gets packet bits 32–35, LTLCCCSR[MI] gets 0's For Large Transport type packets r. LTLACCSR[XA] gets packet bits 94–95, LTLACCSR[A] gets packet bits 64–92, LTLTLTLDIDCCSR[DIDMSB] gets 16–23, LTLDIDCCSR[DID] LTL gets packet bits 24–31, LTLDIDCCSR[SIDMSB] gets bits 32–39, LTLDIDCCSR[SID] gets bits 40–47, LTLCCCSR[FT] gets packet bits 12–15, LTLCCCSR[TT] gets packet bits 48–51, LTLCCCSR[MI] gets 0's	RapidIO packet is dropped.
TransportType Received reserved TT	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	Same as first entry	RapidIO packet is dropped
Received TT which is not enabled. - Error valid when pass_through is disabled and accept_all is disabled Or when accept_all is enabled.	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	Same as first entry	RapidIO packet is dropped

Table 15-107. Hardware Errors For NRead Transaction (continued)

Error	Interrupt Generated	Status Bit Set	RapidIO Error Response Generated	Logical/Transport Layer Capture Register	Comments
DestID DestID does not match this port's DeviceID if Alternate DeviceID is disabled or DestId does not match either Alternate DeviceID or DeviceId if Alternate DeviceID is enabled. Error valid when (pass_through or accept_all) is false	Yes if LTLEECSR[ITTE] is set	LTLEDCSR[ITTE]	Yes	Same as first entry	OCN error response is generated to self
SourceID Not Checked for error.	—	—	—	—	—
TransactionType Received RapidIO packet with reserved TType for this ftype	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	Yes	Same as first entry	OCN error response is generated to self
RdSize Not Checked for error.	—	—	—	—	—
SrcTID Not Checked for error.	—	—	—	—	—
Address:WdPtr:Xambs Read request hits overlapping ATMU windows Refer to 15.8.6.2/15-97	Yes if LTLEECSR[IACB] is set	LTLEDCSR[IACB]	Yes	Same as first entry	OCN error response is generated to self
Address:WdPtr:Xambs Request hits a protected ATMU window	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	Yes	Same as first entry	OCN error response is generated to self
Address:WdPtr:Xambs Beginning address matches LCSBA1CSR with non 32 bit read request. Performed only when ttype == 4'b0100	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	Yes	Same as first entry	OCN error response is generated to self
Header Size Header size is not 12 Bytes for small Transport packet or not 16 Bytes for Large Transport packet. Large Transport packet has 14 valid bytes and two bytes of padding of 0's. Padding of 0's is not checked.	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	Yes	Same as first entry	OCN error response is generated to self
PayloadSize Not Applicable	—	—	—	—	—

Table 15-108. Hardware Errors For Maintenance Read/Write Req Transaction

Error	Interrupt Generated	Status Bit Set	RapidIO Error Response Generated	Logical/Transport Layer Capture Register	Comments
Priority Priority of maintenance read or write request transaction is 3	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No	Using the incoming RapidIO packet, for Small Transport type packets, LTLACCSR[XA] gets packet bits 78–79, LTLACCSR[A] gets packet bits 48–76, LTLDIDCCSR[DIDMSB] gets 0's, LTLDIDCCSR[DID] gets packet bits 16–23, LTLDIDCCSR[SIDMSB] gets 0's, LTLDIDCCSR[SID] gets packet bits 24–31, LTLCCCSR[FT] gets packet bits 12–15, LTLCCCSR[TT] gets packet bits 32–35, LTLCCCSR[MI] gets 0's For Large Transport type packets. LTLACCSR[XA] gets packet bits 94–95, LTLACCSR[A] gets packet bits 64–92, LTLTLTLDIDCCSR[DIDMSB] gets 16–23, LTLDIDCCSR[DID]LTL gets packet bits 24–31, LTLDIDCCSR[SIDMSB] gets bits 32–39, LTLDIDCCSR[SID] gets bits 40–47, LTLCCCSR[FT] gets packet bits 12–15, LTLCCCSR[TT] gets packet bits 48–51, LTLCCCSR[MI] gets 0's	RapidIO packet is dropped
TransportType Received reserved TT	Yes	LTLEDCSR[TSE]	No	Same as first entry	RapidIO packet is dropped
Received TT which is not enabled. - Error valid when passthrough is disabled and accept_all is disabled Or when accept_all is enabled.	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	Same as first entry	RapidIO packet is dropped

Table 15-108. Hardware Errors For Maintenance Read/Write Req Transaction (continued)

Error	Interrupt Generated	Status Bit Set	RapidIO Error Response Generated	Logical/Transport Layer Capture Register	Comments
DestID DestID does not match this port's DeviceID if Alternate DeviceID is disabled or DestId does not match either Alternate DeviceID or DeviceId if Alternate DeviceID is enabled. Error valid when (passthrough or accept_all) is false	Yes if LTLEECSR[ITTE] is set	LTLEDCSR[ITTE]	Yes	Same as first entry	OCN error response is generated to self
SourceID Not Checked for error.	—	—	—	—	—
TransactionType Reserved Transaction Type for this ftype	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	Yes	Same as first entry	RapidIO packet is dropped
RdSize Read/Write request size is not for 4 bytes	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	Yes	Same as first entry	OCN error response is generated to self
SrcTID Not Checked for error.	—	—	—	—	—
HopCount Not Checked for error.	—	—	—	—	—
Config Offset Not Checked for error.	—	—	—	—	—
Header Size Maintenance Read request - Header size is not 12 Bytes for small Transport packet or not 16 Bytes for Large Transport packet. Maintenance Write request - total header size is not 12 Bytes for Small Transport packet or not 16 Bytes for Large Transport packet. Padding of 0's in last two bytes of Large Transport packet is not checked.	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	Yes	Same as first entry	OCN error response is generated to self
PayloadSize Write request with payload not equal to 8 bytes. Read request with payload not 0 bytes	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	Yes	Same as first entry	OCN error response is generated to self

Table 15-109. Hardware Errors For Atomic (inc, dec, set, or clr) Read Transaction

Error	Interrupt Generated	Status Bit Set if Corresp-onding Bit is Enabled	RapidIO Error Response Generated	Logical/Transport Layer Capture Register	Comments
Priority Priority of read transaction is 3	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No	Using the incoming RapidIO packet, for Small Transport type packets, LTLACCSR[XA] gets packet bits 78–79, LTLACCSR[A] gets packet bits 48–76, LTLDIDCCSR[DIDMSB] gets 0's, LTLDIDCCSR[DID] gets packet bits 16–23, LTLDIDCCSR[SIDMSB] gets 0's, LTLDIDCCSR[SID] gets packet bits 24–31, LTLCCCSR[FT] gets packet bits 12–15, LTLCCCSR[TT] gets packet bits 32–35, LTLCCCSR[MI] gets 0's For Large Transport type packets.LTLACCSR[XA] gets packet bits 94–95, LTLACCSR[A] gets packet bits 64–92, LTLTLTLDIDCCSR[DIDMSB] gets 16–23, LTLDIDCCSR[DID] gets packet bits 24–31, LTLDIDCCSR[SIDMSB] gets bits 32–39, LTLDIDCCSR[SID] gets bits 40–47, LTLCCCSR[FT] gets packet bits 12–15, LTLCCCSR[TT] gets packet bits 48–51, LTLCCCSR[MI] gets 0's	RapidIO packet is dropped
TransportType Received reserved TT	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	Same as first entry	RapidIO packet is dropped
Received TT which is not enabled. - Error valid when passthrough is disabled and accept_all is disabled Or when accept_all is enabled.	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	Same as first entry	RapidIO packet is dropped
DestID DestID does not match this port's DeviceID if Alternate DeviceID is disabled or DestId does not match either Alternate DeviceID or DeviceId if Alternate DeviceID is enabled. Error valid when (passthrough or accept_all) is false	Yes if LTLEECSR[ITTE] is set	LTLEDCSR[ITTE]	Yes	Same as first entry	OCN error response is generated to self

Table 15-109. Hardware Errors For Atomic (inc, dec, set, or clr) Read Transaction (continued)

Error	Interrupt Generated	Status Bit Set if Corresp-onding Bit is Enabled	RapidIO Error Response Generated	Logical/Transport Layer Capture Register	Comments
SourceID Not Checked for error.	—	—	—	—	—
TransactionType Non-Atomic ttype is tested with Nread	—	—	—	—	—
TransactionType Received Atomic Increment request with DOCAR[AI] disabled. Received Atomic decrement request with DOCAR[AD] disabled. Received Atomic Set request with DOCAR[AS] disabled. Received Atomic Clear request with DOCAR[AC] disabled.	Yes if LTLEECSR[UT] is se	LTLEDCSR[UT]	Yes	Same as second entry	Error response is generated to self
RdSize Not unsupported RdSize request is not for contiguous one, two or four bytes	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	Yes	Same as first entry	Error response is generated to self
SrcTID Not Checked for error.	—	—	—	—	—
Address:WdPtr:Xambs Not unsupported Request hits a protected ATMU window or the LCSBA1CSR Refer to 15.8.5.2/15-96	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	Yes	Same as first entry	Error response is generated to self
Address:WdPtr:Xambs Read request hits overlapping ATMU windows Refer to 15.8.5.2/15-96	Yes if LTLEECSR[IACB] is set	LTLEDCSR[IACB]	Yes	Same as first entry	Error response is generated to self
Header Size Not unsupported Header size is not 12 Bytes for small Transport packet and not 16 Bytes for Large Transport packet Padding of 0's in last two bytes of Large Transport packet is not checked	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	Yes	Same as first entry	Error response is generated to self

Table 15-110. Hardware Errors For NWrite, NWrite_r, and Unsupported Atomic Test-and-Swap Transactions

Error	Interrupt Generated	Status Bit Set	RapidIO Error Response Generated	Logical/Transport Layer Capture Register	Comments
Priority Not unsupported	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No	Using the incoming RapidIO packet, for Small Transport type packets, LTLACCSR[XA] gets packet bits 78–79, LTLACCSR[A] gets packet bits 48–76, LTLDIDCCSR[DIDMSB] gets 0's, LTLDIDCCSR[DID] gets packet bits 16–23, LTLDIDCCSR[SIDMSB] gets 0's, LTLDIDCCSR[SID] gets packet bits 24–31, LTLCCCSR[FT] gets packet bits 12–15, LTLCCCSR[TT] gets packet bits 32–35, LTLCCCSR[MI] gets 0's For Large Transport type packets.LTLACCSR[XA] gets packet bits 94–95, LTLACCSR[A] gets packet bits 64–92, LTLTLTLDIDCCSR[DIDMSB] gets 16–23, LTLDIDCCSR[DID] gets packet bits 24–31, LTLDIDCCSR[SIDMSB] gets bits 32–39, LTLDIDCCSR[SID] gets bits 40–47, LTLCCCSR[FT] gets packet bits 12–15, LTLCCCSR[TT] gets packet bits 48–51, LTLCCCSR[MI] gets 0's	RapidIO packet is dropped.
TransportType Received reserved TT	Yes	LTLEDCSR[TSE]	No	Same as first entry	RapidIO packet is dropped
Received TT which is not enabled. - Error valid when passthrough is disabled and accept_all is disabled Or when accept_all is enabled.	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	Same as first entry	RapidIO packet is dropped

Table 15-110. Hardware Errors For NWrite, NWrite_r, and Unsupported Atomic Test-and-Swap Transactions (continued)

Error	Interrupt Generated	Status Bit Set	RapidIO Error Response Generated	Logical/Transport Layer Capture Register	Comments
DestID DestID does not match this port's DeviceID if Alternate DeviceID is disabled or DestId does not match either Alternate DeviceID or DeviceId if Alternate DeviceID is enabled. Error valid when (passthrough or accept_all) is false	Yes if LTLEECSR[ITTE] is set	LTLEDCSR[ITTE]	Yes for Nwrite_r. No for Nwrite.	Same as first entry	OCN error response is generated to self for Nwrite_r. RapidIO packet is dropped for Nwrite
SourceID Not applicable	—	—	—	—	—
TransactionType Received RapidIO packet for Atomic test-and-swap transaction	Yes if LTLEECSR[UT] is set	LTLEDCSR[UT]	Yes	Same as first entry	OCN error response is generated to self
TransactionType Received RapidIO packet with reserved TType for this ftype Packet is treated as Nwrite Transaction	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No	Same as first entry	RapidIO packet is dropped
WrSize Not unsupported transaction WrSize request is for one of reserved sizes	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	Yes for Nwrite_r. No for Nwrite.	Same as first entry	OCN error response is generated to self for Nwrite_r. RapidIO packet is dropped for Nwrite
Address:WdPtr:Xambs Not unsupported transaction Nwrite request hits LCSBA1CSR	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No for Nwrite.	Same as first entry	RapidIO packet is dropped for Nwrite
Address:WdPtr:Xambs Not unsupported transaction Request hits a protected ATMU window	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	Yes for Nwrite_r. No for Nwrite.	Same as first entry	OCN error response is generated to self for Nwrite_r. RapidIO packet is dropped for Nwrite

Table 15-110. Hardware Errors For NWrite, NWrite_r, and Unsupported Atomic Test-and-Swap Transactions (continued)

Error	Interrupt Generated	Status Bit Set	RapidIO Error Response Generated	Logical/Transport Layer Capture Register	Comments
Address:WdPtr:Xambs Write request hits overlapping ATMU windows Refer to 15.8.5.2/15-96	Yes if LTLEECSR[IACB] is set	LTLEDCSR[IACB]	Yes for Nwrite_r. No for Nwrite.	Same as first entry	OCN error response is generated to self for Nwrite_r. RapidIO packet is dropped for Nwrite
SrcTID Not Checked for error.	—	—	—	—	—
Address:WdPtr:Xambs Nwrite_r address matches LCSBA1CSR with non 32 bit read request. Performed only for Nwrite_r packet	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	Yes for Nwrite_r.	Same as first entry	OCN error response is generated to self
Header Size Not unsupported transaction Header size is less than 12 bytes for small Transport packet or less than 16 bytes for Large Transport packet - that is, no payload present. Large Transport packet has 14 valid bytes and 2 bytes of padding of 0s. Padding of 0s is not checked.	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	Yes for Nwrite_r. No for Nwrite.	Same as first entry	OCN error response is generated to self for Nwrite_r. RapidIO packet is dropped for Nwrite
PayloadSize Not unsupported transaction Payload is greater than that indicated by {wdptr:wrsize} field, payload is not double word aligned or does not have any payload	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	Yes for Nwrite_r. No for Nwrite.	Same as first entry	OCN error response is generated to self for Nwrite_r. RapidIO packet is dropped for Nwrite

Table 15-111. Hardware Errors For SWrite Transactions

Error	Interrupt Generated	Status Bit Set	Error Response Generated	Logical/Transport Layer Capture Register	Comments
Priority Swrite transaction priority is 3	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No	Same as third entry	RapidIO packet is dropped
TransportType Received reserved TT	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	Using the incoming RapidIO packet, for Small Transport type packets, LTLACCSR[XA] gets packet bits 62–63, LTLACCSR[A] gets packet bits 32–60, LTLDIDCCSR[DIDMSB] gets 0's, LTLDIDCCSR[DID] gets packet bits 16–23, LTLDIDCCSR[SIDMSB] gets 0's, LTLDIDCCSR[SID] gets packet bits 24–31, LTLCCCSR[FT] gets packet bits 12–15, LTLCCCSR[TT] gets packet bits 32–35, LTLCCCSR[MI] gets 0's For Large Transport type packets.LTLACCSR[XA] gets packet bits 78–79, LTLACCSR[A] gets packet bits 48–76, LTLTLTLDIDCCSR[DIDMSB] gets 16–23, LTLDIDCCSR[DID] gets packet bits 24–31, LTLDIDCCSR[SIDMSB] gets bits 32–39, LTLDIDCCSR[SID] gets bits 40–47, LTLCCCSR[FT] gets packet bits 12–15, LTLCCCSR[TT] gets packet bits 48–51, LTLCCCSR[MI] gets 0's	RapidIO packet is dropped
Received TT which is not enabled. - Error valid when passthrough is disabled and accept_all is disabled Or when accept_all is enabled.	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	Same as third entry	RapidIO packet is dropped
DestID DestID does not match this port's DeviceID if Alternate DeviceID is disabled or DestId does not match either Alternate DeviceID or DeviceId if Alternate DeviceID is enabled. Error valid when (passthrough or accept_all) is false	Yes if LTLEECSR[ITTE] is set	LTLEDCSR[ITTE]	No	Same as third entry	RapidIO packet is dropped
SourceID Not Checked for error.	—	—	—	—	—

Table 15-111. Hardware Errors For SWrite Transactions (continued)

Error	Interrupt Generated	Status Bit Set	Error Response Generated	Logical/Transport Layer Capture Register	Comments
Address:WdPtr:Xambs Swrite request hits overlapping ATMU windows. Refer to 15.8.5.2/15-96	Yes if LTLEECSR[IACB] is set	LTLEDCSR[IACB]	No	Same as third entry	RapidIO packet is dropped
Address:WdPtr:Xambs Request hits a protected ATMU window or the LCSBA1CSR	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No	Same as third entry	RapidIO packet is dropped
PayloadSize Payload size is not in DWs, has exceeded 256 bytes or has no payload.	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No	Same as third entry	RapidIO packet is dropped

Table 15-112. Hardware Errors For Maintenance Response Transactions

Error	Interrupt Generated	Status Bit Set	OCN Error Response Generated	Logical/Transport Layer Capture Register	Comments
Priority Not UR Response priority is not higher than RapidIO maintenance request priority	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No	Using the incoming RapidIO packet, for Small Transport type packets, LTLACCSR[XA] gets packet bits 78–79, LTLACCSR[A] gets packet bits 48–76, LTLDIDCCSR[DIDMSB] gets 0's, LTLDIDCCSR[DID] gets packet bits 16–23, LTLDIDCCSR[SIDMSB] gets 0's, LTLDIDCCSR[SID] gets packet bits 24–31, LTLCCCSR[FT] gets packet bits 12–15, LTLCCCSR[TT] gets packet bits 32–35, LTLCCCSR[MI] gets 0's For Large Transport type packets.LTLACCSR[XA] gets packet bits 94–95, LTLACCSR[A] gets packet bits 64–92, LTLTLTLDIDCCSR[DIDMSB] gets 16–23, LTLDIDCCSR[DID] gets packet bits 24–31, LTLDIDCCSR[SIDMSB] gets bits 32–39, LTLDIDCCSR[SID] gets bits 40–47, LTLCCCSR[FT] gets packet bits 12–15, LTLCCCSR[TT] gets packet bits 48–51, LTLCCCSR[MI] gets 0's	RapidIO packet is dropped and ignored
TransportType Received reserved TT	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	Same as first entry	RapidIO packet is dropped and ignored

Table 15-112. Hardware Errors For Maintenance Response Transactions (continued)

Error	Interrupt Generated	Status Bit Set	OCN Error Response Generated	Logical/Transport Layer Capture Register	Comments
Received TT which is not enabled. - Error valid when passthrough is disabled and accept_all is disabled Or when accept_all is enabled.	Yes if LTLEECSR [TSE] is set	LTLEDCSR[TSE]	No	Same as first entry	RapidIO packet is dropped and ignored
DestID DestID does not match this port's DeviceID if Alternate DeviceID is disabled or DestId does not match either Alternate DeviceID or DeviceId if Alternate DeviceID is enabled. Error valid when (passthrough or accept_all) is false	Yes if LTLEECSR [ITTE] is set	LTLEDCSR[ITTE]	No	Same as first entry	RapidIO packet is dropped and ignored
SourceID Does not match the request's DestID	Yes if LTLEECSR [UR] is set	LTLEDCSR[UR]	No	Same as first entry	RapidIO packet is dropped and ignored
TransactionType Not UR Received RapidIO packet with reserved TType for this ftype	Yes if LTLEECSR [ITD] is set	LTLEDCSR[ITD]	No	Same as first entry	RapidIO packet is dropped and ignored
Not UR Maintenance read/write response does not correspond to an outstanding valid message read/write request.	Yes if LTLEECSR [ITD] is set	LTLEDCSR[ITD]	No	Same as first entry	RapidIO packet is dropped and ignored
HopCount Not Checked for error.	—	—	—	—	—
Status Not UR Is not "Done" or "Error" Not "Done" status for "read_response" transaction type with payload "Error" status with payload.	Yes if LTLEECSR [ITD] is set	LTLEDCSR[ITD]	No	Same as first entry	RapidIO packet is dropped and ignored

Table 15-112. Hardware Errors For Maintenance Response Transactions (continued)

Error	Interrupt Generated	Status Bit Set	OCN Error Response Generated	Logical/Transport Layer Capture Register	Comments
Status Not UR Error Response	Yes if LTLEECSR[IER] is set	LTLEDCSR[IER]	Yes	Same as first entry except error capture is done from original request	OCN error response is generated to requestor.
TargetTID No outstanding transaction for this TargetTID	Yes if LTLEECSR[UR] is set	LTLEDCSR[UR]	No	Same as first entry	RapidIO packet is dropped and ignored
Header Size Not UR Maintenance Read response - total payload size with done status is not greater than 4 Bytes. Maintenance Write response - total header size is less than 12 Bytes for Small Transport packet or is less than 16 Bytes for Large Transport packet. Padding of 0's for Small or Large Transport packets is not verified.	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No	Same as first entry	RapidIO packet is dropped and ignored
PayloadSize Not UR Maintenance write response has payload. Maintenance read response with done status and payload not matching valid request size, request size for the response is invalid or payload size is not dword aligned.	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No	Same as first entry	RapidIO packet is dropped and ignored
Packet response time-out Response is not received by configured time	Yes if LTLEECSR[PRT] is set	LTLEDCSR[PRT]	Yes	Same as first entry except error capture is done from original request.	OCN error response is generated to requestor.

Table 15-113. Hardware Errors For IO/GSM Response Transactions (Not Maintenance)

Error	Interrupt Generated	Status Bit Set	OCN Error Response Generated	Logical/Transport Layer Capture Register	Comments
Priority Not UR Response priority is not higher than RapidIO request priority	Yes if LTLEECSR [ITD] is set	LTLEDCSR [ITD]	No	Using the incoming RapidIO packet, for Small Transport type packets, LTLACCSR[XA] gets packet bits 78–79 (if available), LTLACCSR[A] gets packet bits 48–76 (if available), LTLDIDCCSR[DIDMSB] gets 0's, LTLDIDCCSR[DID] gets packet bits 16–23, LTLDIDCCSR[SIDMSB] gets 0's, LTLDIDCCSR[SID] gets packet bits 24–31, LTLCCCSR[FT] gets packet bits 12–15, LTLCCCSR[TT] gets packet bits 32–35, LTLCCCSR[MI] gets 0's For Large Transport type packets.LTLACCSR[XA] gets packet bits 94–95 (if available), LTLACCSR[A] gets packet bits 64–92 (if available), LTLTLTLDIDCCSR[DIDMSB] gets 16–23, LTLDIDCCSR[DID] gets packet bits 24–31, LTLDIDCCSR[SIDMSB] gets bits 32–39, LTLDIDCCSR[SID] gets bits 40–47, LTLCCCSR[FT] gets packet bits 12–15, LTLCCCSR[TT] gets packet bits 48–51, LTLCCCSR[MI] gets 0's	RapidIO packet is dropped and ignored
TransportType Received reserved TT for this ftype	Yes if LTLEECSR [TSE] is set	LTLEDCSR [TSE]	No	Same as first entry	RapidIO packet is dropped and ignored
Received TT which is not enabled. - Error valid when passthrough is disabled and accept_all is disabled Or when accept_all is enabled.	Yes if LTLEECSR [TSE] is set	LTLEDCSR [TSE]	No	Same as first entry	RapidIO packet is dropped and ignored
DestID DestID does not match this port's DeviceID if Alternate DeviceID is disabled or DestId does not match either Alternate DeviceID or DeviceId if Alternate DeviceID is enabled. Error valid when (passthrough or accept_all) is false	Yes if LTLEECSR [ITTE] is set	LTLEDCSR [ITTE]	No	Same as first entry	RapidIO packet is dropped and ignored

Table 15-113. Hardware Errors For IO/GSM Response Transactions (Not Maintenance) (continued)

Error	Interrupt Generated	Status Bit Set	OCN Error Response Generated	Logical/Transport Layer Capture Register	Comments
SourceID Does not match the request's DestID	Yes if LTLEECSR [UR] is set	LTLEDCSR [UR]	No	Same as first entry	RapidIO packet is dropped and ignored
TransactionType Not UR Received RapidIO packet with reserved TType	Yes if LTLEECSR [ITD] is set	LTLEDCSR [ITD]	No	Same as first entry	RapidIO packet is dropped and ignored
Not UR IO read response does not correspond to an outstanding valid IO/GSM read request. IO write response does not correspond to an outstanding valid IO/GSM write request.	Yes if LTLEECSR [ITD] is set	LTLEDCSR [ITD]	No	Same as first entry	RapidIO packet is dropped and ignored
Status Not UR IO transaction - Is not "Done" or "Error" GSM transaction IO_Read_Home Is not "Done - Data-Only", "Done - Done-Intervention", "Done", "Retry" or "Error". Flush_w_Data response is not "Done", "Retry" or "Error" Transaction type of "Response_with_data" and status is not done	Yes if LTLEECSR [ITD] is set	LTLEDCSR [ITD]	No	Same as first entry	RapidIO packet is dropped and ignored.
Status Not UR GSM Error response	Yes if LTLEECSR [GER] is set	LTLEDCSR [GER]	Yes if data is not received for this request.	Same as first entry except error capture is done from original request packet.	OCN error response is generated to requestor if data is not forwarded to it. Else the RapidIO packet is dropped.
Status Not UR IO Error Response	Yes if LTLEECSR [IER] is set	LTLEDCSR [IER]	Yes	Same as first entry except error capture is done from original request packet.	OCN error response is generated to requestor.

Table 15-113. Hardware Errors For IO/GSM Response Transactions (Not Maintenance) (continued)

Error	Interrupt Generated	Status Bit Set	OCN Error Response Generated	Logical/Transport Layer Capture Register	Comments
TargetTID No outstanding transaction for this TargetTID	Yes if LTLEECSR [UR] is set	LTLEDCSR [UR]	No	Same as first entry	RapidIO packet is dropped and ignored.
Packet Size Not UR (All non-maintenance and non-message) Write response - Header size is not 8 Bytes for Small Transport packet or not 12 Bytes for Large Transport packet GSM - “Done” response packet size to “Flush” is not 8 Bytes for Small Transport packet or not 12 Bytes for Large Transport packet. “Done-Intervention” is not 8 Bytes for Small Transport and 12 Bytes for Large Transport field. Two byte padding of 0’s in Large Transport field packet is not checked.	Yes if LTLEECSR [ITD] is set	LTLEDCSR [ITD]	No	Same as first entry	RapidIO packet is dropped and ignored.
Payload Size Not UR IO - Read Response - total payload is not of the size requested. “Done” or “Done-Data_Only” response to IO_Read_Home with incorrect payload size. Response with transaction type “response_with_no_data” has payload	Yes if LTLEECSR [ITD] is set	LTLEDCSR [ITD]	No	Same as first entry	RapidIO packet is dropped and ignored.

Table 15-113. Hardware Errors For IO/GSM Response Transactions (Not Maintenance) (continued)

Error	Interrupt Generated	Status Bit Set	OCN Error Response Generated	Logical/Transport Layer Capture Register	Comments
Retry Not UR GSM request has had one more than configured number of retries for non misaligned request. The misaligned GSM request has had one to four (cumulative for the corresponding child requests) more than configured number of retries.	Yes if LTLEECSR [RETE] is set	LTLEDCSR [RETE]	Yes	Same as first entry except error capture is done from original request.	OCN error response is generated to requestor.
Packet response time-out Response is not received by configured time for packets requiring RapidIO response. “GSM - IO_Read_Home” - Done_With_Data is not received in configured time when Done_Intervention is received for non misaligned request or last child of misaligned request. Done response is not received in configured time for non misaligned request or last child of misaligned request. EME capture occurs for each child packet response time-out.	Yes if LTLEECSR [PRT] is set	LTLEDCSR [PRT]	Yes	Same as first entry except error capture is done from original request.	OCN error response is generated to requestor.

Table 15-113. Hardware Errors For IO/GSM Response Transactions (Not Maintenance) (continued)

Error	Interrupt Generated	Status Bit Set	OCN Error Response Generated	Logical/Transport Layer Capture Register	Comments
Packet response time-out Response is not received by configured time for packets requiring RapidIO response. “GSM - IO_Read_Home” - Done_Intervention is not received in configured time when Done_With_Data is received. This is true for both non misaligned or misaligned requests.	Yes if LTLEECSR [PRT] is set	LTLEDCSR [PRT]	No	Same as first entry except error capture is done from original request.	An OCN done response is generated when the Done_With _Data is received for non misaligned requests or the last child of a misaligned request. Therefore, an error response cannot be sent when the packet response time-out occurs.
GSM - IO_Read_Home Not UR Done response, Retry response, or Error response is after Done_Intervention response or Data_only is received.	Yes if LTLEECSR [ITD] is set	LTLEDCSR [ITD]	No	Same as first entry	RapidIO packet is dropped and ignored.
Not UR Response for OCN packets not requiring response, but converted to “response” type packet by ATMU receives Error response. For example, NWrite converted to NWrite_r received “Error” response	Yes if LTLEECSR [IER]/[GER] /[RETE] is set	LTLEDCSR [IER]/[GER]/[RETE]	No	Same as first entry except error capture is done from original request.	RapidIO packet is dropped and ignored.
Response for OCN packets not requiring response, but converted to “response” type packet by ATMU is not received by configured time.	Yes if LTLEECSR [PRT] is set	LTLEDCSR [PRT]	No	Same as first entry except error capture is done from original request.	No error response is generated.

Table 15-114. Hardware Errors For DMA Message Response Transactions

Error	Interrupt Generated	Status Bit Set	OCN Error Response Generated	Logical/Transport Layer Capture Register	Comments
Priority Not UR Response priority is not higher than RapidIO request priority	Yes if LTLEECSR [ITD] is set	LTLEDCSR [ITD]	No	Using the incoming RapidIO packet, for Small Transport type packets, LTLACCSR[XA] gets packet bits 78–79, LTLACCSR[A] gets packet bits 48–76, LTLDIDCCSR[DIDMSB] gets 0's, LTLDIDCCSR[DID] gets packet bits 16–23, LTLDIDCCSR[SIDMSB] gets 0's, LTLDIDCCSR[SID] gets packet bits 24–31, LTLCCCSR[FT] gets packet bits 12–15, LTLCCCSR[TT] gets packet bits 32–35, LTLCCCSR[MI] gets packet bits 40–47 For Large Transport type packets. LTLACCSR[XA] gets packet bits 94–95, LTLACCSR[A] gets packet bits 64–92, LTLTLTLDIDCCSR[DIDMSB] gets 16–23, LTLDIDCCSR[DID] gets packet bits 24–31, LTLDIDCCSR[SIDMSB] gets bits 32–39, LTLDIDCCSR[SID] gets bits 40–47, LTLCCCSR[FT] gets packet bits 12–15, LTLCCCSR[TT] gets packet bits 48–51, LTLCCCSR[MI] gets packet bits 56–63	RapidIO packet is dropped and ignored
TransportType Received reserved TT	Yes if LTLEECSR [TSE] is set	LTLEDCSR [TSE]	No	Same as first entry	RapidIO packet is dropped and ignored
Received TT which is not enabled. Error valid when passthrough is disabled and accept_all is disabled Or when accept_all is enabled.	Yes if LTLEECSR [TSE] is set	LTLEDCSR [TSE]	No	Same as first entry	RapidIO packet is dropped and ignored
DestID DestID does not match this port's DeviceID if Alternate DeviceID is disabled or DestId does not match either Alternate DeviceID or DeviceId if Alternate DeviceID is enabled. Error valid when (passthrough or accept_all) is false	Yes if LTLEECSR [ITTE] is set	LTLEDCSR [ITTE]	No	Same as first entry	RapidIO packet is dropped and ignored

Table 15-114. Hardware Errors For DMA Message Response Transactions (continued)

Error	Interrupt Generated	Status Bit Set	OCN Error Response Generated	Logical/Transport Layer Capture Register	Comments
SourceID Does not match the request's DestID	Yes if LTLEECSR [UR] is set	LTLEDCSR [UR]	No	Same as first entry	RapidIO packet is dropped and ignored
TransactionType Not checked. To be a message response TType has to be 0x1	—	—	—	—	—
Status Not UR DMA Message Error response	Yes if LTLEECSR [DMAMER] is set	LTLEDCSR [DMAMER]	Yes	Same as first entry except error capture is done from original request	OCN error response is generated to requestor.
Status Not UR Received status of reserved type	Yes if LTLEECSR [ITD] is set	Yes if LTLEDCSR [ITD] is set	No	Same as first entry	RapidIO packet is dropped and ignored
No outstanding transaction for this letter, mailbox and message segment	Yes if LTLEECSR [UR] is set	LTLEDCSR [UR]	No	Same as first entry	RapidIO packet is dropped and ignored.
Packet Size Not UR (All non-maintenance and non-message) DMA response - Header size is not 8 Bytes for Small Transport packet or not 12 Bytes for Large Transport packet Two byte padding of 0's in Large Transport field packet is not checked.	Yes if LTLEECSR [ITD] is set	LTLEDCSR [ITD]	No	Same as first entry	RapidIO packet is dropped and ignored.
Payload Size Not UR Payload size is not zero	Yes if LTLEECSR [ITD] is set	LTLEDCSR [ITD]	No	Same as first entry	RapidIO packet is dropped and ignored.

Table 15-114. Hardware Errors For DMA Message Response Transactions (continued)

Error	Interrupt Generated	Status Bit Set	OCN Error Response Generated	Logical/Transport Layer Capture Register	Comments
Retry Not UR DMA request - has had more than configured number of retries	Yes if LTLEECSR [RETE] is set	LTLEDCSR[RETE]	Yes	Same as first entry except error capture is done from original request	OCN error response is generated to requestor.
Packet response time-out Response is not received by configured time.	Yes if LTLEECSR [PRT] is set	LTLEDCSR[PRT]	Yes	Same as first entry except error capture is done from original request	OCN error response is generated to requestor.

Table 15-115. Hardware Errors For Message Request Transactions

Error	Interrupt Generated	Status Bit Set	RapidIO Error Response Generated	Logical/Transport Layer Capture Register	Comments
Priority Not Applicable	—	—	—	—	—
TransportType Received reserved TT	Yes if LTLEECSR [TSE] is set	LTLEDCSR[TSE]	No	Using the original request RapidIO packet, for small Transport type, LTLACCSR[XA] gets packet bits 78–79, LTLACCSR[A] gets packet bits 48–76, LTLDIDCCSR[DIDMSB] gets 0's, LTLDIDCCSR[DID] gets packet bits 16–23, LTLDIDCCSR[SIDMSB] gets 0's, LTLDIDCCSR[SID] gets packet bits 24–31, LTLCCCSR[FT] gets packet bits 12–15, LTLCCCSR[TT] gets packet bits 32–35, LTLCCCSR[MI] gets packet bits 40–47. For Large Transport type packets. LTLACCSR[XA] gets packet bits 94–95, LTLACCSR[A] gets packet bits 64–92, LTLTLTLDIDCCSR[DIDMSB] gets 16–23, LTLDIDCCSR[DID] LTL gets packet bits 24–31, LTLDIDCCSR[SIDMSB] gets bits 32–39, LTLDIDCCSR[SID] gets bits 40–47, LTLCCCSR[FT] gets packet bits 12–15, LTLCCCSR[TT] gets packet bits 48–51, LTLCCCSR[MI] gets packet bits 56–63.	RapidIO packet is dropped
Received TT which is not enabled. - Error valid when passthrough is disabled and accept_all is disabled Or when accept_all is enabled.	Yes if LTLEECSR [TSE] is set	LTLEDCSR[TSE]	No	Same as third entry	RapidIO packet is dropped

Table 15-115. Hardware Errors For Message Request Transactions (continued)

Error	Interrupt Generated	Status Bit Set	RapidIO Error Response Generated	Logical/Transport Layer Capture Register	Comments
DestID DestID does not match this port's DeviceID if Alternate DeviceID is disabled or DestId does not match either Alternate DeviceID or DeviceId if Alternate DeviceID is enabled. Error valid when (passthrough or accept_all) is false	Yes if LTLEECSR [ITTE] is set	LTLEDCSR[ITTE]	Yes if priority is not 3. Else packet is dropped	Same as third entry	OCN error response is sent to self if request priority is not 3. Else packet is dropped.
SourceID Not Checked for error.	—	—	—	—	—
MsgLen,Ssize, Ltr,Mbox,MsgSeg Not Checked for error.	—	—	—	—	—
PayloadSize Message payload size is larger than the specified ssize, or is of size 0 when seg_len == msg_len. Or Message payload size is not equal to specified ssize when seg_len != msg_len.	Yes if LTLEECSR [MFE] is set	LTLEDCSR[MFE]	Yes if priority is not 3. Else packet is dropped	Same as third entry.	OCN error response is sent to self if request priority is not 3. Else packet is dropped.
Reserved ssize field	Yes if LTLEECSR [MFE] is set	LTLEDCSR[MFE]	Yes if priority is not 3. Else packet is dropped	Same as third entry.	OCN error response is sent to self if request priority is not 3. Else packet is dropped.
Other Received Message request with SOCAR[M] disabled	Yes if LTLEECSR [UT] is se	LTLEDCSR[UT]	Yes if priority is not 3. Else packet is dropped	Same as third entry	OCN error response is sent to self if request priority is not 3. Else packet is dropped.

Table 15-116. Hardware Errors For Message Response Transactions

Error	Interrupt Generated	Status Bit Set	OCN Error Response Generated	Logical/Transport Layer Capture Register	Comments
Priority Not Checked for error.	—	—	—	Using the original request RapidIO packet, for small Transport type, LTLACCSR[XA] gets packet bits 78–79, LTLACCSR[A] gets packet bits 48–76, LTLDIDCCSR[DIDMSB] gets 0's, LTLDIDCCSR[DID] gets packet bits 16–23, LTLDIDCCSR[SIDMSB] gets 0's, LTLDIDCCSR[SID] gets packet bits 24–31, LTLCCCSR[FT] gets packet bits 12–15, LTLCCCSR[TT] gets packet bits 32–35, LTLCCCSR[MI] gets packet bits 40–47. For Large Transport type packets. LTLACCSR[XA] gets packet bits 94–95, LTLACCSR[A] gets packet bits 64–92, LTLTLTLDIDCCSR[DIDMSB] gets 16–23, LTLDIDCCSR[DID] LTL gets packet bits 24–31, LTLDIDCCSR[SIDMSB] gets bits 32–39, LTLDIDCCSR[SID] gets bits 40–47, LTLCCCSR[FT] gets packet bits 12–15, LTLCCCSR[TT] gets packet bits 48–51, LTLCCCSR[MI] gets packet bits 56–63.	—
TransportType Received reserved TT	Yes if LTLEECSR[TSE] is set	LTLEDCSR [TSE]	No	Same as first entry except capture registers are loaded from the response RapidIO packet	RapidIO packet is dropped and ignored
Received TT which is not enabled. - Error valid when passthrough is disabled and accept_all is disabled Or when accept_all is enabled.	Yes if LTLEECSR[TSE] is set	LTLEDCSR [TSE]	No	Same as first entry except capture registers are loaded from the response RapidIO packet	RapidIO packet is dropped and ignored
DestID (All non-maintenance) DestID does not match this port's DeviceID if Alternate DeviceID is disabled or DestId does not match either Alternate DeviceID or DeviceId if Alternate DeviceID is enabled. Error valid when (passthrough or accept_all) is false	Yes if LTLEECSR[ITTE] is set	LTLEDCSR [ITTE]	No	Same as first entry except capture registers are loaded from the response RapidIO packet	RapidIO packet is dropped and ignored
SourceID Not Checked for error.	—	—	—	—	—

Table 15-116. Hardware Errors For Message Response Transactions (continued)

Error	Interrupt Generated	Status Bit Set	OCN Error Response Generated	Logical/Transport Layer Capture Register	Comments
Status Not Checked for error.	—	—	—	—	—
Other Received Message response with SOCAR[M] disabled.	Yes if LTLEECSR[UR] is se	LTLEDCSR[UR]	No	Same as first entry except capture registers are loaded from the response RapidIO packet	RapidIO packet is dropped and ignored

Table 15-117. Hardware Errors For Doorbell Request Transaction

Error	Interrupt Generated	Status Bit Set	Error Response Generated	Logical/Transport Layer Capture Register	Comments
Priority Not Applicable	—	—	—	—	—
TransportType Received reserved TT	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	Using the original request RapidIO packet, for small Transport type, LTLACCSR[XA] gets packet bits 78–79, LTLACCSR[A] gets packet bits 48–76, LTLDIDCCSR[DIDMSB] gets 0's, LTLDIDCCSR[DID] gets packet bits 16–23, LTLDIDCCSR[SIDMSB] gets 0's, LTLDIDCCSR[SID] gets packet bits 24–31, LTLCCCSR[FT] gets packet bits 12–15, LTLCCCSR[TT] gets packet bits 32–35, LTLCCCSR[MI] gets 0's For Large Transport type packets. LTLACCSR[XA] gets packet bits 94–95, LTLACCSR[A] gets packet bits 64–92, LTLTLTLDIDCCSR[DIDMSB] gets 16–23, LTLDIDCCSR[DID] LTL gets packet bits 24–31, LTLDIDCCSR[SIDMSB] gets bits 32–39, LTLDIDCCSR[SID] gets bits 40–47, LTLCCCSR[FT] gets packet bits 12–15, LTLCCCSR[TT] gets packet bits 48–51, LTLCCCSR[MI] gets 0's	RapidIO packet is dropped
Received TT which is not enabled. - Error valid when passthrough is disabled and accept_all is disabled Or when accept_all is enabled.	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	Same as third entry	RapidIO packet is dropped

Table 15-117. Hardware Errors For Doorbell Request Transaction (continued)

Error	Interrupt Generated	Status Bit Set	Error Response Generated	Logical/Transport Layer Capture Register	Comments
DestID DestID does not match this port's DeviceID if Alternate DeviceID is disabled or DestId does not match either Alternate DeviceID or DeviceId if Alternate DeviceID is enabled. Error valid when (passthrough or accept_all) is false	Yes if LTLEECSR[ITTE] is set	LTLEDCSR[ITTE]	Yes if priority is not 3. Else packet is dropped	Same as third entry	OCN error response is sent to self if request priority is not 3. Else packet is dropped.
SourceID Not Checked for error.	—	—	—	—	—
SrcTID Not Checked for error.	—	—	—	—	—
Other Received Doorbell request with DOCAR[D] disabled.	Yes if LTLEECSR[UT] is se	LTLEDCSR[UT]	Yes if priority is not 3. Else packet is dropped	Same as third entry	OCN error response is sent to self if request priority is not 3. Else packet is dropped.

Table 15-118. Hardware Errors For Doorbell Response Transactions

Error	Interrupt Generated	Status Bit Set	OCN Error Response Generated	Logical/Transport Layer Capture Register	Comments
Priority Not UR or UT Response priority is not higher than RapidIO request priority	Yes if LTLEECSR [ITD] is se	LTLEDCSR [ITD]	No	Using the incoming RapidIO packet, for small Transport type, LTLACCSR[XA] gets packet bits 78–79, LTLACCSR[A] gets packet bits 48–76, LTLDIDCCSR[DIDMSB] gets 0's, LTLDIDCCSR[DID] gets packet bits 16–23, LTLDIDCCSR[SIDMSB] gets 0's, LTLDIDCCSR[SID] gets packet bits 24–31, LTLCCCSR[FT] gets packet bits 12–15, LTLCCCSR[TT] gets packet bits 32–35, LTLCCCSR[MI] gets 0's For Large Transport type packets r. LTLACCSR[XA] gets packet bits 94–95, LTLACCSR[A] gets packet bits 64–92, LTLTLTLDIDCCSR[DIDMSB] gets 16–23, LTLDIDCCSR[DID] LTL gets packet bits 24–31, LTLDIDCCSR[SIDMSB] gets bits 32–39, LTLDIDCCSR[SID] gets bits 40–47, LTLCCCSR[FT] gets packet bits 12–15, LTLCCCSR[TT] gets packet bits 48–51, LTLCCCSR[MI] gets 0's	RapidIO packet is dropped and ignored
TransportType Received reserved TT	Yes if LTLEECSR [TSE] is set	LTLEDCSR [TSE]	No	Same as first entry	RapidIO packet is dropped and ignored
Received TT Error valid when passthrough is disabled and accept_all is disabled Or when accept_all is enabled.	Yes if LTLEECSR [TSE] is set	LTLEDCSR [TSE]	No	Same as first entry	RapidIO packet is dropped and ignored
DestID DestID does not match this port's DeviceID if Alternate DeviceID is disabled or DestId does not match either Alternate DeviceID or DeviceId if Alternate DeviceID is enabled. Error valid when (passthrough or accept_all) is false	Yes if LTLEECSR [ITTE] is set	LTLEDCSR [ITTE]	No	Same as first entry	RapidIO packet is dropped and ignored
SourceID Does not match the request's DestID	Yes if LTLEECSR [UR] is set	LTLEDCSR [UR]	No	Same as first entry	RapidIO packet is dropped and ignored

Table 15-118. Hardware Errors For Doorbell Response Transactions (continued)

Error	Interrupt Generated	Status Bit Set	OCN Error Response Generated	Logical/Transport Layer Capture Register	Comments
Status Not UR or UT Not one of Done/Error/Retry	Yes if LTLEECSR[ITD] is se	LTLEDCSR[ITD]	No	Same as first entry	RapidIO packet is dropped and ignored
TransactionType Not UR or UT Anything other than Done_No_Data	Yes if LTLEECSR[ITD] is se	LTLEDCSR[ITD]	No	Same as first entry	RapidIO packet is dropped and ignored
TargetTID No outstanding transaction for this TargetTID	Yes if LTLEECSR[UR] is set	LTLEDCSR[UR]	No	Same as first entry	RapidIO packet is dropped and ignored.
Packet Size Not UR or UT Header size is not 8 Bytes for Small Transport packet or not 12 Bytes for Large Transport packet Note: Two byte padding of 0's in Large Transport field packet is not checked.	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No	Same as first entry	RapidIO packet is dropped and ignored.
Packet response time-out Response is not received by configured time	Yes if LTLEECSR[PRT] is set	LTLEDCSR[PRT]	Yes	Same as first entry except capture registers are loaded from original request RapidIO packet.	OCN doorbell PRT response is generated to requestor.

Table 15-119. Hardware Errors for PortWrite Transaction

Error	Interrupt Generated	Status Bit Set	Error Response Generated	Logical/Transport Layer Capture Register	Comments
Priority Not Applicable	—	—	—	—	—
TransportType Received reserved TT	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	Using the original request RapidIO packet, for small Transport type, LTLACCSR[XA] gets packet bits 78–79, LTLACCSR[A] gets packet bits 48–76, LTLDIDCCSR[DIDMSB] gets 0's, LTLDIDCCSR[DID] gets packet bits 16–23, LTLDIDCCSR[SIDMSB] gets 0's, LTLDIDCCSR[SID] gets packet bits 24–31, LTLCCCSR[FT] gets packet bits 12–15, LTLCCCSR[TT] gets packet bits 32–35, LTLCCCSR[MI] gets 0's Large Transport type packets. LTLACCSR[XA] gets packet bits 94–95, LTLACCSR[A] gets packet bits 64–92, LTLTLTLDIDCCSR[DIDMSB] gets 16–23, LTLDIDCCSR[DID] LTL gets packet bits 24–31, LTLDIDCCSR[SIDMSB] gets bits 32–39, LTLDIDCCSR[SID] gets bits 40–47, LTLCCCSR[FT] gets packet bits 12–15, LTLCCCSR[TT] gets packet bits 48–51, LTLCCCSR[MI] gets 0's	RapidIO packet is dropped
Received TT which is not enabled. - Error valid when passthrough is disabled and accept_all is disabled Or when accept_all is enabled.	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	Same as third entry	RapidIO packet is dropped
DestID DestID does not match this port's DeviceID if Alternate DeviceID is disabled or DestId does not match either Alternate DeviceID or DeviceId if Alternate DeviceID is enabled. Error valid when (passthrough or accept_all) is false	Yes if LTLEECSR[ITTE] is set	LTLEDCSR[ITTE]	No	Same as third entry	RapidIO packet is dropped
SourceID Not Checked for error.	—	—	—	—	—
TransactionType Not Checked for error.	—	—	—	—	—

Table 15-119. Hardware Errors for PortWrite Transaction (continued)

Error	Interrupt Generated	Status Bit Set	Error Response Generated	Logical/Transport Layer Capture Register	Comments
WrSize Not unsupported transaction Is one of reserved sizes or less than 4 bytes	Yes if LTLEECSR[ITD] is se	LTLEDCSR[ITD]	No	Same as third entry	RapidIO packet is dropped
SrcTID Not Checked for error.	—	—	—	—	—
HopCount Not Checked for error.	—	—	—	—	—
ConfigOffset Not Checked for error.	—	—	—	—	—
PayloadSize Not unsupported transaction An incorrect port-write wr_size encoding (not 4, 8, 16, 24, 32, 40, 48, 56 or 64 bytes). Payload size is greater than the value defined by wr_size. Payload size is not dword aligned when the wr_size is not 4 bytes.	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No	Same as third entry	RapidIO packet is dropped
Other Received PortWrite transaction with DOCAR[PW] disabled.	Yes if LTLEECSR[UT] is set	LTLEDCSR[UT]	No	Same as third entry	RapidIO packet is dropped

Table 15-120. Hardware Errors for Reserved Ftype

Error	Interrupt Generated	Status Bit Set if Corresponding Bit is Enabled	RapidIO Error Response Generated	Logical/Transport Layer Capture Register	Comments
Ftype Ftype is not IO Read, IO Write, SWrite, Maintenance request, Maintenance Response, Response (Ftype 13), Doorbell or Message class and it is not a passthrough transaction. (passthrough is not enabled or accept_all is enabled or transaction is addressed to this port)	Yes if LTLEECSR[IUT] is set	LTLEDCSR[UT]	No	Using the original request RapidIO packet, for small Transport type, LTLACCSR[XA] gets packet bits 78–79, LTLACCSR[A] gets packet bits 48–76, LTLDIDCCSR[DIDMSB] gets 0's, LTLDIDCCSR[DID] gets packet bits 16–23, LTLDIDCCSR[SIDMSB] gets 0's, LTLDIDCCSR[SID] gets packet bits 24–31, LTLCCCSR[FT] gets packet bits 12–15, LTLCCCSR[TT] gets packet bits 32–35, LTLCCCSR[MI] gets 0's Large Transport type packets. LTLACCSR[XA] gets packet bits 94–95, LTLACCSR[A] gets packet bits 64–92, LTLTLTLDIDCCSR[DIDMSB] gets 16–23, LTLDIDCCSR[DID] LTL gets packet bits 24–31, LTLDIDCCSR[SIDMSB] gets bits 32–39, LTLDIDCCSR[SID] gets bits 40–47, LTLCCCSR[FT] gets packet bits 12–15, LTLCCCSR[TT] gets packet bits 48–51, LTLCCCSR[MI] gets 0's	RapidIO packet is dropped
TransportType Received reserved TT	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	Same as first entry	RapidIO packet is dropped
Received TT which is not enabled. - Error valid when passthrough is disabled and accept_all is disabled Or when accept_all is enabled.	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	Same as first entry	RapidIO packet is dropped
DestID DestID does not match this port's DeviceID if Alternate DeviceID is disabled or DestId does not match either Alternate DeviceID or DeviceId if Alternate DeviceID is enabled. Error valid when (passthrough or accept_all) is false	Yes if LTLEECSR[ITTE] is set	LTLEDCSR[ITTE]	No	Same as first entry	RapidIO packet is dropped

Table 15-120. Hardware Errors for Reserved Ftype (continued)

Error	Interrupt Generated	Status Bit Set if Corresponding Bit is Enabled	RapidIO Error Response Generated	Logical/Transport Layer Capture Register	Comments
Address:WdPtr:Xambs Request hits overlapping ATMU windows. Refer to 15.8.5.2/15-96 Packet checked as non Swrite packet	Yes if LTLEECSR[IACB] is set	LTLEDCSR[IACB]	No	Same as first entry	RapidIO packet is dropped
Address:WdPtr:Xambs Not unsupported transaction Request hits a protected ATMU window or the LCSBA1CSR Packet checked as non Swrite packet	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No	Same as first entry	RapidIO packet is dropped

Table 15-121. Hardware Errors for Outbound Transaction Crossed ATMU Boundary

Error	Interrupt Generated	Status Bit Set if corresponding bit is enabled	Logical/Transport Layer Capture Register	Comments
OCN Address and payload size OCN address range for Outbound RapidIO transaction hits multiple ATMU windows. Refer to 15.8.5.2/15-96	Yes if LTLEECSR[OACB] is set and/or LTLEECSR[PRT] is set	LTLEDCSR[OACB], LTLEDCSR[PRT]	Using the original request RapidIO packet send out by OB, for small Transport type, LTLACCSR[XA] gets packet bits 78–79, LTLACCSR[A] gets packet bits 48–76, LTLDIDCCSR[DIDMSB] gets 0's, LTLDIDCCSR[DID] gets packet bits 16–23, LTLDIDCCSR[SIDMSB] gets 0's, LTLDIDCCSR[SID] gets packet bits 24–31, LTLCCCSR[FT] gets packet bits 12–15, LTLCCCSR[TT] gets packet bits 32–35, LTLCCCSR[MI] gets 0's For Large Transport type packets, LTLACCSR[XA] gets packet bits 94–95, LTLACCSR[A] gets packet bits 64–92, LTLTLTLDIDCCSR[DIDMSB] gets 16–23, LTLDIDCCSR[DID] LTL gets packet bits 24–31, LTLDIDCCSR[SIDMSB] gets bits 32–39, LTLDIDCCSR[SID] gets bits 40–47, LTLCCCSR[FT] gets packet bits 12–15, LTLCCCSR[TT] gets packet bits 48–51, LTLCCCSR[MI] gets 0's	—

Table 15-122. Hardware Errors for Outbound Packet Time-to-live Errors

Error	Interrupt Generated	Status Bit Set if corresponding bit is enabled	Logical/Transport Layer Capture Register	Comments
Packet time-to-live error	Yes if LTLEECSR[PTTL] is set	LTLEDCSR[PTTL]	Using the RapidIO packet attempted to send outbound, if configured in small transport mode, LTLACCSR[XA] gets packet bits 78–79, LTLACCSR[A] gets packet bits 48–76, LTLDIDCCSR[DIDMSB] gets 0's, LTLDIDCCSR[DID] gets packet bits 16–23, LTLDIDCCSR[SIDMSB] gets 0's, LTLDIDCCSR[SID] gets packet bits 24–31, LTLCCCSR[FT] gets packet bits 12–15, LTLCCCSR[TT] gets packet bits 32–35, LTLCCCSR[MI] gets 0's For large transport mode, LTLACCSR[XA] gets packet bits 94-95, LTLACCSR[A] gets packet bits 64-92, LTLTLTLDIDCCSR[DIDMSB] gets 16-23, LTLDIDCCSR[DID] LTL gets packet bits 24–31, LTLDIDCCSR[SIDMSB] gets bits 32-39, LTLDIDCCSR[SID] gets bits 40-47, LTLCCCSR[FT] gets packet bits 12–15, LTLCCCSR[TT] gets packet bits 48–51, LTLCCCSR[MI] gets 0's	—

15.9 RapidIO Message Unit

This message unit supports multicast and non-multicast single segment messages and non-multicast multiple segment messages.

15.9.1 Overview

The RapidIO message unit supports a message passing programming model for inter-processor and inter-device communication. This model enables a producer to send a message across the interconnect fabric to a consumer's message hardware, called a mailbox. The receiving mailbox hardware places the message in a queue located in local memory. A message may consist of one to sixteen segments. When a configured number of messages have been received, an interrupt (if enabled) is generated to the interrupt controller for the processor to process the messages. Messages can be queued for transmission in the producer's memory, and the message hardware will process them sequentially. Messages can also be queued in the consumer's memory while software processes them sequentially. The depths of the queues in the producer and consumer are configurable by software. A multicast function allows single segment messages to be sent to multiple consumers.

The message unit is compliant with the message passing logical specification contained in the *RapidIO Interconnect Specification, Revision 1.2* (but assumes the msgseg field is redefined for more mailboxes when single segment messages are being sent). The most common use of the message passing model is in systems where a processing element is only allowed to access memory that is local to itself, and communication between processing elements is achieved through message passing and communication is address independent.

Each inbound message controller has a dedicated interrupt that can be used to notify software that a configured number of messages have been received. Similarly, each outbound message controller has a dedicated interrupt that can be used to notify software that there are no more messages for the outbound mailbox controller to process.

The message controller is managed through a set of run-time registers.

15.9.2 Features

- Support for one or more outbound message controllers with the following features
  - chaining and direct modes
  - extended mailboxes (XMBOX) for single segment messages
  - multicast up to 32 RapidIO destinations for single segment messages
  - transmitting to any mailbox and extended mailbox for a single segment message (letter 3 is reserved for an alternative message source like the DMA but can be used if the DMA does not generate messages and the RapidIO endpoint is configured appropriately)
  - transmitting to any mailbox for multi segment message (letter 3 is reserved for an alternative message source like the DMA, but can be used if the DMA does not generate messages and the RapidIO endpoint is configured appropriately)
  - segment size up to 256 bytes
  - up to sixteen segment messages with a total payload of up to 4 Kbytes
  - one entire message (up to a 16 message segments) can be transmitted before receiving any response
  - one entire single segment message to all multicast destinations (up to 32) can be transmitted before receiving any response
  - all message segment transfers for a message transaction must complete before the next message transaction begins
  - pipelined transmission of a full message in each message controller but all responses must be received before the next message can be transmitted
  - in chaining mode the next descriptor can be fetched before the current message completes (descriptor prefetching)
- Support for one or more inbound message controllers with the following features
  - reception of any mailbox and letter for a single or multi segment message
  - segment size up to 256 bytes
  - up to sixteen segment messages with a total payload of up to 4 Kbytes
  - full inbound line rate performance

— back-to-back message reception of the same or lower priority
— out-of-order message segment reception
— concurrent inbound message controller operation

15.9.3 Outbound Modes of Operation

- Direct mode: Software is expected to program all the necessary registers for sending an outbound message.
- Chaining mode: A descriptor that describes the message information is fetched from local memory before a message is sent.
- Multicast mode: A single segment message (256 bytes or less) is sent to multiple destinations. This mode is supported in direct or chaining mode.

The following sections describe the structure and operation of the outbound message controller and inbound message controller hardware in the data message controller.

15.9.4 Outbound Message Controller Operation

The outbound message controller is responsible for sending messages stored in local memory. The outbound message controller supports three modes of operation, direct, chaining, and multicast mode. In direct mode, software programs the necessary registers to point to the beginning of the message in memory. In chaining mode, software programs the necessary registers to point to the beginning of the first valid descriptor in memory. The descriptor provides all the necessary registers to start the message transfer. In multicast mode, a single segment message can be sent to multiple destinations. Multicast mode is supported in direct or chaining mode.

Each outbound message controller uses a unique letter number. For example, if there are two outbound message controllers, message controller 0 uses letter 0 and message controller 1 uses letter 1.

15.9.4.1 Direct Mode Operation

In direct mode (OM*n*MR[MUTM] is set in the outbound message mode register) the outbound message controller does not read descriptors from memory, but instead uses the current parameters programmed in the outbound message controller registers to start the transfer. In direct mode, software is responsible for initializing all the parameters in all the necessary registers to start the message transmission. The message transfer is started when the outbound message controller start bit, OM*n*MR[MUS], in the outbound message mode register transitions from a 0 to 1 and the outbound message controller is not already busy. If the outbound message controller is already busy, OM*n*MR[MUS] transitioning from 0 to 1 is ignored. Software is expected to program all the appropriate registers before setting OM*n*MR[MUS].

There are many ways in which software can interact with the message controller. One sequence of events to start and complete a transfer in direct mode is as follows:

- Poll the status register message unit busy bit, OM*n*SR[MUB], to make sure the outbound message controller is not busy with a previously initiated message.
- Clear the status bits (OM*n*SR[MER], OM*n*SR[RETE], OM*n*SR[PRT], OM*n*SR[TE], OM*n*SR[QOI], OM*n*SR[QFI], OM*n*SR[EOMI], and OM*n*SR[QEI])

- Initialize the source address (EOM*n*SAR, OM*n*SAR), destination port (OM*n*DPR), destination attributes (OM*n*DATR), retry error threshold (OM*n*RETCR), and double-word count (OM*n*DCR) registers. If multicast mode is enabled (OM*n*DATR[MM]) initialize the multicast group and list (OM*n*MGR, OM*n*MLR).
- Initialize the outbound message mode register message unit transfer mode bit, OM*n*MR[MUTM] = 1, to indicate direct mode. Other control parameters must also be initialized in the mode register.
- Clear, then set the mode register message unit start bit, OM*n*MR[MUS], to start the message transfer.
- OM*n*SR[MUB] bit is set by the outbound message controller to indicate the message transfer is in progress.
- The outbound message controller reads a message segment from local memory using the source address register (EOM*n*SAR, OM*n*SAR).
- If a message has multiple segments, the outbound message controller reads the other message segments from local memory.
- After the message read to local memory completes, the message is sent.
- The OM*n*SR[MUB] is cleared by the outbound message controller after the message operation completes. A non multicast message transfer completes after all message segments complete. A multicast message transfer completes after all message segments complete for each destination. A message segment completes when one of the following occurs:
  - done response received
  - error response
  - a packet response time-out received
  - retry error threshold exceeded
  - an internal error occurs during the local memory access
- After the outbound message operation completes, the outbound message interrupt is generated if the end of message outbound message interrupt event is enabled (OM*n*DATR[EOMIE]).

15.9.4.1.1 Interrupts

The outbound message controller interrupt can be generated for one reason in direct mode.

- End-Of-Message — generates an interrupt after the completion of a message if this interrupt event is enabled (OM*n*DATR[EOMIE] is a 1). OM*n*SR[EOMI] provides the event that caused this interrupt. The interrupt is held until the OM*n*SR[EOMI] bit is cleared by writing a 1.

The error/port-write interrupt can be generated for the following reasons in direct mode.

- message error response — an interrupt is generated after a message error response is received and this interrupt event is enabled (OM*n*MR[EIE])
- packet response time-out — an interrupt is generated after a packet response time-out occurs and this interrupt event is enabled (OM*n*MR[EIE])
- retry error threshold exceeded — an interrupt is generated after a retry threshold exceeded error occurs and this interrupt event is enabled (OM*n*MR[EIE])

- transaction error — an interrupt is generated after an OCN error response is received and this interrupt event is enabled (OM*n*MR[EIE])

15.9.4.1.2 Message Error Response Errors

When a message error response is received by the message controller the following occurs.

- The message controller sets the message error response status bit (OM*n*SR[MER])
- If OM*n*MR[EIE] is set, the interrupt SRIO error/write-port is generated.
- After the message operation completes (indicated by OM*n*SR[MUB]) the message controller stops

15.9.4.1.3 Packet Response Time-out Errors

When a packet response time-out occurs for a message segment the following occurs.

- The message controller sets the packet response time-out status bit (OM*n*SR[PRT])
- If OM*n*MR[EIE] is set, the interrupt SRIO error/write-port is generated.
- After the message operation completes (indicated by OM*n*SR[MUB]) the message controller stops

15.9.4.1.4 Retry Error Threshold Exceeded Errors

When a retry error threshold exceeded error occurs for a message segment the following occurs.

- The message controller sets the retry threshold exceed status bit (OM*n*SR[RETE])
- If OM*n*MR[EIE] is set, the interrupt SRIO error/write-port is generated.
- After the message operation completes (indicated by OM*n*SR[MUB]) the message controller stops

15.9.4.1.5 Transaction Errors

When an internal error occurs during a local memory read by the message controller the following occurs.

- The message controller sets the transaction error bit (OM*n*SR[TE])
- Message segments that have an internal error are not sent because the message data is not available
- Memory reads that already were generated before the internal error occurred are also not transferred.
- Additional memory reads for the same message operation are generated but not transferred.
- All subsequent message segments for the same message operation are not transferred. This includes retried message segments.
- After the message operation completes (indicated by OM*n*SR[MUB]) the message controller stops
- If OM*n*MR[EIE] is set, the interrupt SRIO error/write-port is generated.

15.9.4.1.6 Error Handling

When an error occurs and the SRIO error/write-port interrupt is generated, the following occurs.

- Software determines the cause of the interrupt and processes the error
- Software verifies the message controller has stopped operation by polling OM*n*SR[MUB]
- Software disables the message controller by clearing OM*n*MR[MUS]

- Software clears the error by writing a 1 to the corresponding status bit (OM*n*SR[MER], OM*n*SR[PRT], OM*n*SR[RETE], or OM*n*SR[TE])

When an error occurs and the SRIO error/write-port interrupt is not enabled, the following occurs.

- Software determines that an error has occurred by polling the status bits (OM*n*SR[MER], OM*n*SR[PRT], OM*n*SR[RETE], or OM*n*SR[TE])
- Software verifies the message controller has stopped operation by polling OM*n*SR[MUB]
- Software disables the message controller by clearing OM*n*MR[MUS]
- Software clears the error by writing a 1 to the corresponding status bit (OM*n*SR[MER], OM*n*SR[PRT], OM*n*SR[RETE], or OM*n*SR[TE])

15.9.4.1.7 Disabling and Enabling the Message Controller

Once the message controller is started, it cannot be stopped.

15.9.4.1.8 Hardware Error Handling

The following list possible error conditions and what occurs when they are encountered. The error checking level indicates the order in which errors are checked. Multiple errors can be checked at an error checking level. Once an error is detected no additional error checking beyond the current level is performed. The first error detected in the processing pipeline updates the error management extensions registers.

These error condition checks are provided by the messaging unit. These check are in addition to the error condition checks provided by the RapidIO port given in Section 15.8.12, “Errors and Error Handling.”

Table 15-123. Outbound Message Direct Mode Hardware Errors

Transaction	Error	Error Checking Level	Interrupt Generated	Status Bit Set	Message Segment Sent	Logical/Transport Layer Capture Register	Comments
Message Request	An internal error occurred during a read of the message segment from local memory	1	SRIO error/write-port if OM*n*MR[EIE] set	Transaction error in the outbound message status register (OM*n*SR[TE]). Message Failed in the Mailbox CSR (MCSR[FA]).	No	—	Message controller stops after the current message operation completes. The descriptor dequeue pointer is not incremented in chaining mode.
Message Request	An internal error occurred for an earlier message segment local memory read. An internal error for a subsequent message segment local memory read for the same message may or may not occur.	2	No	None	Yes	—	Message controller stops after the current message operation completes.
Undefined Packet	Reserved ftype encoding[1]	3	SRIO error/write-port if LTLEECSR[UT] set	Unsupported transaction in the Logical/Transport Layer Error Detect CSR LTLEDCSR[UT]	Yes	Updated with the packet[2]	Packet is ignored and discarded.

Table 15-123. Outbound Message Direct Mode Hardware Errors (continued)

Transaction	Error	Error Checking Level	Interrupt Generated	Status Bit Set	Message Segment Sent	Logical/Transport Layer Capture Register	Comments
Message Response	Reserved tt encoding[1]	3	SRIO error/write-port if LTLEECSR[TSE] set	Illegal transaction target in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITTE] Transport size error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[TSE]	Yes	Updated with the packet[2]	Packet is ignored and discarded.
Message Response	Large transport size when operating in small transport size or small transport size when operating in large transport size[1]	3	SRIO error/write-port if LTLEECSR[TSE] set	Transport size error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[TSE]	Yes	Updated with the packet[2]	Packet is ignored and discarded. An error or illegal transaction target error response is not generated.
Message Response	Illegal Destination ID[1]	3	SRIO error/write-port if LTLEECSR[ITTE] set	Illegal transaction target in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITTE]	Yes	Updated with the packet[2]	Packet is ignored and discarded.
Message Response	ttype (transaction field) is not message response[1]	3	SRIO error/write-port if LTLEECSR[ITD] set	Illegal transaction decode in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITD]	Yes	Updated with the packet[2]	Packet is ignored and discarded.
Message Response	Message response received and no outbound mailboxes are supported[1]	3	SRIO error/write-port if LTLEECSR[UR] set	Unsupported transaction in the Logical/Transport Layer Error Detect CSR LTLEDCSR[UR]	No	Updated with the packet[2]	Packet is ignored and discarded.

Table 15-123. Outbound Message Direct Mode Hardware Errors (continued)

Transaction	Error	Error Checking Level	Interrupt Generated	Status Bit Set	Message Segment Sent	Logical/Transport Layer Capture Register	Comments
Message Response	reserved response status (not done, retry, or error)	4a	SRIO error/write-port if LTLEECSR[ITD] set	Illegal transaction decode in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITD]	Yes	Updated with the packet[2]	Packet is ignored and discarded.
Message Response	message response packet size is incorrect	4a	SRIO error/write-port if LTLEECSR[ITD] set	Message Format error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITD]	Yes	Updated with the packet[2]	Packet is ignored and discarded.
Message Response	Incorrect Source ID	4b	SRIO error/write-port if LTLEECSR[UR] set	Illegal transaction decode in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITD]	Yes	Updated with the packet[2]	Packet is ignored and discarded.
Message Response	letter, mbox and msgseg not outstanding or letter, mbox and xmbox not outstanding	4b	SRIO error/write-port if LTLEECSR[UR] set	Unsolicited response in the Logical/Transport Layer Error Detect CSR LTLEDCSR[UR]	Yes	Updated with the packet[2]	Packet is ignored and discarded.
Message Response	RapidIO priority is less than or equal to message request	4c	SRIO error/write-port if LTLEECSR[ITD] set	Illegal transaction decode in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITD]	Yes	Updated with the packet[2]	Packet is ignored and discarded.
Message Response	error response	5	SRIO error/write-port if LTLEECSR[MER] set. SRIO error/write-port if OM*n*MR[EIE].	Message error response in the Logical/Transport Layer Error Detect CSR LTLEDCSR[MER]. OM*n*SR[MER] bit set if in direct mode or if in chaining mode.	Yes	Updated with the corresponding message request packet[2]	Message segment transfer complete. The descriptor dequeue pointer is not incremented in chaining mode.

Table 15-123. Outbound Message Direct Mode Hardware Errors (continued)

Transaction	Error	Error Checking Level	Interrupt Generated	Status Bit Set	Message Segment Sent	Logical/Transport Layer Capture Register	Comments
Message Response	number of retries exceeds limit	5	SRIO error/write-port if LTLEECSR[RETE] set. SRIO error/write-port if OM*n*MR[EIE].	Retry error threshold exceeded in the Logical/Transport Layer Error Detect CSR LTLEDCSR[RETE] OM*n*SR[RETE] bit set if in direct mode or if in chaining mode.	Yes	Updated with the corresponding message request packet[2]	Message segment transfer complete. The descriptor dequeue pointer is not incremented in chaining mode.
Message Response	packet response time-out	unrelated	SRIO error/write-port if LTLEECSR[PRT] set. SRIO error/write-port if OM*n*MR[EIE].	Packet response time-out in the Logical/Transport Layer Error Detect CSR LTLEDCSR[PRT]. OM*n*SR[PRT] bit set if in direct mode or if in chaining mode.	Yes	Updated with the corresponding message request packet.[2] The LTLDIDCCSR[SIDMSB] and LTLDIDCCSR[SID] field is 0.	Message segment transfer complete. The descriptor dequeue pointer is not incremented in chaining mode.

Note:

1. These error types are actually detected in the RapidIO port, not in the message controller
2. If operating in small transport size configuration using the packet LTLACCSR[XA] gets the extended address (packet bits 78–79), LTLACCSR[A] gets the address (packet bits 48–76), LTLDIDCCSR[MDID] gets 0, LTLDIDCCSR[DID] gets the least significant byte of the destination ID (packet bits 16–23), LTLDIDCCSR[MSID] gets 0, LTLDIDCCSR[SID] gets the least significant byte of the source ID (packet bits 24–31), LTLCCCSR[FT] gets the ftype (packet bits 12–15), LTLCCCSR[TT] gets the ttype (packet bits 32–35), LTLCCCSR[MI] gets the msg info (packet bits 40–47). If operating in large transport size configuration using the packet LTLACCSR[XA] gets the extended address (packet bits 94–95), LTLACCSR[A] gets the address (packet bits 64–92), LTLDIDCCSR[MDID] gets the most significant byte of the destination ID (packet bits 16–23), LTLDIDCCSR[DID] gets the least significant byte of the destination ID (packet bits 24–31), LTLDIDCCSR[MSID] gets the most significant byte of the source ID (packet bits 32–39), LTLDIDCCSR[SID] gets the least significant byte of the source ID (packet bits 40–47), LTLCCCSR[FT] gets the ftype (packet bits 12–15), LTLCCCSR[TT] gets the ttype (packet bits 48–51) if the message request packet is captured or 0 if the message response packet is captured, LTLCCCSR[MI] gets the msg info (packet bits 56–63).

15.9.4.1.9 Programming Errors

The following is the list of programming errors that result in undefined or undesired hardware operation.

Table 15-124. Outbound Message Direct Mode Programming Errors

Error	Interrupt Generated	Status Bit Set	Comments
double-word count greater than 256 bytes when multicast mode selected	No	None	Undefined operation results
double-word count set to a reserved value	No	None	Undefined operation results
Transaction flow level set to 3	No	None	Undefined operation results
Target interface set to an invalid RapidIO port	No	None	Undefined operation results
Source address for message read is invalid	No	No	Local memory captures the transaction and generates an interrupt.
address for error enqueue address pointer is invalid	No	No	Local memory captures the transaction and generates an interrupt.
register values changed during operation	No	No	Undefined operation results

15.9.4.2 Chaining Mode

In chaining mode, OM*n*MR[MUTM] = 0 in the outbound message mode register, message descriptors are built in local memory in a circular queue. There are several options available to the programmer in chaining mode. Software can build one or more descriptors in memory before initializing the outbound message controller registers, or it can initialize the outbound message controller registers and then build the descriptors. The enqueue pointer (OM*n*DQEPAR, EOM*n*DQEPAR) is maintained by software. The outbound message controller dequeues descriptors, processes them and increments the dequeue pointer (OM*n*DQDPAR, EOM*n*DQDPAR) to point to the next descriptor in the queue.

Figure 15-99 below depicts a sample structure of the outbound portion of the message controller, and a descriptor queue, with each valid descriptor queue entry pointing to a valid message. In this example, the descriptor queue has eight entries, four of which are currently valid. The local processor enqueues descriptors and the outbound message controller dequeues the descriptors.

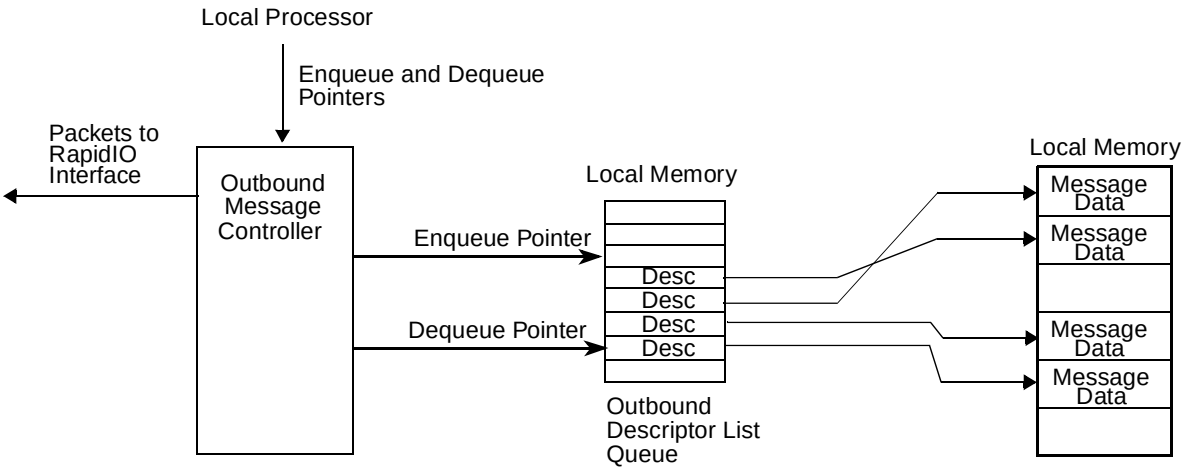


Figure 15-99. Outbound Frame Queue Structure

15.9.4.2.1 Message Controller Initialization

There are many ways in which software can interact with the message controller. One method to initialize the message controller is as follows:

- Poll the status register message unit busy bit, OM*n*SR[MUB], to make sure the outbound message controller is not busy with a previously initiated message.
- Clear the message unit start bit (OM*n*MR[MUS]).
- Initialize the descriptor queue dequeue pointer address registers (OM*n*DQDPAR, EOM*n*DQDPAR) and the descriptor queue enqueue pointer address registers (OM*n*DQEPAR, EOM*n*DQEPAR). These need to be initialized to the same value for proper operation.

  These pairs of registers must also be queue size aligned (that is, the queue must be aligned on a boundary equal to number of queue entries × 32 bytes (size of each queue descriptor)). For example, if there are 16 entries in the queue, the queue must be 512-byte aligned.

  The number of queue entries is set in OM*n*MR[CIRQ_SIZ]; see Section 15.7.1.1, “Outbound Message n Mode Registers (OMnMR)”).
- Initialize the retry error threshold in the outbound message retry error threshold configuration register (OM*n*RETCR).
- Clear OM*n*MR[MUTM] for chaining mode.
- If using single segment multicast mode, set OM*n*DATR[MM].
- Configure the other control parameters in the mode register (OM*n*MR).
- Clear OM*n*SR[MER, PRT, RETE, TE, QOI, QFI, EOMI and QEI]. If OM*n*SR[MER, PRT, RETE, TE or QOI] are not cleared, the message controller can not start a new message operation. Incorrect status will be indicated if the other status bits are not cleared.
- Set the message unit start (OM*n*MR[MUS]). This enables the outbound message controller and causes the descriptor dequeue pointer (OM*n*DQDPAR, EOM*n*DQDPAR) to be saved off as the base address of the descriptor queue.

15.9.4.2.2 Chaining Mode Operation

The method to start and complete transfers by adding descriptors after initializing the message unit is as follows:

- Create one or more descriptors in local memory starting at the address pointed to by the descriptor queue enqueue pointer address register (OM*n*DQEPAR, EOM*n*DQEPAR)
- Either increment the enqueue pointer address registers (OM*n*DQEPAR, EOM*n*DQEPAR) by setting OM*n*MR[MUI] for each descriptor entry added or directly change the enqueue pointer address register (OM*n*DQEPAR). If OM*n*MR[MUI] is set by software, the message controller clears this bit after successfully incrementing the enqueue pointer.
- When the descriptor queue is not empty, the message controller reads the descriptor from local memory using the address pointed to by the dequeue pointer (OM*n*DQDPAR, EOM*n*DQDPAR) and sets the busy bit (OM*n*SR[MUB]).
- If another descriptor is available, the message controller reads the next descriptor from local memory using the address pointed to by the dequeue pointer (OM*n*DQDPAR, EOM*n*DQDPAR). The message controller can not prefetch more than one descriptor.
- The message controller sets OM*n*SR[MUB] to indicate that the message transfer is in progress. OM*n*SR[MUB] remains set until the descriptor queue is empty or a transaction error occurs.
- Additional descriptors can be created and enqueued by software while the message controller is busy (OM*n*SR[MUB]). Software can continue adding descriptors as long as the descriptor queue is not full. If software is adding descriptors using the OM*n*MR[MUI] bit, overflowing the queue can be prevented by polling the queue full bit (OM*n*SR[QF]) before creating and enqueueing the next descriptor.
- After the descriptor memory read completes, the corresponding message segment is read from local memory.
- If a message has multiple segments, the outbound message controller reads the other message segments from local memory.
- After the message read to local memory completes, the message is sent.
- If multicast is enabled, all of the indicated targets are sent the same message.
- A non multicast message transfer completes after all message segments complete. A multicast message transfer completes after all message segments complete for each destination. A message segment completes when one of the following occurs:
  - done response received
  - error response received
  - a packet response time-out occurred
  - retry error threshold exceeded
  - an internal error occurred during the descriptor (all message segments complete) or message read of local memory
- When processing for the current descriptor completes, if another descriptor is available the above steps are repeated.

- If a RapidIO error response is received, the message error response bit is set (OM*n*SR[MER]) and outbound message controller operation stops after all message segments complete. If OM*n*MR[EIE] is set, the interrupt SRIO error/write-port is generated.
- If a packet response time-out occurs, the packet response time-out bit is set (OM*n*SR[PRT]). If OM*n*MR[EIE] is set, the interrupt SRIO error/write-port is generated.
- If the retry error threshold value is exceeded for a specific segment, the retry error threshold exceeded bit is set (OM*n*SR[RETE]) and outbound message controller operation stops after all message segments complete. If OM*n*MR[EIE] is set, the interrupt SRIO error/write-port is generated.
- If an internal error occurs while reading local memory, the transaction error bit is set (OM*n*SR[TE]) and outbound message controller operation stops after all message segments complete. If OM*n*MR[EIE] is set, the interrupt SRIO error/write-port is generated.
- The above process continues until the descriptor queue is empty (dequeue pointer equals the enqueue pointer).
- The message unit clears OM*n*SR[MUB] after completing the processing of the last descriptor or a transaction error occurs.
- If an error occurs the message unit must be disabled, re-initialized and re-enabled before another message can be sent.

15.9.4.2.3 Changing Descriptor Queues in Chaining Mode

When software wants to switch to another descriptor queue in local memory, it must wait for the processing of the current queue to complete as indicated by the busy bit (OM*n*SR[MUB]). Software then disables the message controller by clearing OM*n*MR[MUS], changes the enqueue and dequeue descriptor pointers (EOM*n*DQEPAR, OM*n*DQEPAR and EOM*n*DQDPAR, OM*n*DQDPAR) and re-enables the message unit by setting OM*n*MR[MUS].

15.9.4.2.4 Preventing Queue Overflow in Chaining Mode

Software must guarantee that descriptors are not added to an already full queue. When the increment bit is used (OM*n*MR[MUI]) software can poll the queue full bit (OM*n*SR[QF]) before enqueueing another descriptor. When software sets the enqueue pointer directly, software is responsible for not overflowing the descriptor queue.

15.9.4.2.5 Switching Between Direct and Chaining Modes

The message unit architecture allows switching from direct mode to chaining mode and vice-versa once all the required parameters have been initialized in the appropriate registers and when the message unit is not busy with a current transfer as indicated by OM*n*SR[MUB] being cleared. When switching from direct mode to chaining mode, if OM*n*MR[MUS] is cleared and set, the message unit is re-initialized in chaining mode, and the outbound message descriptor queue dequeue pointer address is saved off as the new base address of the circular queue in memory. When switching from chaining mode to direct mode, OM*n*MR[MUS] must also be cleared and set.

15.9.4.2.6 Chaining Mode Descriptor Format

The descriptor contains information for the message unit controller to transfer data. Software must ensure that each descriptor is aligned on a 32-byte boundary and that the descriptor queue is on a queue boundary (in other words, on a boundary equal to number of queue entries × 32 bytes (size of each queue descriptor)). For each descriptor in the queue, the message unit controller starts a new message operation with the control parameters specified by the descriptor.

Table 15-125. Outbound Message Unit Descriptor Summary

Descriptor Field	Description
Source Extended Address	Contains the source address of the message operation for local addresses of greater than 32 bits. After the message controller reads the descriptor from memory, this field is loaded into the source extended address register.
Source Address	Contains the source address of the message operation. After the message controller reads the descriptor from memory, this field is loaded into the source address register.
Destination Port	Contains the destination port of the message operation. After the message controller reads the descriptor from memory, this field is loaded into the destination port register.
Destination Attributes	Contains transaction attributes of the message operation. After the message controller reads the descriptor from memory, this field is loaded into the destination attributes register.
Multicast Group	Contains the logical multicast group. Groups are defined a list of 32 numerically consecutive destinations by deviceID.
Multicast List	Contains a bit vector list of consecutive destinations by deviceID.
Double-word Count	Contains the number of doublewords for the message operation. After the message controller reads the descriptor from memory, this field is loaded into the double-word count register.
Reserved	—

Figure 15-100 depicts the queue dequeue pointer and an associated descriptor. The descriptor is only valid if the enqueue and dequeue pointers are not equal.

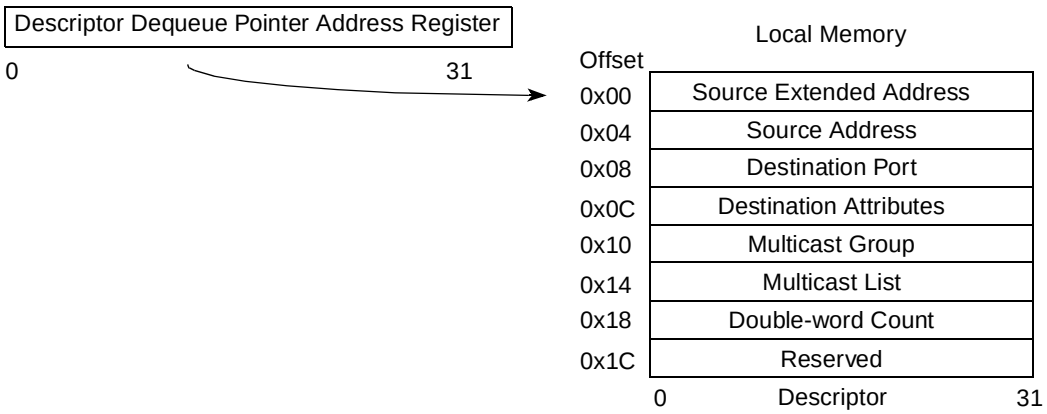


Figure 15-100. Descriptor Dequeue Pointer and Descriptor

15.9.4.2.7 Chaining Mode Controller Interrupts

The outbound message interrupt can be generated for several reasons.

- Queue empty—an interrupt is generated to the interrupt controller if the queue goes empty and the interrupt event is enabled (OM*n*MR[QEIE] is a 1). The event that caused the outbound message

interrupt is indicated by OM*n*SR[QEI]. The interrupt is held until the queue is no longer empty and the OM*n*SR[QEI] bit is cleared by writing a 1.

- Queue full—an interrupt is generated to the interrupt controller if the queue is full and the interrupt event is enabled (OM*n*MR[QFIE] is a 1). The event that caused the outbound message interrupt is indicated by OM*n*SR[QFI]. The interrupt is held until the queue is no longer full and the OM*n*SR[QFI] bit is cleared by writing a 1.
- Queue overflow—an interrupt is generated to the interrupt controller if the queue is full, the increment bit is set (OM*n*MR[MUI]) and the interrupt event is enabled (OM*n*MR[QOIE] is a 1). The event that caused the outbound message interrupt is indicated by OM*n*SR[QOI]. The message unit must be re-initialized. The interrupt is held until the OM*n*SR[QOI] bit has been cleared by writing a 1. This interrupt is also generated if the enqueue pointer is directly written and causes an overflow.
- End-of-message—an interrupt is generated after the completion of the message if this interrupt event is enabled (OM*n*DATR[EOMIE] is a 1). The event that caused this interrupt is indicated by OM*n*SR[EOMI]. The interrupt is held until the OM*n*SR[EOMI] bit is cleared by writing a 1. This allows an interrupt to be generated after a particular descriptor has been processed.

The error/port-write interrupt can be generated for the following reasons in direct mode.

- Message error response—an interrupt is generated after a message error response is received and this interrupt event is enabled (OM*n*MR[EIE])
- Packet response time-out—an interrupt is generated after a packet response time-out occurs and this interrupt event is enabled (OM*n*MR[EIE])
- Retry error threshold exceeded—an interrupt is generated after a retry threshold exceeded error occurs and this interrupt event is enabled (OM*n*MR[EIE])
- Transaction error—an interrupt is generated after a message error response is received and this interrupt event is enabled (OM*n*MR[EIE])

15.9.4.2.8 Message Error Response Errors

When a message error response is received by the message controller the following occurs.

- The message controller sets the message error response status bit (OM*n*SR[MER])
- If OM*n*MR[EIE] is set, the interrupt SRIO error/write-port is generated.
- After the message operation completes (indicated by OM*n*SR[MUB]) the message controller stops

15.9.4.2.9 Packet Response Time-out Errors

When a packet response time-out occurs for a message segment the following occurs.

- The message controller sets the packet response time-out status bit (OM*n*SR[PRT])
- If OM*n*MR[EIE] is set, the interrupt SRIO error/write-port is generated.
- After the message operation completes (indicated by OM*n*SR[MUB]) the message controller stops

15.9.4.2.10 Retry Error Threshold Exceeded Errors

When a retry error threshold exceeded error occurs for a message segment the following occurs.

- The message controller sets the retry threshold exceed status bit (OM*n*SR[RETE])
- If OM*n*MR[EIE] is set, the interrupt SRIO error/write-port is generated.
- After the message operation completes (indicated by OM*n*SR[MUB]) the message controller stops

15.9.4.2.11 Transaction Errors

When an internal error occurs during a local memory read by the message controller the following occurs.

- The message controller sets the transaction error bit (OM*n*SR[TE])
- If the internal error occurs during the descriptor memory read by the message controller, no message segment memory reads are generated and no message segments are sent
- Message segments that have an internal error are not sent because the message data is not available
- Memory reads that already were generated before the internal error occurred are also not transferred.
- Additional memory reads for the same message operation are generated but not transferred.
- All subsequent message segments for the same message operation are not transferred. This includes retried message segments.
- After the message operation completes (indicated by OM*n*SR[MUB]) the message controller stops
- If OM*n*MR[EIE] is set, the interrupt SRIO error/write-port is generated.

15.9.4.2.12 Error Handling

When an error occurs and the SRIO error/write-port interrupt is generated, the following occurs.

- Software determines the cause of the interrupt and processes the error
- Software verifies the message controller has stopped operation by polling OM*n*SR[MUB]
- Software disables the message controller by clearing OM*n*MR[MUS]
- Software clears the error by writing a 1 to the corresponding status bit (OM*n*SR[MER], OM*n*SR[PRT], OM*n*SR[RETE], or OM*n*SR[TE])

When an error occurs and the SRIO error/write-port interrupt is not enabled, the following occurs.

- Software determines that an error has occurred by polling the status bits (OM*n*SR[MER], OM*n*SR[PRT], OM*n*SR[RETE], or OM*n*SR[TE])
- Software verifies the message controller has stopped operation by polling OM*n*SR[MUB]
- Software disables the message controller by clearing OM*n*MR[MUS]
- Software clears the error by writing a 1 to the corresponding status bit (OM*n*SR[MER], OM*n*SR[PRT], OM*n*SR[RETE], or OM*n*SR[TE])

15.9.4.2.13 Hardware Error Handling

The following list additional error conditions beyond the errors listed for direct mode. All the errors listed in the direct mode section can also occur. The error checking level indicates the order in which errors are checked. Multiple errors can be checked at an error checking level. Once an error is detected no additional error checking beyond the current level is performed. The first error detected in the processing pipeline updates the error management extensions registers.

These error condition checks are provided by the messaging unit. These checks are in addition to the error condition checks provided by the RapidIO port given in Section 15.8.12, “Errors and Error Handling.”

Table 15-126. Outbound Message Chaining Mode Hardware Errors

Transaction	Error	Error Checking Level	Interrupt Generated	Status Bit Set	Message Segment Sent	Logical/Transport Layer Capture Register	Comments
Message Request	An internal error occurred during a read of the descriptor from local memory	0	SRIO error/write-port if OM*n*MR[EIE] set	Transaction error in the outbound message status register (OM*n*SR[TE]). Message Failed in the Mailbox CSR (MCSR[FA]).	No	—	Message controller stops. Note that the descriptor dequeue pointer is not incremented.

15.9.4.2.14 Programming Errors

The following is the list of programming errors that result in undefined or undesired hardware operation. These errors are in addition to the programming errors listed for direct mode.

Table 15-127. Outbound Message Chaining Mode Programming Errors

Error	Interrupt Generated	Status Bit Set	Comments
Enqueued descriptor address is invalid	No	No	Local memory captures the transaction and generate an interrupt.
Address for descriptor enqueue address pointer is invalid	No	No	Local memory captures the transaction and generate an interrupt.
Descriptor queue enqueue and dequeue pointers are not initialized to the same value	No	No	Undefined operation results
Descriptor queue size set to a reserved value	No	No	Undefined operation results
Address of descriptor enqueue pointer set to a value outside of queue	No	No	Undefined operation results
Enqueueing of descriptors causes descriptor queue overflow	Outbound message interrupt enable set (OM*n*MR[QOIE])	Queue overflow (OM*n*SR[QOI])	Message controller stops.
Queue misaligned	No	No	May result in duplicate messages being sent

15.9.4.3 Message Controller Arbitration

Service control defines the order in which each message controller sends messages from its message queues when there are more than one outbound message unit. There are two options:

- Fixed priority—the lowest numbered message unit has the highest priority. Message unit 0 has the highest priority.
- Rotating priority— in this mode the order in which the message units take turns sending messages is round robin (message units 0, 1, 2, 3, 0, and so on). The number of messages a message unit can send is from 1 to 64.

OM*n*MR[SCNTL] configures the message units for these modes and defines operation in direct or chaining mode.

In fixed priority mode, all message segments for message unit 0 must complete before another lower priority message unit (message units 1, 2, 3, and so on) can start. However, in rotating priority mode, another message unit can start processing a message as soon as all message segments have been transmitted. Also, if in fixed priority mode and a message unit other than message unit 0 is processing a message, message unit 0 can start processing a message as soon as all of the message segments have been transmitted.

15.9.5 Inbound Message Controller Operation

The inbound message controller is responsible for receiving messages and placing them in a circular frame queue in local memory. The inbound message controller can receive segments of a message in any order. The address of where to write the message is computed as: Base address + (msgseg × ssize in double-words). Unlike the outbound message controller where the enqueue pointer is controlled by software and the dequeue pointer is controlled by hardware, the inbound message controller controls the enqueue pointer and the software controls the dequeue pointer.

Figure 15-101 depicts a sample structure of the inbound message frames and the frame pointers. In this example, the frame queue has eight entries, three of which are currently valid. The inbound controller controls the enqueue pointer and the software controls the dequeue pointer. After a configured number of messages have been received, an interrupt is generated to the processor. After processing a received message, the local processor can either write the inbound message mode register mailbox increment bit (IM*n*MR[MI]]) causing the dequeue pointer to point to the next message frame in the queue or wait until all the received messages have been processed and write the dequeue pointer.

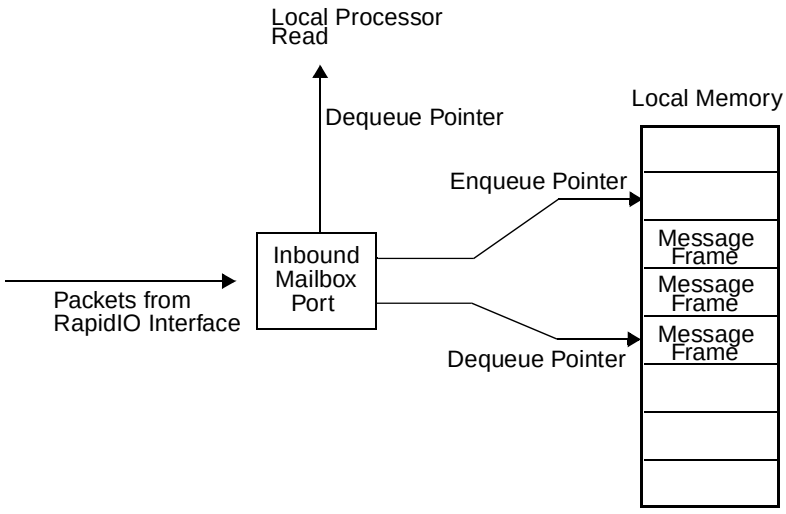


Figure 15-101. Inbound Message Structure

15.9.5.1 Inbound Message Controller Initialization

The sequence of events to initialize the message controller is as follows:

- Initialize the frame queue dequeue pointer address registers (IM*n*FQDPAR, EIM*n*FQDPAR) and the frame queue enqueue pointer address registers (EIM*n*FQEPAR, IM*n*FQEPAR). These need to be initialized to the same value for proper operation.

  These also must be queue size aligned (in other words, the queue to which they point must be aligned on a boundary equal to number of queue entries × frame size). For example, if there are eight entries in the queue and the frame size is 128 bytes, the queue must be 1024-byte aligned.
- Clear the status register (IM*n*SR).
- Set the mailbox enable bit (IM*n*MR[ME]) along with the other control parameters (frame queue size, message-in-queue threshold, frame size, snoop enable, and the various interrupt enables) in the inbound message mode register (IM*n*MR).

15.9.5.2 Inbound Controller Operation

There are many ways in which software can interact with the message controller. One method is as follows:

- The inbound message controller receives a message segment request from the RapidIO port. If the inbound message controller is enabled (IM*n*MR[ME] =1), the inbound message controller has received all the segments for the previous message, and the frame queue is not full then the message segment is accepted.
- The inbound message controller computes the address for each segment of the message (up to 16 segments per message) using the value of the inbound message frame queue enqueue pointer address registers and the segment number.
- The inbound message controller writes each segment to the circular queue in local memory at the computed address.

- Once the entire message is received and the write of all message segments have completed, the enqueue pointer is incremented to point to the next message frame in local memory by the inbound message controller. A message operation completes after all message segments for the message complete. A message segment completes when one of the following occurs:
  - the memory write completes (either successfully or an internal error occurred)
  - a message request time-out occurs (all message segments that have not yet been received are now complete)
- The message segments for a message can immediately be followed by the message segments for another message if certain rules are followed. If a message segment arrives for a new message before all of the previous message memory writes complete for the previous message and the RapidIO priority (the prio field value in the message packet) of the message is equal to or lower than the RapidIO priority of all of the previous messages, the message is processed by the message controller and a memory write is generated to the appropriate frame queue entry. If the RapidIO priority of the message is higher than any of the previous message memory writes that have not completed, the message controller generates a retry. Also, if a message segment arrives before all of the previous message memory writes for the same message complete and the new message segment is higher priority than the previous message segments, then the message controller generates a retry. The IM*n*SR[MB] is cleared by the message controller when all message operations complete.
- An inbound message interrupt is generated to the local processor if the number of messages in the queue is greater than or equal to the configured message-in-queue threshold (IM*n*MR[MIQ_THRESH) and this event is enabled to generate the interrupt (IM*n*MR[MIQIE]).
- Software determines that the message-in-queue event caused the interrupt by detecting that the message-in-queue interrupt bit is set when reading the inbound message status register (IM*n*SR[MIQI]).
- Software processes the frame queue entry pointed to by the frame dequeue pointer address registers (IM*n*FQDPAR, EIM*n*FQDPAR).
- Software increments the dequeue pointer address registers (IM*n*FQDPAR, EIM*n*FQDPAR) by setting the message increment bit (IM*n*MR[MI]). Software determines if there are any more messages to process by reading the queue empty bit (IM*n*SR[QE]). If the queue is not empty, the previous two steps are repeated.
- Optionally, software reads the enqueue pointer address registers (IM*n*FQEPAR, EIM*n*FQEPAR) and processes all the received messages. After the message processing is complete the dequeue pointer address registers (IM*n*FQDPAR, EIM*n*FQDPAR) are written.
- Software clears the message-in-queue interrupt bit (IM*n*SR[MIQI]) by writing a 1 to the IM*n*SR[MIQI] bit.

15.9.5.3 Message Steering

Messages are forwarded to the inbound message controllers as follows:

- Messages directed to mailbox 0 are forwarded to message controller 0
- Messages directed to mailbox 1, 2 or 3 are forwarded to message controller 1

15.9.5.4 Retry Response Conditions

The conditions to generate a logical layer retry (response retry) are:

- Local memory circular queue is full and a message is received.
- The inbound message controller has already received at least one segment of a multi-segment message but has not received all message segments (determined by a different RapidIO source ID, RapidIO destination ID or mailbox)
- Message received with a higher priority than all of the previous messages that are being written to memory but have not completed.

Note that if all inbound messages are the same RapidIO priority, the third condition for generating a retry can not occur.

15.9.5.5 Inbound Message Controller Interrupts

The inbound message controller generates the inbound message interrupt for several different events. Each event can be individually enabled.

- Message-in-queue—an interrupt is generated whenever
  - The circular queue has accumulated the specified number of messages and this interrupt event is enabled (IM*n*MR[MIQIE]). The event that caused this interrupt is indicated by IM*n*SR[MIQI]. The interrupt is held until the dequeue pointer and enqueue pointer indicate that the specified number of messages is not in the frame queue and the IM*n*SR[MIQI] bit has been cleared by writing a 1.
  - The circular queue has one or message in it, the specified number of message has not accumulated, a message has not been dequeued for the maximum interrupt report interval and this interrupt event is enabled (IM*n*SR[MIQIE]). The event that caused this interrupt is indicated by IM*n*SR[MIQI]. The interrupt is held until the IM*n*SR[MIQI] bit has been cleared by writing a 1.
- Queue full—an interrupt is generated whenever the circular queue becomes full and this interrupt event is enabled (IM*n*SR[QFIE]). The event that caused this interrupt is indicated by IM*n*SR[QFI]. The interrupt is held until the queue is not full and the IM*n*SR[QFI] bit has been cleared by writing a 1.

The error/port-write interrupt can be generated for the following reasons.

- Message request time-out—an interrupt is generated after a message request time-out occurs and this interrupt event is enabled (IM*n*MR[EIE])
- Transaction error— an interrupt is generated after a OCN error response is received and this interrupt event is enabled (IM*n*MR[EIE])

15.9.5.5.1 Message Request Time-out Errors

The message request time-out counter starts after the first valid segment of a multisegment message is received and the time-out counter is enabled. When a message request time-out occurs the following occurs.

- The message controller sets the message request time-out status bit (IM*n*SR[MRT])

- All message segments that have not yet been received are considered complete
- If IM*n*MR[EIE] is set, the interrupt SRIO error/write-port is generated
- Once all message segments complete (indicated by IM*n*SR[MB]), the message controller stops

15.9.5.5.2 Transaction Errors

When an internal error occurs during a local memory write by the message controller the following occurs.

- The message controller sets the transaction error bit (IM*n*SR[TE]) and enters the error state
- The message controller returns an error response
- Memory writes that already were generated before the internal error occurred also return an error response
- After the message operation completes (indicated by IM*n*SR[MB]) the message controller stops
- If IM*n*MR[EIE] is set, the interrupt SRIO error/write-port is generated.

15.9.5.5.3 Error Handling

When an error occurs and the SRIO error/write-port interrupt is generated, the following occurs.

- Software determines the cause of the interrupt and processes the error
- Software verifies the message controller has stopped operation by polling IM*n*SR[MB]
- Software clears the error by writing a 1 to the corresponding status bit (IM*n*SR[MRT], and/or IM*n*SR[TE])
- The message unit must be disabled, re-initialized and re-enabled before another message can be received.

When an error occurs and the SRIO error/write-port interrupt is not enabled, the following occurs.

- Software determines that an error has occurred by polling the status bits (IM*n*SR[MRT] and/or IM*n*SR[TE])
- Software verifies the message controller has stopped operation by polling IM*n*SR[MB]
- Software disables the message controller by clearing IM*n*MR[ME]
- Software clears the error by writing a 1 to the corresponding status bit (IM*n*SR[MRT] and/or IM*n*SR[TE])

15.9.5.5.4 Hardware Error Handling

The following lists possible error conditions and what occurs when they are encountered. The error checking level indicates the order in which errors are checked. Multiple errors can be checked at an error checking level. Once an error is detected no additional error checking beyond the current level is performed. Note that messages are processed in a pipeline fashion. The first error detected in the processing pipeline updates the error management extensions registers.

These error condition checks are provided by the messaging unit. These check are in addition to the error condition checks provided by the RapidIO port given in Table 15-128.

Table 15-128. Inbound Message Hardware Errors

Error	Error Checking Level	Interrupt Generated	Status Bit Set	Queue Entry Written in Local Memory	Response Status	Logical/ Transport Layer Capture Register	Comments
A reserved ftype[1]	1	SRIO error/write-port if LTLEECSR[UT] set	Unsupported transaction in the Logical/Transport Layer Error Detect CSR LTLEDCSR[UT]	No	No Response	Updated with the packet[2]	Packet is ignored and discarded.
Reserved tt encoding[1]	1	SRIO error/write-port if LTLEECSR[TSE] set	Transport size error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[TSE].	No	No Response	Updated with the packet[2]	Packet is ignored and discarded.
Large transport size when operating in small transport size or small transport size when operating in large transport size[1]	1	SRIO error/write-port if LTLEECSR[TSE] set	Transport size error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[TSE]	No	No Response	Updated with the packet[2]	Packet is ignored and discarded. An error or illegal transaction target error response is not generated.
Illegal Destination ID[1]	1	SRIO error/write-port if LTLEECSR[ITTE] set	Illegal transaction target in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITTE]	No	Error	Updated with the packet[2]	Packet is ignored and discarded.
An incorrect message packet size (payload is not than the specified ssize except for the last segment. The last segment's payload can be less than or equal to the segment size but not 0. Note that payload sizes are dword multiples.)[1]	1	SRIO error/write-port if LTLEECSR[MFE] set	Message Format error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[MFE]	No	Error	Updated with the packet[2]	Packet is ignored and discarded.
Reserved ssize field[1]	1	SRIO error/write-port if LTLEECSR[MFE] set	Message Format error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[MFE]	No	Error	Updated with the packet[2]	Packet is ignored and discarded.

Table 15-128. Inbound Message Hardware Errors (continued)

Error	Error Checking Level	Interrupt Generated	Status Bit Set	Queue Entry Written in Local Memory	Response Status	Logical/ Transport Layer Capture Register	Comments
Message received and no mailboxes are supported as indicated by DOCAR[M][1]	1	SRIO error/write-port if LTLEECSR[UT] set	Unsupported transaction in the Logical/Transport Layer Error Detect CSR LTLEDCSR[UT]	No	Error	Updated with the packet[2]	Packet is ignored and discarded.
Message packet size larger than the configured frame size and message controller is enabled	2	SRIO error/write-port if LTLEECSR[MFE] set	Message Format error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[MFE]	No	Error	Updated with the packet[2]	Packet is ignored and discarded.
An inbound message packet with a RapidIO priority of 3	2	SRIO error/write-port if LTLEECSR[ITD] set	Illegal transaction decode in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITD]	No	No Response	Updated with the packet[2]	Packet is ignored and discarded.
Not an error - The RapidIO priority is not consistent for all message segments of a message	2	—	—	Yes	Done or Retry Response	—	Retry response occurs if the higher priority message segment is received while the memory write for a corresponding lower priority message segment is outstanding
message segment number is larger than the number of message segments in the message	2	SRIO error/write-port if LTLEECSR[MFE] set	Message format error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[MFE]	No	Error	Updated with the packet[2]	Packet is ignored and discarded.

Table 15-128. Inbound Message Hardware Errors (continued)

Error	Error Checking Level	Interrupt Generated	Status Bit Set	Queue Entry Written in Local Memory	Response Status	Logical/ Transport Layer Capture Register	Comments
duplicate message segment is received (Note that all segments of a multi segment message must be received before the next message begins).	2	SRIO error/write-port if LTLEECSR[MFE] set	Message format error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[MFE]	No	Error	Updated with the packet[2]	Packet is ignored and discarded.
msglen (number of segments in a message) is not consistent in all segments of a multi segment message	2	SRIO error/write-port if LTLEECSR[MFE] set	Message format error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[MFE]	No	Error	Updated with the packet[2]	Packet is ignored and discarded.
ssize (segment size) is not consistent in all segments of a multi segment message	2	SRIO error/write-port if LTLEECSR[MFE] set	Message format error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[MFE]	No	Error	Updated with the packet[2]	Packet is ignored and discarded.
Message received for an unsupported mailbox but at least one mailbox is supported	2	SRIO error/write-port if LTLEECSR[UT] set	Unsupported transaction in the Logical/Transport Layer Error Detect CSR LTLEDCSR[UT]	No	Error	—	Packet is ignored and discarded. This error only applies if a mailbox is not supported. This error is not currently supported since all mailboxes are supported.
Message Controller Disabled and Message Received	2	No	None	No	Error	—	Packet is ignored and discarded.
Message Controller Enabled but in the Error State and Message Received	2	No	None	No	Error	—	Packet is ignored and discarded.

Table 15-128. Inbound Message Hardware Errors (continued)

Error	Error Checking Level	Interrupt Generated	Status Bit Set	Queue Entry Written in Local Memory	Response Status	Logical/ Transport Layer Capture Register	Comments
Internal error occurred during the write of the frame queue entry to memory	3	SRIO error/write-port if IM*n*MR[EIE] set	Transaction error in the Message status register (IM*n*SR[TE]). Message Failed in the Message CSR (MCSR[FA])	No	Error	—	Message controller stops after the current message operation completes. The enqueue pointer is not incremented.
An internal error occurred for an earlier posted frame queue entry memory write and a subsequent frame queue entry memory write is posted before the internal error is detected. An internal error may or may not occur during the subsequent frame queue entry memory write. The frame queue could be for the same message or a different message.	4	No	None	Yes	Error	—	—

Table 15-128. Inbound Message Hardware Errors (continued)

Error	Error Checking Level	Interrupt Generated	Status Bit Set	Queue Entry Written in Local Memory	Response Status	Logical/ Transport Layer Capture Register	Comments
message request to request time-out for multi segment messages	unrelated	SRIO error/write-port if LTLEECSR[MRT] set. SRIO error/write-port if OM*n*MR[EIE].	Message request time-out in the Logical/Transport Layer Error Detect CSR LTLEDCSR[MRT]. IM*n*SR[MRT] bit set.	No	No	Updated with the previous message request packet except that the message segment field (bits 4 to 7 of LTLCCCSR[MI]) is updated with the lowest message segment number that has not yet been received.[2]	All message segments received before the time-out update memory. The enqueue pointer is not incremented. The message operation completes.

Note:

1. These error types are actually detected in the RapidIO port, not in the message controller.
2. If operating in small transport size configuration using the packet LTLACCSR[XA] gets the extended address (packet bits 78–79), LTLACCSR[A] gets the address (packet bits 48–76), LTLDIDCCSR[MDID] gets 0, LTLDIDCCSR[DID] gets the least significant byte of the destination ID (packet bits 16–23), LTLDIDCCSR[MSID] gets 0, LTLDIDCCSR[SID] gets the least significant byte of the source ID (packet bits 24–31), LTLCCCSR[FT] gets the ftype (packet bits 12–15), LTLCCCSR[TT] gets the ttype (packet bits 32–35), LTLCCCSR[MI] gets the msg info (packet bits 40–47). If operating in large transport size configuration using the packet LTLACCSR[XA] gets the extended address (packet bits 94–95), LTLACCSR[A] gets the address (packet bits 64–92), LTLDIDCCSR[MDID] gets the most significant byte of the destination ID (packet bits 16–23), LTLDIDCCSR[DID] gets the least significant byte of the destination ID (packet bits 24–31), LTLDIDCCSR[MSID] gets the most significant byte of the source ID (packet bits 32–39), LTLDIDCCSR[SID] gets the least significant byte of the source ID (packet bits 40–47), LTLCCCSR[FT] gets the ftype (packet bits 12–15), LTLCCCSR[TT] gets the ttype (packet bits 48–51), LTLCCCSR[MI] gets the msg info (packet bits 56–63).

15.9.5.5.5 Programming Errors

The following is a partial list of programming errors that result in undefined or undesired hardware operation.

Table 15-129. Inbound Message Programming Errors

Error	Interrupt Generated	Status Bit Set	Comments
Reserved value of the message in queue threshold (IM*n*MR[MIQ_THRESH]) or reserved value of the circular frame queue size (IM*n*MR[CIRQ_SIZE])	No	No	Undefined operation results
The message in-queue threshold is equal to the frame queue size	No	No	Message in queue interrupt occurs when queue is full
The message in-queue threshold is greater than the frame queue size	No	No	Message in queue interrupt never occurs

Table 15-129. Inbound Message Programming Errors (continued)

Error	Interrupt Generated	Status Bit Set	Comments
Frame queue entry written to non-existent memory	No	No	Memory controller causes the interrupt and updates capture registers. IM*n*SR[TE] is set due to the internal error.
Message enqueue and dequeue pointers are not initialized to the same value	No	No	Undefined operation results
The dequeue frame pointer register is set incorrectly.	No	No	Undefined operation results
Queue misaligned	No	No	May cause unpredictable behavior

15.9.5.5.6 Disabling and Enabling the Inbound Message Controller

When the message controller is disabled by clearing IM*n*MR[ME] the following occurs.

- Queue full clears (IM*n*SR[QF])
- Message-in-queue clears (IM*n*SR[MIQ])
- Queue empty is set (IM*n*SR[QE])

Once the message controller is disabled, an error response is generated for all new message packets. If the message controller is disabled before all of the message segments for a multisegment message are received, a message request time-out must occur and all pending frame queue writes must complete before message busy clears (IM*n*SR[MB]).

Before the message controller is re-enabled the message busy bit must be clear (IM*n*SR[MB) and the (IM*n*MR[ME]) the frame queue dequeue pointer address registers (IM*n*FQDPAR, EIM*n*FQDPAR) and the frame queue enqueue pointer address registers (IM*n*FQEPAR, EIM*n*FQEPAR) must be initialized to the same value for proper message controller operation.

15.9.6 RapidIO Message Passing Logical Specification Registers

The mailbox command and status register (MCSR) provides the status for the four inbound and the four outbound controllers. These read-only status bits indicate the state of each of the message controllers.

- Available (MCSR[A])—Indicates that the inbound message controller is enabled (IM*n*MR[ME]), the inbound message controller is not in the internal error state (IM*n*SR[TE] = 0) and the inbound message controller did not detect a message request time-out (IM*n*SR[MRT] = 0)
- Full (MCSR[FU])—This bit reflects the inbound message controller queue full status
- Empty (MCSR[EM])—This bit reflects the state of the outbound message controller message empty status
- Busy (MCSR[B])—This bit reflects the state of the inbound message controller busy bit IM*n*SR[MB]
- Failed (MCSR[FA])—This bit is set if any of the following bits are set:
  - The inbound message controller transaction error status bit IM*n*SR[TE]
  - The inbound message controller message request time-out status bit IM*n*SR[MRT]
  - The outbound message controller transaction error status bit OM*n*SR[TE] is set
  - The outbound message controller packet response time-out bit OM*n*SR[PRT] is set
  - The outbound message controller message error response received status bit OM*n*SR[MER] is set
  - The outbound message controller retry error threshold exceeded status bit OM*n*SR[RETE] is set
- Error (MCSR[ERR])—This bit is always 0

15.10 RapidIO Doorbell and Port-Write Unit

This section describes the operation of the doorbell and port-write controllers, which are part of the RapidIO message unit. The doorbell and port-write controllers are compliant with the message passing logical specification contained in the *RapidIO Interconnect Specification, Revision 1.2*. The doorbell controller generates and receives doorbells. The port-write controller receives but does not generate port-writes.

The doorbell and port-write controllers are controlled through a set of run-time registers.

15.10.1 Features

- Support for one outbound doorbell controller
- Support for one inbound doorbell controller
  - The doorbell controller can sustain back-to-back inbound doorbells
- Support for one inbound port-write controller with the following features
  - Up to 64 bytes of payload
  - Only one inbound port-write queue entry

15.10.2 Doorbell Controller

RapidIO supports a doorbell type that contains no data payload. The RapidIO architecture references outbound and inbound doorbell controllers. This doorbell unit supports both inbound and outbound doorbells. Inbound doorbells are handled by the doorbell controller similar to how the message controller handles inbound data messages. The doorbell controller receives the doorbells and places it in a circular queue located in local memory. Additional doorbells can be received and forwarded to memory before the previous doorbell memory write completes as long as the RapidIO priority is the same or lower than the previous doorbell. The doorbell is retried if the RapidIO priority is higher than the previous doorbell. Outbound doorbells are generated through a memory-mapped write port rather than a queue. An outbound doorbell must complete before another outbound doorbell can be generated.

Figure 15-102 depicts an example of the structure of the inbound doorbell queue and pointers. The doorbell queue has eight entries, three of which are currently valid. The doorbell controller enqueues doorbells and the local processor dequeues doorbells.

The doorbell entry size is fixed at 64 bits because doorbell packets only pass a small amount of information, making the enqueue and dequeue pointers double-word addresses.

The outbound doorbell controller has a dedicated interrupt that can be used to notify software that a doorbell has been sent. Each inbound doorbell controller has a dedicated interrupt that can be used for notification of incoming doorbells.

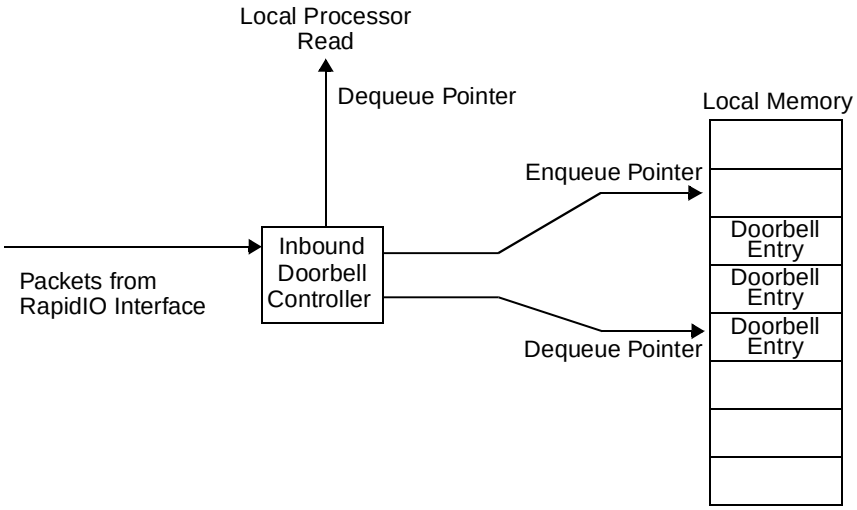


Figure 15-102. Inbound Doorbell Queue and Pointer Structure

15.10.2.1 Outbound Doorbell Controller

The outbound doorbell controller is used to generate doorbells. Software is responsible for initializing all the parameters in all the necessary registers to start the doorbell transmission. The doorbell transfer is started when the doorbell start bit, ODMR[DUS], in the outbound doorbell mode register transitions from a 0 to 1 and the doorbell controller is not already busy. Software is expected to program all the appropriate registers before setting ODMR[DUS].

There are many ways in which software can interact with the doorbell controller. One method to generate a doorbell is as follows:

- Poll the status register doorbell unit busy bit, ODSR[DUB], to make sure the outbox is not busy with a previously initiated doorbell.
- Clear the status bits (ODSR[MER], ODSR[RETE], ODSR[PRT], and ODSR[EODI])
- Initialize the destination port (ODDPR), destination attributes (ODDATR) and retry error threshold (ODRETCR) registers
- Clear, then set the doorbell start bit, ODMR[DUS], to start the doorbell transfer.
- ODSR[DUB] is set when ODMR[DUS] transitions from a 0 to a 1 to indicate the doorbell transfer is in progress.
- The outbound doorbell controller sends the doorbell
- The ODSR[DUB] is cleared by the outbound doorbell controller after the doorbell operation completes. A doorbell completes when one of the following occurs:
  - done response received
  - error response received
  - a packet response time-out occurred
  - the retry limit was exceeded
- The outbound doorbell interrupt is generated if the end of doorbell outbound doorbell interrupt event is enabled (ODDATR[EODIE]).

15.10.2.1.1 Interrupts

The outbound doorbell controller interrupt can be generated for one reason.

- End-Of-Doorbell. An interrupt is generated after the completion of a doorbell (done, error, packet response time-out or retry limit exceeded) if this interrupt event is enabled (ODDATR[EODIE] is a 1). The event that caused this interrupt is indicated by ODSR[EODI]. The interrupt is held until the ODSR[EODI] bit has been cleared by writing a 1.

The error/port-write interrupt can be generated for the following reasons.

- RapidIO error response. An interrupt is generated after a RapidIO error response is received and this interrupt event is enabled (ODMR[EIE])
- Packet response time-out. An interrupt is generated after a packet response time-out occurs and this interrupt event is enabled (ODMR[EIE])
- Retry error threshold exceeded. An interrupt is generated after a retry threshold exceeded error occurs and this interrupt event is enabled (ODMR[EIE])

15.10.2.1.2 Error Response Errors

When a RapidIO error response is received by the doorbell controller the following occurs.

- The doorbell controller sets the message error response status bit (ODSR[MER])
- If ODMR[EIE] is set, the interrupt SRIO error/write-port is generated.
- After the doorbell operation completes (indicated by ODSR[DUB]) the doorbell controller stops

15.10.2.1.3 Packet Response Time-Out Errors

When a packet response time-out occurs for a doorbell the following occurs.

- The doorbell controller sets the packet response time-out status bit (ODSR[PRT])
- If ODMR[EIE] is set, the interrupt SRIO error/write-port is generated.
- After the doorbell operation completes (indicated by ODSR[DUB]) the doorbell controller stops

15.10.2.1.4 Retry Error Threshold Exceeded Errors

When a retry error threshold exceeded error occurs for a doorbell the following occurs.

- The doorbell controller sets the retry threshold exceed status bit (ODSR[RETE])
- If ODMR[EIE] is set, the interrupt SRIO error/write-port is generated.
- After the doorbell operation completes (indicated by ODSR[DUB]) the doorbell controller stops

15.10.2.1.5 Error Handling

When an error occurs and the SRIO error/write-port interrupt is generated, the following occurs.

- Software determines the cause of the interrupt and processes the error
- Software verifies the doorbell controller has stopped operation by polling ODSR[DUB]
- Software disables the doorbell controller by clearing ODMR[DUS]
- Software clears the error by writing a 1 to the corresponding status bit (ODSR[PRT], ODSR[PRT], and/or ODSR[RETE])

When an error occurs and the SRIO error/write-port interrupt is not enabled, the following occurs.

- Software determines that an error has occurred by polling the status bits (ODSR[MER], ODSR[PRT], and/or ODSR[RETE])
- Software verifies the doorbell controller has stopped operation by polling ODSR[DUB]
- Software disables the doorbell controller by clearing ODMR[DUS]
- Software clears the error by writing a 1 to the corresponding status bit (ODSR[MER], ODSR[PRT], and/or ODSR[RETE])

15.10.2.1.6 Disabling and Enabling the Doorbell Controller

Once the doorbell controller is started, it cannot be stopped.

15.10.2.1.7 Hardware Error Handling

The following is a list possible error conditions and what occurs when they are encountered. The error checking level indicates the order in which errors are checked. Multiple errors can be checked at an error checking level. Once an error is detected, no additional error checking beyond the current level is performed. Note that outbound doorbell responses are processed in a pipeline fashion. The first error detected in the processing pipeline updates the error management extensions registers.

These error condition checks are provided by the messaging unit. These check are in addition to the error condition checks provided by the RapidIO port given in Section 15.8.12, “Errors and Error Handling.”

Table 15-130. Outbound Doorbell Hardware Errors

Transaction	Error	Error Checking Level	Interrupt Generated	Status Bit Set	Doorbell Sent	Logical/Transport Layer Capture Register	Comments
Undefined Packet	Reserved ftype encoding[1]	1	SRIO error/write-port if LTLEECSR[UT] set	Unsupported transaction in the Logical/Transport Layer Error Detect CSR LTLEDCSR[UT]	Yes	Updated with the packet[2]	Packet is ignored and discarded.
Doorbell Response	Reserved tt encoding[1]	1	SRIO error/write-port if LTLEECSR[TSE] set	Transport size error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[TSE.	Yes	Updated with the packet[2]	Packet is ignored and discarded.
Doorbell Response	Large transport size when operating in small transport size or small transport size when operating in large transport size[1]	1	SRIO error/write-port if LTLEECSR[TSE] set	Transport size error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[TSE.	Yes	Updated with the packet[2]	Packet is ignored and discarded. An error or illegal transaction target error response is not generated.
Doorbell Response	Illegal Destination ID[1]	1	SRIO error/write-port if LTLEECSR[ITTE] set	Illegal transaction target in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITTE]	Yes	Updated with the packet[2]	Packet is ignored and discarded.
Doorbell Response	doorbell not outstanding[1]	1	SRIO error/write-port if LTLEECSR[UR] set	Unsolicited response in the Logical/Transport Layer Error Detect CSR LTLEDCSR[UR]	Yes	Updated with the packet[2]	Packet is ignored and discarded.
Doorbell Response	ttype (transaction field) is not doorbell response[1]	1	SRIO error/write-port if LTLEECSR[ITD] set	Illegal transaction decode in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITD]	Yes	Updated with the packet[2]	Packet is ignored and discarded.
Doorbell Response	RapidIO priority is less than or equal to outbound request[1]	2	SRIO error/write-port if LTLEECSR[ITD] set	Illegal transaction decode in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITD]	Yes	Updated with the packet[2]	Packet is ignored and discarded.

Table 15-130. Outbound Doorbell Hardware Errors (continued)

Transaction	Error	Error Checking Level	Interrupt Generated	Status Bit Set	Doorbell Sent	Logical/Transport Layer Capture Register	Comments
Doorbell Response	Incorrect Source ID[1]	2	SRIO error/write-port if LTLEECSR[ITD] set	Illegal transaction decode in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITD]	Yes	Updated with the packet[2]	Packet is ignored and discarded.
Doorbell Response	reserved response status [1]	2	SRIO error/write-port if LTLEECSR[ITD] set	Illegal transaction decode in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITD]	Yes	Updated with the packet[2]	Packet is ignored and discarded.
Doorbell Response	doorbell response packet size is incorrect[1]	2	SRIO error/write-port if LTLEECSR[MFE] set	Message Format error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[MFE]	Yes	Updated with the packet[2]	Packet is ignored and discarded.
Doorbell Response	error response	3	SRIO error/write-port if LTLEECSR[MER] set. SRIO error/write-port if OM*n*MR[EIE].	Message error response in the Logical/Transport Layer Error Detect CSR LTLEDCSR[MER]. ODSR[MER] bit set	Yes	Updated with the corresponding doorbell request packet[2]	doorbell transfer complete

Table 15-130. Outbound Doorbell Hardware Errors (continued)

Transaction	Error	Error Checking Level	Interrupt Generated	Status Bit Set	Doorbell Sent	Logical/Transport Layer Capture Register	Comments
Doorbell Response	number of retries exceeds limit	3	SRIO error/write-port if LTLEECSR[RETE] set. SRIO error/write-port if OM*n*MR[EIE].	Retry limit exceeded in the Logical/Transport Layer Error Detect CSR LTLEDCSR[RETE] ODSR[RETE] bit set.	Yes	Updated with the corresponding doorbell request packet[2]	doorbell transfer complete
Doorbell Response	packet response time-out[1]	unrelated	SRIO error/write-port if LTLEECSR[PRT] set. SRIO error/write-port if OM*n*MR[EIE].	Packet response time-out in the logical/transport layer error detect CSR LTLEDCSR[PRT] in RapidIO endpoint. ODSR[PRT] bit set.	Yes	Updated with the doorbell request packet in the RapidIO endpoint[2]	doorbell transfer complete. Note that the RapidIO endpoint sends special priority 3 pkt indicating doorbell time-out.

1) These error types are actually detected in the RapidIO port, not in the doorbell controller

2) If operating in small transport size configuration using the packet LTLACCSR[XA] gets the extended address (packet bits 78–79), LTLACCSR[A] gets the address (packet bits 48–76), LTLDIDCCSR[MDID] gets 0, LTLDIDCCSR[DID] gets the least significant byte of the destination ID (packet bits 16–23), LTLDIDCCSR[MSID] gets 0, LTLDIDCCSR[SID] gets the least significant byte of the source ID (packet bits 24–31), LTLCCCSR[FT] gets the ftype (packet bits 12–15), LTLCCCSR[TT] gets the ttype (packet bits 32–35), LTLCCCSR[MI] gets 0. If operating in large transport size configuration using the packet LTLACCSR[XA] gets the extended address (packet bits 94–95), LTLACCSR[A] gets the address (packet bits 64–92), LTLDIDCCSR[MDID] gets the most significant byte of the destination ID (packet bits 16–23), LTLDIDCCSR[DID] gets the least significant byte of the destination ID (packet bits 24–31), LTLDIDCCSR[MSID] gets the most significant byte of the source ID (packet bits 32–39), LTLDIDCCSR[SID] gets the least significant byte of the source ID (packet bits 40–47), LTLCCCSR[FT] gets the ftype (packet bits 12–15), LTLCCCSR[TT] gets the ttype (packet bits 48–51), LTLCCCSR[MI] gets 0.

15.10.2.1.8 Programming Errors

The following is the list of programming errors that result in undefined or undesired hardware operation.

Table 15-131. Outbound Doorbell Programming Errors

Error	Interrupt Generated	Status Bit Set	Comments
Transaction flow level set to reserved (2'b11)	No	None	Unit hangs. If the transaction flow level is then changed to a value other than reserved, the doorbell operation starts using this new transaction flow level.

Table 15-131. Outbound Doorbell Programming Errors (continued)

Error	Interrupt Generated	Status Bit Set	Comments
Target interface set to an invalid RapidIO port	No	None	Undefined operation results
Register values changed during operation	No	No	Undefined operation results

15.10.2.2 Inbound Doorbell Controller

The inbound doorbell controller is responsible for receiving doorbells and placing them in a circular doorbell queue in local memory. The inbound controller controls the enqueue pointer and the software controls the dequeue pointer. After a configured number of doorbells have been received, an interrupt is generated to the processor. After processing a received doorbell, the local processor can either write the doorbell mode register increment bit (IDMR[DI]]) causing the dequeue pointer to point to the next doorbell in the queue or wait until all the received doorbells have been processed and write the dequeue pointer.

15.10.2.2.1 Inbound Doorbell Controller Initialization

There are many ways in which software can interact with the doorbell controller. One method to initialize the doorbell controller is as follows:

- Initialize the doorbell queue dequeue pointer address registers (DQDPAR, EDQDPAR) and the doorbell queue enqueue pointer address registers (DQEPAR, EDQEPAR). These need to be initialized to the same value for proper operation. These also must be queue size aligned.
- Clear the status register (IDSR).
- Set the doorbell enable bit (IDMR[DE]) along with the other control parameters (doorbell queue size, doorbell-in-queue threshold, snoop enable, and the various interrupt enables) in the doorbell mode register (IDMR).

15.10.2.2.2 Inbound Doorbell Controller Operation

There are many ways in which software can interact with the doorbell controller. One method is as follows:

- The doorbell controller receives a doorbell. If the inbound doorbell controller is enabled (IDMR[DE] = 1) and the doorbell queue is not full, then the doorbell is accepted.
- The 16-bit information field along with the RapidIO source ID and RapidIO destination ID are stored in local memory by the doorbell controller using the value of the doorbell queue enqueue pointer address registers (DQEPAR, EDQEPAR).
- Once the memory write completes the enqueue pointer is incremented to point to the next doorbell queue entry in local memory.
- If another doorbell arrives before all of the previous doorbell memory writes complete and the RapidIO priority of the doorbell is equal to or lower than the RapidIO priority of all of the previous doorbells, the doorbell is processed by the doorbell controller and a memory write is generated to the appropriate doorbell queue entry. If the RapidIO priority of the doorbell is higher than any of the previous doorbell memory writes that have not completed, the doorbell controller generates a retry.

- An inbound doorbell interrupt is generated to the local processor because the number of doorbells in the queue is greater than or equal to the configured doorbell-in-queue threshold (IDMR[DIQ_THRESH) and this event is enabled to generate the interrupt (IDMR[DIQIE]).
- Software determines that the doorbell-in-queue event caused the interrupt by detecting that the doorbell-in-queue interrupt bit is set when reading the doorbell status register (IDSR[DIQI]).
- Software processes the doorbell queue entry pointed to by the doorbell dequeue pointer address registers (DQDPAR, EDQDPAR).
- Software increments the dequeue pointer address registers (DQDPAR, EDQDPAR) by setting the doorbell increment bit (IDMR[DI]).
- Software determines if there are any more doorbells to process by reading the queue empty bit (IDSR[QE]). If the queue is not empty, the previous two steps are repeated.
- Software clears the doorbell-in-queue interrupt bit (IDSR[DIQI]) by writing a 1 to the IDSR[DIQI] bit.

15.10.2.2.3 Inbound Doorbell Queue Entry Format

This section defines the format of the doorbell information written to memory by the doorbell controller. Each doorbell entry in the queue has 2 offsets of 32 bits, one for target information and one for source information. The target information is stored because a RapidIO port can be configured to accept packets from any destination. In addition, when there are multiple RapidIO ports on one device each port may be configured with a different destination ID.

Table 15-132. Inbound Doorbell Target Info Definition

Bits	Name	Description
0–15	—	Reserved
16–23	ETID	Extended target ID when in large transport mode, reserved when in small transport mode
24–31	TID	Target ID field from the received doorbell packet

Table 15-133. Source Info Definition

Bits	Name	Description
0–7	ESID	Extended Source ID when in large transport mode, reserved when in small transport mode
8–15	SID	Source ID field from the received doorbell packet
16–23	INFO MSB	Most significant byte of the info field from the received doorbell packet
24–31	INFO LSB	Least significant byte of the info field from the received doorbell packet

Figure 15-103 depicts the doorbell queue entry fields and their related offsets.

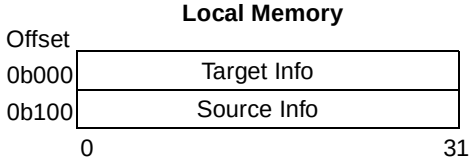


Figure 15-103. Doorbell Entry Format

15.10.2.2.4 Retry Response Conditions

There are two conditions in which a doorbell is retried at the logical layer (response retry).

- Doorbell received and there are no entries available in the doorbell queue.
- Doorbell received with a higher priority than all of the previous doorbells that are being written to memory but have not completed.

If all inbound doorbells are the same RapidIO priority, the second condition for generating a retry can not occur.

15.10.2.2.5 Doorbell Controller Interrupts

There is one doorbell controller interrupt per inbound doorbell controller. The following events can generate this interrupt.

- Doorbell-in-queue—this event generates an interrupt under the following conditions:
  - the circular queue has accumulated the configured number of doorbells specified by the doorbell-in-queue threshold (IDMR[DIQ_THRESH) and this interrupt event is enabled (IDMR[DIQIE]). The event that caused this interrupt is indicated by IDSR[DIQI]. The interrupt is held until the dequeue pointer and enqueue pointer indicate that the specified number of doorbells is not in the doorbell queue and the IDSR[DIQI] bit has been cleared by writing a 1.
  - the circular queue has one or more doorbells in it, the specified number of doorbells has not accumulated, a doorbell has not been dequeued for the maximum interrupt report interval and this interrupt event is enabled (IDMR[DIQIE]). The event that caused this interrupt is indicated by IDSR[DIQI]. The interrupt is held until either IDMR[DI] has been set or DQDPAR[DQDPA] has been written followed by clearing IDSR[DIQI].
- Queue full—an interrupt is generated each time the circular queue becomes full and this interrupt event is enabled (IDSR[QFIE]). The event that caused this interrupt is indicated by IDSR[QFI]. The interrupt is held until the queue is not full and the IDSR[QFI] bit has been cleared by writing a 1.

The error/port-write interrupt can be generated for the following reasons.

- Transaction error—an interrupt is generated after a OCN error response is received and this interrupt event is enabled (IDMR[EIE])

15.10.2.2.6 Transaction Errors

When an internal error occurs during a local memory write by the doorbell controller the following occurs.

- The doorbell controller sets the transaction error bit (IDSR[TE]) and enters the error state
- The doorbell controller returns an error response
- If IDMR[EIE] is set, the interrupt SRIO error/write-port is generated.

15.10.2.2.7 Error Handling

When an error occurs and the SRIO error/write-port interrupt is generated, the following occurs.

- Software determines the cause of the interrupt and processes the error
- Software verifies the doorbell controller has stopped operation by polling IDSR[DB]
- Software clears the error by writing a 1 to the corresponding status bit (IDSR[TE])
- The doorbell unit must be disabled, re-initialized and re-enabled before another doorbell can be received.

15.10.2.2.8 Hardware Error Handling

The following list possible error conditions and what occurs when they are encountered. The error checking level indicates the order in which errors are checked. Multiple errors can be checked at an error checking level. Once an error is detected no additional error checking beyond the current level is performed. Note that doorbells are processed in a pipeline fashion. The first error detected in the processing pipeline updates the error management extensions registers.

These error condition checks are provided by the messaging unit. These check are in addition to the error condition checks provided by the RapidIO port given in Section 15.8.12, “Errors and Error Handling.”

Table 15-134. Inbound Doorbell Hardware Errors

Error	Error Checking Level	Interrupt Generated	Status Bit Set	Queue Entry Written in Local Memory	Response Status	Logical/Transport Layer Capture Register	Comments
Reserved ftype[1]	1	SRIO error/write-port if LTLEECSR[UT] set	Unsupported transaction in the Logical/Transport Layer Error Detect CSR LTLEDCSR[UT]	No	No Response	Updated with the packet[2]	Packet is ignored and discarded.
Reserved tt encoding[1]	1	SRIO error/write-port if LTLEECSR[TSE] set	Transport size error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[TSE].	No	No Response	Updated with the packet[2]	Packet is ignored and discarded.
Large transport size when operating in small transport size or small transport size when operating in large transport size[1]	1	SRIO error/write-port if LTLEECSR[TSE] set	Transport size error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[TSE].	No	No Response	Updated with the packet[2]	Packet is ignored and discarded. An error or illegal transaction target error response is not generated.

Table 15-134. Inbound Doorbell Hardware Errors (continued)

Error	Error Checking Level	Interrupt Generated	Status Bit Set	Queue Entry Written in Local Memory	Response Status	Logical/Transport Layer Capture Register	Comments
Illegal Destination ID[1]	1	SRIO error/write-port if LTLEECSR[ITTE] set	Illegal transaction target in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITTE]	No	Error	Updated with the packet[2]	Packet is ignored and discarded.
Inbound doorbell received and inbound doorbells are not supported as indicated by DOCAR[D][1]	1	SRIO error/write-port if LTLEECSR[UT] set	Unsupported transaction in the Logical/Transport Layer Error Detect CSR LTLEDCSR[UT]	No	Error	Updated with the packet[2]	Packet is ignored and discarded.
An inbound doorbell packet with a RapidIO priority of 3	2	SRIO error/write-port if LTLEECSR[ITD] set	Illegal transaction decode in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITD]	No	No Response	Updated with the packet[2]	Packet is ignored and discarded.
An incorrect doorbell packet size (not one datum in small transport mode or not two datums in large transport mode)	2	SRIO error/write-port if LTLEECSR[MFE] set	Message Format error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[MFE]	No	Error	Updated with the packet[2]	Packet is ignored and discarded.
Doorbell Controller Disabled and Doorbell Received	3	No	None	No	Error	—	Packet is ignored and discarded.
Doorbell Controller Enabled but in the Error State and Doorbell Received	3	No	None	No	Error	—	Packet is ignored and discarded.

Table 15-134. Inbound Doorbell Hardware Errors (continued)

Error	Error Checking Level	Interrupt Generated	Status Bit Set	Queue Entry Written in Local Memory	Response Status	Logical/Transport Layer Capture Register	Comments
Internal error occurred during the write of the doorbell queue entry to memory	4	SRIO error/write-port if OMMR[EIE] set	Transaction error in the Doorbell status register (ISSR[TE]). Doorbell Failed in the Port-write and Doorbell CSR (PWDCSR[FA])	No	Error	—	Doorbell controller stops after the current doorbell operation completes. The enqueue pointer is not incremented.
An internal error occurred for an earlier posted write of a doorbell queue entry to memory and a subsequent write of a doorbell queue entry to memory is posted before the internal error is detected. An internal error may or may not occur during the subsequent write of a doorbell queue entry to memory.	5	No	None	Yes	Error	—	—

1. These error types are actually detected in the RapidIO port, not in the doorbell controller
2. If operating in small transport size configuration using the packet LTLACCSR[XA] gets the extended address (packet bits 78–79), LTLACCSR[A] gets the address (packet bits 48–76), LTLDIDCCSR[MDID] gets 0, LTLDIDCCSR[DID] gets the least significant byte of the destination ID (packet bits 16–23), LTLDIDCCSR[MSID] gets 0, LTLDIDCCSR[SID] gets the least significant byte of the source ID (packet bits 24–31), LTLCCCSR[FT] gets the ftype (packet bits 12–15), LTLCCCSR[TT] gets the ttype (packet bits 32–35), LTLCCCSR[MI] gets 0. If operating in large transport size configuration using the packet LTLACCSR[XA] gets the extended address (packet bits 94–95), LTLACCSR[A] gets the address (packet bits 64–92), LTLDIDCCSR[MDID] gets the most significant byte of the destination ID (packet bits 16–23), LTLDIDCCSR[DID] gets the least significant byte of the destination ID (packet bits 24–31), LTLDIDCCSR[MSID] gets the most significant byte of the source ID (packet bits 32– 39), LTLDIDCCSR[SID] gets the least significant byte of the source ID (packet bits 40–47), LTLCCCSR[FT] gets the ftype (packet bits 12–15), LTLCCCSR[TT] gets the ttype (packet bits 48–51), LTLCCCSR[MI] gets 0.

15.10.2.2.9 Programming Errors

The following is the list of programming errors that result in undefined or undesired hardware operation.

Table 15-135. Inbound Doorbell Programming Errors

Error	Interrupt Generated	Status Bit Set	Comments
Reserved value of the doorbell in queue threshold (ID*n*MR[DIQ_THTRES]) or reserved value of the circular doorbell queue size (ID*n*MR[CIRQ_SIZE])	No	No	Undefined operation results
The doorbell in-queue threshold is equal to the doorbell queue size	No	No	Doorbell in queue interrupt occurs when queue is full
The doorbell in-queue threshold is greater than the doorbell queue size	No	No	Doorbell in queue interrupt never occurs
Doorbell queue entry written to non-existent memory	No	No	Memory controller causes the interrupt and update capture registers
Doorbell enqueue and dequeue pointers are not initialized to the same value	No	No	Undefined operation results
The dequeue pointer register is set incorrectly.	No	No	Undefined operation results

15.10.2.2.10 Disabling and Enabling the Doorbell Controller

When the doorbell controller is disabled by clearing IDMR[DE] the following occurs.

- Queue full clears (IDSR[QF]).
- Doorbell-in-queue clears (IDSR[DIQ]).
- Queue empty is set (IDSR[QE]).
- Doorbell busy clears (IDSR[DUB]) after all pending doorbell queue entry writes to local memory complete.

Before the doorbell controller is re-enabled the doorbell busy bit must be clear (IDSR[DB) and the doorbell queue dequeue pointer address registers (DQDPAR, EDQDPAR) and the doorbell queue enqueue pointer address registers (DQEPAR, EDQEPAR) must be initialized to the same value for proper doorbell controller operation.

15.10.2.3 RapidIO Message Passing Logical Specification Registers

The port-write and doorbell command and status register (PWDCSR) includes several doorbell controller status bits. These read-only status bits indicate the state of doorbell controller 0.

- Available (PWDCSR[A]). Indicates that the inbound doorbell controller is enabled (IDMR[DE]) and the doorbell controller is not in the internal error state (IDSR[TE] = 0).
- Full (PWDCSR[FU]). This bit reflects the inbound doorbell controller queue full status bit (IDSR[QF]).
- Empty (PWDCSR[EM]). This bit reflects the inverted state of the outbound doorbell busy bit (ODSR[DUB] = 0).

- Busy (PWDCSR[B]). This bit reflects the state of the inbound doorbell controller busy bit IDSR[DUB].
- Failed (PWDCSR[FA]). This bit reflects the state of the transaction error status bit IDSR[TE].
- Error (PWDCSR[ERR]). This bit is always a 0.

15.10.3 Port-Write Controller

The implementation of the port-write controller is very similar to the inbound message and doorbell controllers except only one queue entry is supported. The port write is intended as an error reporting mechanism from an end point-less device to a control processor or other system host.

Figure 15-104 depicts an example of the structure of the inbound queue and pointer. The port-write queue only contains one entry with a fixed size of 64 bytes and aligned to a cache line boundary.

The port-write controller uses the error/port-write interrupt (SRIO error/write-port) for notification of incoming port-writes.

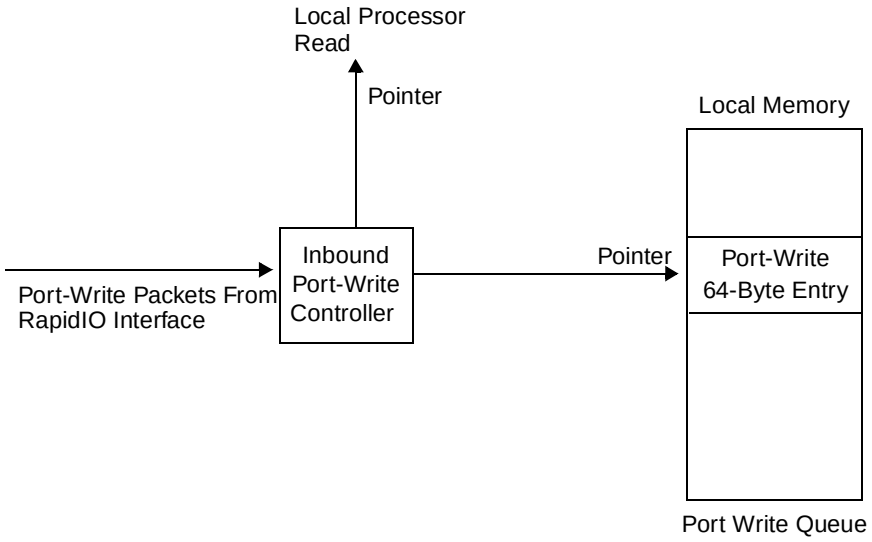


Figure 15-104. Inbound Port-Write Structure

15.10.3.1 Port-Write Controller Initialization

There are many ways in which software can interact with the port-write controller. One method to initialize the port-write controller is as follows:

- Initialize the port-write queue base address registers (IPWQBAR, EIPWQBAR).
- Clear the status register (IPWSR).
- Set the port-write enable bit (IPWMR[PWE]) along with the other control parameters (snoop enable and the interrupt enable) in the inbound port-write mode register (IPWMR).

15.10.3.2 Port-Write Controller Operation

There are several ways in which software can interact with the port-write controller. One method is as follows:

- The port-write controller receives a port-write from the RapidIO port. If the inbound port-write controller is enabled (IPWMR[PWE] = 1) and the port-write queue is not full, then the port-write is accepted.
- 64 bytes of payload is stored by the port-write controller in local memory using the value of the port-write queue base address registers (IPWQBAR, EIPWQBAR). Valid payload sizes include 4, 8, 16, 24, 32, 40, 48, 56 or 64 bytes. Note that 64 bytes are always written to memory. If the actual payload size is less than 64 bytes, the non payload data written is undefined.
- If the queue full interrupt enable bit is set (IPWMR[QFIE]) after the memory write completes the error/port-write interrupt is generated by the port-write controller.
- An inbound error/port-write interrupt is generated to the local processor because a port-write was received and this event is enabled to generate the interrupt (IPWMR[QFIE]). Note that the RMU actually generates the SRIO error/write-port output and this is combined with the error interrupt to generate the error/port-write interrupt.
- Software determines that the queue full event caused the interrupt by detecting that the queue full interrupt bit is set when reading the inbound port-write status register (IPWSR[QFI]). Note there are many events that can generate this interrupt. Software must read several registers to determine that the interrupt was generated due to a port-write.
- Software processes the port-write queue entry pointed to by the port-write base address registers (IPWQBAR, EIPWQBAR).
- Software sets the clear queue bit (IPWMR[CQ]) re-enabling the hardware to receive another port-write.
- Software clears the queue full interrupt bit (IPWSR[QFI]) by setting the IPWSR[QFI].

15.10.3.3 Port-Write Controller Interrupt

The error/port-write interrupt is used by the port-write controller. This interrupt is used to notify the processor that some type of error event has occurred in a RapidIO port, message controller, doorbell controller or port-write controller. There are many events that can generate this interrupt. For example, the error management extensions use this interrupt to notify that error events have occurred. In the port-write controller the following event can generate this interrupt.

- Queue full—an interrupt is generated when this interrupt event is enabled (IPWMR[QFIE]) and a port-write is received and has been written to memory. The event that caused this interrupt is indicated by IPWSR[QFI]. The interrupt is held until the queue is no longer full and the IPWSR[QFI] bit has been cleared by writing a 1.
- Transaction error—an interrupt is generated after a OCN error response is received and this interrupt event is enabled (IPWMR[EIE]).

15.10.3.4 Discarding Port-Writes

While the queue full bit is set or if a port-write is currently being written to memory but has not completed all received port-writes are discarded. When a port-write is discarded for one of these reasons the controller sets the port write discarded bit (IPWSR[PWD]). Note that the port-write busy bit (IPWSR[PWB]) indicates that a port-write is currently being written to memory but has not completed.

15.10.3.5 Transaction Errors

When an internal error occurs during a local memory write by the port-write controller the following occurs:

- The port-write controller sets the transaction error bit (IPWSR[TE]) and enters the error state.
- If IPWMR[EIE] is set, the interrupt SRIO error/write-port is generated.

15.10.3.5.1 Error Handling

When an error occurs and the SRIO error/write-port interrupt is generated, the following occurs:

- Software determines the cause of the interrupt and processes the error.
- Software verifies the port-write controller has stopped operation by polling IPWSR[PWB].
- Software disables the port-write controller by clearing IPWMR[PWE].
- Software clears the error by writing a 1 to the corresponding status bit (IPWSR[TE]).
- The port-write unit must be disabled, re-initialized and re-enabled before another maintenance port-write can be received.

When an error occurs and the SRIO error/write-port interrupt is not enabled, the following occurs.

- Software determines that an error has occurred by polling the status bits (IPWSR[TE]).
- Software verifies the port-write controller has stopped operation by polling IPWSR[PWB].
- Software disables the port-write controller by clearing IPWMR[PWE].
- Software clears the error by writing a 1 to the corresponding status bit (IPWSR[TE]).

15.10.3.6 Hardware Error Handling

The following list possible error conditions and what occurs when they are encountered. The error checking level indicates the order in which errors are checked. Multiple errors can be checked at an error checking level. Once an error is detected no additional error checking beyond the current level is performed. Note that port-writes are processed in a pipeline fashion. The first error detected in the processing pipeline updates the error management extensions registers.

These error condition checks are provided by the messaging unit.

These check are in addition to the error condition checks provided by the RapidIO port given in Section 15.8.12, “Errors and Error Handling.”

Table 15-136. Inbound Port-Write Hardware Errors

Error	Error Checking Level	Interrupt Generated	Status Bit Set	Queue entry written in local memory	Response status	Logical/Transport Layer Capture Register	Comments
reserved ftype[1]	1	SRIO error/write-port if LTLEECSR[UT] set	Unsupported transaction in the Logical/Transport Layer Error Detect CSR LTLEDCSR[UT]	No	No Response	Updated with the packet[2]	Packet is ignored and discarded.
Reserved tt encoding[1]	1	SRIO error/write-port if LTLEECSR[TSE] set	Transport size error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[TSE].	No	No Response	Updated with the packet[2]	Packet is ignored and discarded.
Large transport size when operating in small transport size or small transport size when operating in large transport size[1]	1	SRIO error/write-port if LTLEECSR[TSE] set	Transport size error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[TSE].	No	No Response	Updated with the packet[2]	Packet is ignored and discarded. An error or illegal transaction target error response is not generated.
Illegal Destination ID[1]	1	SRIO error/write-port if LTLEECSR[ITTE] set	Illegal transaction target in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITTE]	No	No Response	Updated with the packet[2]	Packet is ignored and discarded.

Table 15-136. Inbound Port-Write Hardware Errors (continued)

Error	Error Checking Level	Interrupt Generated	Status Bit Set	Queue entry written in local memory	Response status	Logical/Transport Layer Capture Register	Comments
An incorrect wr_size encoding (not 4, 8, 16, 24, 32, 40, 48, 56 or 64 bytes), payload size greater than the size defined by wr_size, or not dword aligned when the size is not 4 bytes.[1]	1	SRIO error/write-port if LTLEECSR[ITD] set	Message Format error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITD]	No	No Response	Updated with the packet[2]	Packet is ignored and discarded.
wr_size value reserved[1]	1	SRIO error/write-port if LTLEECSR[ITD] set	Message Format error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITD]	No	No Response	Updated with the packet[2]	Packet is ignored and discarded.
Inbound maintenance port-write received and inbound maintenance port-writes are not supported as indicated by DOCAR[PW] [1]	1	SRIO error/write-port if LTLEECSR[UT] set	Unsupported transaction in the Logical/Transport Layer Error Detect CSR LTLEDCSR[UT]	No	Error	Updated with the packet[2]	Packet is ignored and discarded.
Not an error - An inbound port-write packet with a RapidIO priority of 3	2	—	—	Yes	No Response	Inbound port write considered priority 2 by the inbound port write controller since response from memory is required at priority 3.	—

Table 15-136. Inbound Port-Write Hardware Errors (continued)

Error	Error Checking Level	Interrupt Generated	Status Bit Set	Queue entry written in local memory	Response status	Logical/Transport Layer Capture Register	Comments
Port-write Controller Disabled and Port-write Received	2	No	None	No	No Response	—	Packet is ignored and discarded.
Port-write Controller Enabled but in the Error State and Port-write Received	2	No	None	No	No Response	—	Packet is ignored and discarded.
Internal error occurred during the write of the port-write queue entry to memory	3	SRIO error/write-port if IPWMR[EIE] set	Transaction error in the Port-write status register (IPWSR[TE]). Port-write Failed in the Port-write and Doorbell CSR (PWDCSR[PFA])	No	No Response	—	Port-write controller stops after the current port-write operation completes.

Note:

1. These error types are actually detected in the RapidIO port, not in the port-write controller
2. If operating in small transport size configuration using the packet LTLACCSR[XA] gets the extended address (packet bits 78–79), LTLACCSR[A] gets the address (packet bits 48–76), LTLDIDCCSR[MDID] gets 0, LTLDIDCCSR[DID] gets the least significant byte of the destination ID (packet bits 16–23), LTLDIDCCSR[MSID] gets 0, LTLDIDCCSR[SID] gets the least significant byte of the source ID (packet bits 24–31), LTLCCCSR[FT] gets the ftype (packet bits 12–15), LTLCCCSR[TT] gets the ttype (packet bits 32–35), LTLCCCSR[MI] gets 0. If operating in large transport size configuration using the packet LTLACCSR[XA] gets the extended address (packet bits 94–95), LTLACCSR[A] gets the address (packet bits 64–92), LTLDIDCCSR[MDID] gets the most significant byte of the destination ID (packet bits 16–23), LTLDIDCCSR[DID] gets the least significant byte of the destination ID (packet bits 24–31), LTLDIDCCSR[MSID] gets the most significant byte of the source ID (packet bits 32–39), LTLDIDCCSR[SID] gets the least significant byte of the source ID (packet bits 40–47), LTLCCCSR[FT] gets the ftype (packet bits 12–15), LTLCCCSR[TT] gets the ttype (packet bits 48–51), LTLCCCSR[MI] gets 0.

15.10.3.6.1 Programming Errors

The following is the list of programming errors.

Table 15-137. Inbound Port-Write Programming Errors

Error	Interrupt Generated	Status Bit Set	Comments
Port-write queue entry written to non-existent memory	No	No	When a write to memory occurs, the memory controller causes its own interrupt and update its own capture registers. An internal error response will be returned. When the port-write controller receives the error response it will set the transaction error bit (IPWSR[TE]) and be in the error state.

15.10.3.7 Disabling and Enabling the Port-Write Controller

When the port-write controller is disabled by clearing IPWMR[PWE] the following occurs:

- Queue full clears (IPWSR[QF]).
- Port-write busy clears (IPWSR[PWB]) after a pending port-write queue entry write completes.

Before the port-write controller is re-enabled (IPWMR[PWE]) the port-write busy bit must be clear (IPWSR[PWB]).

15.10.3.8 RapidIO Message Passing Logical Specification Registers

The port-write and doorbell command and status register (PWDCSR) includes several port-write controller status bits. These read-only status bits only indicate the state of the port-write controller.

- Available (PWDCSR[PA]). Indicates that the port-write controller is enabled (IPWMR[PWE]), the only port-write queue entry is available to be written (IPWSR[QF]) = 0 and the port-write controller is not in the internal error state (IPWSR[TE] = 0).
- Full (PWDCSR[PFU]). This bit reflects the state of the queue full status bit (IPWSR[QF]).
- Empty (PWDCSR[PEM]). This bit always a 1 since a port-write cannot be generated.
- Busy (PWDCSR[PB]). This bit reflects the state of the busy bit IPWSR[PWB].
- Failed (PWDCSR[PFA]). This bit reflects the state of the transaction error status bit IPWSR[TE].
- Error (PWDCSR[PE]). This bit is always a 0.

Chapter 16
PCI Express Interface Controller

The PCI Express interface complies with the *PCI Express™ Base Specification, Revision 1.0a* (available from http://www.pcisig.org). It is beyond the scope of this manual to document the intricacies of the PCI Express protocol. This chapter describes the PCI Express controller of this device and provides a basic description of the PCI Express protocol. The specific emphasis is directed at how the device implements the PCI Express specification. Designers of systems incorporating PCI Express devices should refer to the specification for a thorough description of PCI Express.

NOTE

Much of the available PCI Express literature refers to a 16-bit quantity as a WORD and a 32-bit quantity as a DWORD. Note that this is inconsistent with the terminology in the rest of this manual where the terms ‘word’ and ‘double word’ refer to a 32-bit and 64-bit quantity, respectively. Where necessary to avoid confusion, the precise number of bits or bytes is specified.

16.1 Introduction

The PCI Express controller provides the mechanism to communicate with PCI Express devices. Figure 16-1 is a high-level block diagram of the PCI Express controller.

16.1.1 Overview

The PCI Express controller connects the internal platform to a 2.5- GHz serial interface. The MPC8641D offers two instantiations of this controller yielding up to a x8 link on each SerDes port, 1 and 2.The remainder of this chapter refers to a single PCI Express controller offering up to a x8 link interface. Notes are included to indicate variations for multiple instantiations.

As both an initiator and a target device, the PCI Express interface is capable of high-bandwidth data transfer and is designed to support next generation I/O devices. Upon coming out of reset, the PCI Express interface performs link width negotiation and exchanges flow control credits with its link partner. Once link autonegotiation is successful, the controller is in operation.

Internally, the design contains queues to keep track of inbound and outbound transactions. There is control logic that handles buffer management, bus protocol, transaction spawning and tag generation. In addition, there are memory blocks used to store inbound and outbound data.

The PCI Express controller can be configured to operate as either a PCI Express root complex (RC) or an endpoint (EP) device. An RC device connects the host CPU/memory subsystem to I/O devices while an

EP device typically denotes a peripheral or I/O device. In RC mode, a PCI Express type 1 configuration header is used; in EP mode, a PCI Express type 0 configuration header is used.

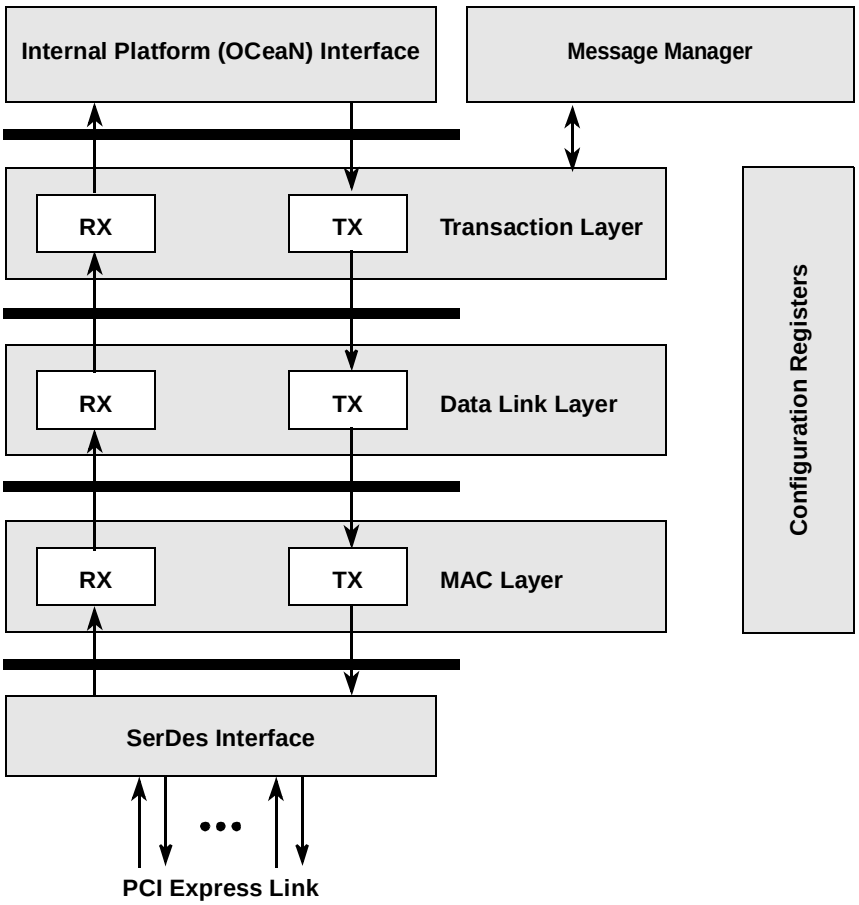


Figure 16-1. PCI Express Controller Block Diagram

As an initiator, the PCI Express controller supports memory read and write operations with a maximum transaction size of 256 bytes. In addition, configuration and I/O transactions are supported if the PCI Express controller is in RC mode. As a target interface, the PCI Express controller accepts read and write operations to local memory space. When configured as an EP device, the PCI Express controller accepts configuration transactions to the internal PCI Express configuration registers. Message generation and acceptance are supported in both RC and EP modes. Locked transactions and inbound I/O transactions are not supported.

16.1.1.1 Outbound Transactions

Outbound internal platform transactions to PCI Express are first mapped to a translation window to determine what PCI Express transactions are to be issued. A transaction from the internal platform can become a PCI Express Memory, I/O, Message, or Configuration transaction depending on the window attributes.

A transaction may be broken up into smaller sized transactions depending on the original request size, transaction type, and either the PCI Express device control register [MAX_PAYLOAD_SIZE] field for write requests or the PCI Express device control register [MAX_READ_SIZE] field for read requests. The controller performs PCI Express ordering rule checking to determine which transaction is to be sent on the PCI Express link.

In general, transactions are serviced in the order that they are received from the internal platform (OCeaN). Only when there is a stalled condition does the controller apply PCI Express ordering rules to outstanding transactions. For posted write transactions, once all data has been received from the internal platform (OCeaN), the data is forwarded to the PCI Express link and the transaction is considered as done. For non-posted write transactions, the controller waits for the completion packets to return before considering the transaction finished. For non-posted read transactions, the controller waits for all completion packets to return and then forwards all data back to the internal platform before terminating the transaction.

Note that after reset or when recovering from a link down condition, external transactions should not be attempted until the link has successfully trained. Software can poll the LTSSM state status register (PEX_LTSSM_STAT) to check the status of link training before issuing external requests.

16.1.1.2 Inbound Transactions

Inbound PCI Express transactions to internal platform are first mapped to a translation window to determine what internal platform transactions are to be issued.

A transaction may be broken up into smaller sized transactions when sending to the internal platform depending on the original request size, byte enables and starting/ending addresses. The controller performs PCI Express ordering rule checking to determine what transaction is to be sent next to the internal platform (OCeaN).

In general, transactions are serviced in the order that they are received from the PCI Express link. Only when there is a stalled condition does the controller apply PCI Express ordering to outstanding transactions. For posted write transactions, once all data has been received from the PCI Express link, the data is forwarded to the internal platform and the transaction is considered as done. For non-posted read transactions, the controller forwards internal platform data back to the PCI Express link.

Note that the controller splits transactions at the crossing of every 256-byte-aligned boundary when sending data back to the PCI Express link.

16.1.2 Features

The following is a list of features supported by the PCI Express controller:

- Complies with the *PCI Express™ Base Specification, Revision 1.0a*
- Supports root complex (RC) and endpoint (EP) configurations
- 32- and 64-bit address support
- x8, x4, x2 and x1 link supported on each of the two SerDes ports
- Supports accesses to all PCI Express memory and I/O address spaces (requestor only)
- Supports posting of processor-to-PCI Express and PCI Express-to-memory writes

- Supports strong and relaxed transaction ordering rules
- PCI Express configuration registers (type 0 in EP mode, type 1 in RC mode)
- Baseline and advanced error reporting support
- One virtual channel (VC0)
- 256-byte maximum payload size (MAX_PAYLOAD_SIZE)
- Supports three inbound general-purpose translation windows and one configuration window
- Supports four outbound translation windows and one default window
- Supports eight non-posted and four posted PCI Express transactions
- Supports up to six priority 0 internal platform reads and eight priority 0 to 2 internal platform writes. (The maximum number of outstanding transactions at any given time is eight.)
- Credit-based flow control management
- Supports PCI Express messages and interrupts
- Accepts up to 256-byte transactions from the internal platform (OCeaN)

16.1.3 Modes of Operation

There is one parameter that affects the mode of operation for the PCI Express controller. It is determined at power-on reset (POR) by a reset configuration signal as described in Chapter 4, “Reset, Clocking, and Initialization.”

Table 16-1. POR Parameters for PCI Express Controller

Parameter	Description	Section/Page
Host/Agent Configuration	Selects between root complex (RC) and endpoint (EP) modes.	4.4.3.9/4-17

16.1.3.1 Root Complex/Endpoint Modes

The PCI Express controller can function as either a root complex (RC) or an endpoint (EP) on the PCI Express link. The host/agent configuration input signals cfg_host_agt[0:1] multiplexed on $\overline{\text{LWE}}$/$\overline{\text{LBS}}$[2:3] determine the RC/EP mode.

16.2 External Signal Descriptions

PCI Express defines the connection between two devices as a link, which can be composed of a single or multiple lanes. Each lane consists of a differential pair for transmitting (TX*n* and $\overline{\text{TX}}$*n*) and a differential pair for receiving (RX*n* and $\overline{\text{RX}}$*n*) with an embedded data clock.

Table 16-2 contains detailed descriptions of the external PCI Express interface signals. Note that there are two SD ports (SD1 and SD2) corresponding to PCI Express Controller 1 and 2 (depending on the high-speed interface port configuration—see Chapter 4, “Reset, Clocking, and Initialization for more information).

Table 16-2. PCI Express Interface Signals—Detailed Signal Descriptions

Signal	I/O	Description	
SD*n*_RX[7:0]	I	Receive data. The receive data signals carry PCI Express packet information. SD*n*_RX[7:0] correspond to PCI Express RX lanes 7:0. Note that lane reversal may affect the logical lane assignment. Refer to Section 16.4.1.3, "Lane Reversal," for more information.	
		State Meaning	Asserted/Negated—Represents data being received from the PCI Express interface.
		Timing	Assertion/Negation—As described in the *PCI Express Base Specification, Revision 1.0a*.
$\overline{\text{SD}n\text{\_RX}}$[7:0]	I	Receive data, inverted. $\overline{\text{SD}n\text{\_RX}}$[7:0] are the inverted forms of the receive data signals (SD*n*_RX[7:0]).	
		State Meaning	Asserted/Negated—Represents the inverse of data being received from the PCI Express interface.
		Timing	Assertion/Negation—As described in the *PCI Express Base Specification, Revision 1.0a*.
SD*n*_TX[7:0]	O	Transmit data. The transmit data signals carry PCI Express packet information. SD*n*_TX[7:0] correspond to PCI Express TX lanes 7:0. Note that lane reversal may affect the logical lane assignment. Refer to Section 16.4.1.3, "Lane Reversal," for more information.	
		State Meaning	Asserted/Negated—Represents data being transmitted to the PCI Express interface.
		Timing	Assertion/Negation—As described in the *PCI Express Base Specification, Revision 1.0a* .
$\overline{\text{SD}n\text{\_TX}}$[7:0]	O	Transmit data, inverted. $\overline{\text{SD}n\text{\_TX}}$[7:0] are the inverted form of the transmit data signals (SD*n*_TX[7:0]).	
		State Meaning	Asserted/Negated—Represents the inverse of data being transmitted to the PCI Express interface.
		Timing	Assertion/Negation—As described in the *PCI Express Base Specification, Revision 1.0a*.

16.3 Memory Map/Register Definitions

The PCI Express interface supports the following register types:

- Memory-mapped registers—these registers control PCI Express address translation, PCI error management, and PCI Express configuration register access. These registers are described in Section 16.3.1, "PCI Express Memory Mapped Registers," and its subsections.
- PCI Express configuration registers contained within the PCI Express configuration space—these registers are specified by the PCI Express specification for every PCI Express device. These registers are described in Section 16.3.7, "PCI Express Configuration Space Access," and its subsections.

16.3.1 PCI Express Memory Mapped Registers

The PCI Express memory mapped registers are accessed by reading and writing to an address comprised of the base address (specified in the CCSRBAR on the local side or the PEXCSRBAR on the PCI Express side) plus the block base address, plus the offset of the specific register to be accessed. Note that all memory-mapped registers (except the PCI Express configuration data register, PEX_CONFIG_DATA) must only be accessed as 32-bit quantities.

Also note that although the table explicitly lists only the registers for the PCI Express controller 1, the register map for PCI Express controller 2 is the same except the for the block base address.

Memory-mapped registers for PCI Express controller 1 begin at block base address 0x0_8000 and the memory-mapped registers for PCI Express controller 2 begin at block base address 0x0_9000.

Table 16-3 lists the memory-mapped registers.

Table 16-3. PCI Express Memory-Mapped Register Map

PCI Express Controller 1 —Block Base Address 0x0_8000 PCI Express Controller 2—Block Base Address 0x0_9000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
PCI Express Controller 1 Memory-Mapped Registers				
PCI Express Configuration Access Registers				
0x000	PEX_CONFIG_ADDR—PCI Express configuration address register	R/W	0x0000_0000	16.3.2.1/16-9
0x004	PEX_CONFIG_DATA—PCI Express configuration data register	R/W	0x0000_0000	16.3.2.2/16-10
0x008	Reserved	—	—	—
0x00C	PEX_OTB_CPL_TOR—PCI Express outbound completion timeout register	R/W	0x0010_FFFF	16.3.2.3/16-11
0x010	PEX_CONF_RTY_TOR—PCI Express configuration retry timeout register	R/W	0x0400_FFFF	16.3.2.4/16-11
0x014	PEX_CONFIG—PCI Express configuration register	R/W	0x0000_0000	16.3.2.5/16-12
0x018–0x01C	Reserved	—	—	—
PCI Express Power Management Event & Message Registers				
0x020	PEX_PME_MES_DR—PCI Express PME & message detect register	w1c	0x0000_0000	16.3.3.1/16-13
0x024	PEX_PME_MES_DISR—PCI Express PME & message disable register	R/W	0x0000_0000	16.3.3.2/16-15
0x028	PEX_PME_MES_IER—PCI Express PME & message interrupt enable register	R/W	0x0000_0000	16.3.3.3/16-16
0x02C	PEX_PMCR—PCI Express power management command register	R/W	0x0000_0000	16.3.3.4/16-18
0x030–0xBF4	Reserved	—	—	—
PCI Express IP Block Revision Registers				
0xBF8	IP block revision register 1 (PEX_IP_BLK_REV1)	R	0x0208_0100	16.3.4.1/16-18
0xBFC	IP block revision register 2 (PEX_IP_BLK_REV2)	R	0x0000_0000	16.3.4.2/16-19
PCI Express ATMU Registers				
Outbound Window 0 (Default)				
0xC00	PEXOTAR0—PCI Express outbound translation address register 0 (default)	R/W	0x0000_0000	16.3.5.1.1/16-20
0xC04	PEXOTEAR0—PCI Express outbound translation extended address register 0 (default)	R/W	0x0000_0000	16.3.5.1.2/16-21
0xC08–0xC0C	Reserved	—	—	—
0xC10	PEXOWAR0—PCI Express outbound window attributes register 0 (default)	Mixed	0x8004_4023	16.3.5.1.4/16-22

Table 16-3. PCI Express Memory-Mapped Register Map (continued)

PCI Express Controller 1 —Block Base Address 0x0_8000 PCI Express Controller 2—Block Base Address 0x0_9000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0xC14–0xC1C	Reserved	—	—	—
Outbound Window 1				
0xC20	PEXOTAR1—PCI Express outbound translation address register 1	R/W	0x0000_0000	16.3.5.1.1/16-20
0xC24	PEXOTEAR1—PCI Express outbound translation extended address register 1	R/W	0x0000_0000	16.3.5.1.2/16-21
0xC28	PEXOWBAR1—PCI Express outbound window base address register 1	R/W	0x0000_0000	16.3.5.1.3/16-21
0xC2C	Reserved	—	—	—
0xC30	PEXOWAR1—PCI Express outbound window attributes register 1	R/W	0x0004_4023	16.3.5.1.4/16-22
0xC34–0xC3C	Reserved	—	—	—
Outbound Window 2				
0xC40	PEXOTAR2—PCI Express outbound translation address register 2	R/W	0x0000_0000	16.3.5.1.1/16-20
0xC44	PEXOTEAR2—PCI Express outbound translation extended address register 2	R/W	0x0000_0000	16.3.5.1.2/16-21
0xC48	PEXOWBAR2—PCI Express outbound window base address register 2	R/W	0x0000_0000	16.3.5.1.3/16-21
0xC4C	Reserved	—	—	—
0xC50	PEXOWAR2—PCI Express outbound window attributes register 2	R/W	0x0004_4023	16.3.5.1.4/16-22
0xC54–0xC5C	Reserved	—	—	—
Outbound Window 3				
0xC60	PEXOTAR3—PCI Express outbound translation address register 3	R/W	0x0000_0000[1]	16.3.5.1.1/16-20
0xC64	PEXOTEAR3—PCI Express outbound translation extended address register 3	R/W	0x0000_0000	16.3.5.1.2/16-21
0xC68	PEXOWBAR3—PCI Express outbound window base address register 3	R/W	0x0000_0000[2]	16.3.5.1.3/16-21
0xC6C	Reserved	—	—	—
0xC70	PEXOWAR3—PCI Express outbound window attributes register 3	R/W	0x0000_0000[3]	16.3.5.1.4/16-22
0xC74–0xC7C	Reserved	—	—	—
Outbound Window 4				
0xC80	PEXOTAR4—PCI Express outbound translation address register 4	R/W	0x0000_0000	16.3.5.1.1/16-20
0xC84	PEXOTEAR4—PCI Express outbound translation extended address register 4	R/W	0x0000_0000	16.3.5.1.2/16-21
0xC88	PEXOWBAR4—PCI Express outbound window base address register 4	R/W	0x0000_0000	16.3.5.1.3/16-21
0xC8C	Reserved	—	—	—

Table 16-3. PCI Express Memory-Mapped Register Map (continued)

PCI Express Controller 1 —Block Base Address 0x0_8000 PCI Express Controller 2—Block Base Address 0x0_9000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0xC90	PEXOWAR4—PCI Express outbound window attributes register 4	R/W	0x0004_4023	16.3.5.1.4/16-22
0xC94–0xC9C	Reserved	—	—	—
0xD14–0xD9C	Reserved	—	—	—
Inbound Window 3				
0xDA0	PEXITAR3—PCI Express inbound translation address register 3	R/W	0x0000_0000	16.3.5.2.3/16-25
0xDA4	Reserved	—	—	—
0xDA8	PEXIWBAR3—PCI Express inbound window base address register 3	R/W	0x0000_0000	16.3.5.2.4/16-26
0xDAC	PEXIWBEAR3—PCI Express inbound window base extended address register 3	R/W	0x0000_0000	16.3.5.2.5/16-27
0xDB0	PEXIWAR3—PCI Express inbound window attributes register 3	R/W	0x20F4_4023	16.3.5.2.6/16-27
0xDB4–0xDBC	Reserved	—	—	—
Inbound Window 2				
0xDC0	PEXITAR2—PCI Express inbound translation address register 2	R/W	0x0000_0000	16.3.5.2.3/16-25
0xDC4	Reserved	—	—	—
0xDC8	PEXIWBAR2—PCI Express inbound window base address register 2	R/W	0x0000_0000	16.3.5.2.4/16-26
0xDCC	PEXIWBEAR2—PCI Express inbound window base extended address register 2	R/W	0x0000_0000	16.3.5.2.5/16-27
0xDD0	PEXIWAR2—PCI Express inbound window attributes register 2	R/W	0x20F4_4023	16.3.5.2.6/16-27
0xDD4–0xDDC	Reserved	—	—	—
Inbound Window 1				
0xDE0	PEXITAR1—PCI Express inbound translation address register 1	R/W	0x0000_0000	16.3.5.2.3/16-25
0xDE4	Reserved	—	—	—
0xDE8	PEXIWBAR1—PCI Express inbound window base address register 1	R/W	0x0000_0000	16.3.5.2.4/16-26
0xDEC	Reserved	—	—	—
0xDF0	PEXIWAR1—PCI Express inbound window attributes register 1	R/W	0x20F4_4023	16.3.5.2.6/16-27
0xDF4–0xDFC	Reserved	—	—	—
PCI Express Error Management Registers				
0xE00	PEX_ERR_DR—PCI Express error detect register	w1c	0x0000_0000	16.3.6.1/16-29
0xE04	Reserved	—	—	—
0xE08	PEX_ERR_EN—PCI Express error interrupt enable register	R/W	0x0000_0000	16.3.6.2/16-32
0xE0C	Reserved	—	—	—
0xE10	PEX_ERR_DISR—PCI Express error disable register	R/W	0x0000_0000	16.3.6.3/16-34

Table 16-3. PCI Express Memory-Mapped Register Map (continued)

PCI Express Controller 1 —Block Base Address 0x0_8000 PCI Express Controller 2—Block Base Address 0x0_9000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0xE14– 0xE1C	Reserved	—	—	—
0xE20	PEX_ERR_CAP_STAT—PCI Express error capture status register	Mixed	0x0000_0000	16.3.6.4/16-35
0xE24	Reserved	—	—	—
0xE28	PEX_ERR_CAP_R0—PCI Express error capture register 0	R/W	0x0000_0000	16.3.6.5/16-36
0xE2C	PEX_ERR_CAP_R1—PCI Express error capture register 1	R/W	0x0000_0000	16.3.6.6/16-38
0xE30	PEX_ERR_CAP_R2—PCI Express error capture register 2	R/W	0x0000_0000	16.3.6.7/16-39
0xE34	PEX_ERR_CAP_R3—PCI Express error capture register 3	R/W	0x0000_0000	16.3.6.8/16-41
0xE38– 0xFFC	Reserved	—	—	—
PCI Express Controller 2 Memory-Mapped Registers				
0x000– 0xFFC	PCI Express Controller 2 registers **Note:** All registers defined for PCI Express Controller 1 are also defined for PCI Express Controller 2; the offsets of PCI Express Controller 2 registers are the same except they have a different block base address.			

1 If the device is configured to use the alternate boot vector (cfg_boot_vec = 0), the reset value for PCI controller 1 PEXOTAR3 is 0x000F_FFF0.

2 If the device is configured to use the alternate boot vector (cfg_boot_vec = 0), the reset value for PCI controller 1 PEXOWBAR3 is 0x000F_FF00.

3 If the device is configured to use the alternate boot vector (cfg_boot_vec = 0), the reset value for PCI controller 1 PEXOWAR3 is 0x8004_400F.

16.3.2 PCI Express Configuration Access Registers

16.3.2.1 PCI Express Configuration Address Register (PEX_CONFIG_ADDR)

The PCI Express configuration address register, shown in Figure 16-2, contains address information for accesses to PCI Express internal and external configuration registers.

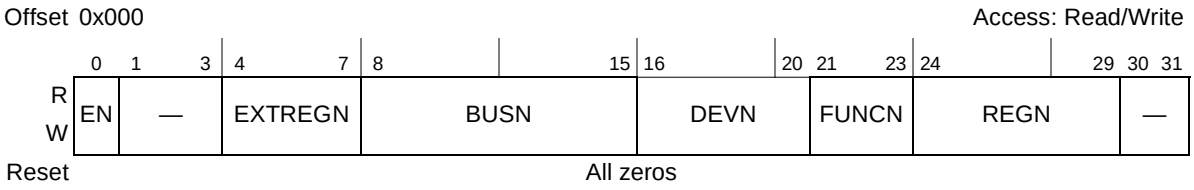


Figure 16-2. PCI Express Configuration Address Register (PEX_CONFIG_ADDR)

The fields of the PCI Express configuration address register are described in Table 16-4.

Table 16-4. PEX_CONFIG_ADDR Field Descriptions

Bits	Name	Description
0	EN	Enable. This bit allows a PCI Express configuration access when PEX_CONFIG_DATA is accessed. If this bit is cleared, writing to PEX_CONFIG_DATA has no effect and reading PEX_CONFIG_DATA returns unknown data.
1–3	—	Reserved
4–7	EXTREGN	Extended register number. This field allows access to extended PCI Express configuration space (that is, the registers in the offset range from 0x100 to 0xFFF).
8–15	BUSN	Bus number. PCI bus number to access
16–20	DEVN	Device number. Device number to access on specified bus
21–23	FUNCN	Function number. Function to access within specified device
24–29	REGN	Register number. 32-bit register to access within specified device
30–31	—	Reserved

Both root complex (RC) and endpoint (EP) configuration headers contain 4096 bytes of address space. To access a register within the header, both the extended register number and the register number fields are concatenated to form the 4-byte aligned address of the register. That is, the register address is extended register number || register number || 0b00.

16.3.2.2 PCI Express Configuration Data Register (PEX_CONFIG_DATA)

The PCI Express configuration data register, show in Figure 16-3, is a 32-bit port for internal and external configuration access. Note that accesses of 1, 2, or 4 bytes to the PCI Express configuration data register are allowed. Also note that accesses to the little-endian PCI Express configuration space must be properly formatted. See Section 16.4.1.2.1, “Byte Order for Configuration Transactions,” for more information.

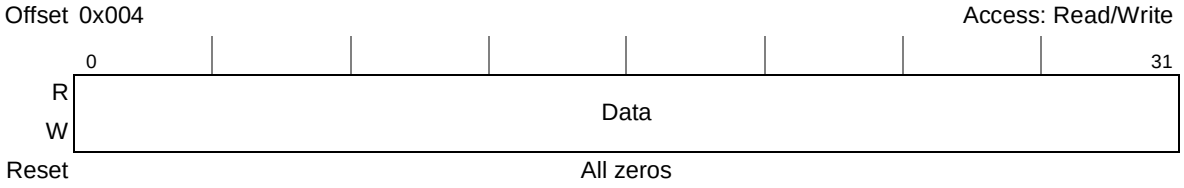


Figure 16-3. PCI Express Configuration Data Register (PEX_CONFIG_DATA)

The fields of the PCI Express configuration data register are described in Table 16-5.

Table 16-5. PEX_CONFIG_DATA Field Descriptions

Bits	Name	Description
0–31	Data	A read or write to this register starts a PCI Express configuration cycle if the PEX_CONFIG_ADDR enable bit is set (PEX_CONFIG_ADDR[EN] = 1).

16.3.2.3 PCI Express Outbound Completion Timeout Register (PEX_OTB_CPL_TOR)

The PCI Express outbound completion timeout register, shown in Figure 16-4, contains the maximum wait time for a response to come back as a result of an outbound non-posted request before a timeout condition occurs.

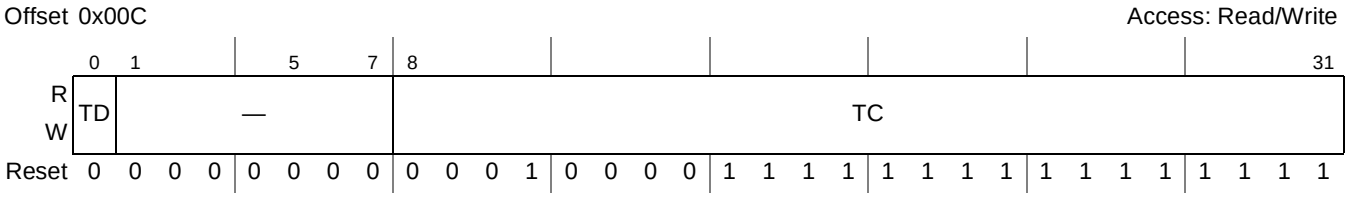


Figure 16-4. PCI Express Outbound Completion Timeout Register (PEX_OTB_CPL_TOR)

The fields of the PCI Express outbound completion timeout register are described in Table 16-6.

Table 16-6. PEX_OTB_CPL_TOR Field Descriptions

Bits	Name	Description
0	TD	Timeout disable. This bit controls the enabling/disabling of the timeout function. 0 Enable completion timeout 1 Disable completion timeout
1–7	—	Reserved
8–31	TC	Timeout counter. This is the value that is used to load the response counter of the completion timeout. One TC unit is 8× the PCI Express controller clock period; that is, one TC unit is 20 ns at 400 MHz, and 30 ns at 266.66 MHz. The following are examples of timeout periods based on different TC settings: 0x00_0000 Reserved 0x10_FFFF 22.28 ms at 400 MHz controller clock; 33.34 ms at 266.66 MHz controller clock 0xFF_FFFF 335.54 ms at 400 MHz controller clock; 503.31 ms at 266.66 MHz controller clock

16.3.2.4 PCI Express Configuration Retry Timeout Register (PEX_CONF_RTY_TOR)

The PCI Express configuration retry timeout register, shown in Figure 16-5, contains the maximum time period during which retries of configuration transactions which resulted in a CRS response occur.

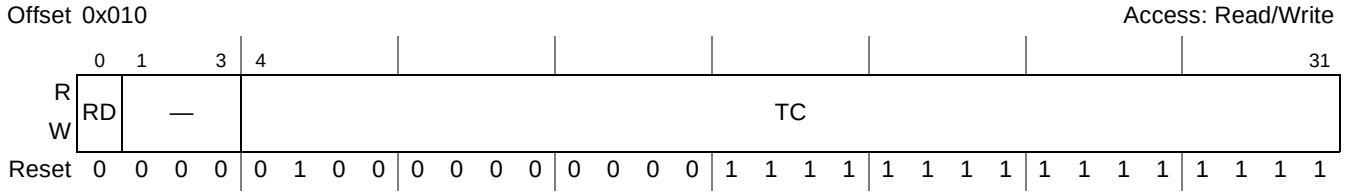


Figure 16-5. PCI Express Configuration Retry Timeout Register (PEX_CONF_RTY_TOR)

The fields of the PCI Express configuration retry timeout register are described in Table 16-7.

Table 16-7. PEX_CONF_RTY_TOR Field Descriptions

Bits	Name	Description
0	RD	Retry disable. This bit disables the retry of a configuration transaction that receives a CRS status response packet. 0 Enable retry of a configuration transaction in response to receiving a CRS status response until the timeout counter (defined by the PEX_CONF_RTY_TOR[TC] field) has expired. 1 Disable retry of a configuration transaction regardless of receiving a CRS status response.
1–3	—	Reserved
4–31	TC	Timeout counter. This is the value that is used to load the CRS response counter. One TC unit is 8× the PCI Express controller clock period; that is, one TC unit is 20 ns at 400 MHz and 30 ns at 266.66 MHz. Timeout period based on different TC settings: 0x000_0000 Reserved 0x400_FFFF 1.34 s at 400 MHz controller clock, 2.02 s at 266.66 MHz controller clock 0xFFF_FFFF 5.37 s at 400 MHz controller clock, 8.05 s at 266.66 MHz controller clock

16.3.2.5 PCI Express Configuration Register (PEX_CONFIG)

The PCI Express configuration register, shown in Figure 16-6, contains various control switches for the controller.

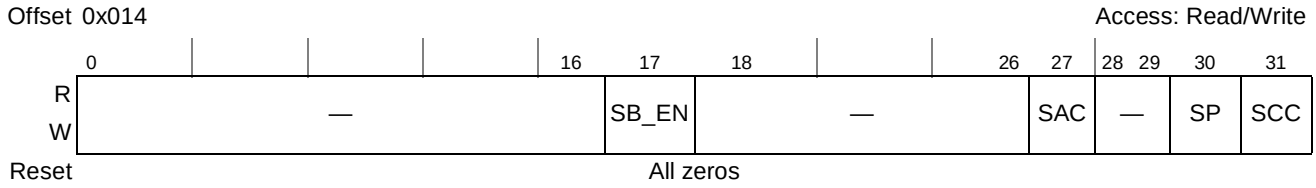


Figure 16-6. PCI Express Configuration Register (PEX_CONFIG)

The fields of the PCI Express configuration register are described in Table 16-8.

Table 16-8. PEX_CONFIG Field Descriptions

Bits	Name	Description
0–16	—	Reserved
17	SB_EN	Southbridge enable. Enables the ability to operate with certain southbridge devices that require being configured on bus 0. Note that this bit is for use in RC mode only; this bit must be cleared in EP mode. 0 Configuration transactions targeting bus 0 are handled internally and are not allowed to pass to the external link. 1 Configuration transactions targeting bus 0 are allowed to pass to the external link.
18–26	—	Reserved
27	SAC	Sense ASPM Control. This bit controls the default value of ASPM of PEX Link Control Register's bit 0. See Section 16.3.9.11, "PCI Express Link Control Register—0x5C," for more information.
28–29	—	Reserved

Table 16-8. PEX_CONFIG Field Descriptions

Bits	Name	Description
30	SP	Slot Present. This bit controls the default value of the PCI Express capabilities register [slot] bit. See Section 16.3.9.6, “PCI Express Capabilities Register—0x4E,” for more information.
31	SCC	Slot Clock Configuration. This bit controls the default value of the PCI Express link status register [SCC] bit. See Section 16.3.9.12, “PCI Express Link Status Register—0x5E,” for more information.

16.3.3 PCI Express Power Management Event and Message Registers

16.3.3.1 PCI Express PME and Message Detect Register (PEX_PME_MES_DR)

The PCI Express PME and message detect register, shown in Figure 16-7, logs inbound messages and PME events that are detected by the PCI Express controller. This register is a write-1-to-clear type register.

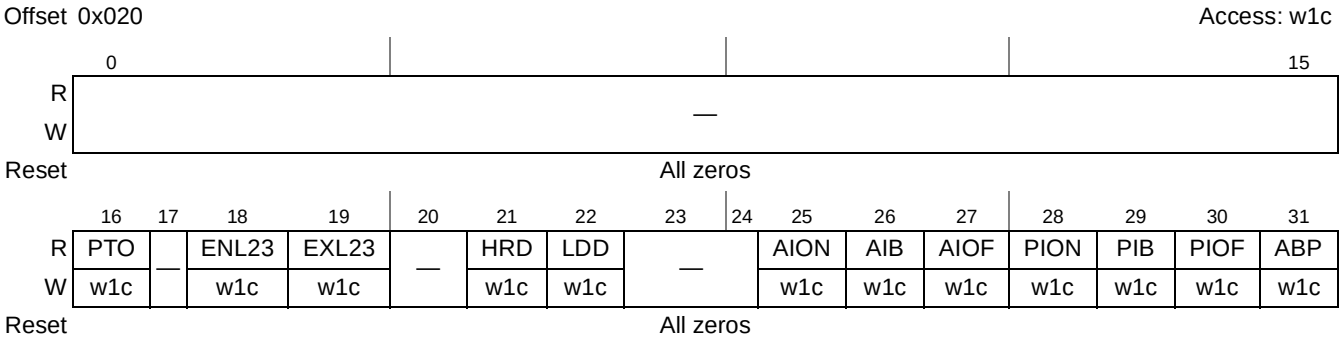


Figure 16-7. PCI Express PME and Message Detect Register (PEX_PME_MES_DR)

The fields of the PCI Express PME and message detect register are described in Table 16-9.

Table 16-9. PEX_PME_MES_DR Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16	PTO	PME turn off. This bit indicates the detection of a PME_Turn_Off message. This bit is only valid in EP mode. 1 A PME_Turn_Off_message is detected 0 No PME_Turn_Off message detected
17	—	Reserved. Note that during normal operation, this bit may be set (falsely). The bit may be ignored and cleared (w1c) without consequence.
18	ENL23	Entered L2/L3 ready state. This bit indicates that the PCI Express controller has entered L2/L3 state. This is only valid in RC mode. 1 L2/L3 ready state has been entered 0 L2/L3 ready state has not been entered
19	EXL23	Exit L2/L3 ready state. This bit indicates that the PCI Express controller has exited the L2/L3 state. This is only valid in RC mode. 1 Exit L2/L3 state has been detected 0 Exit L2/L3 state not detected

Table 16-9. PEX_PME_MES_DR Field Descriptions (continued)

Bits	Name	Description
20	—	Reserved. Note that during normal operation, this bit may be set (falsely). The bit may be ignored and cleared (w1c) without consequence.
21	HRD	Hot reset detected. This bit indicates that the PCI Express controller has detected a hot reset condition on the link. The controller is reset and cleans up all outstanding transactions. Link retraining takes place once hot reset state is exited. This is valid only in EP mode. 1 Hot reset request has been detected 0 Hot reset request not detected
22	LDD	Link down detected. This bit indicates that a link down condition has been detected. The controller is reset and then cleans up all outstanding transactions. Link retraining takes place once the controller has cleaned itself up. Note that for EP, this bit and HRD are typically set when a hot reset event is detected. 1 Link down has been detected 0 Link down not detected
23–24	—	Reserved
25	AION	Attention indicator on. This bit indicates the detection of an Attention_Indicator_On message. This bit is only valid in EP mode. 1 Attention indicator on message is detected 0 No attention indicator on message detected
26	AIB	Attention indicator blink. This bit indicates the detection of an Attention_Indicator_Blink message. This bit is only valid in EP mode. 1 Attention indicator blink message is detected 0 No attention indicator blink message detected
27	AIOF	Attention indicator off. This bit indicates the detection of an Attention_Indicator_Off message. This bit is only valid in EP mode. 1 Attention indicator off message is detected 0 No attention indicator off message detected
28	PION	Power indicator on. This bit indicates the detection of a Power_Indicator_On message. This bit is only valid in EP mode. 1 Power indicator on message is detected 0 No power indicator on message detected
29	PIB	Power indicator blink. This bit indicates the detection of an Power_Indicator_Blink message. This bit is only valid in EP mode. 1 Power indicator blink message is detected 0 No power indicator blink message detected
30	PIOF	Power indicator off. This bit indicates the detection of an Power_Indicator_Off message. This bit is only valid in EP mode. 1 Power indicator off message is detected 0 No power indicator off message detected
31	ABP	Attention button pressed. This bit indicates the detection of an Attention_Button_Pressed message. This bit is only valid in EP mode. 1 Attention button press message is detected 0 No attention button press message detected

16.3.3.2 PCI Express PME and Message Disable Register (PEX_PME_MES_DISR)

The PCI Express PME and message disable register, shown in Figure 16-8, when set, prevents the detection of the corresponding bits in the PCI Express PME and message detect register.

Figure 16-8. PCI Express PME and Message Disable Register (PEX_PME_MES_DISR)

The fields of the PCI Express PME and message disable register are described in Table 16-10.

Table 16-10. PEX_PME_MES_DISR Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16	PTOD	PME turn off disable. When set disables the setting of PEX_PME_MES_DR[PTO] bit. 1 Disable PME_Turn_Off_message detection 0 Enable PME_Turn_Off message detection
17	—	Reserved
18	ENL23D	Entered_L2/L3 ready disable. When set disables the setting of PEX_PME_MES_DR[ENL23] bit. 1 Disable Entered_L2/L3 ready state detection 0 Enable Entered_L2/L3 ready state detection
19	EXL23D	Exited_L2/L3 ready disable. When set disables the setting of PEX_PME_MES_DR[EXL23] bit. 1 Disable Exited_L2/L3 ready state detection 0 Enable Exited_L2/L3 ready state detection
20	—	Reserved
21	HRDD	Hot reset detected disable. When set disables the setting of PEX_PME_MES_DR[HRD] bit. 1 Disable hot reset state detection 0 Enable hot reset state detection
22	LDDD	Link down detected disable. When set disables the setting of PEX_PME_MES_DR[LDD] bit. 1 Disable link down state detection 0 Enable link down state detection
23–24	—	Reserved
25	AIOND	Attention indicator on disable. When set disables the setting of PEX_PME_MES_DR[AION] bit. 1 Disable attention indicator on message detection 0 Enable attention indicator on message detection
26	AIBD	Attention indicator blink disable. When set disables the setting of PEX_PME_MES_DR[AIB] bit. 1 Disable attention indicator blink message detection 0 Enable attention indicator blink message detection

Table 16-10. PEX_PME_MES_DISR Field Descriptions (continued)

Bits	Name	Description
27	AIOFD	Attention indicator off disable. When set disables the setting of PEX_PME_MES_DR[AIOF] bit. 1 Disable attention indicator off message detection 0 Enable attention indicator off message detection
28	PIOND	Power indicator on disable. When set disables the setting of PEX_PME_MES_DR[PION] bit. 1 Disable power indicator on message detection 0 Enable power indicator on message detection
29	PIBD	Power indicator blink disable. When set disables the setting of PEX_PME_MES_DR[PIB] bit. 1 Disable power indicator blink message detection 0 Enable power indicator blink message detection
30	PIOFD	Power indicator off disable. When set disables the setting of PEX_PME_MES_DR[PIOF] bit. 1 Disable power indicator off message detection 0 Enable power indicator off message detection
31	ABPD	Attention button pressed disable. When set disables the setting of PEX_PME_MES_DR[ABP] bit. 1 Disable attention button press message detection 0 Enable attention button press message detection

16.3.3.3 PCI Express PME and Message Interrupt Enable Register (PEX_PME_MES_IER)

The PCI Express PME and message interrupt enable register, shown in Figure 16-9, allows for the detection of a message or a PME event to generate an interrupt, provided that the corresponding bit in the PCI Express PME and message detect register is set.

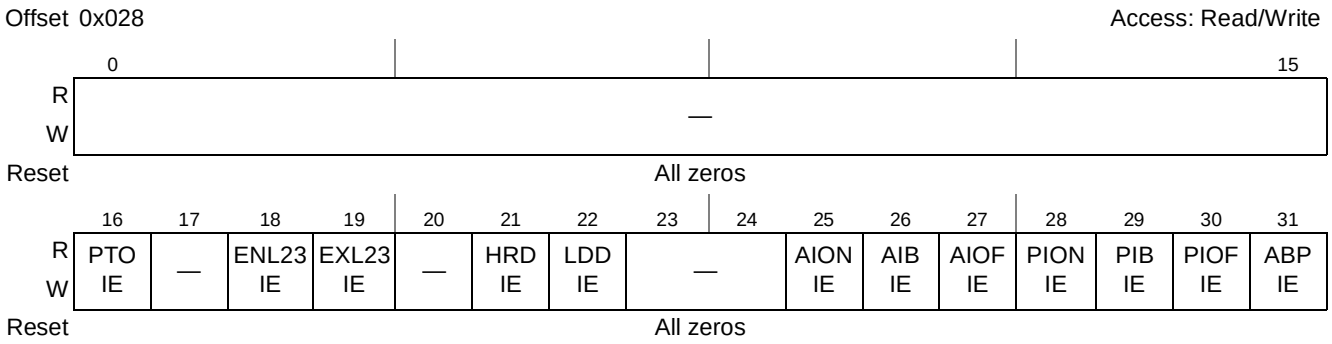


Figure 16-9. PCI Express PME and Message Interrupt Enable Register (PEX_PME_MES_IER)

Table 16-11 shows the fields of the PCI Express PME and message interrupt enable register.

Table 16-11. PEX_PME_MES_IER Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16	PTOIE	PME turn off interrupt enable. When set and PEX_PME_MES_DR[PTO]=1 generates an interrupt. 1 Enable PME_Turn_Off_message interrupt generation 0 Disable PME_Turn_Off message interrupt generation
17	—	Reserved

Table 16-11. PEX_PME_MES_IER Field Descriptions (continued)

Bits	Name	Description
18	ENL23IE	Entered L2/L3 ready interrupt enable. When set and PEX_PME_MES_DR[ENL23]=1 generates an interrupt. 1 Enable Entered_L2/L3 ready state interrupt generation 0 Disable Entered_L2/L3 ready state interrupt generation
19	EXL23IE	Exited L2/L3 ready interrupt enable. When set and PEX_PME_MES_DR[EXL23]=1 generates an interrupt. 1 Enable Exited_L2/L3 ready state interrupt generation 0 Disable Exited_L2/L3 ready state interrupt generation
20	—	Reserved
21	HRDIE	Hot reset detected interrupt enable. When set and PEX_PME_MES_DR[HRD]=1 generates an interrupt. 1 Enable hot reset state interrupt generation 0 Disable hot reset state interrupt generation
22	LDDIE	Link down detected interrupt enable. When set and PEX_PME_MES_DR[LDD]=1 generates an interrupt. 1 Enable link down state interrupt generation 0 Disable link down state interrupt generation
23–24	—	Reserved
25	AIONIE	Attention indicator on interrupt enable. When set and PEX_PME_MES_DR[AION]=1 generates an interrupt. 1 Enable attention indicator on message interrupt generation 0 Disable attention indicator on message interrupt generation
26	AIBIE	Attention indicator blink interrupt enable. When set and PEX_PME_MES_DR[AIB]=1 generates an interrupt. 1 Enable attention indicator blink message interrupt generation 0 Disable attention indicator blink message interrupt generation
27	AIOFIE	Attention indicator off interrupt enable. When set and PEX_PME_MES_DR[AIOF]=1 generates an interrupt. 1 Enable attention indicator off message interrupt generation 0 Disable attention indicator off message interrupt generation
28	PIONIE	Power indicator on interrupt enable. When set and PEX_PME_MES_DR[PION]=1 generates an interrupt. 1 Enable power indicator on message interrupt generation 0 Disable power indicator on message interrupt generation
29	PIBIE	Power indicator blink interrupt enable. When set and PEX_PME_MES_DR[PIB]=1 generates an interrupt. 1 Enable power indicator blink message interrupt generation 0 Disable power indicator blink message interrupt generation
30	PIOFIE	Power indicator off interrupt enable. When set and PEX_PME_MES_DR[PIOF]=1 generates an interrupt. 1 Enable power indicator off message interrupt generation 0 Disable power indicator off message interrupt generation
31	ABPIE	Attention button pressed interrupt enable. When set and PEX_PME_MES_DR[ABP]=1 generates an interrupt. 1 Enable attention button press message interrupt generation 0 Disable attention button press message interrupt generation

16.3.3.4 PCI Express Power Management Command Register (PEX_PMCR)

The PCI Express power management command register, shown in Figure 16-10, provides software a mechanism to allow the PCI Express controller to get back to L0 link state.

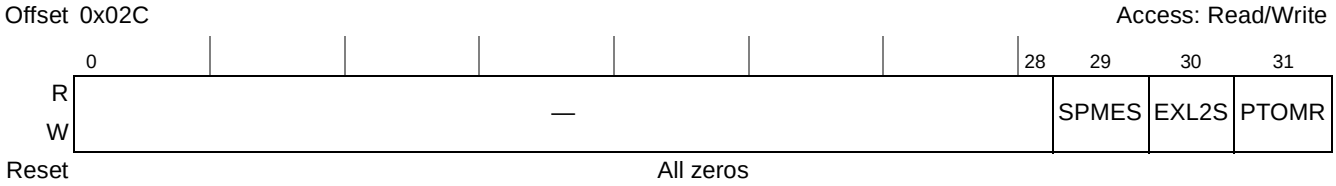


Figure 16-10. PCI Express Power Management Command Register (PEX_PMCR)

The fields of the PCI Express power management command register are described in Table 16-12.

Table 16-12. PEX_PMCR Field Descriptions

Bits	Name	Description
0–28	—	Reserved
29	SPMES	Set PME status. This sets the PME status bit and if PME is enabled (see Section 16.3.9.3, "PCI Express Power Management Status and Control Register—0x48," on page 16-70 for more information) it transmits a PM_PME message upstream. This bit should not be used when in RC mode. This bit is self-clearing.
30	EXL2S	Exit L2 state. When set exits the link state out of L2/L3 ready state in order to send new requests. The request is only made when entered_L2/L3 ready state is active. This bit is self-clearing. When the link has exited L2/L3 ready state, the status bit Exit_L2/L3 ready state is set. This bit should not be used when in EP mode.
31	PTOMR	PME_Turn_Off message request. When set broadcasts a PME turn_off message. This bit should not be used when in EP mode. This bit is self-clearing

16.3.4 PCI Express IP Block Revision Registers

16.3.4.1 IP Block Revision Register 1 (PEX_IP_BLK_REV1)

The IP block revision register 1 is shown in Figure 16-11.

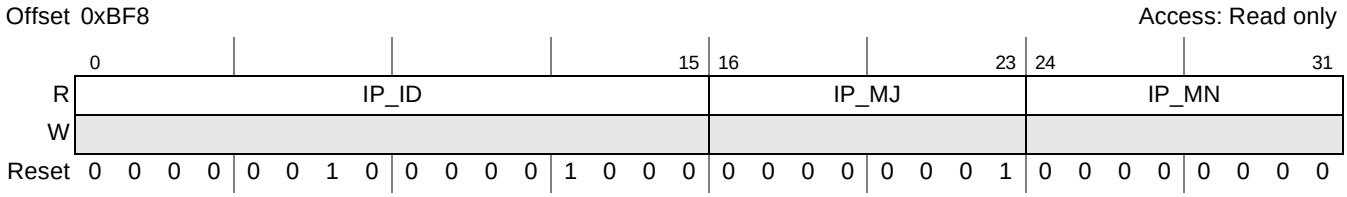


Figure 16-11. IP Block Revision Register 1

Table 16-13 contains descriptions of the fields of the IP block revision register 1.

Table 16-13. PCI Express IP Block Revision Register 1 Field Descriptions

Bits	Name	Description
0–15	IP_ID	Block ID

Table 16-13. PCI Express IP Block Revision Register 1 Field Descriptions (continued)

Bits	Name	Description
16–23	IP_MJ	Block Major Revision
24–31	IP_MN	Block Minor Revision

16.3.4.2 IP Block Revision Register 2 (PEX_IP_BLK_REV2)

The IP block revision register 2 is shown in Figure 16-12.

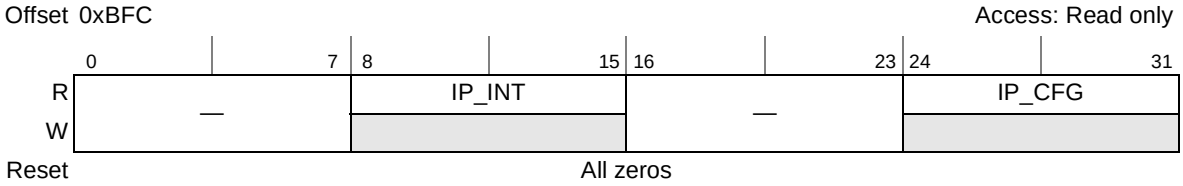


Figure 16-12. IP Block Revision Register 2

Table 16-14 contains descriptions of the fields of the IP block revision register 2.

Table 16-14. PCI Express IP Block Revision Register 2 Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–15	IP_INT	Block integration option
16–23	—	Reserved
24–31	IP_CFG	Block configuration option

16.3.5 PCI Express ATMU Registers

16.3.5.1 PCI Express Outbound ATMU Registers

The outbound address translation windows must be aligned based on the granularity selected by the size fields. Outbound window misses use the default outbound register set (outbound ATMU window 0). Overlapping outbound windows are not supported and will cause undefined behavior. Note that for RC mode, all outbound transactions post ATMU must hit either into the memory base/limit range or the prefetchable memory base/limit range defined in the PCI Express type 1 header. For EP mode, there is no such requirement.

Note that in RC mode, there is no checking on whether the translated address actually hits into the memory base/limit range. It will just pass it through as is.

Figure 16-13 shows the outbound transaction flow.

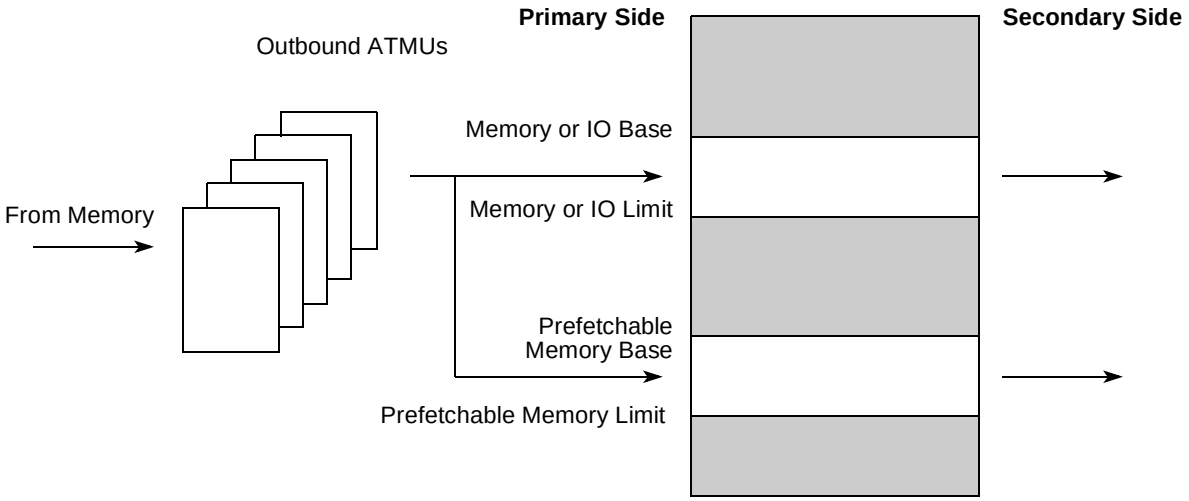


Figure 16-13. RC Outbound Transaction Flow

16.3.5.1.1 PCI Express Outbound Translation Address Registers (PEXOTAR*n*)

The PCI Express outbound translation address registers, shown in Figure 16-14, select the starting addresses in the system address space for window hits within the PCI Express outbound address translation windows. The new translated address is created by concatenating the transaction offset to this translation address.

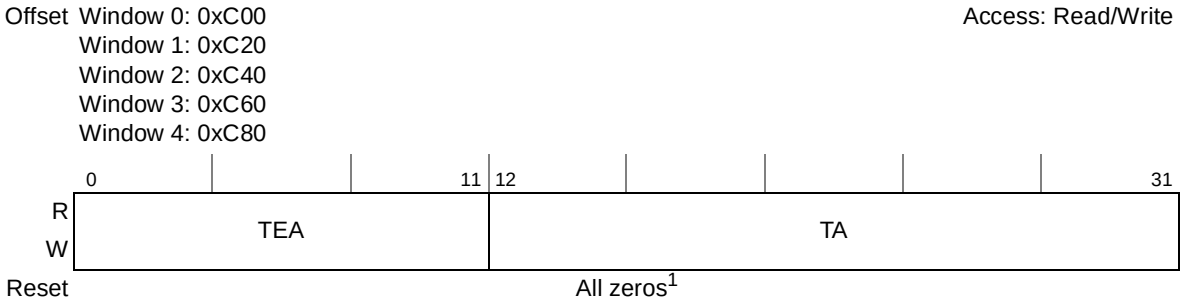


Figure 16-14. PCI Express Outbound Translation Address Registers (PEXOTAR*n*)

[1] If the device is configured to use the alternate boot vector (cfg_boot_vec = 0), the reset value for PCI controller 1 PEXOTAR3 is 0x000F_FFF0.

Table 16-15 describes the fields of the PCI Express outbound translation address registers.

Table 16-15. PEXOTAR*n* Field Descriptions

Bits	Name	Description
0–11	TEA	Translation extended address. System address which indicates the starting point of the outbound translated address. The translation address must be aligned based on the size field. Corresponds to PCI Express address bits [43:32] (bit 32 is the lsb).
12–31	TA	Translation address. System address which indicates the starting point of the outbound translated address. The translation address must be aligned based on the size field. This corresponds to PCI Express address bits [31:12].

16.3.5.1.2 PCI Express Outbound Translation Extended Address Registers (PEXOTEAR*n*)

The PCI Express outbound translation extended address registers, shown in Figure 16-15, contain the most-significant bits of a 64 bit translation address.

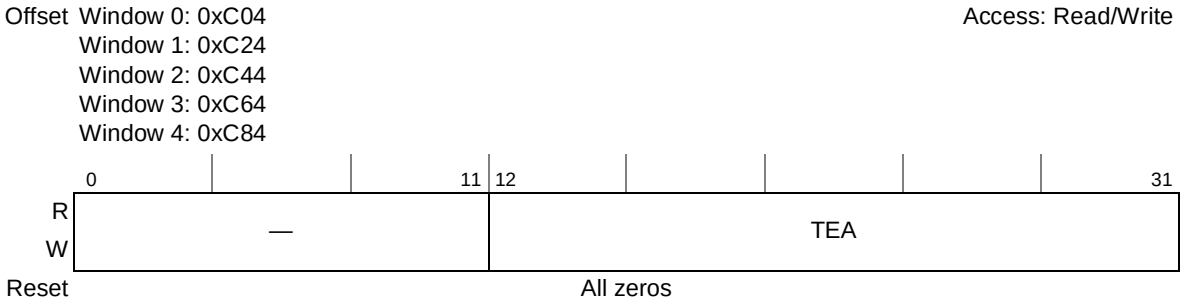


Figure 16-15. PCI Express Outbound Translation Extended Address Registers (PEXOTEAR*n*)

Table 16-16 describes the fields of the PCI Express outbound translation extended address registers.

Table 16-16. PCI Express Outbound Extended Address Translation Register *n* Field Descriptions

Bits	Name	Description
0–11	—	Reserved
12–31	TEA	Translation extended address. System address which indicates the starting point of the outbound translated address. The translation address must be aligned based on the size field. Corresponds to PCI Express address bits [63:44].

16.3.5.1.3 PCI Express Outbound Window Base Address Registers (PEXOWBAR*n*)

The PCI Express outbound window base address registers, shown in Figure 16-16, select the base address for the windows which are translated to the external address space. Addresses for outbound transactions are compared to these windows. If such a transaction does not fall within one of these spaces the transaction is forwarded through a default register set.

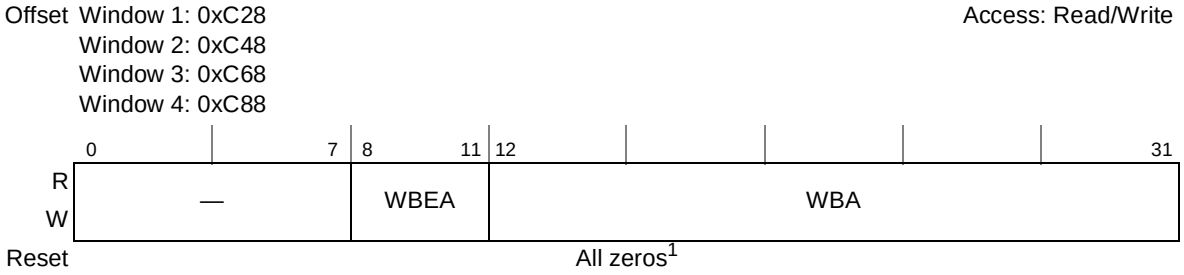


Figure 16-16. PCI Express Outbound Window Base Address Registers (PEXOWBAR*n*)

[1] If the device is configured to use the alternate boot vector (cfg_boot_vec = 0), the reset value for PCI controller 1 PEXOWBAR3 is 0x000F_FF00.

Table 16-17 describes the fields of the PCI Express outbound window base address registers.

Table 16-17. PCI Express Outbound Window Base Address Register *n* Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–11	WBEA	Window base extended address. Source address which is the starting point for the outbound translation window. The window must be aligned based on the size selected in the window size bits. Correspond to internal platform address bits [0:3]. (where 0 is the msb of the internal platform address)
12–31	WBA	Window base address. Source address which is the starting point for the outbound translation window. The window must be aligned based on the size selected in the window size bits. This corresponds to internal platform address bits [4:23].

16.3.5.1.4 PCI Express Outbound Window Attributes Registers (PEXOWAR*n*)

The PCI Express outbound window attributes registers, shown in Figure 16-17 and Figure 16-18, define the window sizes to translate and other attributes for the translations. 64 Gbytes is the largest window size allowed. Figure 16-17 shows the outbound window attributes register 0 (PEXOWAR0).

Offset 0xC10 Access: Mixed

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EN	—			ROE	NS	—		TC			—	RTT				WTT				—							OWS					
W																																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	1	1

Figure 16-17. PCI Express Outbound Window Attributes Register 0 (PEXOWAR0)

Figure 16-18 shows the PCI Express outbound window attributes registers 1–4 (PEXOWAR*n*).

Offset Window 1: 0xC30
Window 2: 0xC50
Window 3: 0xC70
Window 4: 0xC90
Access: Mixed

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EN	—			ROE	NS	—		TC			—	RTT				WTT				—							OWS					
W																																
Reset[1]	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	1	1

Figure 16-18. PCI Express Outbound Window Attributes Registers 1–4 (PEXOWAR*n*)

[1] If the device is configured to use the alternate boot vector (cfg_boot_vec = 0), the reset value for PCI controller 1 PEXOWAR3 is 0x8004_400F. If the device is not configured to use the alternate boot vector (cfg_boot_vec = 1), the reset value for PCI controller 1 PEXOWAR3 is 0x0000_0000.

Table 16-18 describes the fields of the PCI Express outbound window attributes registers.

Table 16-18. PEXOWAR*n* Field Descriptions

Bits	Name	Description
0	EN	Enable. This bit enables this address translation window. For the default window, this bit is read-only and always hardwired to 1. 0 Disable outbound translation window 1 Enable outbound translation window
1–2	—	Reserved
3	ROE	Relaxed ordering enable. This bit when set and the PCI Express device control register[Enable Relaxed] bit is set enables the Relaxed Ordering bit for the packet. This bit only applies to memory transactions. 0 Default ordering 1 Relaxed ordering
4	NS	No snoop enable. This bit when set and the PCI Express device control register[Enable No Snoop] bit is set enables the no snoop bit for the packet. This bit only applies to memory transactions. 0 Snoopable 1 No snoop
5–7	—	Reserved
8–10	TC	Traffic class. This field indicates the traffic class of the outbound packet. This field only applies to memory transaction. All other transaction types should set the TC field to 0. 000 TC0 001 TC1 010 TC2 011 TC3 100 TC4 101 TC5 110 TC6 111 TC7 **Note:** Traffic class settings are passed through to the PCI Express link, but no specific actions are taken in the device based on traffic class.
11	—	Reserved
12–15	RTT	Read transaction type. Read transaction type to run on the PCI Express link 0000 Reserved 0000 Reserved 0010 Configuration read. Supported only when in RC mode and size of less than or equal to 4 bytes and not crossing 4-byte address boundary. 0100 Memory read ... Reserved 1000 IO read. Supported only when in RC mode and size of less than or equal to 4 bytes and not crossing 4-byte address boundary. ... Reserved 1111 Reserved

Table 16-18. PEXOWAR*n* Field Descriptions (continued)

Bits	Name	Description
16–19	WTT	Write transaction type. Write transaction type to run on the PCI Express link. 0000 Reserved 0001 Reserved 0010 Configuration write. Supported only when in RC mode and size of less than or equal to 4 bytes and not crossing 4-byte address boundary. Note that inbound write transactions on one PCI express port must not translate to outbound configuration write transactions on another PCI Express port. 0100 Memory write 0101 Message write. Only support 4-byte size access on a 4-byte address boundary. ... Reserved 1000 IO Write. Supported only when in RC mode and size of less than or equal to 4 bytes and not crossing 4-byte address boundary. Note that inbound write transactions on one PCI express port must not translate to outbound I/O write transactions on another PCI Express port. ... Reserved 1111 Reserved
20–25	—	Reserved
26–31	OWS	Outbound window size. Outbound translation window size N which is the encoded $2^{(N + 1)}$-byte window size. The smallest window size is 4 Kbytes. Note that for the default window (window 0), the outbound window size may be programmed less than the 64-Gbyte maximum. However, accesses that miss all other windows and hit outside the default window is aliased to the default window. 000000Reserved ... 0010114-Kbyte window size 0011008-Kbyte window size ... 0111114-Gbyte window size 1000008-Gbyte window size 10000116-Gbyte window size 10001032-Gbyte window size 10001164-Gbyte window size 100100Reserved ... 111111Reserved

16.3.5.2 PCI Express Inbound ATMU Registers

There are differences between RC and EP implementations of inbound ATMU registers as described in the following sections.

16.3.5.2.1 EP Inbound ATMU Implementation

All base address registers (BARs) reside in the PCI Express type 0 configuration header space which is accessible through PEX_CONFIG_ADDR/PEX_CONFIG_DATA mechanism. Note that host software must program these BAR using configuration type 0 cycles. There are 4 inbound BARs.

- Default inbound window BAR0 at configuration address 0x10 (32-bit). Also known as PEXCSRBAR. This is a fixed 1-Mbyte window used for inbound memory transactions that access memory-mapped registers.
- Inbound window BAR1 at configuration address 0x14 (32-bit)
- Inbound window BAR2 at configuration address 0x18-0x1c (64-bit)

- Inbound window BAR3 at configuration address 0x20-0x24 (64-bit)

The PCI Express controller does not implement a shadow of the inbound BARs in the memory-mapped register set. However, when there is a hit to the BAR(s), the PCI Express controller uses the corresponding translation and attribute registers in the memory-mapped register set for the translation. If the transaction hits multiple BARs, then the lowest-numbered BAR is used.

16.3.5.2.2 RC Inbound ATMU Implementation

In RC mode, the PEXIWBAR[1–3] registers reside outside of the type 1 header; PEXIWBAR0 is the only inbound BAR that resides in the Type 1 header (at offset 0x10).

If the transaction hits any window, the translation is performed and then the transaction is sent to memory. If there is no hit to any one of the BARs, then an UR completion is returned for non-posted transactions. All posted transactions with no BAR hit are ignored.

Figure 16-19 shows the inbound transaction flow in RC mode.

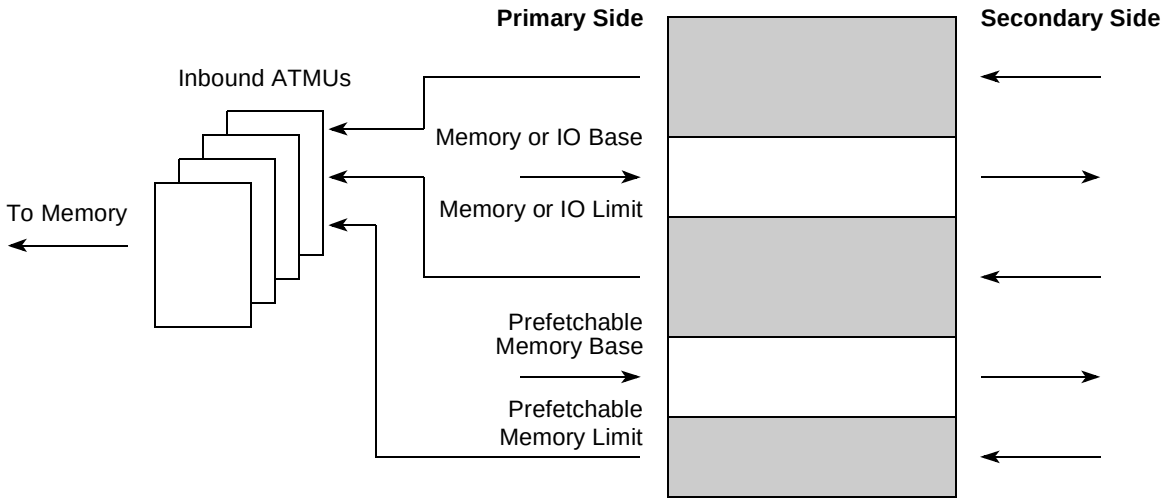


Figure 16-19. RC Inbound Transaction Flow

16.3.5.2.3 PCI Express Inbound Translation Address Registers (PEXITAR*n*)

The PCI Express inbound translation address registers, shown in Figure 16-20, contain the translated internal platform address to be used. Note that PEXITAR0 does not exist in the memory-mapped space; it is a fixed 1-Mbyte translation to the internal configuration (CCSRBAR) space.

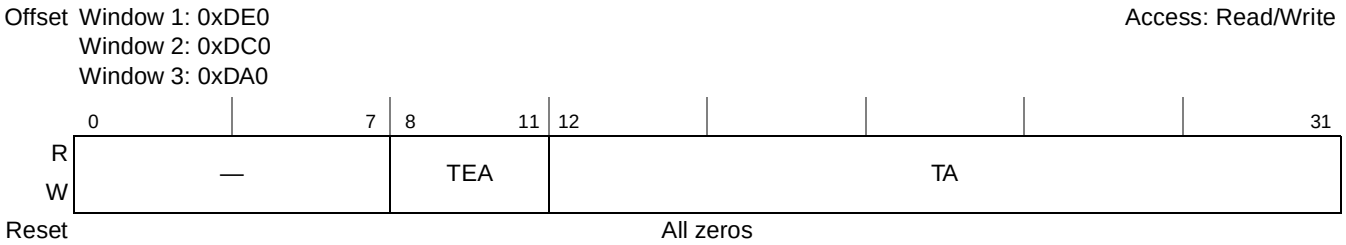


Figure 16-20. PCI Express Inbound Translation Address Registers (PEXITAR*n*)

Table 16-19 describes the fields of the PCI Express inbound translation address registers.

Table 16-19. PCI Express Inbound Translation Address Registers Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–11	TEA	Translation extended address. Target address which indicates the starting point of the inbound translated address. The translation address must be aligned based on the size field. Corresponds to internal platform address bits [0:3] where bit 0 is the msb of the internal platform address.
12–31	TA	Translation address. Target address which indicates the starting point of the inbound translated address. The translation address must be aligned based on the size field. This corresponds to internal platform address bits [4:23].

16.3.5.2.4 PCI Express Inbound Window Base Address Registers (PEXIWBAR*n*)

The PCI Express inbound window base address registers, shown in Figure 16-21, select the base address for the windows which are translated to an alternate target address space. In root complex (RC) mode, addresses for inbound transactions are compared to these windows. In RC mode, PEXIWBAR0 is located in the PCI Express type 1 configuration header space and PEXIWBAR[1–3] registers are implemented as described in this section. In endpoint (EP) mode, these registers are not implemented in the memory-mapped space. Reading these registers in EP mode returns all zeros and writing to these offsets has no consequences. All base address registers in EP are located in the PCI Express type 0 configuration header space. Note that PEXIWBAR1 only supports 32-bit PCI Express address space.

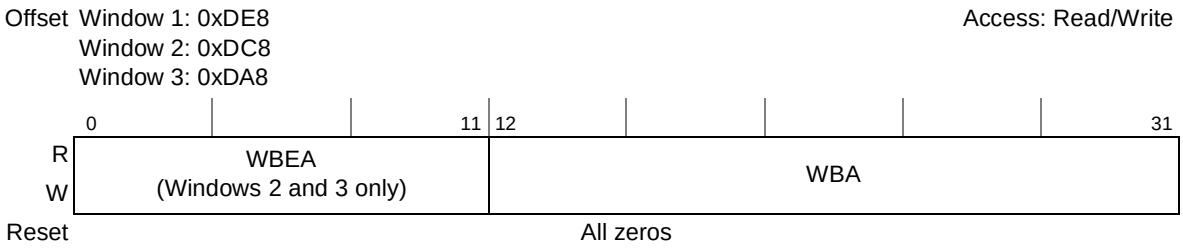


Figure 16-21. PCI Express Inbound Window Base Address Registers (PEXIWBAR*n*)

Table 16-20 describes the fields of the PCI Express inbound window base address registers.

Table 16-20. PCI Express Inbound Window Base Address Register Field Descriptions

Bits	Name	Description
0–11	WBEA	Window base extended address. This field corresponds to PCI Express address bits [43:32]. Note that the extended address is supported for windows 2 and 3 only; for PEXIWBAR1, these bits are reserved and must be 0.
12–31	WBA	Window base address. Source address which is the starting point for the inbound translation window. The window must be aligned based on the size selected in the window size bits. This corresponds to PCI Express address bits [31:12].

16.3.5.2.5 PCI Express Inbound Window Base Extended Address Registers (PEXIWBEAR*n*)

The PCI Express inbound window base extended address registers, shown in Figure 16-22, contain the most-significant bits of a 64 bit base address.

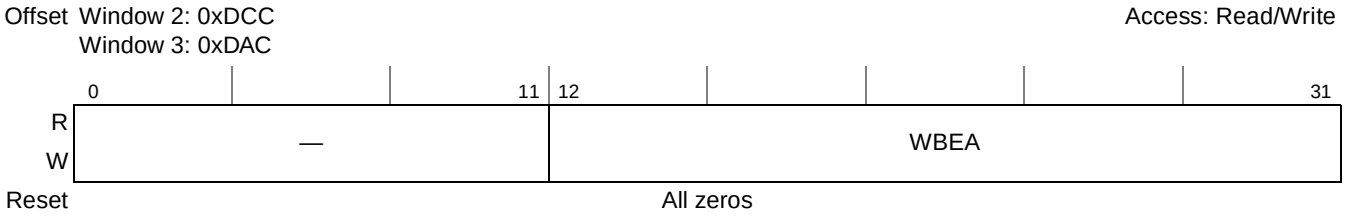


Figure 16-22. PCI Express Inbound Window Base Extended Address Registers (PEXIWBEAR*n*)

Table 16-21 describes the fields of the PCI Express inbound window base extended address registers.

Table 16-21. PCI Express Inbound Window Base Extended Address Register Field Descriptions

Bits	Name	Description
0–11	—	Reserved
12–31	WBEA	Window base extended address. This field corresponds to PCI Express address bits [63:44]

16.3.5.2.6 PCI Express Inbound Window Attributes Registers (PEXIWAR*n*)

The PCI Express inbound window attributes registers, shown in Figure 16-23, define the window sizes to translate and other attributes for the translations. 64 Gbytes is the largest window size allowed.

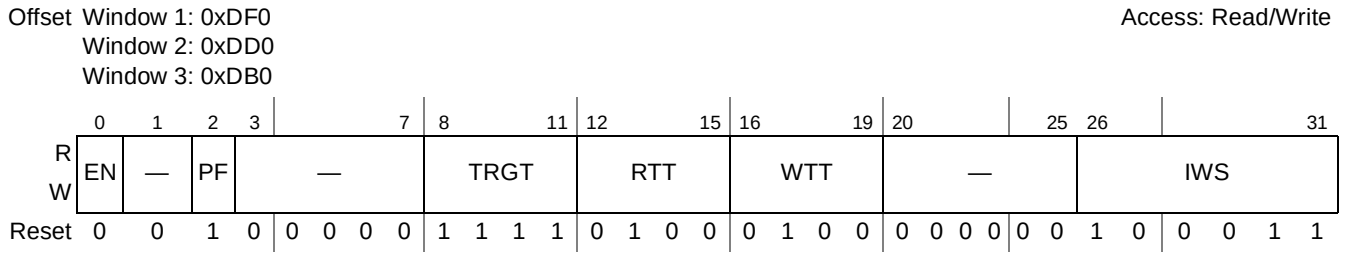


Figure 16-23. PCI Express Inbound Window Attributes Registers (PEXIWAR*n*)

Table 16-22 describes the fields of the PCI Express inbound window attributes registers.

Table 16-22. PCI Express Inbound Window Attributes Registers Field Descriptions

Bits	Name	Description
0	EN	Enable. This bit controls the enabling/disabling of the translation window. 0 Disable inbound window translation 1 Enable inbound window translation
1	—	Reserved

Table 16-22. PCI Express Inbound Window Attributes Registers Field Descriptions (continued)

Bits	Name	Description
2	PF	Prefetchable. This bit indicates that the address space is prefetchable. This bit corresponds to the prefetchable bit in the BAR in the PCI Express type 0 header. This bit drives the BAR's prefetchable bit in EP mode. 0 Not prefetchable 1 Prefetchable
3–7	—	Reserved
8–11	TRGT	Target interface. If this field is set to anything other than local memory space, the attributes for the transaction must be assigned in a corresponding outbound window at the target. 0000 PCI Express 1 — PCI Express controller 1 should not use this encoding 0001 PCI Express 2 — PCI Express controller 2 should not use this encoding 0010-1011Reserved 1100 Serial RapidIO 1101-1110 Reserved 1111 Local memory space **Note:** Inbound write transactions on one PCI Express port must not translate to outbound configuration or I/O write transactions on another PCI Express port.
12–15	RTT	Read transaction type. Read transaction type to send to target interface. If the transaction is not going to local memory space 0000 Reserved ... 0100 Read 0101 Reserved 0110 Reserved 0111 Reserved 1000 Reserved ... 1111 Reserved If the transaction is going to local memory space 0000 Reserved ... 0100 Read, don't snoop local processor 0101 Read, snoop local processor 0110 Reserved 0111 Reserved 1000 Reserved ... 1111 Reserved

Table 16-22. PCI Express Inbound Window Attributes Registers Field Descriptions (continued)

Bits	Name	Description
16–19	WTT	Write transaction type. Write transaction type to send to target interface. If the transaction is not going to local memory space 0000 Reserved ... 0100 Write 0101 Reserved 0110 Reserved 0111 Reserved 1000 Reserved ... 1111 Reserved If the transaction is going to local memory space 0000 Reserved ... 0100 Write, don't snoop local processor 0101 Write, snoop local processor 0110 Reserved 0111 Reserved 1000 Reserved ... 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 size is 4 Kbytes. For EP mode, this field directly controls the size of the BARs. 000000 Reserved ... 001010 Reserved 001011 4-Kbyte window size 001100 8-Kbyte window size ... 011111 4-Gbyte window size 100000 8-Gbyte window size 100001 16-Gbyte window size 100010 32-Gbyte window size 100011 64-Gbyte window size 100100 Reserved ... 111111 Reserved

16.3.6 PCI Express Error Management Registers

16.3.6.1 PCI Express Error Detect Register (PEX_ERR_DR)

The PCI Express error detect register, shown in Figure 16-24, contains error status bits that are detected by hardware.The detected error bits are write-1-to-clear type registers. Reading from these registers occurs

normally; however, write operations can clear but not set bits. A bit is cleared whenever the register is written, and the data in the corresponding bit location is a 1. For example, to clear bit 6 and not affect any other bits in the register, the value 0b0200_0000 is written to the register. When an error is detected the appropriate error bit is set. Subsequent errors sets the appropriate error bits in the error detection registers, but only the first error for a particular unit have any relevant information captured. The interrupt enable bits are used to allow or block the error reporting to the interrupt mechanism while the disable bits are used to prevent or allow the setting of the status bits.

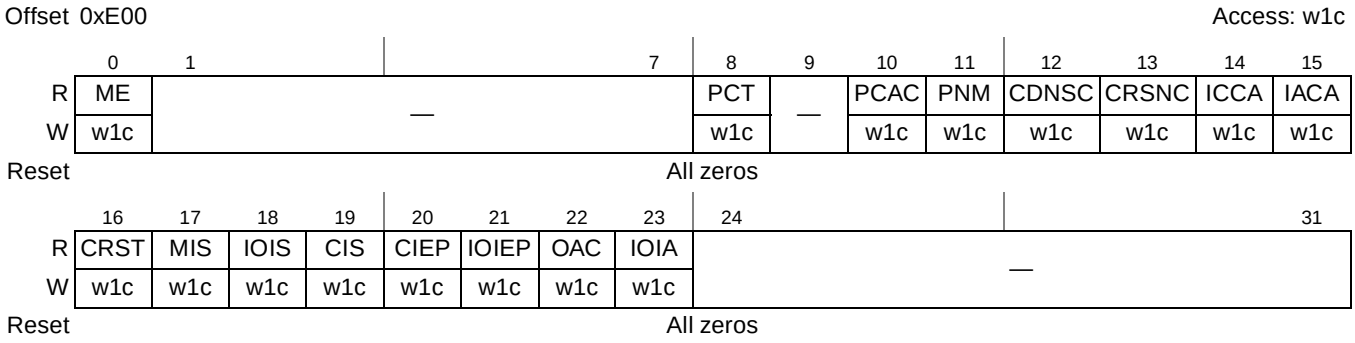


Figure 16-24. PCI Express Error Detect Register (PEX_ERR_DR)

Table 16-23 describes the fields of the PCI Express error detect register.

Table 16-23. PCI Express Error Detect Register Field Descriptions

Bits	Name	Description
0	ME	Multiple errors. Detecting multiple errors of the same type. An error is considered as multiple error when its detect bit is set and the same error is occurring again. Note that this bit does not track the ordering of when the error occurs. 1 Multiple errors were detected. 0 Multiple errors were not detected.
1–7	—	Reserved
8	PCT	PCI Express completion time-out. A completion time-out condition was detected for a non-posted, outbound PCI Express transaction. An error response is sent back to the requestor. Note that a completion timeout counter only starts when the non-posted request was able to send to the link partner. 1 A completion time-out on the PCI Express link was detected. Note that a completion timeout error is a fatal error. If a completion timeout error is detected, the system has become unstable. Hot reset is recommended to restore stability of the system. 0 No completion time-out on the PCI Express link detected.
9	—	Reserved
10	PCAC	PCI Express completer abort (CA) completion. A completion with CA status was received. 1 A completion with CA status was detected. 0 No completion with CA status detected.
11	PNM	PCI Express no map. Detect an inbound transaction that was not mapped to any inbound windows. In RC mode, a completion without data (Cpl) packet with a UR completion status is sent back to the requester and this bit is set. For EP mode, a Cpl packet with a UR completion status is sent back to the requester but does not set this bit. 1 A no-map transaction was detected in RC mode. 0 No no-map transaction detected.

Table 16-23. PCI Express Error Detect Register Field Descriptions (continued)

Bits	Name	Description
12	CDNSC	Completion with data not successful. A completion with data packet was received with a non successful status (that is, UR, CA or CRS status). 1 Completion with data non successful packet was detected. 0 No completion with data non successful packet detected.
13	CRSNC	CRS non configuration. A completion was detected for a non configuration cycle and with CRS status. 1 CRS non configuration packet was detected. 0 No CRS non configuration packet detected.
14	ICCA	Invalid PEX_CONFIG_ADDR/PEX_CONFIG_DATA configuration access. Access to an illegal configuration space from PEX_CONFIG_ADDR/PEX_CONFIG_DATA was detected. 1 Invalid CONFIG_ADDR/PEX_CONFIG_DATA access detected 0 No invalid PEX_CONFIG_ADDR/PEX_CONFIG_DATA access detected
15	IACA	Invalid ATMU configuration access. Access to an illegal configuration space from an ATMU window was detected. 1 Invalid ATMU configuration access was detected 0 No invalid ATMU configuration access detected
16	CRST	CRS thresholded. An outbound configuration transaction was retried and thresholded due to a CRS completion status. An error response is sent back to the requestor. See Section 16.3.2.4, “PCI Express Configuration Retry Timeout Register (PEX_CONF_RTY_TOR),” for more information. 1 A CRS threshold condition was detected for an outbound configuration transaction 0 No CRS threshold condition detected
17	MIS	Message invalid size. An outbound message transaction that is greater than 4 bytes or crosses a 4-byte boundary was detected. See Section 16.4.1.9.1, “Outbound ATMU Message Generation,” for more information. 1 An invalid size outbound message transaction was detected 0 No invalid size outbound message transaction detected
18	IOIS	I/O invalid size. An outbound I/O transaction that is greater than 4 bytes or crosses a 4-byte boundary was detected. 1 an invalid size outbound I/O transaction was detected 0 no invalid size outbound I/O transaction detected
19	CIS	Configuration invalid size. An outbound configuration transaction that is greater than 4 bytes or crosses a 4-byte boundary was detected. 1 An invalid size outbound configuration transaction was detected 0 No invalid size outbound configuration transaction detected
20	CIEP	Configuration invalid EP. An outbound ATMU configuration transaction request was seen when in EP mode. 1 An outbound configuration transaction while in EP was detected 0 No outbound configuration transaction in EP detected
21	IOIEP	I/O invalid EP. An outbound I/O transaction request was seen when in EP mode. 1 An outbound I/O transaction while in EP was detected 0 No outbound I/O transaction in EP detected
22	OAC	Outbound ATMU crossing. An outbound crossing ATMU transaction was detected. 1 An outbound transaction that hits in one window and crosses overing it was detected 0 No outbound ATMU crossing condition detected

Table 16-23. PCI Express Error Detect Register Field Descriptions (continued)

Bits	Name	Description
23	IOIA	I/O invalid address. An outbound I/O transaction with a translated address of greater than 4 Gbytes was detected. 1 A greater than 4-Gbyte I/O address was detected 0 No greater than 4-Gbyte I/O address detected
24–31	—	Reserved

16.3.6.2 PCI Express Error Interrupt Enable Register (PEX_ERR_EN)

The PCI Express error interrupt enable register, shown in Figure 16-25, allows interrupts to be generated when the corresponding PCI Express error detect register bits are set.

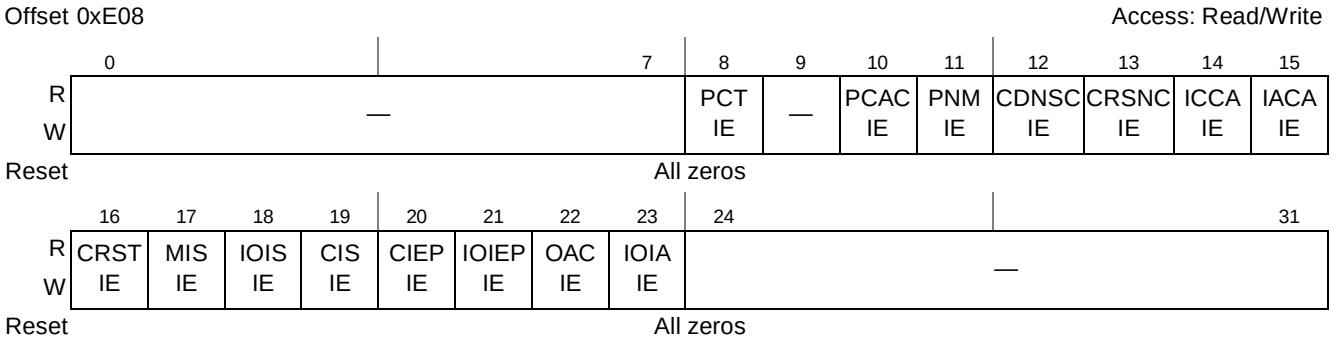


Figure 16-25. PCI Express Error Interrupt Enable Register (PEX_ERR_EN)

Table 16-24 describes the fields of the PCI Express error interrupt enable register.

Table 16-24. PCI Express Error Interrupt Enable Register Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8	PCTIE	PCI Express completion time-out interrupt enable. When set and PEX_ERR_DR[PCT]=1 generates an interrupt. 1 Enable PCI Express completion time-out interrupt generation 0 Disable PCI Express completion time-out interrupt generation
9	—	Reserved
10	PCACIE	PCI Express CA completion interrupt enable. When set and PEX_ERR_DR[PCAC]=1 generates an interrupt. 1 Enable completion with CA status interrupt generation 0 Disable completion with CA status interrupt generation
11	PNMIE	PCI Express no map interrupt enable. When set and PEX_ERR_DR[PNM]=1 generates an interrupt. 1 Enable no map PCI Express packet interrupt generation 0 Disable no map PCI Express packet interrupt generation
12	CDNSCIE	Completion with data not successful interrupt enable. When this bit is set and PEX_ERR_DR[CDNSC] = 1 generates an interrupt. 1 Enable completion with data non successful interrupt generation 0 Disable completion with data non successful interrupt generation

Table 16-24. PCI Express Error Interrupt Enable Register Field Descriptions (continued)

Bits	Name	Description
13	CRSNCIE	CRS non configuration interrupt enable. When this bit is set and PEX_ERR_DR[CRSNC] = 1 generates an interrupt. 1 Enable CRS non configuration interrupt generation 0 Disable CRS non configuration interrupt generation
14	ICCAIE	Invalid PEX_CONFIG_ADDR/PEX_CONFIG_DATA configuration access interrupt enable. When set and PEX_ERR_DR[ICCA]=1 generates an interrupt. 1 Enable invalid PEX_CONFIG_ADDR/PEX_CONFIG_DATA access interrupt generation 0 Disable invalid PEX_CONFIG_ADDR/PEX_CONFIG_DATA access interrupt generation.
15	IACAIE	Invalid ATMU configuration access. When set and PEX_ERR_DR[IACA]=1 generates an interrupt. 1 Enable invalid ATMU configuration access interrupt generation 0 Disable invalid ATMU configuration access interrupt generation
16	CRSTIE	CRS thresholded interrupt enable. When set and PEX_ERR_DR[CRST]=1 generates an interrupt. 1 Enable CRS threshold interrupt generation 0 Disable CRS threshold interrupt generation
17	MISIE	Message invalid size interrupt enable. When set and PEX_ERR_DR[MIS]=1 generates an interrupt. 1 Enable invalid outbound message size interrupt generation 0 Disable invalid outbound message size interrupt generation
18	IOISIE	I/O invalid size interrupt enable. When set and PEX_ERR_DR[IOIS]=1 generates an interrupt. 1 Enable invalid outbound I/O size interrupt generation 0 Disable invalid outbound I/O size interrupt generation
19	CISIE	Configuration invalid size interrupt enable. When set and PEX_ERR_DR[CIS]=1 generates an interrupt. 1 Enable invalid outbound configuration size interrupt generation 0 Disable invalid outbound configuration size interrupt generation
20	CIEPIE	Configuration invalid EP interrupt enable. When set and PEX_ERR_DR[CIEP]=1 generates an interrupt. 1 Enable outbound configuration transaction while in EP mode interrupt generation 0 Disable outbound configuration transaction in EP mode interrupt generation
21	IOIEPIE	I/O invalid EP interrupt enable. When set and PEX_ERR_DR[IOIEP]=1 generates an interrupt. 1 Enable outbound I/O transaction EP mode interrupt generation 0 Disable outbound I/O transaction EP mode interrupt generation
22	OACIE	Outbound ATMU crossing interrupt enable. When set and PEX_ERR_DR[OAC]=1 generates an interrupt. 1 Enable outbound crossing ATMU interrupt generation 0 Disable outbound crossing ATMU interrupt generation
23	IOIAIE	I/O address invalid enable. When set and PEX_ERR_DR[IOIA]=1 generates an interrupt. 1 Enable greater than 4G I/O address interrupt generation 0 Disable greater than 4G I/O address interrupt generation
24–31	—	Reserved

16.3.6.3 PCI Express Error Disable Register (PEX_ERR_DISR)

The PCI Express error disable register, shown in Figure 16-26, controls the setting of the PCI Express error detect register's bits.

Offset 0xE10 Access: Read/Write

	0	1–7	8	9	10	11	12	13	14	15
R	MED	—	PCTD	—	PCACD	PNMD	CDNSCD	CRSNCD	ICCAD	IACAD
W										
Reset	All zeros									

	16	17	18	19	20	21	22	23	24–31
R	CRSTD	MISD	IOISD	CISD	CIEPD	IOIEPD	OACD	IOIAD	—
W									
Reset	All zeros								

Figure 16-26. PCI Express Error Disable Register (PEX_ERR_DISR)

Table 16-25 describes the fields of the PCI Express error disable register.

Table 16-25. PCI Express Error Disable Register Field Descriptions

Bits	Name	Description
0	MED	Multiple errors disable. When set disables the setting of PEX_ERR_DR[ME] bit. 1 Disable multiple errors detection 0 Enable multiple errors detection
1–7	—	Reserved
8	PCTD	PCI Express completion time-out disable. When set disables the setting of PEX_ERR_DR[PCT] bit. 1 Disable PCI Express completion time-out detection 0 Enable PCI Express completion time-out detection
9	—	Reserved
10	PCACD	PCI Express CA completion disable. When set disables the setting of PEX_ERR_DR[PCAC] bit. 1 Disable completion with CA status detection 0 Enable completion with CA status detection
11	PNMD	PCI Express no map disable. When set disables the setting of PEX_ERR_DR[PNM] bit. 1 Disable no map PCI Express packet detection 0 Enable no map PCI Express packet detection
12	CDNSCD	Completion with data not successful disable. When set disables the setting of PEX_ERR_DR[CDNSC] bit. 1 Disable completion with data not successful detection 0 Enable completion with data not successful detection
13	CRSNCD	CRS non configuration disable. When set disables the setting of PEX_ERR_DR[CRSNC] bit. 1 Disable CRS non configuration detection 0 Enable CRS non configuration detection
14	ICCAD	Invalid PEX_CONFIG_ADDR/PEX_CONFIG_DATA configuration access disable. When set disables the setting of PEX_ERR_DR[ICCA] bit. 1 Disable invalid PEX_CONFIG_ADDR/PEX_CONFIG_DATA access detection 0 Enable invalid PEX_CONFIG_ADDR/PEX_CONFIG_DATA access detection

Table 16-25. PCI Express Error Disable Register Field Descriptions (continued)

Bits	Name	Description
15	IACAD	Invalid ATMU configuration access. When set disables the setting of PEX_ERR_DR[IACA] bit. 1 Disable invalid ATMU configuration access detection 0 Enable invalid ATMU configuration access detection
16	CRSTD	CRS thresholded disable. When set disables the setting of PEX_ERR_DR[CRST] bit. 1 Disable CRS threshold detection 0 Enable CRS threshold detection
17	MISD	Message invalid size disable. When set disables the setting of PEX_ERR_DR[MIS] bit. 1 Disable invalid outbound message size detection 0 Enable invalid outbound message size detection
18	IOISD	I/O invalid size disable. When set disables the setting of PEX_ERR_DR[IOIS] bit. 1 Disable invalid outbound I/O size detection 0 Enable invalid outbound I/O size detection
19	CISD	Configuration invalid size disable. When set disables the setting of PEX_ERR_DR[CIS] bit. 1 Disable invalid outbound configuration size detection 0 Enable invalid outbound configuration size detection
20	CIEPD	Configuration invalid EP disable. When set disables the setting of PEX_ERR_DR[CIEP] bit. 1 Disable outbound configuration transaction EP mode detection 0 Enable outbound configuration transaction EP mode detection
21	IOIEPD	I/O invalid EP disable. When set disables the setting of PEX_ERR_DR[IOEP] bit. 1 Disable outbound I/O transaction EP mode detection 0 Enable outbound I/O transaction EP mode detection
22	OACD	Outbound ATMU crossing disable. When set disables the setting of PEX_ERR_DR[OAC] bit. 1 Disable outbound crossing ATMU detection 0 Enable outbound crossing ATMU detection
23	IOIAD	I/O invalid address disable. When set disables the setting of PEX_ERR_DR[IOIA] bit. 1 Disable greater than 4G I/O address detection 0 Enable greater than 4G I/O address detection
24–31	—	Reserved

16.3.6.4 PCI Express Error Capture Status Register (PEX_ERR_CAP_STAT)

The PCI Express error capture status register, shown in Figure 16-27, allows vital error information to be captured when an error occurs. Note that no further error capturing is performed until the ECV bit is cleared.

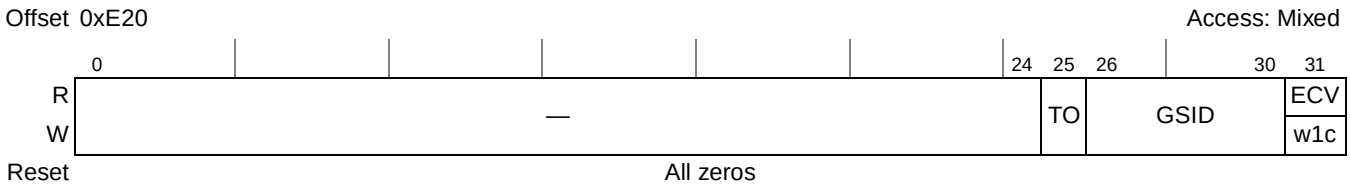


Figure 16-27. PCI Express Error Capture Status Register (PEX_ERR_CAP_STAT)

Table 16-26 describes the fields of the PCI Express error capture status register.

Table 16-26. PCI Express Error Capture Status Register Field Descriptions

Bits	Name	Description
0–24	—	Reserved
25	TO	Transaction originator. This field Indicates whether the originator of the transaction is from PEX_CONFIG_ADDR/PEX_CONFIG_DATA. 1 Transaction originated from PEX_CONFIG_ADDR/PEX_CONFIG_DATA. 0 Transaction not originated from PEX_CONFIG_ADDR/PEX_CONFIG_DATA.
26–30	GSID	Global source ID. This field indicates the internal platform global source ID that the error transaction originates. This field only applies to non PEX_CONFIG_ADDR/PEX_CONFIG_DATA transactions. 00000 PCI Express 1 00001 PCI Express 2/RapidIO 00010 Reserved 01010 Boot sequencer 10000 Core 0 instruction/data 10010 Core 1 instruction/data 10101 DMA 11000 eTSEC1 11001 eTSEC2 11010 eTSEC3 11011 eTSEC4 11100 RapidIO message/doorbell/port write with responses and reads All other settings reserved.
31	ECV	Error capture valid. This bit indicates that the capture registers 0-3 contain valid info. This bit when set indicates that the captured registers contain valid capturing information. No new capturing is done unless this bit is cleared by writing a 1 to it.

16.3.6.5 PCI Express Error Capture Register 0 (PEX_ERR_CAP_R0)

Together with the other PCI Express error capture registers, PEX_ERR_CAP_R0 allows vital error information to be captured when an error occurs. Different error information is reported depending on whether the error source is from an outbound transaction from an internal source or from an inbound transaction from an external source; the source of the captured error is reflected in PEX_ERR_CAP_STAT[GSID]. Note that after the initial error is captured, no further capturing is performed until the PEX_ERR_CAP_STAT[ECV] bit is clear.

16.3.6.5.1 PEX_ERR_CAP_R0—Outbound Case

PEX_ERR_CAP_R0 for the case when the error is caused by an outbound transaction from an internal source (that is, PEX_ERR_CAP_STAT[GSID] ≠ 0h02), is shown in Figure 16-28.

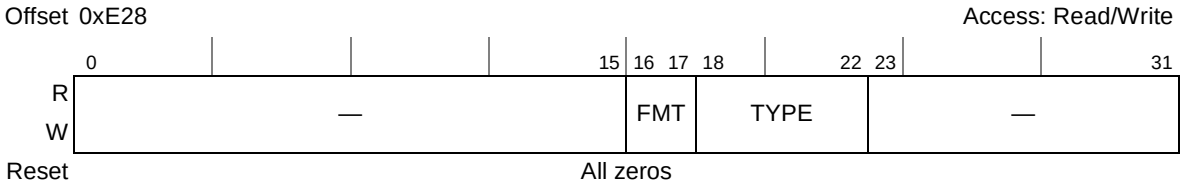


Figure 16-28. PCI Express Error Capture Register 0 (PEX_ERR_CAP_R0) Internal Source Outbound Transaction

Table 16-27 describes the fields of the PCI Express error capture register 0 for the case when the error is caused by an outbound transaction from an internal source.

Table 16-27. PCI Express Error Capture Register 0 Field Descriptions Internal Source Outbound Transaction

Bits	Name	Description
0–15	—	Reserved
16-17	FMT	PCI Express format. This field indicates the PCI Express packet format. See PCI Express Spec 1.0a for more information on 3 or 4 DW (4-byte) header format.
18–22	TYPE	PCI Express type. This field indicates the PCI express packet type. See PCI Express Spec 1.0a for more information on 3 or 4 DW (4-byte) header format.
23–31	—	Reserved

16.3.6.5.2 PEX_ERR_CAP_R0—Inbound Case

PEX_ERR_CAP_R0 for the case when the error is caused by an inbound transaction from an external source (that is, PEX_ERR_CAP_STAT[GSID] = 0h02 for controller 1), is shown in Figure 16-29.

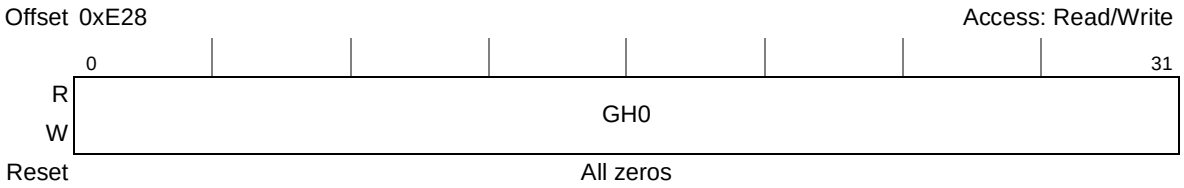


Figure 16-29. PCI Express Error Capture Register 0 (PEX_ERR_CAP_R0) External Source, Inbound Transaction

Table 16-28 describes the fields of PEX_ERR_CAP_R0 for the case when the error is caused by an inbound transaction from an external source.

Table 16-28. PCI Express Error Capture Register 0 Field Descriptions External Source, Inbound Transaction

Bits	Name	Description
0–31	GH0	PCI Express first DW (4-byte) header. This field contains the first DW (4-byte) of the captured PCI Express packet header. 27–31 TYPE 25–26 FMT 20–24 Reserved 17–19 TC 16 Reserved 14–15 LENGTH[9:8] 12–13 Reserved 10–11 ATTR 9 EP 8 TD 0–7 LENGTH[7:0]

16.3.6.6 PCI Express Error Capture Register 1 (PEX_ERR_CAP_R1)

Together with the other PCI Express error capture registers, PEX_ERR_CAP_R1 allows vital error information to be captured when an error occurs. Different error information is reported depending on whether the error source is from an outbound transaction from an internal source or from an inbound transaction from an external source; the source of the captured error is reflected in PEX_ERR_CAP_STAT[GSID]. Note that after the initial error is captured, no further capturing is performed until the PEX_ERR_CAP_STAT[ECV] bit is clear.

16.3.6.6.1 PEX_ERR_CAP_R1—Outbound Case

PEX_ERR_CAP_R1 for the case when the error is caused by an outbound transaction from an internal source (that is, PEX_ERR_CAP_STAT[GSID] ≠ 0h02), is shown in Figure 16-30.

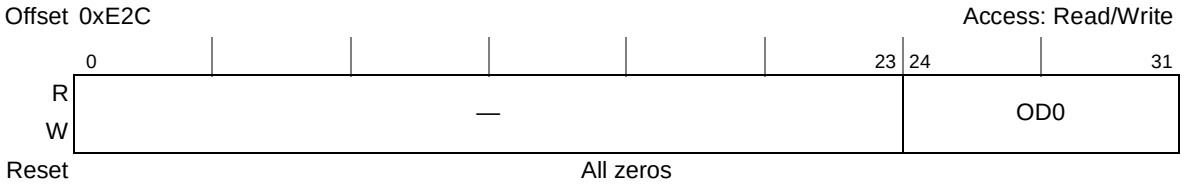


Figure 16-30. PCI Express Error Capture Register 1 (PEX_ERR_CAP_R1)
Internal Source, Outbound Transaction

Table 16-29 describes the fields of PEX_ERR_CAP_R1 for the case when the error is caused by an outbound transaction from an internal source.

Table 16-29. PCI Express Error Capture Register 1 Field Descriptions
Internal Source, Outbound Transaction

Bits	Name	Description
0–23	—	Reserved
24–31	OD0	Internal platform transaction information. Reserved for factory debug.

16.3.6.6.2 PEX_ERR_CAP_R1—Inbound Case

PEX_ERR_CAP_R1 for the case when the error is caused by an inbound transaction from an external source (that is, PEX_ERR_CAP_STAT[GSID] = 0h02 for controller 1), is shown in Figure 16-31.

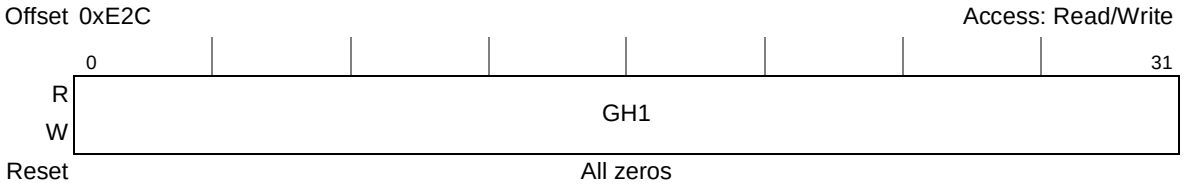


Figure 16-31. PCI Express Error Capture Register 1 (PEX_ERR_CAP_R1)
External Source, Inbound Transaction

Table 16-30 describes the fields of PEX_ERR_CAP_R1 for the case when the FMT and TYPE subfields in PEX_ERR_CAP_R0 (see Table 16-28) indicate the error was caused by an inbound completion transaction.

Table 16-30. PCI Express Error Capture Register 1 Field Descriptions External Source, Inbound CompletionTransaction

Bits	Name	Description
0–31	GH1	PEX second DW (4-byte) header. This field contains the second DW (4-byte) of the captured PCI Express packet header. 24–31 Comp ID[15:8] 16–23 Comp ID[7:0] 12–15 Byte Count[11:8] 11 BCM 8–10 Comp Status 0–7 Byte Count[7:0]

Table 16-31 describes the fields of PEX_ERR_CAP_R1 for the case when the FMT and TYPE subfields in PEX_ERR_CAP_R0 (see Table 16-28) indicate the error was caused by an inbound memory request transaction.

Table 16-31. PCI Express Error Capture Register 1 Field Descriptions External Source, Inbound Memory Request Transaction

Bits	Name	Description
0–31	GH1	PEX second DW (4-byte) header. This field contains the second DW (4-byte) of the captured PCI Express packet header. 24–31 Requester ID[15:8] 16–23 Requester ID[7:0] 8–15 Tag[7:0] 4–7 First DW BE[3:0] 0–3 Last DW BE[3:0]

16.3.6.7 PCI Express Error Capture Register 2 (PEX_ERR_CAP_R2)

Together with the other PCI Express error capture registers, PEX_ERR_CAP_R2 allows vital error information to be captured when an error occurs. Different error information is reported depending on whether the error source is from an outbound transaction from an internal source or from an inbound transaction from an external source; the source of the captured error is reflected in PEX_ERR_CAP_STAT[GSID]. Note that after the initial error is captured, no further capturing is performed until the PEX_ERR_CAP_STAT[ECV] bit is clear.

16.3.6.7.1 PEX_ERR_CAP_R2—Outbound Case

PEX_ERR_CAP_R2 for the case when the error is caused by an outbound transaction from an internal source (that is, PEX_ERR_CAP_STAT[GSID] ≠ 0h02), is shown in Figure 16-32.

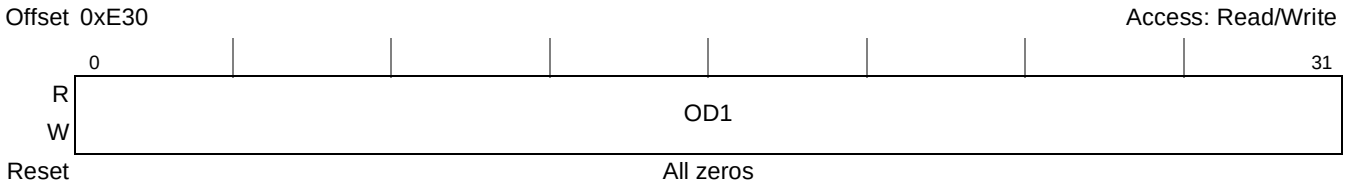


Figure 16-32. PCI Express Error Capture Register 2 (PEX_ERR_CAP_R2) Internal Source Outbound Transaction

Table 16-32 describes the fields of PEX_ERR_CAP_R2 for the case when the error is caused by an outbound transaction from an internal source.

Table 16-32. PCI Express Error Capture Register 2 Field Descriptions Internal Source Outbound Transaction

Bit	Name	Description
0–31	OD1	Internal platform transaction information. Reserved for factory debug.

16.3.6.7.2 PEX_ERR_CAP_R2—Inbound Case

PEX_ERR_CAP_R2 for the case when the error is caused by an inbound transaction from an external source (that is, PEX_ERR_CAP_STAT[GSID] = 0h02 for controller 1), is shown in Figure 16-33.

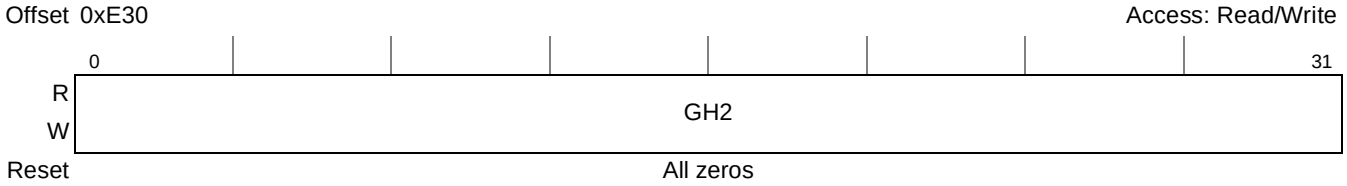


Figure 16-33. PCI Express Error Capture Register 2 (PEX_ERR_CAP_R2) External Source Inbound Transaction

Table 16-33 describes the fields of PEX_ERR_CAP_R2 for the case when the FMT and TYPE subfields in PEX_ERR_CAP_R0 (see Table 16-28) indicate the error was caused by an inbound completion transaction.

Table 16-33. PCI Express Error Capture Register 2 Field Descriptions External Source Inbound Completion Transaction

Bits	Name	Description
0–31	GH2	PEX third DW (4-byte) header. This field contains the third DW (4-byte) of the captured PCI Express packet header. 24–31 Req ID[15:8] 16–23 Req ID[7:0] 8–15 Tag[7:0] 1–7 Lower Address[6:0] 0 Reserved

Table 16-34 describes the fields of PEX_ERR_CAP_R2 for the case when the FMT and TYPE subfields in PEX_ERR_CAP_R0 (see Table 16-28) indicate the error was caused by an inbound memory request transaction. Note that PEX_ERR_CAP_R2 captures the 32-bit address for a 3 DW memory request header or the upper half of the 64-bit address for a 4 DW memory request header; the lower half of the 64-bit address for a 4 DW memory request header is captured in PEX_ERR_CAP_R3.

Table 16-34. PCI Express Error Capture Register 2 Field Descriptions
External Source, Inbound Memory Request Transaction

Bits	Name	Description	
		3 DW Header	4 DW Header
0–31	GH2	PEX third DW (4-byte) header. This field contains the third DW (4-byte) of the captured PCI Express packet header.	
		24–31 Address[31:24] 16–23 Address[23:16] 8–15 Address[15:8] 6–7 Reserved 0-5 Address[7:2]	24–31 Address[63:56] 16–23 Address[55:48] 8–15 Address[47:40] 0-7 Address[39:32]

16.3.6.8 PCI Express Error Capture Register 3 (PEX_ERR_CAP_R3)

Together with the other PCI Express error capture registers, PEX_ERR_CAP_R3 allows vital error information to be captured when an error occurs. Different error information is reported depending on whether the error source is from an outbound transaction from an internal source or from an inbound transaction from an external source; the source of the captured error is reflected in PEX_ERR_CAP_STAT[GSID]. Note that after the initial error is captured, no further capturing is performed until the PEX_ERR_CAP_STAT[ECV] bit is clear.

16.3.6.8.1 PEX_ERR_CAP_R3—Outbound Case

PEX_ERR_CAP_R3 for the case when the error is caused by an outbound transaction from an internal source (that is, PEX_ERR_CAP_STAT[GSID] ≠ 0h02), is shown in Figure 16-34.

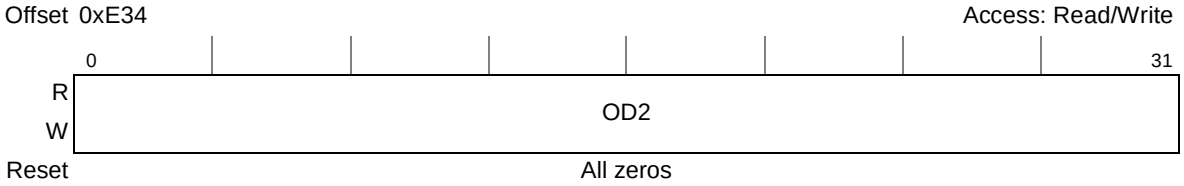


Figure 16-34. PCI Express Error Capture Register 3 (PEX_ERR_CAP_R3)
Internal Source, Outbound Transaction

Table 16-35 describes the fields of PEX_ERR_CAP_R3 for the case when the error is caused by an outbound transaction from an internal source.

Table 16-35. PCI Express Error Capture Register 3 Field Descriptions Internal Source Outbound Transaction

Bits	Name	Description
0–31	OD2	Internal platform transaction information. Reserved for factory debug.

16.3.6.8.2 PEX_ERR_CAP_R3—Inbound Case

PEX_ERR_CAP_R3 for the case when the error is caused by an inbound transaction from an external source (that is, PEX_ERR_CAP_STAT[GSID] = 0h02 for controller 1), is shown in Figure 16-35.

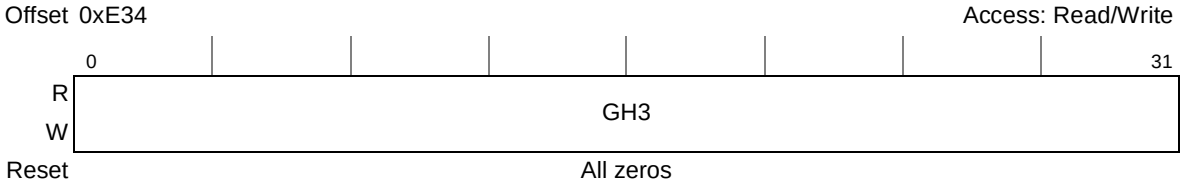


Figure 16-35. PCI Express Error Capture Register 3 (PEX_ERR_CAP_R3) External Source Inbound Transaction

Table 16-36 describes the fields of PEX_ERR_CAP_R3 for the case when the FMT and TYPE subfields in PEX_ERR_CAP_R0 (see Table 16-28) indicate the error was caused by an inbound memory request transaction. Note that PEX_ERR_CAP_R3 captures the lower half of the 64-bit address for a 4 DW memory request header; the upper half of the 64-bit address for a 4 DW memory request header or the 32-bit address for a 3 DW memory request header is captured in PEX_ERR_CAP_R2.

Table 16-36. PEX Error Capture Register 3 Field Descriptions External Source Inbound Memory Request Transaction

Bits	Name	Description
0–31	GH3	PEX fourth DW (4-byte) header. This field contains the fourth DW (4-byte) of the captured PCI Express packet header. 24–31 Address[31:24] 16–23 Address[23:16] 8–15 Address[15:8] 6–7 Reserved 0-5 Address[7:2]

16.3.7 PCI Express Configuration Space Access

There are two methods of accessing the PCI Express configuration header:

- PCI Express outbound ATMU window
- PCI Express configuration access registers (PEX_CONFIG_ADDR/PEX_CONFIG_DATA)

16.3.7.1 RC Configuration Register Access

To access internal configuration space, software must rely on the PCI Express configuration access register (PEX_CONFIG_ADDR/ PEX_CONFIG_DATA) mechanism. To access external configuration space, software can either use configuration access registers or the outbound ATMU mechanism. For the configuration access register method, a value must be written to the PEX_CONFIG_ADDR register that specifies the PCI Express bus, the device on that bus, the function within the device, and the configuration register in that device that should be accessed. The PCI Express controller's bus number is obtained from the PCI Express configuration header (type 1). Then either a write or a read to the PEX_CONFIG_DATA register triggers the actual write or read cycle to the configuration space. Note that accesses to the little-endian PCI Express configuration space must be properly formatted. See Section 16.4.1.2.1, "Byte Order for Configuration Transactions," for more information.

Note that external configuration transactions should not be attempted until the link has successfully trained. Software can poll the LTSSM state status register (PEX_LTSSM_STAT) to check the status of link training before issuing external configuration requests.

16.3.7.1.1 PCI Express Configuration Access Register Mechanism

There are two types of configuration transactions (Type 0 and Type 1) needed to support hierarchical bridges.

- If the bus number, and device number equal to the PCI Express controller's bus number and device number, and the function number is zero, then an internal PCI Express configuration cycle access is performed.
- If the bus number does not equal the PCI Express controller's bus number, but does equal the secondary bus number (from the type 1 header) and the device number is 0, then a Type 0 configuration transaction is sent to the PCI Express link.
- If the bus number does not equal the PCI Express controller's bus number, and does not equal the secondary bus number (from the type 1 header), and the bus number is less than or equal to the subordinate bus number (from the type 1 header), then a Type 1 configuration transaction is sent to the PCI Express link.
- If none of the above conditions occur, then the PCI Express controller returns all 1s for reads and ignores writes.

16.3.7.1.2 Outbound ATMU Configuration Mechanism (RC-Only)

Software can also program one of the outbound ATMU windows to perform a configuration access. This is accomplished by programming the ReadTType or WriteTType field of the desired PEXOWAR to 0x2. Software must only issue 4-byte or less access to the ATMU configuration window and the access cannot cross a 4-byte boundary. The bus number, device number, function number, register, and extended register number sent are decoded from the outbound translated PCI Express address.

- bus number[7:0] = PCI Express address[27:20]
- device number[4:0] = PCI Express address[19:15]
- function number[2:0] = PCI Express address[14:12]
- extended register number[3:0] = PCI Express address[11:8]

- register number[5:0] = PCI Express address[7:2]

A Type 0 configuration cycle is sent to the link if the bus number equals the secondary bus number (from the type 1 header) and device number is 0. A Type 1 configuration cycle is sent to the link if bus number does not equal primary bus and secondary bus numbers and it is less than or equal to the subordinate bus number (from the type 1 header). For all other cases, the PCI Express controller squashes the write and read results in a response with error returned.

Note that the PCI Express controller does not support access to its internal configuration registers using the outbound ATMU mechanism. That is, the outbound ATMU mechanism must not be used to program the internal registers.

16.3.7.2 EP Configuration Register Access

When the PCI Express controller is configured as an EP device it responds to remote host generated configuration cycles. This is indicated by decoding the configuration command along with type 0 access in the packet. A remote host can access all of the PCI Express configuration area except the PCI Express Controller Internal CSR registers in the extended PCI Express configuration space at offsets 0x400–0x6FF. The PCI Express Controller Internal CSR registers are not accessible by inbound PCI Express configuration transactions. Attempts to access these registers return all zeros.

While in EP mode, the PCI Express controller does not support generating configuration accesses as a master. All accesses to PEX_CONFIG_ADDR/PEX_CONFIG_DATA cause the device to access the internal configuration registers regardless of the bus number or device number programmed in the PEX_CONFIG_ADDR register. There is no configuration mechanism supported in EP mode using the ATMU window. If the outbound ATMU window is configured to issue a configuration transaction, all posted transactions hitting this window are ignored and all non-posted transactions get a response with an error.

16.3.8 PCI Compatible Configuration Headers

The first 64 bytes of the 256-byte PCI compatible configuration space consists of a predefined header that every PCI device must support. The first 16 bytes of the predefined header are defined the same for all PCI Express devices. These common registers are shown in Figure 16-36.

Reserved

				Address Offset (Hex)
Device ID		Vendor ID		00
Status		Command		04
Class Code			Revision ID	08
BIST	Header Type	Latency Timer	Cache Line Size	0C

Figure 16-36. PCI Express PCI-Compatible Configuration Header Common Registers

The remaining 48 bytes of the header may have differing layouts depending on the function of the device. There are two header types applicable to PCI Express. Type 0 headers are typically used by endpoints; Type 1 headers are used by root complexes and switches/bridges.

16.3.8.1 Common PCI Compatible Configuration Header Registers

This section details the registers that are common to both type 0 and type 1 configuration headers.

16.3.8.1.1 PCI Express Vendor ID Register—Offset 0x00

The vendor ID register, shown in Figure 16-37, is used to identify the manufacturer of the device.

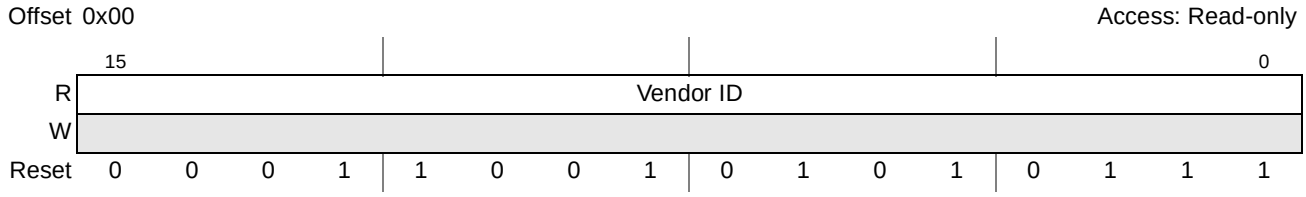


Figure 16-37. PCI Express Vendor ID Register

Table 16-37 describes the vendor ID register fields.

Table 16-37. PCI Express Vendor ID Register Field Description

Bits	Name	Description
15–0	Vendor ID	0x1957 (Freescale)

16.3.8.1.2 PCI Express Device ID Register—Offset 0x02

The device ID register, shown in Figure 16-38, is used to identify the device.

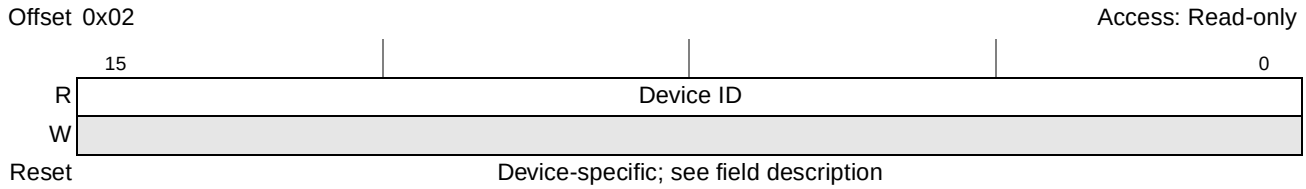


Figure 16-38. PCI Express Device ID Register

Table 16-38 describes the device ID register fields.

Table 16-38. PCI Express Device ID Register Field Description

Bits	Name	Description
15–0	Device ID	0x7010 MPC8641 0x7011MPC8641D

16.3.8.1.3 PCI Express Command Register—Offset 0x04

The command register, shown in Figure 16-39, provides control over the ability to generate and respond to PCI Express cycles.

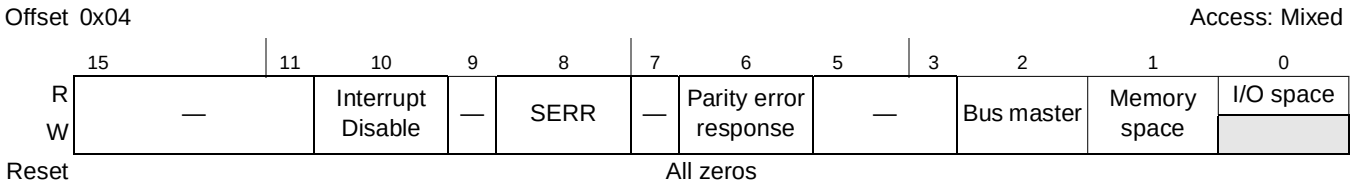


Figure 16-39. PCI Express Command Register

Table 16-39 describes the bits of the command register.

Table 16-39. PCI Express Command Register Field Descriptions

Bits	Name	Description
15–11	—	Reserved
10	Interrupt Disable	Controls the ability to generate INTx interrupt messages. 0 Enables INTx interrupt messages 1 Disables INTx interrupt messages Any INTx emulation interrupts already asserted by this device must be deasserted when this bit is set.
9	—	Reserved
8	SERR	Controls the reporting of fatal and non-fatal errors detected by the device to the root complex. 0 Disables reporting 1 Enables reporting **Note:** The error control and status bits in the command and status registers control PCI-compatible error reporting. PCI Express advanced error reporting is controlled by the PCI Express device control register described in Section 16.3.9.8, “PCI Express Device Control Register—0x54,” and the advance error reporting capability structure described in sections 16.3.10.1 through 16.3.10.12.
7	—	Reserved
6	Parity error response	Controls whether this PCI Express controller responds to parity errors. 0 Parity errors are ignored and normal operation continues. 1 Parity errors cause the appropriate bit in the PCI Express status register to be set. However, note that errors are reported based on the values set in the PCI Express error enable and detection registers. **Note:** The error control and status bits in the command and status registers control PCI-compatible error reporting. PCI Express advanced error reporting is controlled by the PCI Express device control register described in Section 16.3.9.8, “PCI Express Device Control Register—0x54,” and the advance error reporting capability structure described in sections 16.3.10.1 through 16.3.10.12.
5–3	—	Reserved
2	Bus master	Indicates whether this PCI Express device is configured as a master. 0 Disables the ability to generate PCI Express accesses 1 Enables this PCI Express controller to behave as a PCI Express bus master EP mode: Clearing this bit prevent the device from issuing any memory or I/O transactions. Because MSI interrupts are effectively memory writes, clearing this bit also disables the ability of the device to issue MSI interrupts. RC mode: Clearing this bit disables the ability of the device to forward memory transactions upstream. This causes any inbound memory transaction to be treated as an unsupported request.

Table 16-39. PCI Express Command Register Field Descriptions (continued)

Bits	Name	Description
1	Memory space	Controls whether this PCI Express device (as a target) responds to memory accesses. 0 This PCI Express device does not respond to PCI Express memory space accesses. 1 This PCI Express device responds to PCI Express memory space accesses. EP mode: Clearing this bit prevents the device from accepting any memory transaction. RC mode: This bit is ignored. It does not affect outbound memory transaction
0	I/O space	I/O space. 0 This PCI Express device (as a target) does not respond to PCI Express I/O space accesses. 1 This PCI Express device (as a target) does respond to PCI Express I/O space accesses. EP mode: Clearing this bit prevents the device from accepting any IO transaction. Note that this bit is a don't care in EP mode since the device does not support IO transaction. RC mode: This bit is ignored. It does not affect outbound IO transaction.

16.3.8.1.4 PCI Express Status Register—Offset 0x06

The status register, shown in Figure 16-40, is used to record status information for PCI Express related events.

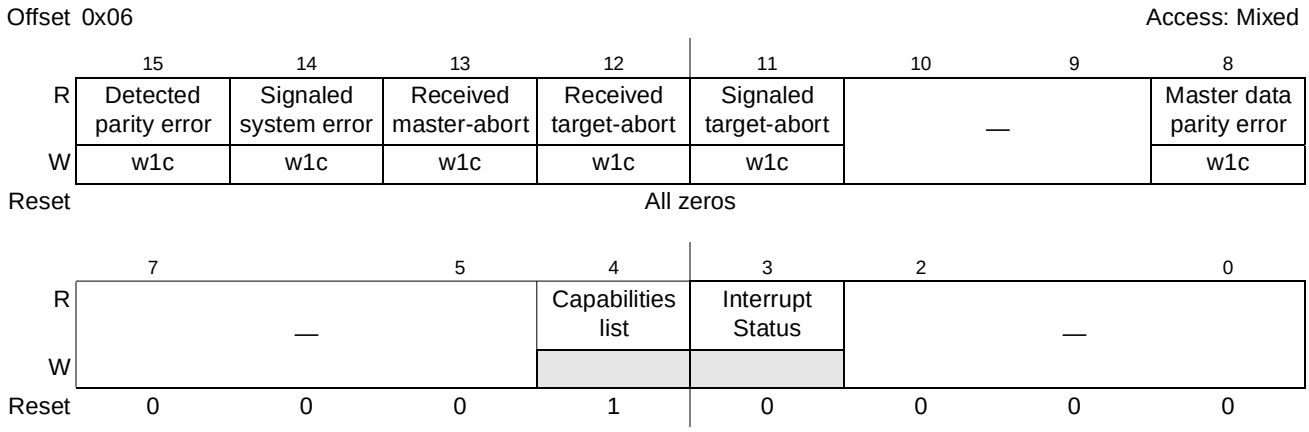


Figure 16-40. PCI Express Status Register

The definition of each bit is given in Table 16-40.

Table 16-40. PCI Express Status Register Field Descriptions

Bits	Name	Description
15	Detected parity error[1]	Set whenever a device receives a poisoned TLP regardless of the state of bit 6 in the command register.
14	Signaled system error[1]	Set whenever a device sends a ERR_FATAL or ERR_NONFATAL message and the SERR enable bit in the command register is set.
13	Received master-abort[1]	Set whenever a requestor receives a completion with unsupported request completion status.
12	Received target-abort[1]	Set whenever a device receives a completion with completer abort completion status.
11	Signaled target-abort[1]	Set whenever a device completes a request using completer abort completion status.

Table 16-40. PCI Express Status Register Field Descriptions (continued)

Bits	Name	Description
10–9	—	Reserved
8	Master data parity error detected[1]	Set by the requestor (primary side for Type1 headers) when either the requestor receives a completion marked poisoned or the requestor poisons a write request. Note that the parity error enable bit (bit 6) in the command register must be set for this bit to be set.
7–5	—	Reserved
4	Capabilities List	All PCI Express devices are required to implement the PCI Express capability structure.
3	Interrupt Status	Set when an INTx interrupt message is pending internally to the device. Note that this bit is associated with INTx messages and not Message Signaled Interrupts.
2–0	—	Reserved

1 The error control and status bits in the command and status registers control PCI-compatible error reporting. PCI Express advanced error reporting is controlled by the PCI Express device control register described in Section 16.3.9.8, "PCI Express Device Control Register—0x54," and the advance error reporting capability structure described in sections 16.3.10.1 through 16.3.10.12.

16.3.8.1.5 PCI Express Revision ID Register—Offset 0x08

The revision ID register, shown in Figure 16-41, is used to identify the revision of the device.

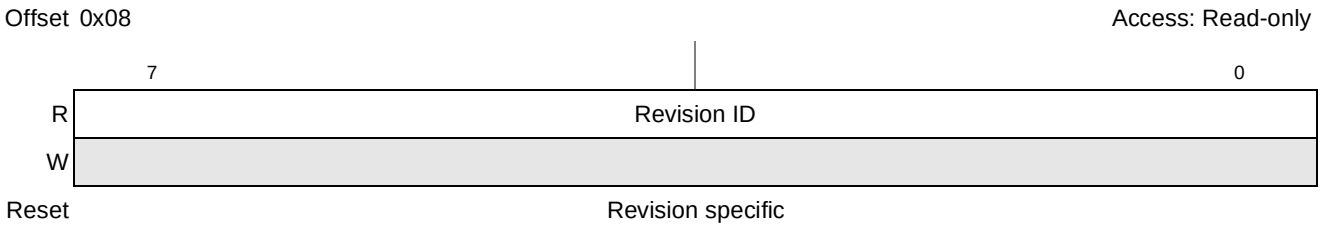


Figure 16-41. PCI Express Revision ID Register

Table 16-41 describes the revision ID register fields.

Table 16-41. PCI Express Revision ID Register Field Descriptions

Bits	Name	Description
7–0	Revision ID	Revision specific.

16.3.8.1.6 PCI Express Class Code Register—Offset 0x09

The class code register, shown in Figure 16-42, comprises three single-byte fields—base class (offset 0x0B), sub-class (offset 0x0A), and programming interface (offset 0x09)—that indicate the basic functionality of the function.

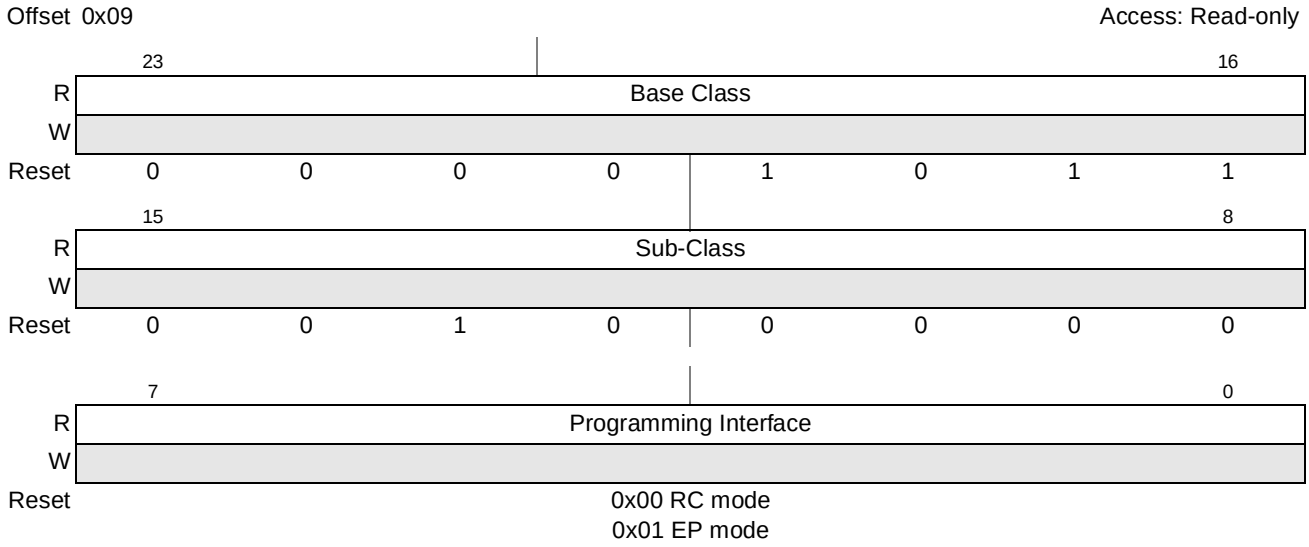


Figure 16-42. PCI Express Class Code Register

Table 16-42 describes the class code register fields.

Table 16-42. PCI Express Class Code Register Field Descriptions

Bits	Name	Description
23–16	Base Class	0x0B—Processor
15–8	Sub-Class	0x20—PowerPC
7–0	Programming Interface	0x00—RC mode 0x01—EP mode

16.3.8.1.7 PCI Express Cache Line Size Register—Offset 0x0C

The cache line size register, shown in Figure 16-43, is provided for legacy compatibility purposes (PCI 2.3); it is not used for PCI Express device functionality.

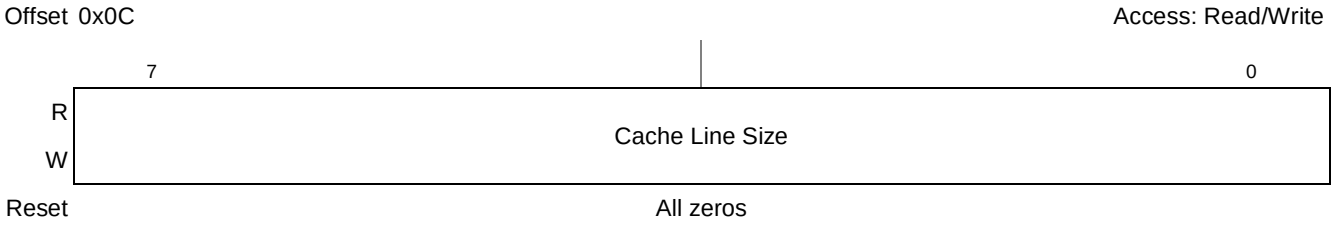


Figure 16-43. PCI Express Bus Cache Line Size Register

Table 16-43 describes the cache line size register.

Table 16-43. PCI Express Bus Cache Line Size Register Field Descriptions

Bits	Name	Description
7–0	Cache Line Size	Represents the cache line size of the processor in terms of 32-bit words (8 32-bit words = 32 bytes). Note that for PCI Express operation this register is ignored.

16.3.8.1.8 PCI Express Latency Timer Register—0x0D

The latency timer register, shown in Figure 16-44, is provided for legacy compatibility purposes (PCI 2.3); it is not used for PCI Express device functionality.

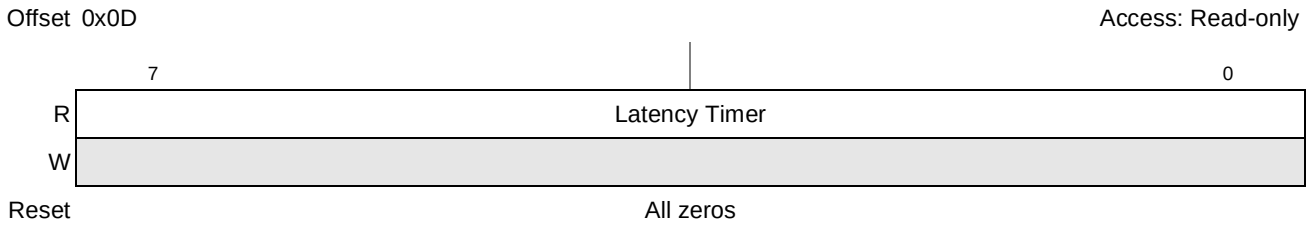


Figure 16-44. PCI Express Bus Latency Timer Register

Table 16-44 describes the PCI Express latency timer register (PLTR).

Table 16-44. PCI Express Bus Latency Timer Register Field Descriptions

Bits	Name	Description
7–0	Latency Timer	Note that for PCI Express operation this register is ignored.

16.3.8.1.9 PCI Express Header Type Register—0x0E

The PCI Express header type register, shown in Figure 16-43, is used to identify the layout of the PCI compatible header.

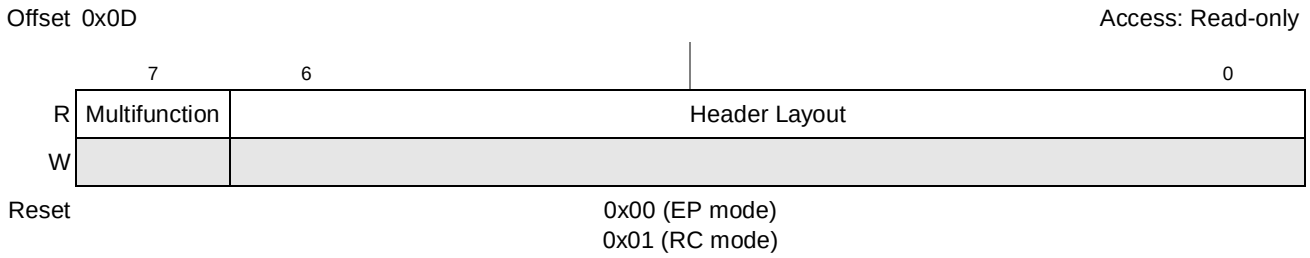


Figure 16-45. PCI Express Bus Latency Timer Register

Table 16-44 describes the PCI Express header type register.

Table 16-45. PCI Express Bus Latency Timer Register Field Descriptions

Bits	Name	Description
7	Multifunction	Identifies whether a device supports multiple functions 0 Single function device 1 Multiple function device
6–0	Header Layout	0x00 Endpoint. See Figure 16-46 for type 0 layout. 0x01 Root Complex. See Figure 16-58 for type 1 layout. All other encodings reserved.

16.3.8.1.10 PCI Express BIST Register—0x0F

The BIST register is optional and reserved on the PCI Express controller.

16.3.8.2 Type 0 Configuration Header

The type 0 header is shown in Figure 16-46.

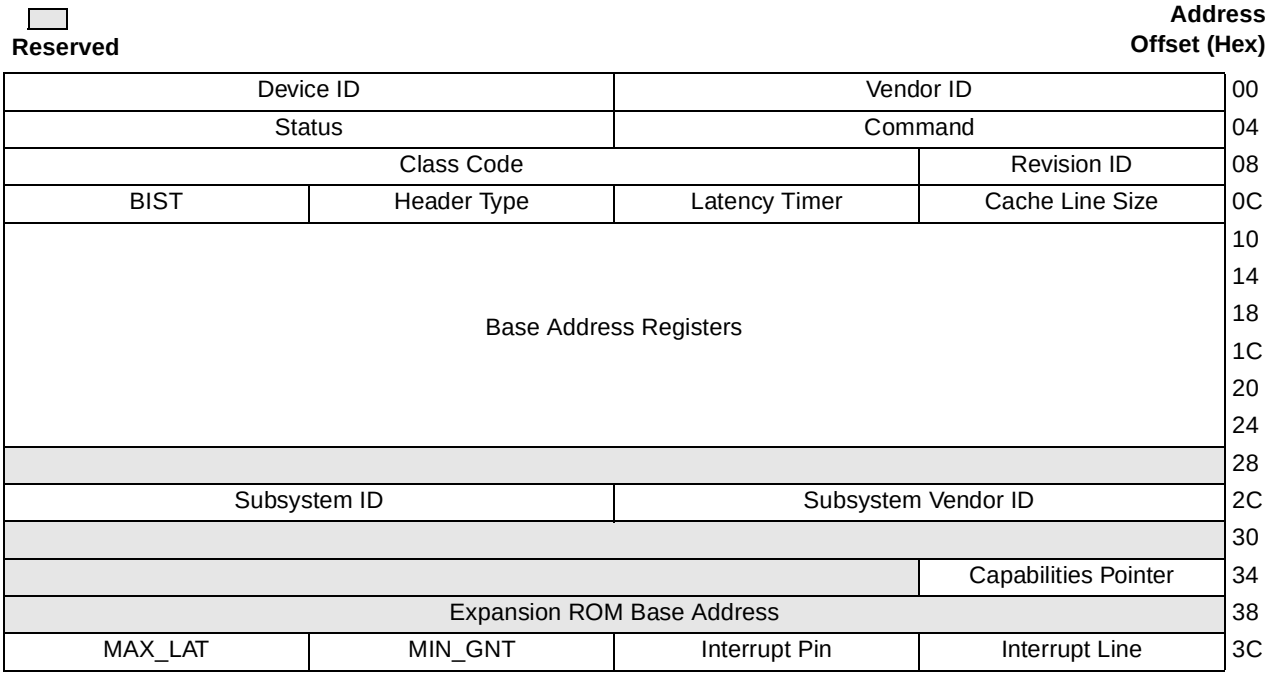


Figure 16-46. PCI Express PCI-Compatible Configuration Header—Type 0

Section 16.3.8.1, "Common PCI Compatible Configuration Header Registers," describes the registers in the first 16 bytes of the header. This section describes the registers that are unique to the type 0 header beginning at offset 0x10.

16.3.8.2.1 PCI Express Base Address Registers—0x10–0x27

The PCI Express base address registers (BARs) point to the beginning of distinct address ranges which the device should claim. In EP mode, the device supports a configuration space BAR, a 32-bit memory space

BAR, and two 64-bit memory space BARs. In RC mode, the device only supports the configuration space BAR in the header; the other memory spaces are defined by the inbound ATMUs. Refer to Section 16.3.5.2, "PCI Express Inbound ATMU Registers," for more information.

Base address register 0 at offset 0x10 is a special fixed 1-Mbyte window that is used for inbound configuration accesses. This window is called the PCI Express configuration and status register base address register (PEXCSRBAR). Note that PEXCSRBAR cannot be updated through the inbound ATMU registers. The PEXCSRBAR is shown in Figure 16-47.

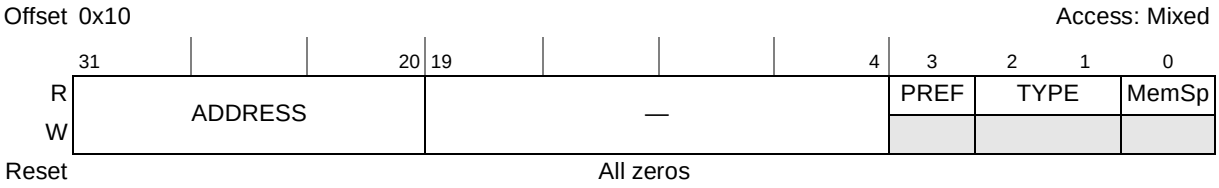


Figure 16-47. PCI Express Base Address Register 0 (PEXCSRBAR)

Table 16-46 describes the PCI Express configuration and status register base address register.

Table 16-46. PEXCSRBAR Field Descriptions

Bits	Name	Description
31–20	ADDRESS	Indicates the base address that the inbound configuration window occupies. This window is fixed at 1 Mbyte.
19–4	—	Reserved
3	PREF	Prefetchable
2–1	TYPE	Type. 00 Locate anywhere in 32-bit address space.
0	MemSp	Memory space indicator

Base address register 1 at offset 0x14 is used to define the inbound memory window in the 32-bit memory space. The 32-bit memory BAR is shown in Figure 16-48.

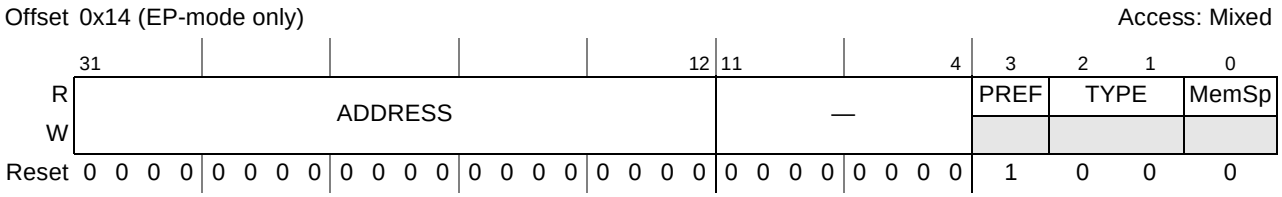


Figure 16-48. 32-Bit Memory Base Address Register (BAR1)

Table 16-47 describes the PCI Express 32-bit memory BAR fields.

Table 16-47. 32-Bit Memory Base Address Register (BAR1) Field Descriptions

Bits	Name	Description
31–12	ADDRESS	Indicates the base address where the inbound memory window begins. The number of upper bits that the device allows to be writable is selected through the inbound window size in the inbound window attributes register (PEXIWAR1).
11–4	—	Reserved. The device allows a 4 Kbyte window minimum.
3	PREF	Prefetchable. This bit is determined by PEXIWAR1[PF].
2–1	TYPE	Type. 00 Locate anywhere in 32-bit address space.
0	MemSp	Memory space indicator.

Base address register 2 at offset 0x18 and base address register 4 at offset 0x20 are used to define the lower portion of the 64-bit inbound memory windows. The 64-bit low memory BARs are shown in Figure 16-49.

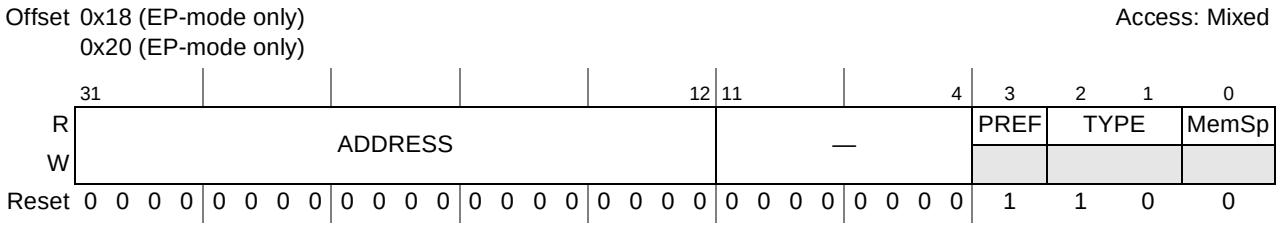


Figure 16-49. 64-Bit Low Memory Base Address Register

Table 16-48 describes the PCI Express 64-bit low memory BAR fields.

Table 16-48. 64-Bit Low Memory Base Address Register Field Descriptions

Bits	Name	Description
31–12	ADDRESS	Indicates the lower portion of the base address where the inbound memory window begins. The number of bits that the device allows to be writable is selected through the inbound window size in the inbound window attributes registers (PEXIWAR2 for offset 0x18 and PEXIWAR3 for offset 0x20).
11–4	—	Reserved. The device allows a 4 Kbyte window minimum.
3	PREF	Prefetchable. This bit is determined by PEXIWAR*n*[2].
2–1	TYPE	Type. 0b10 Locate anywhere in 64-bit address space.
0	MemSp	Memory space indicator

Base address register 3 at offset 0x1C and base address register 5 at offset 0x24 are used to define the upper portion of the 64-bit inbound memory windows. The 64-bit high memory BARs are shown in Figure 16-50.

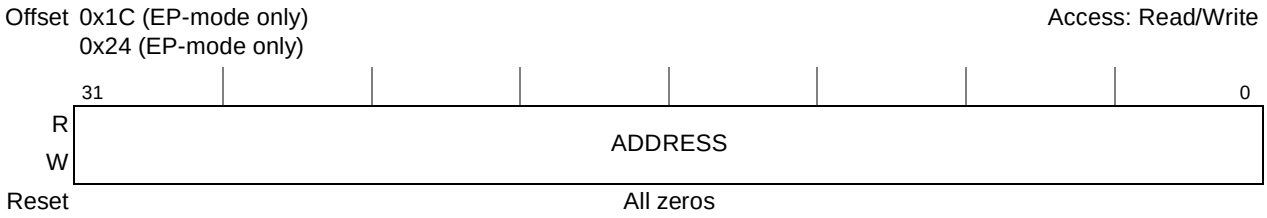


Figure 16-50. 64-Bit High Memory Base Address Register

Table 16-49 describes the PCI Express 64-bit low memory BAR fields.

Table 16-49. Bit Setting for 64-Bit High Memory Base Address Register

Bits	Name	Description
31–0	ADDRESS	Indicates the upper portion of the base address where the inbound memory window begins. The number of bits that the device allows to be writable is selected through the inbound window size in the inbound window attributes registers (PEXIWAR2 for offset 0x1C and PEXIWAR3 for offset 0x24). If no access to local memory is to be permitted by external requestors, then all bits are programmed.

16.3.8.2.2 PCI Express Subsystem Vendor ID Register (EP-Mode Only)—0x2C

The PCI Express subsystem vendor ID register is used to identify the subsystem.

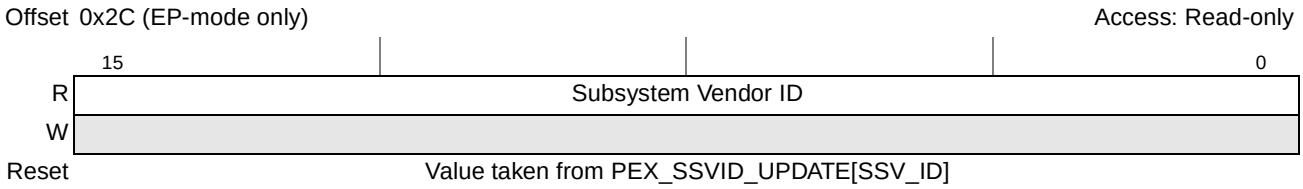


Figure 16-51. PCI Express Subsystem Vendor ID Register

Table 16-50. PCI Express Subsystem Vendor ID Register Field Description

Bits	Name	Description
15–0	Subsystem Vendor ID	The value for subsystem vendor ID is determined by the PCI Express subsystem vendor ID update register. See Section 16.3.10.17, “PCI Express Subsystem Vendor ID Update Register (EP Mode Only)—0x478,” for more information.

16.3.8.2.3 PCI Express Subsystem ID Register (EP-Mode Only)—0x2E

The PCI Express subsystem ID register is used to identify the subsystem.

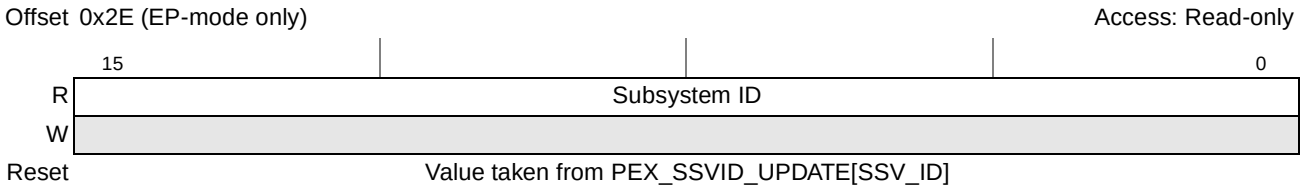


Figure 16-52. PCI Express Subsystem ID Register

Table 16-51. PCI Express Subsystem ID Register Field Description

Bits	Name	Description
15–0	Subsystem ID	The value for subsystem ID is determined by the PCI Express subsystem vendor ID update register. See Section 16.3.10.17, "PCI Express Subsystem Vendor ID Update Register (EP Mode Only)—0x478," for more information.

16.3.8.2.4 Capabilities Pointer Register—0x34

The capabilities pointer identifies additional functionality supported by the device.

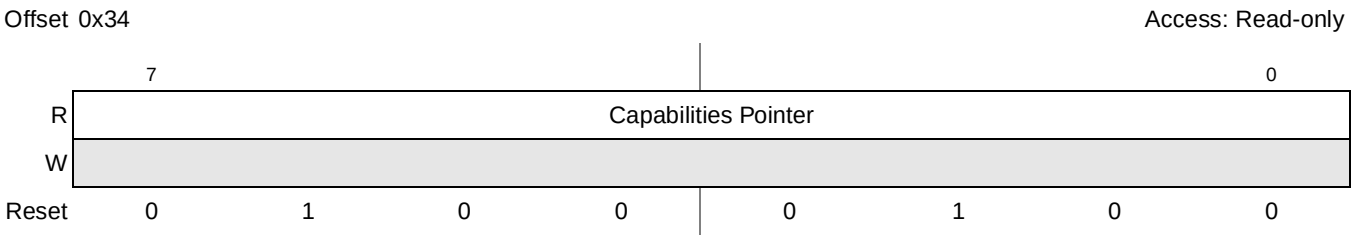


Figure 16-53. Capabilities Pointer Register

Table 16-52. Capabilities Pointer Register Field Description

Bits	Name	Description
7–0	Capabilities Pointer	The capabilities pointer provides the offset (0x44) for additional PCI-compatible registers above the common 64-byte header. Refer to Section 16.3.9, "PCI Compatible Device-Specific Configuration Space," for more information.

16.3.8.2.5 PCI Express Interrupt Line Register (EP-Mode Only)—0x3C

The interrupt line register is used by device drivers and OS software to communicate interrupt line routing information. Values in this register are programmed by system software and are system specific.

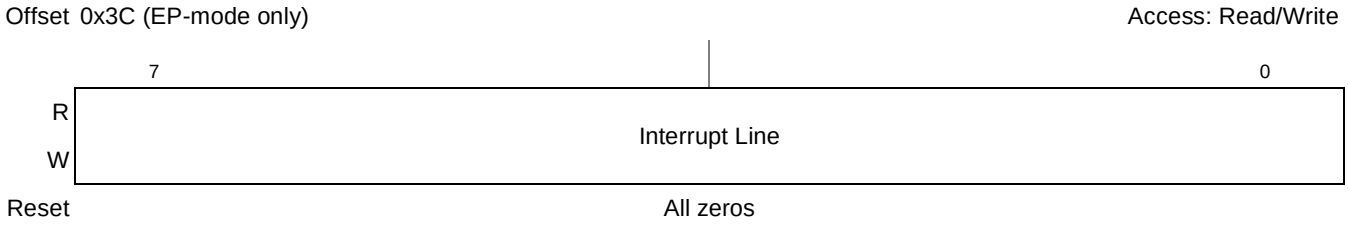


Figure 16-54. PCI Express Interrupt Line Register

Table 16-53. PCI Express Interrupt Line Register Field Description

Bits	Name	Description
7–0	Interrupt Line	Used to communicate interrupt line routing information.

16.3.8.2.6 PCI Express Interrupt Pin Register—0x3D

The interrupt pin register identifies the legacy interrupt (INTx) messages the device (or function) uses.

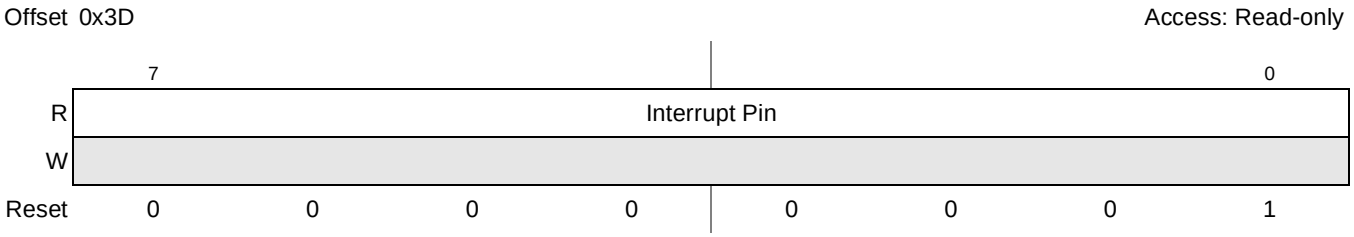


Figure 16-55. PCI Express Interrupt Pin Register

Table 16-54. PCI Express Interrupt Pin Register Field Description

Bits	Name	Description
7–0	Interrupt pin	Legacy INTx message used by this device. 0x00 This device does not use legacy interrupt (INTx) messages. 0x01 INTA 0x02 INTB 0x03 INTC 0x04 INTD all others Reserved.

16.3.8.2.7 PCI Express Minimum Grant Register (EP-Mode Only)—0x3E

This register does not apply to PCI Express. It is present for legacy purposes.

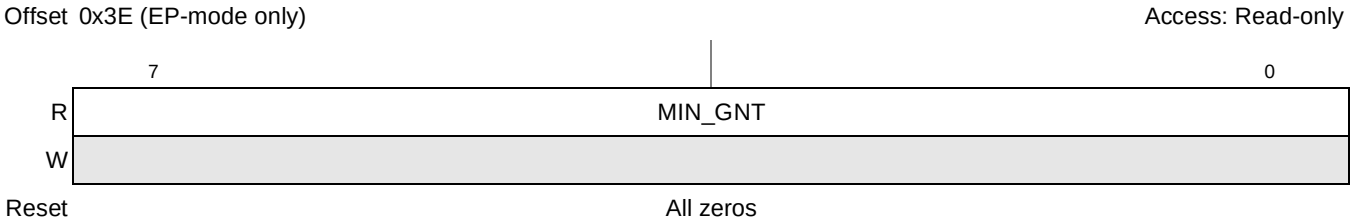


Figure 16-56. PCI Express Maximum Grant Register (MAX_GNT)

Table 16-55. PCI Express Maximum Grant Register Field Description

Bits	Name	Description
7–0	MIN_GNT	Does not apply for PCI Express.

16.3.8.2.8 PCI Express Maximum Latency Register (EP-Mode Only)—0x3F

This register does not apply to PCI Express. It is present for legacy purposes.

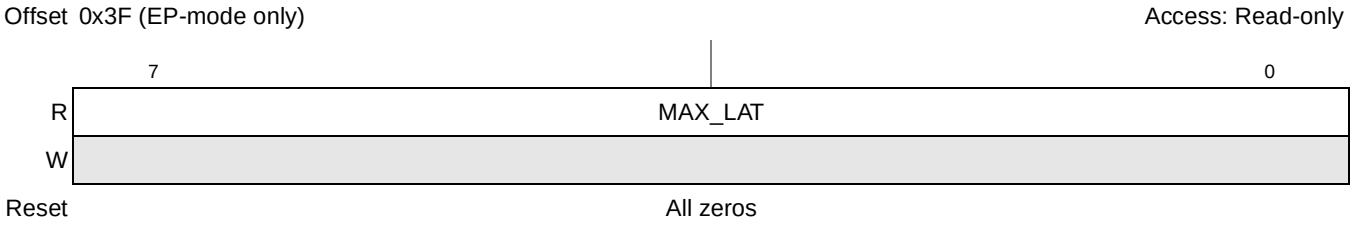


Figure 16-57. PCI Express Maximum Latency Register (MAX_LAT)

Table 16-56. PCI Express Maximum Latency Register Field Description

Bits	Name	Description
7–0	MAX_LAT	Does not apply for PCI Express.

16.3.8.3 Type 1 Configuration Header

The type 1 header is shown in Figure 16-58.

Reserved

				Address Offset (Hex)
Device ID		Vendor ID		00
Status		Command		04
Class Code			Revision ID	08
BIST	Header Type	Latency Timer	Cache Line Size	0C
Base Address Register 0				10
				14
Secondary Latency Timer	Subordinate Bus Number	Secondary Bus Number	Primary Bus Number	18
Secondary Status		I/O Limit	I/O Base	1C
Memory Limit		Memory Base		20
Prefetchable Memory Limit		Prefetchable Memory Base		24
Prefetchable Base Upper 32 Bits				28
Prefetchable Limit Upper 32 Bits				2C
I/O Limit Upper 16 Bits		I/O Base Upper 16 Bits		30
			Capabilities Pointer	34
Bridge Control		Interrupt Pin	Interrupt Line	3C

Figure 16-58. PCI Express PCI-Compatible Configuration Header—Type 1

Section 16.3.8.1, "Common PCI Compatible Configuration Header Registers," describes the registers in the first 16 bytes of the header. This section describes the registers that are unique to the type 1 header beginning at offset 0x10.

16.3.8.3.1 PCI Express Base Address Register 0—0x10

Base address register 0 at offset 0x10 is a special fixed 1-Mbyte window that is used for inbound configuration accesses. This window is called the PCI Express configuration and status register base address register (PEXCSRBAR). Note that PEXCSRBAR cannot be updated through the inbound ATMU registers. The PEXCSRBAR is shown in Figure 16-47.

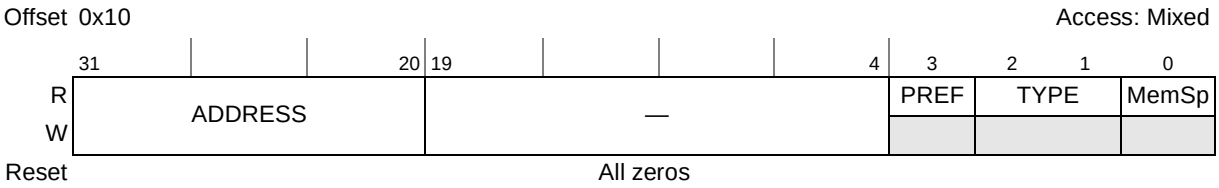


Figure 16-59. PCI Express Base Address Register 0 (PEXCSRBAR)

Table 16-46 describes the PCI Express configuration and status register base address register.

Table 16-57. PEXCSRBAR Field Descriptions

Bits	Name	Description
31–20	ADDRESS	Indicates the base address that the inbound configuration window occupies. This window is fixed at 1 Mbyte.
19–4	—	Reserved
3	PREF	Prefetchable
2–1	TYPE	Type. 00 Locate anywhere in 32-bit address space.
0	MemSp	Memory space indicator

16.3.8.3.2 PCI Express Primary Bus Number Register—Offset 0x18

The primary bus number register is shown in Figure 16-60.

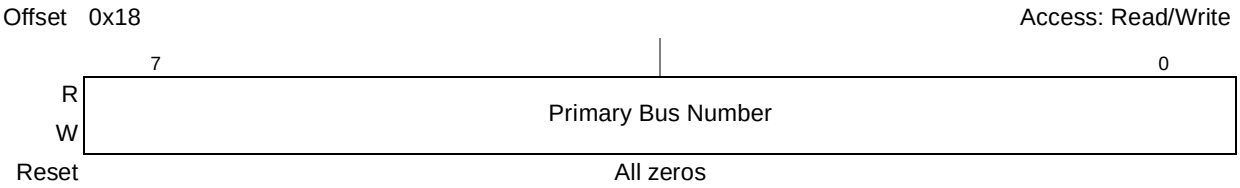


Figure 16-60. PCI Express Primary Bus Number Register

Table 16-58 describes the primary bus number register fields.

Table 16-58. PCI Express Primary Bus Number Register Field Description

Bits	Name	Description
7–0	Primary Bus Number	Bus that is connected to the upstream interface. Note that this register is programmed during system enumeration; in RC mode this register should remain 0x00.

16.3.8.3.3 PCI Express Secondary Bus Number Register—Offset 0x19

The secondary bus number register is shown in Figure 16-61.

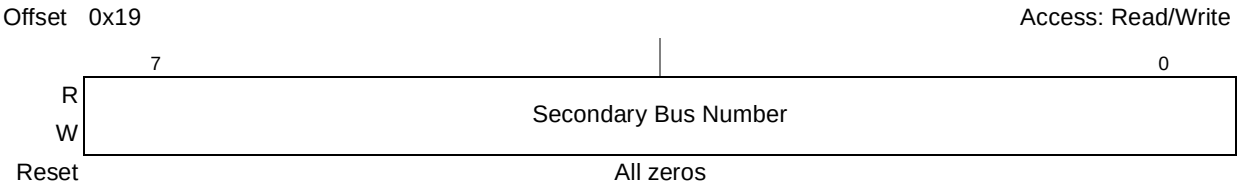


Figure 16-61. PCI Express Secondary Bus Number Register

Table 16-59 describes the secondary bus number register fields.

Table 16-59. PCI Express Secondary Bus Number Register Field Description

Bits	Name	Description
7–0	Secondary Bus Number	Bus that is directly connected to the downstream interface. Note that this register is programmed during system enumeration; in RC mode, this register is typically programmed to 0x01.

16.3.8.3.4 PCI Express Subordinate Bus Number Register—Offset 0x1A

The subordinate bus number register is shown in Figure 16-62.

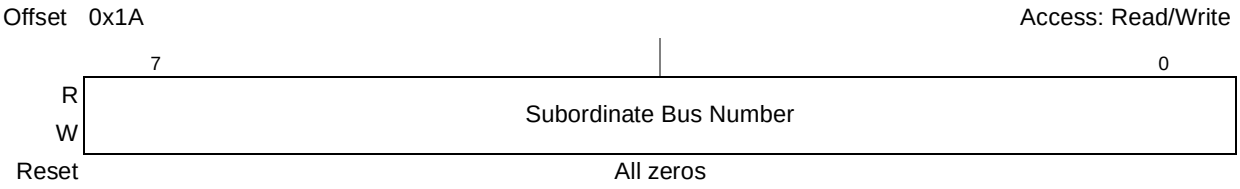


Figure 16-62. PCI Express Subordinate Bus Number Register

Table 16-60 describes the subordinate bus number register fields.

Table 16-60. PCI Express Subordinate Bus Number Register Field Description

Bits	Name	Description
7–0	Subordinate Bus Number	Highest bus number that is on the downstream interface.

16.3.8.3.5 PCI Express Secondary Latency Timer Register—0x1B

The secondary latency timer register does not apply to PCI Express. It must be read-only and return all zeros when read.

16.3.8.3.6 PCI Express I/O Base Register—0x1C

Note that this device does not support inbound I/O transactions. The I/O base register is shown in Figure 16-62.

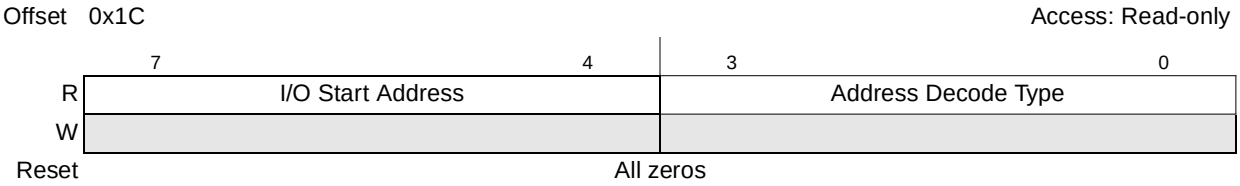


Figure 16-63. PCI Express I/O Base Register

Table 16-60 describes the I/O base register fields.

Table 16-61. PCI Express I/O Base Register Field Description

Bits	Name	Description
7–4	I/O Start Address	Specifies bits 15:12 of the I/O space start address
3–0	Address Decode Type	Specifies the number of I/O address bits. 0x00 16-bit I/O address decode 0x01 32-bit I/O address decode All other settings reserved.

16.3.8.3.7 PCI Express I/O Limit Register—0x1D

Note that this device does not support inbound I/O transactions. The I/O limit register is shown in Figure 16-62.

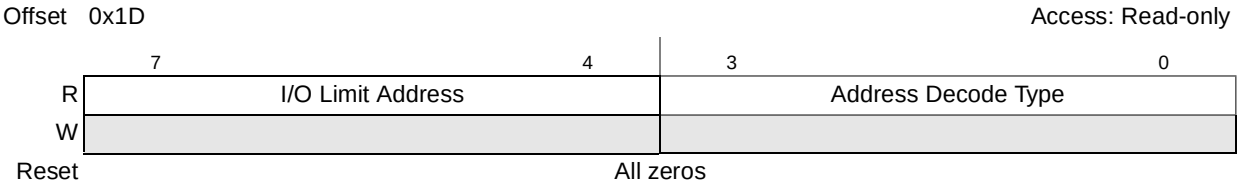


Figure 16-64. PCI Express I/O Limit Register

Table 16-60 describes the I/O limit register fields.

Table 16-62. PCI Express I/O Limit Register Field Description

Bits	Name	Description
7–4	I/O Limit Address	Specifies bits 15:12 of the I/O space ending address
3–0	Address Decode Type	Specifies the number of I/O address bits. 0x00 16-bit I/O address decode 0x01 32-bit I/O address decode All other settings reserved.

16.3.8.3.8 PCI Express Secondary Status Register—0x1E

The PCI Express secondary status register is shown in Figure 16-65. Note that the errors in this register may be masked by corresponding bits in the secondary status interrupt mask register (PEX_SS_INTR_MASK) and that by default all of the errors are masked. See Section 16.3.10.20, "Secondary Status Interrupt Mask Register (RC-Mode Only)—0x5A0," for more information.

Offset 0x1E Access: w1c

	15	14	13	12	11	10	9	8	7	0
R	DPE	SSE	RMA	RTA	STA	—		MDPE	—	
W	w1c	w1c	w1c	w1c	w1c			w1c		
Reset	All zeros									

Figure 16-65. PCI Express Secondary Status Register

Table 16-63 describes the PCI Express secondary status register fields.

Table 16-63. PCI Express Secondary Status Register Field Description

Bits	Name	Description
15	DPE	Detected parity error. This bit is set whenever the secondary side receives a poisoned TLP regardless of the state of the parity error response bit.
14	SSE	Signaled system error. This bit is set when a device sends a ERR_FATAL or ERR_NONFATAL message, provided the SERR enable bit in the command register is set to enable reporting.
13	RMA	Received master abort. This bit is set when the secondary side receives an unsupported request (UR) completion.
12	RTA	Received target abort. This bit is set when the secondary side receives a completer abort (CA) completion.
11	STA	Signaled target abort. This bit is set when the secondary side issues a CA completion.
10–9	—	Reserved
8	MDPE	Master data parity error. This bit is set when the parity error response bit is set and the secondary side requestor receives a poisoned completion or poisons a write request. If the parity error response bit is cleared, this bit is never set.
7–0	—	Reserved

16.3.8.3.9 PCI Express Memory Base Register—0x20

The memory base register is shown in Figure 16-66.

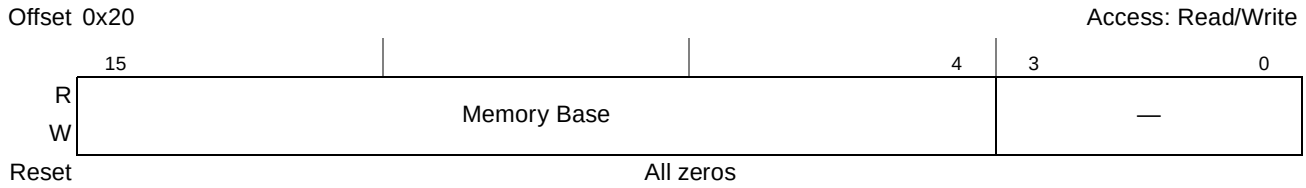


Figure 16-66. PCI Express Memory Base Register

Table 16-64 describes the memory base register fields.

Table 16-64. PCI Express Memory Base Register Field Description

Bits	Name	Description
15–4	Memory Base	Specifies bits 31:20 of the non-prefetchable memory space start address. Typically used for specifying memory-mapped I/O space. **Note:** Inbound posted transactions hitting into the mem base/limit range are ignored; inbound non-posted transactions hitting into the mem base/limit range results in an unsupported request response.
3–0	—	Reserved

16.3.8.3.10 PCI Express Memory Limit Register—0x22

The memory limit register is shown in Figure 16-67.

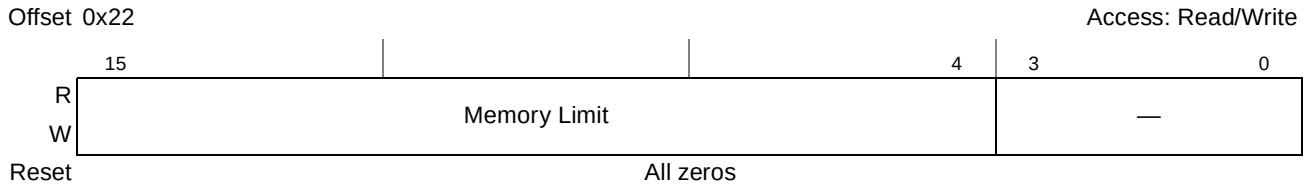


Figure 16-67. PCI Express Memory Limit Register

Table 16-65 describes the memory base register fields.

Table 16-65. PCI Express Memory Limit Register Field Description

Bits	Name	Description
15–4	Memory Limit	Specifies bits 31:20 of the non-prefetchable memory space ending address. Typically used for specifying memory-mapped I/O space. **Note:** Inbound posted transactions hitting into the mem base/limit range are ignored; inbound non-posted transactions hitting into the mem base/limit range results in unsupported request response.
3–0	—	Reserved

16.3.8.3.11 PCI Express Prefetchable Memory Base Register—0x24

The prefetchable memory base register is shown in Figure 16-68.

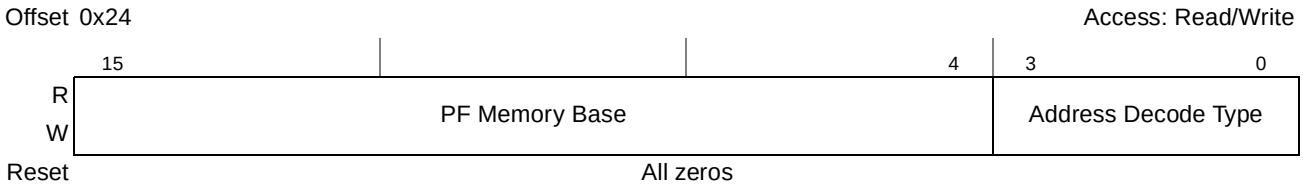


Figure 16-68. PCI Express Prefetchable Memory Base Register

Table 16-66 describes the prefetchable memory base register fields.

Table 16-66. PCI Express Prefetchable Memory Base Register Field Description

Bits	Name	Description
15–4	PF Memory Base	Specifies bits 31:20 of the prefetchable memory space start address.
3–0	Address Decode Type	Specifies the number of prefetchable memory address bits. 0x00 32-bit memory address decode 0x01 64-bit memory address decode All other settings reserved.

16.3.8.3.12 PCI Express Prefetchable Memory Limit Register—0x26

The prefetchable memory limit register is shown in Figure 16-69.

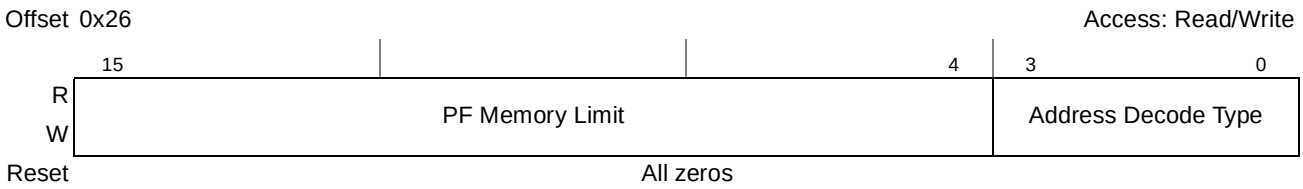


Figure 16-69. PCI Express Prefetchable Memory Limit Register

Table 16-67 describes the prefetchable memory limit register fields.

Table 16-67. PCI Express Prefetchable Memory Limit Register Field Description

Bits	Name	Description
15–4	PF Memory Limit	Specifies bits 31:20 of the prefetchable memory space ending address.
3–0	Address Decode Type	Specifies the number of prefetchable memory address bits. 0x00 32-bit memory address decode 0x01 64-bit memory address decode All other settings reserved.

16.3.8.3.13 PCI Express Prefetchable Base Upper 32 Bits Register—0x28

The PCI Express prefetchable memory base upper 32 bits register is shown in Figure 16-70.

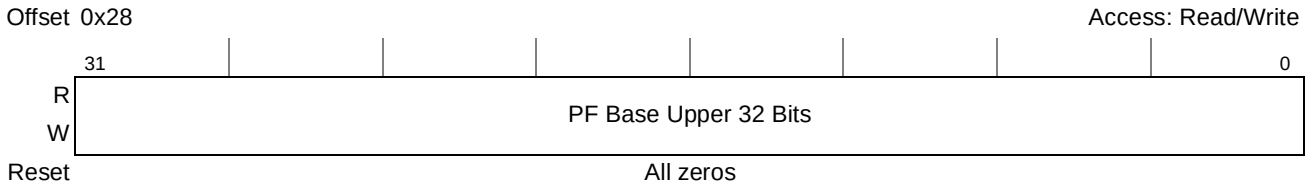


Figure 16-70. PCI Express Prefetchable Base Upper 32 Bits Register

Table 16-68 describes the PCI Express prefetchable memory base upper 32 bits register fields.

Table 16-68. PCI Express Prefetchable Base Upper 32 Bits Register

Bits	Name	Description
31–0	PF Base Upper 32 Bits	Specifies bits 64:32 of the prefetchable memory space start address when the address decode type field in the prefetchable memory base register is 0x01.

16.3.8.3.14 PCI Express Prefetchable Limit Upper 32 Bits Register—0x2C

The PCI Express prefetchable memory base upper 32 bits register is shown in Figure 16-71.

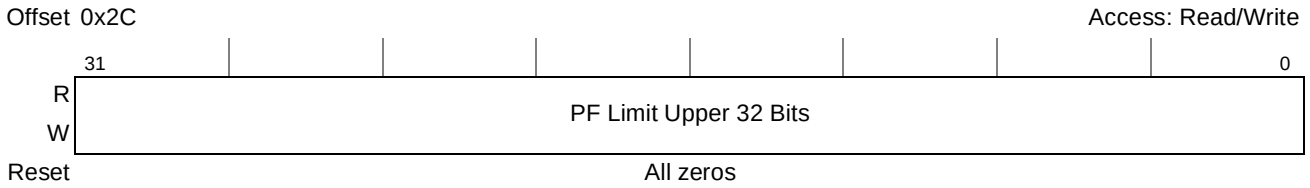


Figure 16-71. PCI Express Prefetchable Limit Upper 32 Bits Register

Table 16-69 describes the PCI Express prefetchable memory limit upper 32 bits register fields.

Table 16-69. PCI Express Prefetchable Limit Upper 32 Bits Register

Bits	Name	Description
31–0	PF Limit Upper 32 Bits	Specifies bits 64–32 of the prefetchable memory space ending address when the address decode type field in the prefetchable memory limit register is 0x01.

16.3.8.3.15 PCI Express I/O Base Upper 16 Bits Register—0x30

Note that this device does not support inbound I/O transactions. The I/O base upper 16 bits register is shown in Figure 16-72.

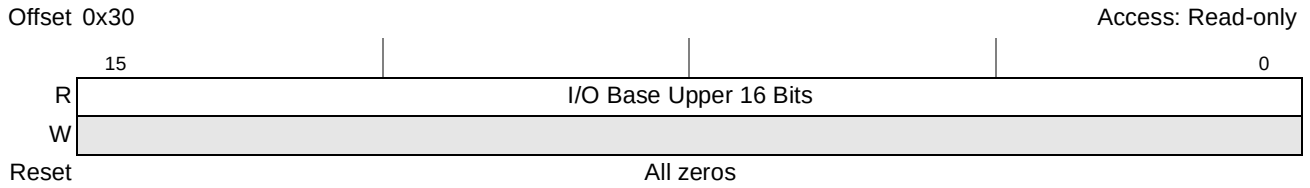


Figure 16-72. PCI Express I/O Base Upper 16 Bits Register

Table 16-70 describes the I/O base upper 16 bits register fields.

Table 16-70. PCI Express I/O Base Upper 16 Bits Register Field Description

Bits	Name	Description
15–0	I/O Base Upper 16 Bits	Specifies bits 31–16 of the I/O space start address when the address decode type field in the I/O base register is 0x01.

16.3.8.3.16 PCI Express I/O Limit Upper 16 Bits Register—0x32

Note that this device does not support inbound I/O transactions. The I/O limit upper 16 bits register is shown in Figure 16-73.

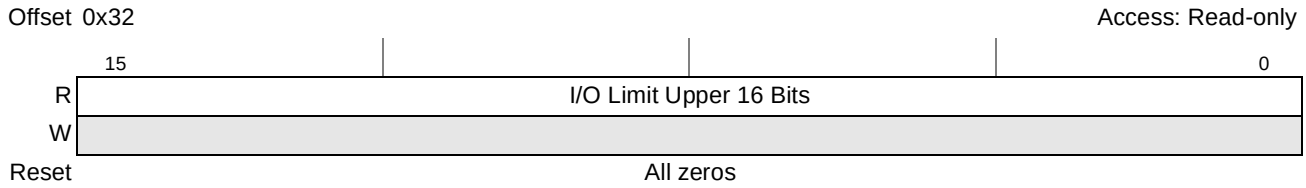


Figure 16-73. PCI Express I/O Limit Upper 16 Bits Register

Table 16-71 describes the I/O limit upper 16 bits register fields.

Table 16-71. PCI Express I/O Limit Upper 16 Bits Register Field Description

Bits	Name	Description
15–0	I/O Limit Upper 16 Bits	Specifies bits 31–16 of the I/O space ending address when the address decode type field in the I/O limit register is 0x01.

16.3.8.3.17 Capabilities Pointer Register—0x34

The capabilities pointer identifies additional functionality supported by the device.

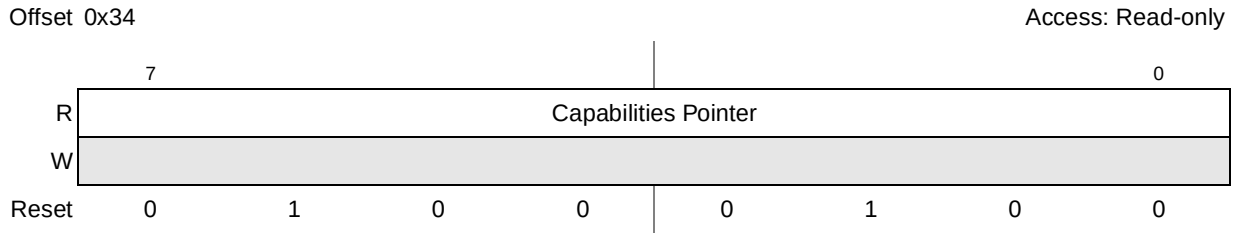


Figure 16-74. Capabilities Pointer Register

Table 16-72. Capabilities Pointer Register Field Description

Bits	Name	Description
7–0	Capabilities Pointer	The capabilities pointer provides the offset (0x44) for additional PCI-compatible registers above the common 64-byte header. Refer to Section 16.3.9, “PCI Compatible Device-Specific Configuration Space,” for more information.

16.3.8.3.18 PCI Express Interrupt Line Register—0x3C

The interrupt line register is used by device drivers and OS software to communicate interrupt line routing information. Values in this register are programmed by system software and are system specific.

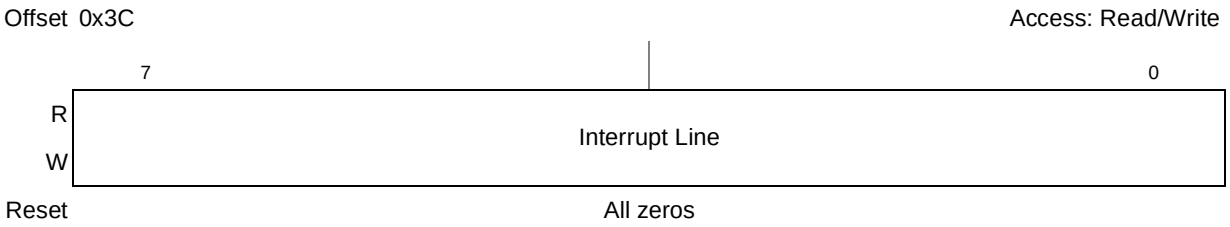


Figure 16-75. PCI Express Interrupt Line Register

Table 16-73. PCI Express Interrupt Line Register Field Description

Bits	Name	Description
7–0	Interrupt Line	Used to communicate interrupt line routing information.

16.3.8.3.19 PCI Express Interrupt Pin Register—0x3D

The interrupt pin register identifies the legacy interrupt (INTx) messages the device (or function) uses.

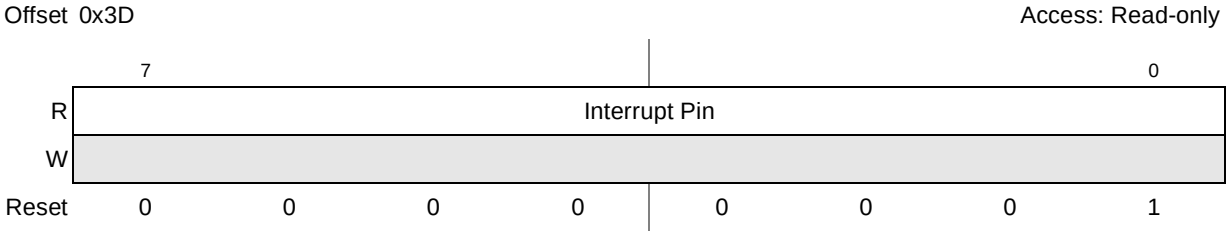


Figure 16-76. PCI Express Interrupt Pin Register

Table 16-74. PCI Express Interrupt Pin Register Field Description

Bits	Name	Description
7–0	Interrupt pin	Legacy INTx message used by this device. 0x00 This device does not use legacy interrupt (INTx) messages. 0x01 INTA 0x02 INTB 0x03 INTC 0x04 INTD all others Reserved.

16.3.8.3.20 PCI Express Bridge Control Register—0x3E

The PCI Express bridge control register is shown in Figure 16-77.

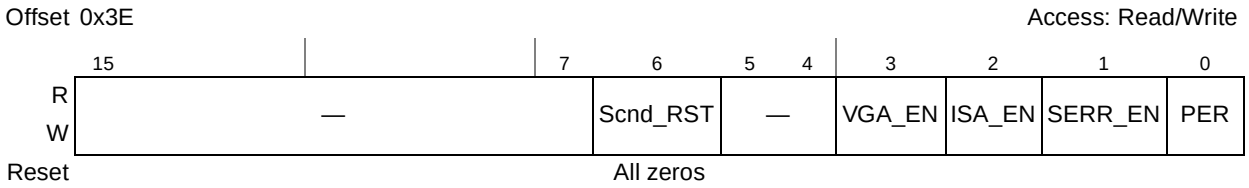


Figure 16-77. PCI Express Bridge Control Register

Table 16-75 describes the PCI Express bridge control register fields.

Table 16-75. PCI Express Bridge Control Register Field Description

Bits	Name	Description
15–7	—	Reserved
6	Scnd_RST	Secondary bus reset
5–4	—	Reserved
3	VGA_EN	VGA enable
2	ISA_EN	ISA enable
1	SERR_EN	SERR enable. This bit controls the propagation of ERR_COR, ERR_NONFATAL, and ERR_FATAL responses received on the secondary side.
0	PER	Parity error response.

16.3.9 PCI Compatible Device-Specific Configuration Space

The PCI compatible device-specific configuration space is a PCI compatible configuration space from 0x40 to 0xFF (just above the 64-byte PCI-compatible configuration header).

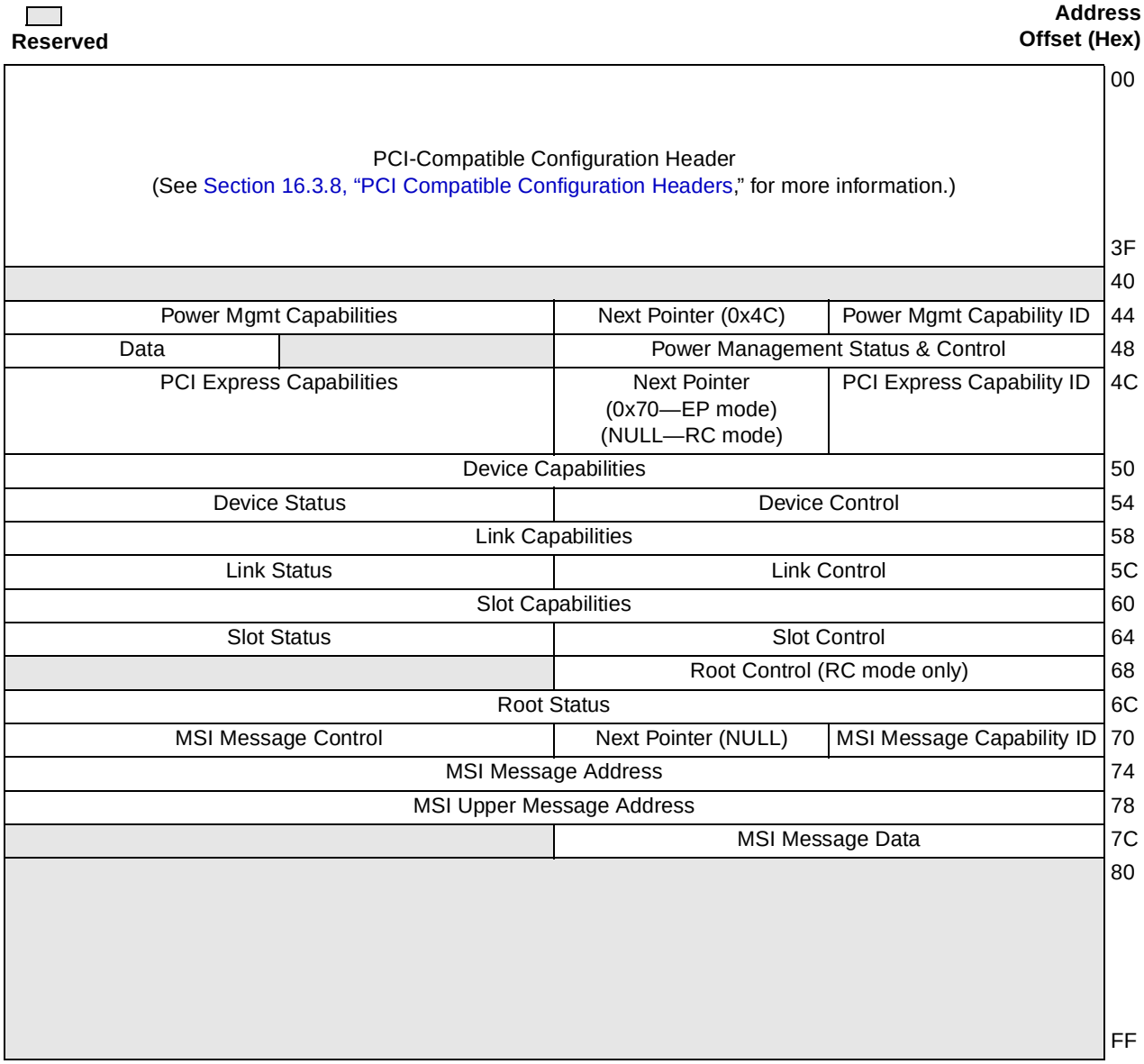


Figure 16-78. PCI Compatible Device-Specific Configuration Space

16.3.9.1 PCI Express Power Management Capability ID Register—0x44

The PCI Express power management capability ID register is shown in Figure 16-79.

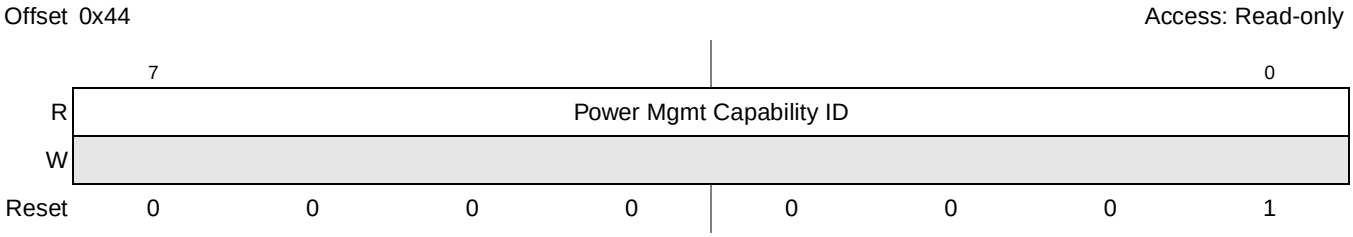


Figure 16-79. PCI Express Power Management Capability ID Register

Table 16-76. PCI Express Power Management Capability ID Register Field Description

Bits	Name	Description
7–0	Power Mgmt Capability ID	Power Management = 0x01

16.3.9.2 PCI Express Power Management Capabilities Register—0x46

The PCI Express power management capabilities register is shown in Figure 16-80.

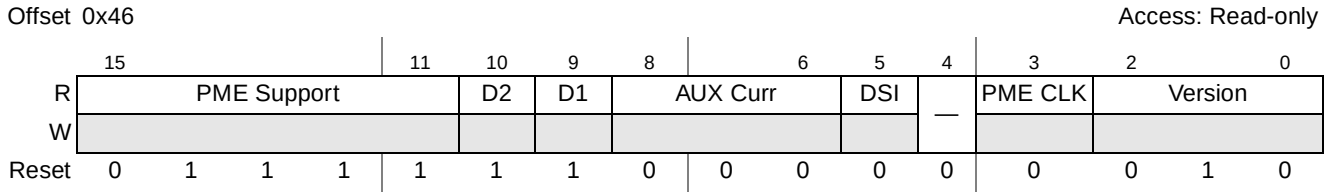


Figure 16-80. PCI Express Power Management Capabilities Register

Table 16-77. PCI Express Power Management Capabilities Register Field Description

Bits	Name	Description
15–11	PME Support	Indicates the power states that this device supports
10	D2	D2 Support
9	D1	D1 Support
8–6	AUX Curr	AUX Current
5	DSI	Device Specific Initialization
4	—	Reserved
3	PME CLK	Does not apply to PCI Express.
2–0	Version	Set to 0x2 for this version of the specification.

16.3.9.3 PCI Express Power Management Status and Control Register—0x48

The PCI Express power management status and control register is shown in Figure 16-81.

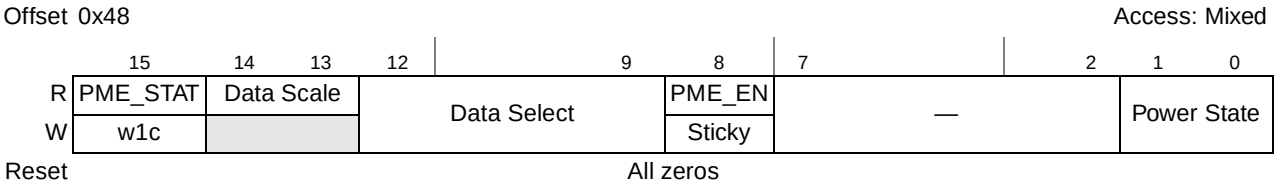


Figure 16-81. PCI Express Power Management Status and Control Register

Table 16-78. PCI Express Status and Control Register Field Description

Bits	Name	Description
15	PME_STAT	PME Status
14–13	Data Scale	Obtained directly from *PCI Express™ Base Specification, Revision 1.0a*
12–9	Data Select	Obtained directly from *PCI Express™ Base Specification, Revision 1.0a*
8	PME_EN	PME Enable
7–2	—	Reserved
1–0	Power State	Power state. Indicates the current power state of the function. 0x00 D0 0x01 D1 0x02 D2 0x03 D3

16.3.9.4 PCI Express Power Management Data Register—0x4B

The PCI Express power management data register is shown in Figure 16-82.

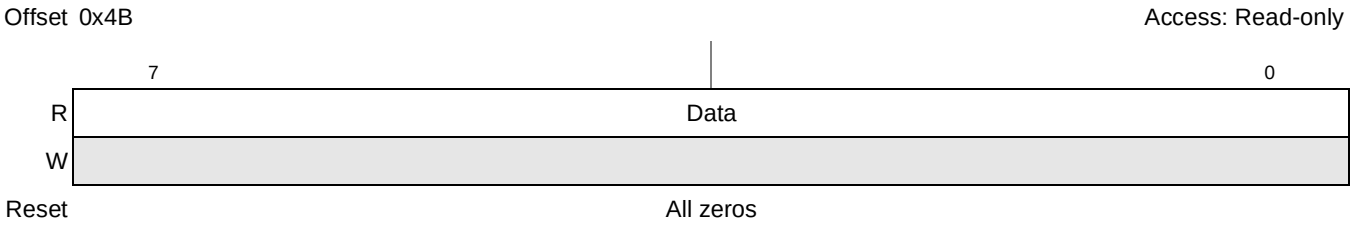


Figure 16-82. PCI Express Power Management Data Register

Table 16-79. PCI Express Power Management Data Register Field Description

Bits	Name	Description
7–0	Data	Obtained from *PCI Express™ Base Specification, Revision 1.0a*

16.3.9.5 PCI Express Capability ID Register—0x4C

The PCI Express capability ID register is shown in Figure 16-83.

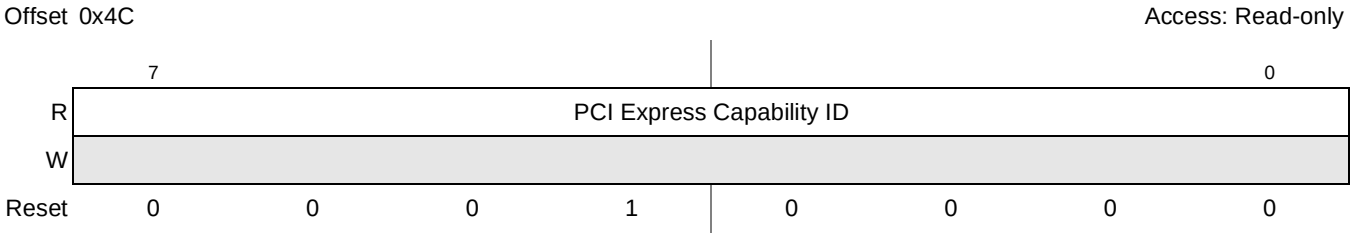


Figure 16-83. PCI Express Capability ID Register

Table 16-80. PCI Express Capability ID Register Field Description

Bits	Name	Description
7–0	PCI Express Capability ID	PCI Express = 0x10

16.3.9.6 PCI Express Capabilities Register—0x4E

The PCI Express capabilities register is shown in Figure 16-84.

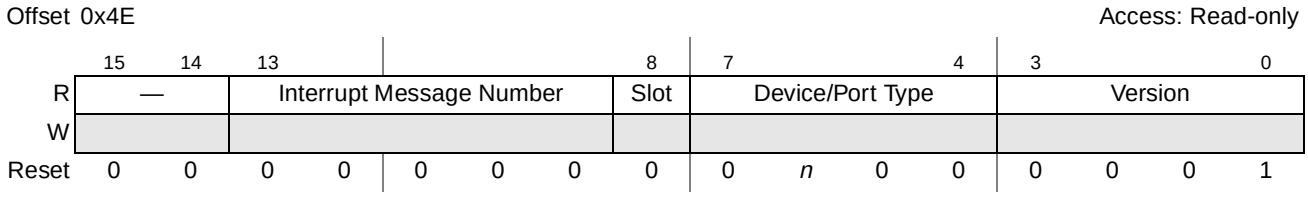


Figure 16-84. PCI Express Capabilities Register

Table 16-81. PCI Express Capabilities Register Field Description

Bits	Name	Description
15–14	—	Reserved
13–9	Interrupt Message Number	If this function is allocated more than one MSI interrupt number, then this register is required to contain the offset between the base Message Data and the MSI Message that is generated when any of the status bits in either the Slot Status register or the Root Port Status register, of this capability structure, are set.
8	Slot	Slot Implemented (RC mode only)
7–4	Device/Port Type	0100 (RC mode) 0000 (EP mode)
3–0	Capability Version	Indicates the defined PCI Express capability structure version number. Must be 1h for 1.0, 1.0a, and 1.1 specification.

16.3.9.7 PCI Express Device Capabilities Register—0x50

The PCI Express device capabilities register is shown in Figure 16-85.

Offset 0x50 Access: Read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	—				CSPLS		CSPLV								—	
W																
Reset	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	—	PIP	AIP	ABP	EP_L1_LAT			EP_L0s_LAT			ET	PHAN_FCT		MAX_PL_SIZE_SUP		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 16-85. PCI Express Device Capabilities Register

Table 16-82. PCI Express Device Capabilities Register Field Description

Bits	Name	Description
31–28	—	Reserved
27–26	CSPLS	Captured Slot Power Limit Scale
25–18	CSPLV	Captured Slot Power Limit Value
17–15	—	Reserved
14	PIP	Power Indicator Present
13	AIP	Attention Indicator Present
12	ABP	Attention Button Present
11–9	EP_L1_LAT	Endpoint L1 Acceptable Latency
8–6	EP_L0s_LAT	Endpoint L0s Acceptable Latency
5	ET	Extended Tag Field Supported
4–3	PHAN_FCT	Phantom Functions Supported
2–0	MAX_PL_SIZE_SUP	Maximum payload size supported. 001 = 256-bytes

16.3.9.8 PCI Express Device Control Register—0x54

The PCI Express device control register is shown in Figure 16-86.

Offset 0x54 Access: Read/write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	—	MAX_READ_SIZE			NSE	APE	PFE	ETE	MAX_PAYLOAD_SIZE			RO	URR	FER	NFER	CER
W																
Reset	0	0	0	1	0	1	0	0	0	0	0	1	0	0	0	0

Figure 16-86. PCI Express Device Control Register

Table 16-83. PCI Express Device Control Register Field Description

Bits	Name	Description
15	—	Reserved
14–12	MAX_READ_SIZE	Maximum read request size
11	NSE	No snoop enable
10	APE	AUX power PM enable
9	PFE	Phantom functions enable
8	ETE	Extended tag field enable
7–5	MAX_PAYLOAD_SIZE	Maximum payload size
4	RO	Relaxed ordering
3	URR	Unsupported request reporting
2	FER	Fatal error reporting
1	NFER	Non-fatal error reporting
0	CER	Correctable error reporting

16.3.9.9 PCI Express Device Status Register—0x56

The PCI Express device status register is shown in Figure 16-87.

Offset 0x56 Access: Mixed

	15–6	5	4	3	2	1	0
R	—	TP	APD	URD	FED	NFED	CED
W				w1c	w1c	w1c	w1c
Reset	All zeros						

Figure 16-87. PCI Express Device Status Register

Table 16-84. PCI Express Device Status Register Field Description

Bits	Name	Description
15–6	—	Reserved
5	TP	Transactions pending
4	APD	AUX power detected
3	URD	Unsupported request detected
2	FED	Fatal error detected
1	NFED	Non-fatal error detected
0	CED	Correctable error detected

16.3.9.10 PCI Express Link Capabilities Register—0x58

The PCI Express link capabilities register is shown in Figure 16-88.

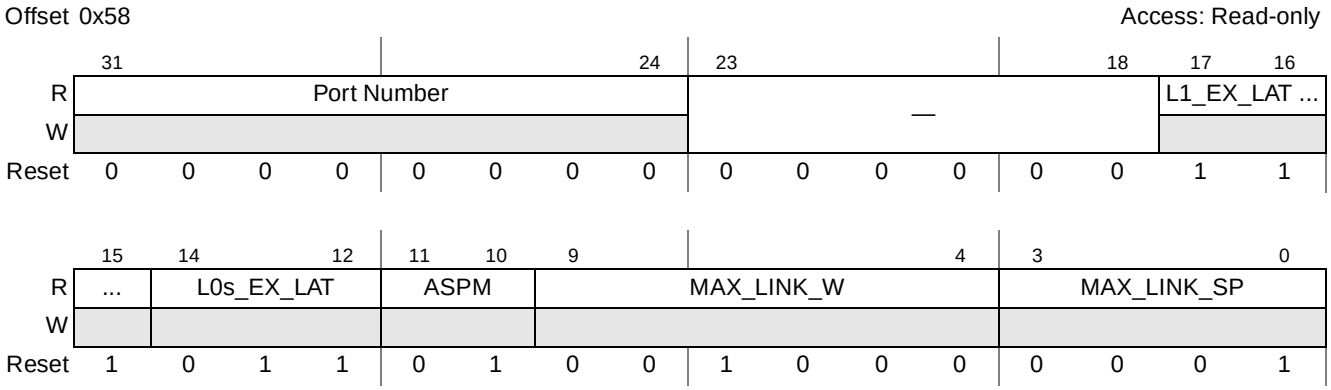


Figure 16-88. PCI Express Link Capabilities Register

Table 16-85. PCI Express Link Capabilities Register Field Description

Bits	Name	Description
31–24	Port Number	—
23–18	—	Reserved
17–15	L1_EX_LAT	L1 exit latency
14–12	L0s_EX_LAT	L0s exit latency
11–10	ASPM	Active state power management (ASPM) Support
9–4	MAX_LINK_W	Maximum link width
3–0	MAX_LINK_SP	Maximum link speed 0001 2.5 GT/s link

16.3.9.11 PCI Express Link Control Register—0x5C

The PCI Express link control register is shown in Figure 16-89.

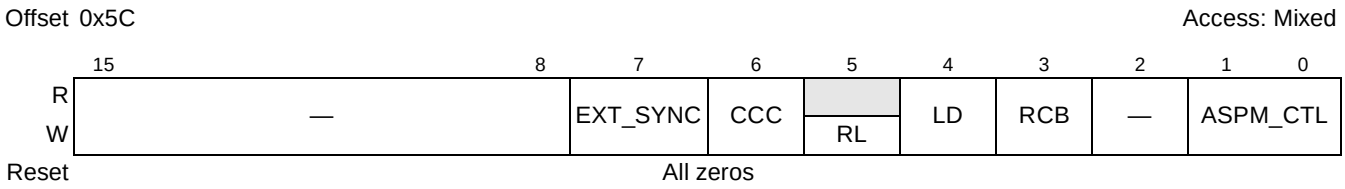


Figure 16-89. PCI Express Link Control Register

Table 16-86. PCI Express Link Control Register Field Description

Bits	Name	Description
15–8	—	Reserved
7	EXT_SYNC	Extended synch

Table 16-86. PCI Express Link Control Register Field Description (continued)

Bits	Name	Description
6	CCC	Common clock configuration
5	RL	Retrain link (Reserved for EP devices). In RC mode, setting this bit initiates link retraining by directing the Physical Layer LTSSM to the Recovery state; reads of this bit always return 0.
4	LD	Link disable (Reserved for EP devices)
3	RCB	Read completion boundary
2	—	Reserved
1–0	ASPM_CTL	Active state power management (ASPM) control

16.3.9.12 PCI Express Link Status Register—0x5E

The PCI Express link status register is shown in Figure 16-90.

Offset 0x5E Access: Read-only

	15 13	12	11	10	9 4	3 0
R	—	SCC	LT	—	NEG_LINK_W	LINK_SP
W						
Reset	0 0 0	0	0	0	0 0 0 0 0 1	0 0 0 1

Figure 16-90. PCI Express Link Status Register

Table 16-87. PCI Express Link Status Register Field Description

Bits	Name	Description
15–13	—	Reserved
12	SCC	Slot clock configuration
11	LT	Link training
10	—	Reserved.
9–4	NEG_LINK_W	Negotiated link width
3–0	LINK_SP	Link speed.

16.3.9.13 PCI Express Slot Capabilities Register—0x60

The PCI Express slot capabilities register is shown in Figure 16-91.

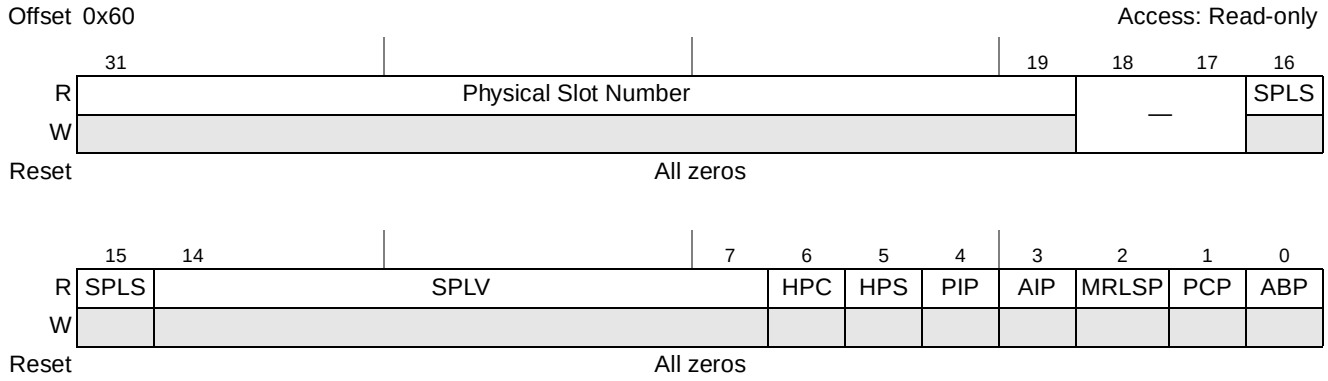


Figure 16-91. PCI Express Slot Capabilities Register

Table 16-88. PCI Express Slot Capabilities Register Field Description

Bits	Name	Description
31–19	Physical Slot Number	This hardware initialized field indicates the physical slot number attached to this Port. This field must be hardware initialized to a value that assigns a slot number that is globally unique within the chassis. These registers should be initialized to 0 for Ports connected to devices that are either integrated on the system board or integrated within the same silicon as the Switch device or Root Port.
18–17	—	Reserved
16–15	SPLS	Slot power limit scale.
14–7	SPLV	Slot power limit value.
6	HPD	Hot plug capable.
5	HPS	Hot plug surprise.
4	PIP	Power indicator present.
3	AIP	Attention indicator present.
2	MRLSP	MRL sensor present.
1	PCP	Power controller present.
0	ABP	Attention button present.

16.3.9.14 PCI Express Slot Control Register—0x64

The PCI Express slot control register is shown in Figure 16-92.

Offset 0x64 Access: Read/Write

	15–11	10	9–8	7–6	5	4	3	2	1	0
R W	—	PCC	PIC	AIC	HPIE	CCIE	PDCE	MRLSCE	PFDE	ABPE
Reset	All zeros									

Figure 16-92. PCI Express Slot Control Register

Table 16-89. PCI Express Slot Control Register Field Description

Bits	Name	Description
15–11	—	Reserved
10	PCC	Power controller control.
9–8	PIC	Power indicator control.
7–6	AIC	Attention indicator control.
5	HPIE	Hot plug interrupt enable.
4	CCIE	Command completed interrupt enable.
3	PDCE	Presence detect changed enable.
2	MRLSCE	MRL sensor changed enable.
1	PFDE	Power fault detected enable.
0	ABPE	Attention button pressed enable.

16.3.9.15 PCI Express Slot Status Register—0x66

The PCI Express slot status register is shown in Figure 16-93.

Offset 0x66 Access: Mixed

	15–7	6	5	4	3	2	1	0
R	—	PDS	MRLSS	CC	PDC	MRLSC	PFD	ABP
W				w1c	w1c	w1c	w1c	w1c
Reset	0 0 0 0 0 0 0 0 0	1	0	0	0	0	0	0

Figure 16-93. PCI Express Slot Status Register

Table 16-90. PCI Express Slot Status Register Field Descriptions

Bits	Name	Description
15–7	—	Reserved
6	PDS	Presence detect state. This bit indicates the presence of a card in the slot. 0 Slot empty 1 Card is present

Table 16-90. PCI Express Slot Status Register Field Descriptions (continued)

Bits	Name	Description
5	MRLSS	MRL sensor state. 0 MRL closed 1 MRL open
4	CC	Command completed.
3	PDC	Presence detect changed.
2	MRLSC	MRL sensor changed.
1	PFD	Power fault detected.
0	ABP	Attention button pressed.

16.3.9.16 PCI Express Root Control Register (RC Mode Only)—0x68

The PCI Express root control register is shown in Figure 16-94.

Offset 0x68 Access: Read/Write

	15–4	3	2	1	0
R W	—	PMEIE	SEFEE	SENFEE	SECEE
Reset	All zeros				

Figure 16-94. PCI Express Root Control Register

Table 16-91. PCI Express Root Control Register Field Description

Bits	Name	Description
15–4	—	Reserved
3	PMEIE	PME interrupt enable.
2	SEFEE	System error on fatal error enable.
1	SENFEE	System error on non-fatal error enable.
0	SECEE	System error on correctable error enable.

16.3.9.17 PCI Express Root Status Register (RC Mode Only)—0x6C

The PCI Express root status register is shown in Figure 16-95.

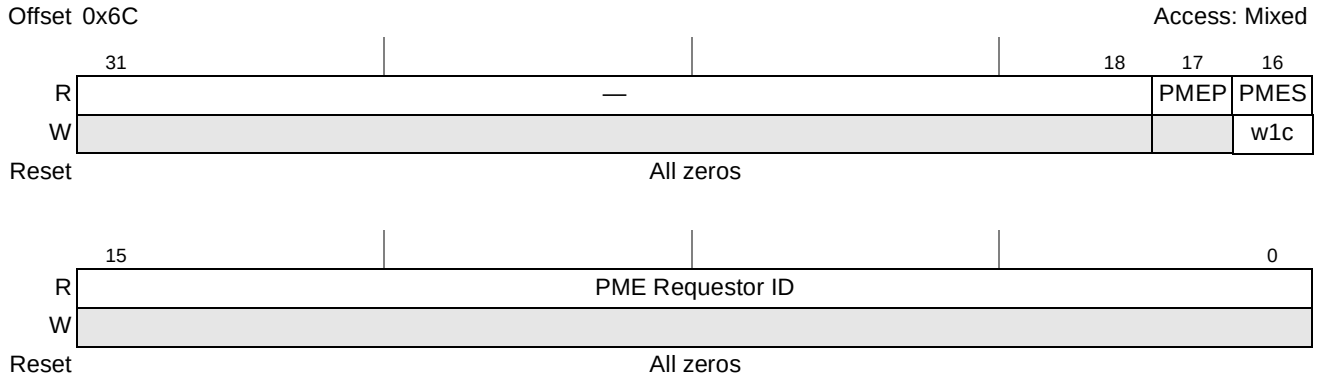


Figure 16-95. PCI Express Root Status Register

Table 16-92. PCI Express Root Status Register Field Description

Bits	Name	Description
31–18	—	Reserved
17	PMEP	PME pending.
16	PMES	PME status.
15–0	PME Requestor ID	PME requestor ID.

16.3.9.18 PCI Express MSI Message Capability ID Register (EP Mode Only)—0x70

The PCI Express MSI message capability ID register is shown in Figure 16-96.

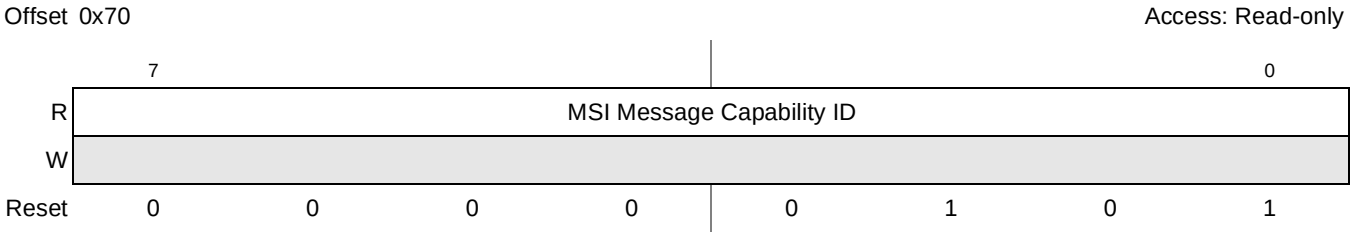


Figure 16-96. PCI Express Capability ID Register

Table 16-93. PCI Express Capability ID Register Field Description

Bits	Name	Description
7–0	MSI Message Capability ID	MSI Message = 0x05

16.3.9.19 PCI Express MSI Message Control Register (EP Mode Only)—0x72

The PCI Express MSI message control register is shown in Figure 16-97.

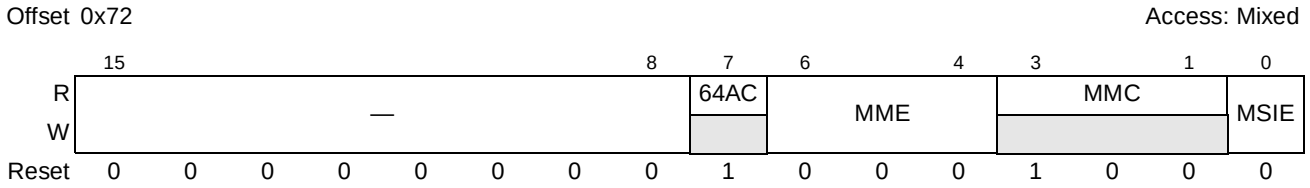


Figure 16-97. PCI Express MSI Message Control Register

Table 16-94. PCI Express MSI Message Control Register Field Description

Bits	Name	Description
15–8	—	Reserved
7	64AC	64-bit address capable.
6–4	MME	Multiple message enable.
3–1	MMC	Multiple message capable.
0	MSIE	MSI enable.

16.3.9.20 PCI Express MSI Message Address Register (EP Mode Only)—0x74

The PCI Express MSI message address register is shown in Figure 16-98.

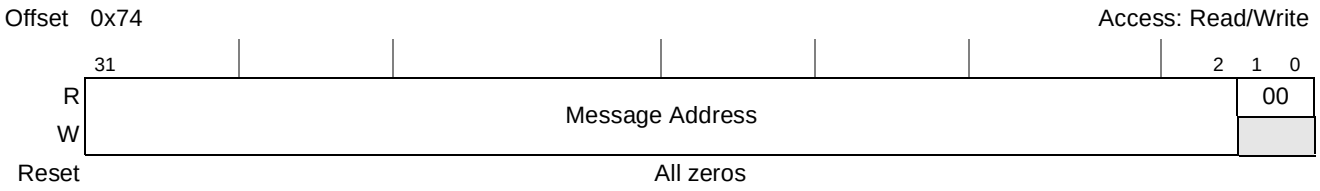


Figure 16-98. PCI Express MSI Message Address Register

Table 16-95. PCI Express MSI Message Address Register Field Description

Bits	Name	Description
31–2	Message Address	System-specified message address
1–0	00	Always returns 00 on reads; write operations have no effect.

16.3.9.21 PCI Express MSI Message Upper Address Register (EP Mode Only)—0x78

The PCI Express MSI message upper address register is shown in Figure 16-99.

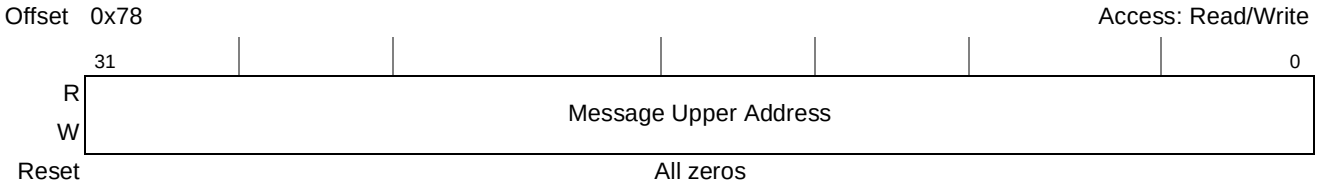


Figure 16-99. PCI Express MSI Message Upper Address Register

Table 16-96. PCI Express MSI Message Upper Address Register Field Description

Bits	Name	Description
31–0	Message Upper Address	System-specified message upper address

16.3.9.22 PCI Express MSI Message Data Register (EP Mode Only)—0x7C

The PCI Express MSI message data register is shown in Figure 16-100.

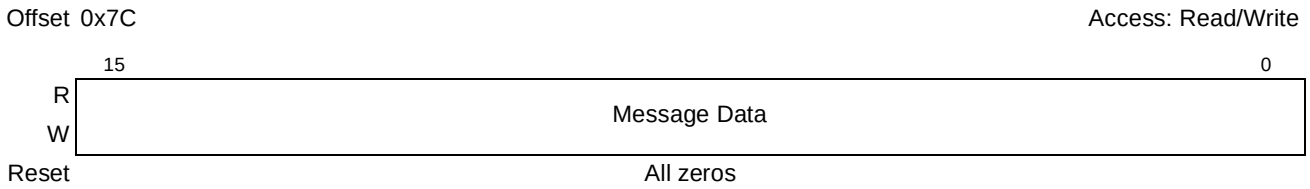


Figure 16-100. PCI Express MSI Message Data Register

Table 16-97. PCI Express MSI Message Data Register Field Description

Bits	Name	Description
15–0	Message Data	System-specified message.

16.3.10 PCI Express Extended Configuration Space

Reserved

		Address Offset (Hex)
PCI Compatible Configuration Header (See Section 16.3.8, “PCI Compatible Configuration Headers,” for more information.)		000 03F
PCI-Compatible Device-Specific Configuration Space (See Section 16.3.9, “PCI Compatible Device-Specific Configuration Space,” for more information.)		040 0FF
Next Capability Offset (NULL)/Capability Version	Advanced Error Reporting Capability ID	100
Uncorrectable Error Status		104
Uncorrectable Error Mask		108
Uncorrectable Error Severity		10C
Correctable Error Status		110
Correctable Error Mask		114
Advanced Error Capabilities and Control		118
Header Log		11C 120 124 128
Root Error Command		12C
Root Error Status		130
Error Source ID	Correctable Error Source ID	134
		138 3FF
PCI Express Controller Internal CSRs[1]		400 6FF
		700 FFF

Figure 16-101. PCI Express Extended Configuration Space

1 Note that the PCI Express Controller Internal CSRs are not accessible by inbound PCI Express configuration transactions. Attempts to access these registers returns all 0s.

16.3.10.1 PCI Express Advanced Error Reporting Capability ID Register—0x100

The PCI Express advanced error reporting capability ID register is shown in Figure 16-102.

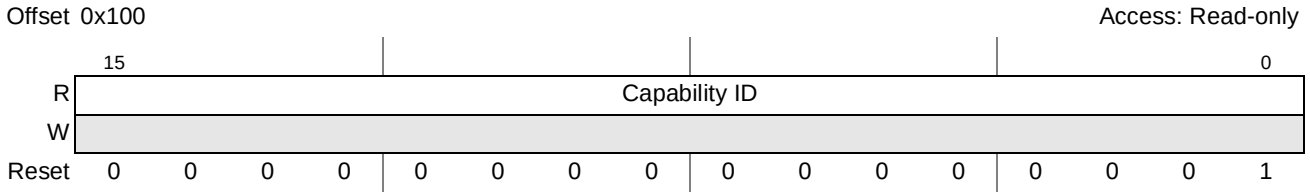


Figure 16-102. PCI Express Advanced Error Reporting Capability ID Register

Table 16-98. PCI Express Advanced Error Reporting Capability ID Register Field Description

Bits	Name	Description
15–0	Capability ID	Advanced error reporting capability = 0x0001

16.3.10.2 PCI Express Uncorrectable Error Status Register—0x104

The PCI Express uncorrectable error status register is shown in Figure 16-103.

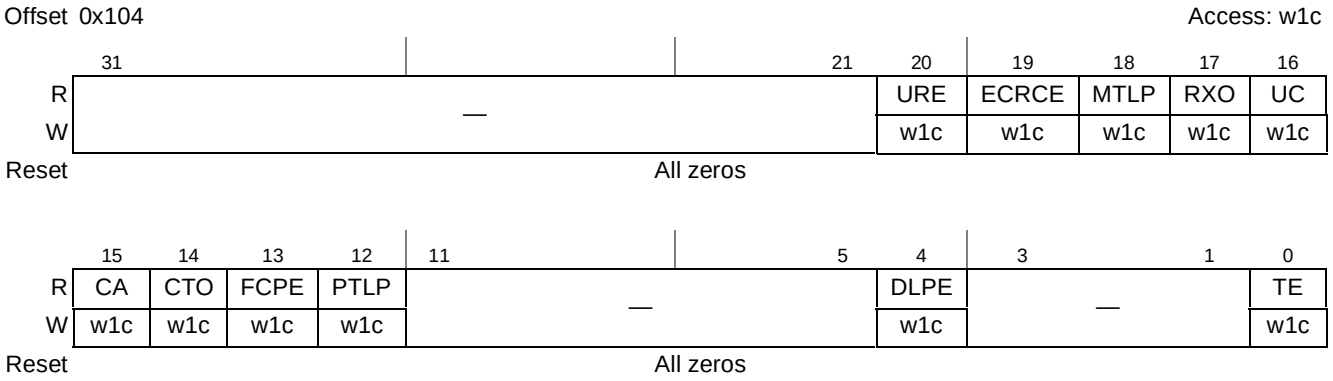


Figure 16-103. PCI Express Uncorrectable Error Status Register

Table 16-99. PCI Express Uncorrectable Error Status Register Field Description

Bits	Name	Description
31–21	—	Reserved
20	URE	Unsupported request error status.
19	ECRCE	ECRC error status.
18	MTLP	Malformed TLP status.
17	RXO	Receiver overflow status.
16	UC	Unexpected completion status.
15	CA	Completer abort status.
14	CTO	Completion timeout status. Note that a completion timeout error is a fatal error. If a completion timeout error is detected, the system has become unstable. Hot reset is recommended to restore stability of the system.

Table 16-99. PCI Express Uncorrectable Error Status Register Field Description (continued)

Bits	Name	Description
13	FCPE	Flow control protocol error status.
12	PTLP	Poisoned TLP status.
11–5	—	Reserved
4	DLPE	Data link protocol error status.
3–1	—	Reserved
0	TE	Training error status.

16.3.10.3 PCI Express Uncorrectable Error Mask Register—0x108

The PCI Express uncorrectable error mask register is shown in Figure 16-104.

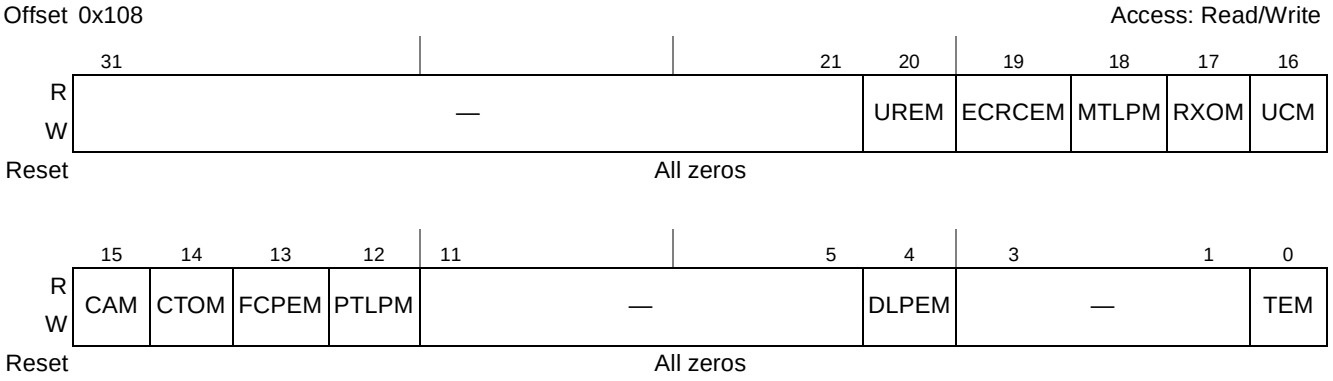


Figure 16-104. PCI Express Uncorrectable Error Mask Register

Table 16-100. PCI Express Uncorrectable Error Mask Register Field Description

Bits	Name	Description
31–21	—	Reserved
20	UREM	Unsupported request error mask.
19	ECRCEM	ECRC error mask.
18	MTLPM	Malformed TLP mask.
17	RXOM	Receiver overflow mask.
16	UCM	Unexpected completion mask.
15	CAM	Completer abort mask.
14	CTOM	Completion timeout mask.
13	FCPEM	Flow control protocol error mask.
12	PTLPM	Poisoned TLP mask.
11–5	—	Reserved
4	DLPEM	Data link protocol error mask.

Table 16-100. PCI Express Uncorrectable Error Mask Register Field Description (continued)

Bits	Name	Description
3–1	—	Reserved
0	TEM	Training error mask.

16.3.10.4 PCI Express Uncorrectable Error Severity Register—0x10C

The PCI Express uncorrectable error severity register is shown in Figure 16-105.

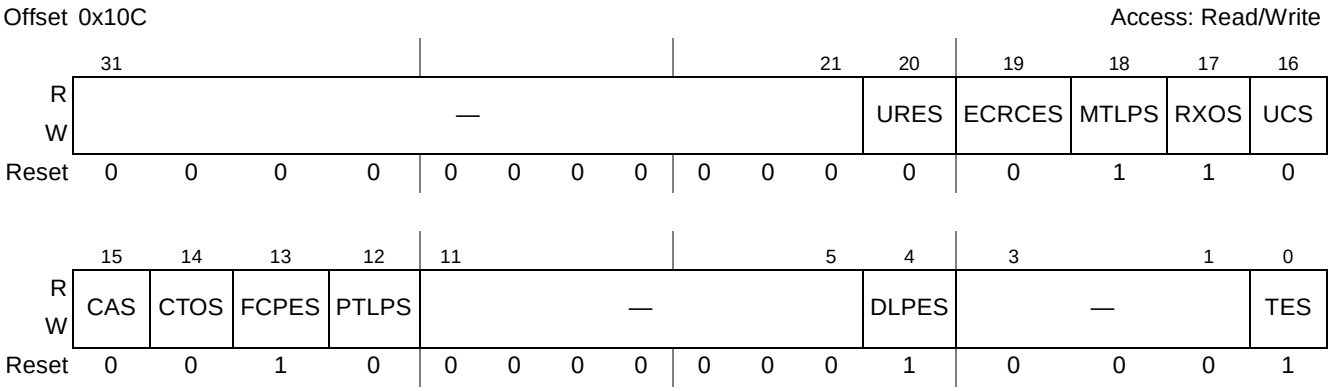


Figure 16-105. PCI Express Uncorrectable Error Severity Register

Table 16-101. PCI Express Uncorrectable Error Severity Register Field Description

Bits	Name	Description
31–21	—	Reserved
20	URES	Unsupported request error severity.
19	ECRCES	ECRC error severity.
18	MTLPS	Malformed TLP severity.
17	RXOS	Receiver overflow severity.
16	UCS	Unexpected completion severity.
15	CAS	Completer abort severity.
14	CTOS	Completion timeout severity.
13	FCPES	Flow control protocol error severity.
12	PTLPS	Poisoned TLP severity.
11–5	—	Reserved
4	DLPES	Data link protocol error severity.
3–1	—	Reserved
0	TES	Training error severity.

16.3.10.5 PCI Express Correctable Error Status Register—0x110

The PCI Express correctable error status register is shown in Figure 16-106.

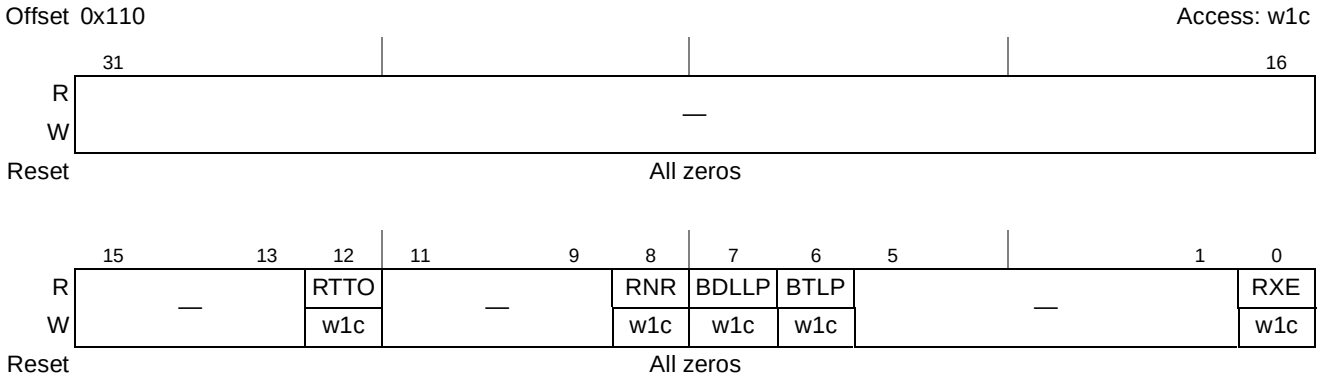


Figure 16-106. PCI Express Correctable Error Status Register

Table 16-102. PCI Express Correctable Error Status Register Field Description

Bits	Name	Description
31–13	—	Reserved
12	RTTO	Replay timer timeout status
11–9	—	Reserved
8	RNR	REPLAY_NUM Rollover status
7	BDLLP	Bad DLLP status
6	BTLP	Bad TLP status
5–1	—	Reserved
0	RXE	Receiver error status

16.3.10.6 PCI Express Correctable Error Mask Register—0x114

The PCI Express correctable error mask register is shown in Figure 16-107.

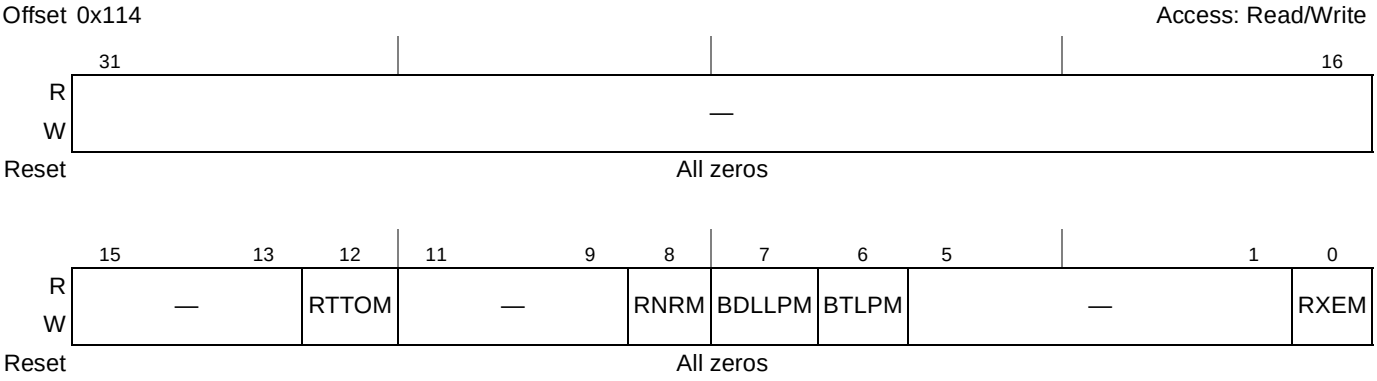


Figure 16-107. PCI Express Correctable Error Mask Register

Table 16-103. PCI Express Correctable Error Mask Register Field Description

Bits	Name	Description
31–13	—	Reserved
12	RTTOM	Replay timer timeout mask
11–9	—	Reserved
8	RNRM	REPLAY_NUM Rollover mask
7	BDLLPM	Bad DLLP mask
6	BTLPM	Bad TLP mask
5–1	—	Reserved
0	RXEM	Receiver error mask

16.3.10.7 PCI Express Advanced Error Capabilities and Control Register—0x118

The PCI Express advanced error capabilities and control register is shown in Figure 16-108.

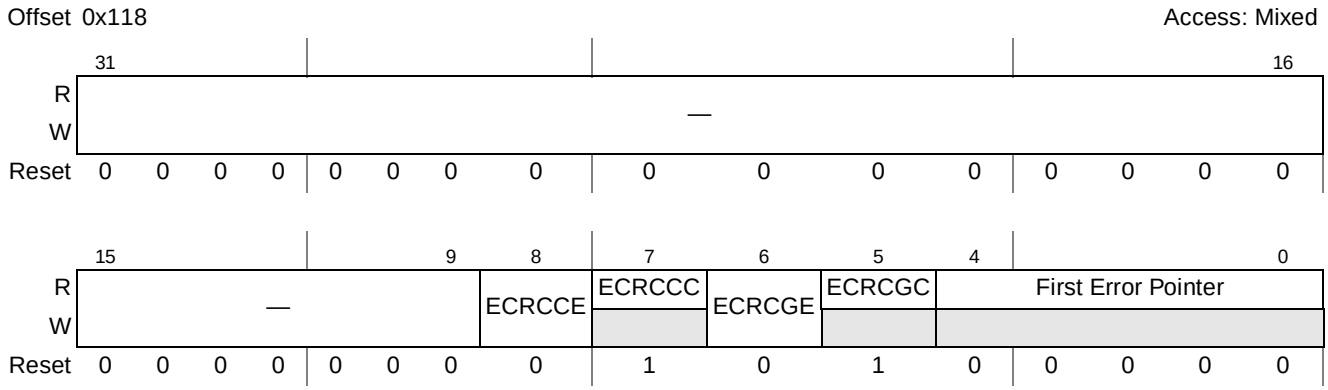


Figure 16-108. PCI Express Advanced Error Capabilities and Control Register

Table 16-104. PCI Express Advanced Error Capabilities and Control Register Field Description

Bits	Name	Description
31–9	—	Reserved.
8	ECRCCE	ECRC checking enable.
7	ECRCCC	ECRC checking capable.
6	ECRCGE	ECRC generation enable.
5	ECRCGC	ECRC generation capable.
4–0	First Error Pointer	The First Error Pointer is a read-only register that identifies the bit position of the first error reported in the Uncorrectable Error Status register.

16.3.10.8 PCI Express Header Log Register—0x11C–0x12B

The PCI Express header log register is shown in Figure 16-109.

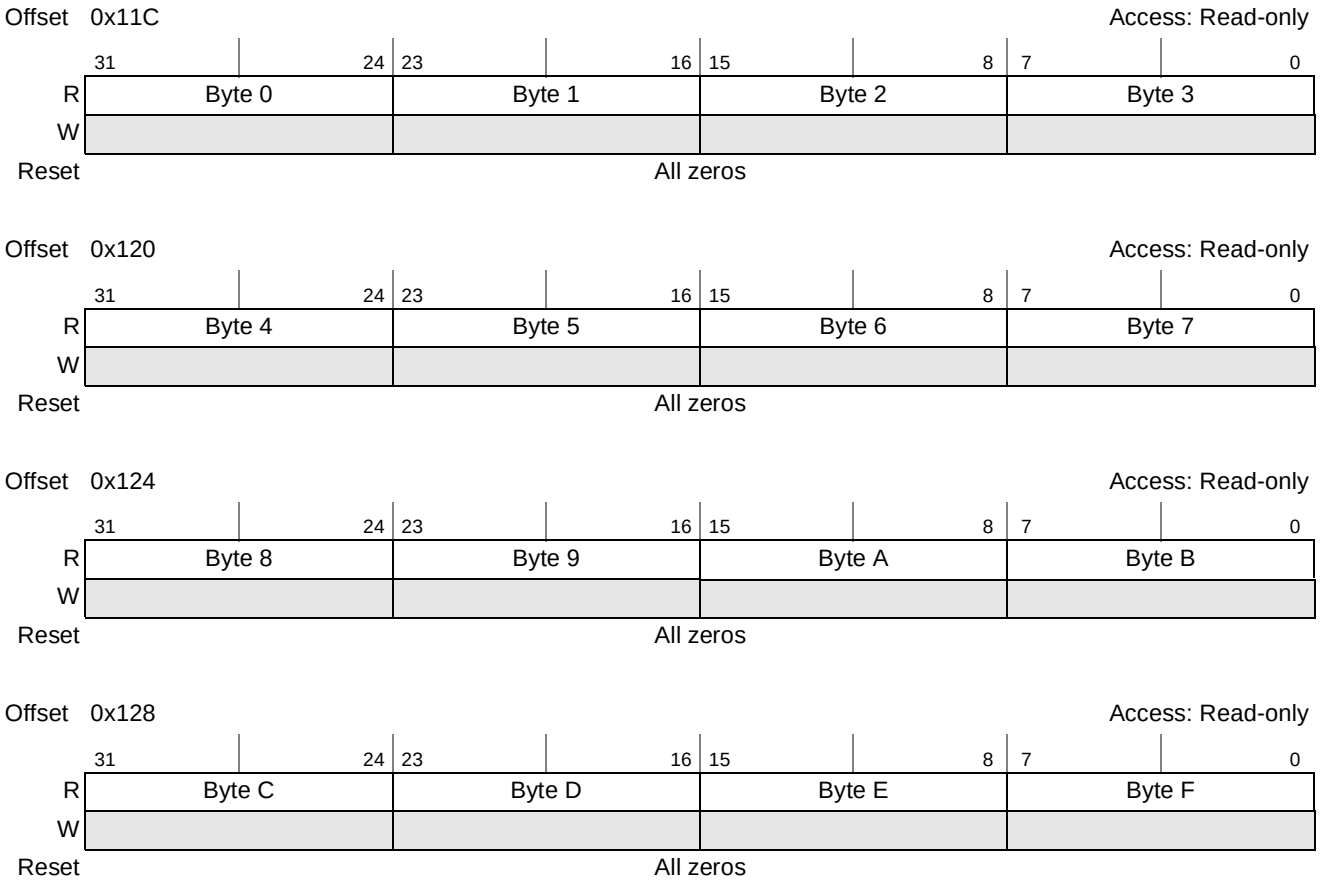


Figure 16-109. PCI Express Header Log Register

Table 16-105. PCI Express Header Log Register Field Description

Bits	Name	Description
127–0	Header Log	Header of TLP associated with error.

16.3.10.9 PCI Express Root Error Command Register—0x12C

The PCI Express root error command register is shown in Figure 16-110.

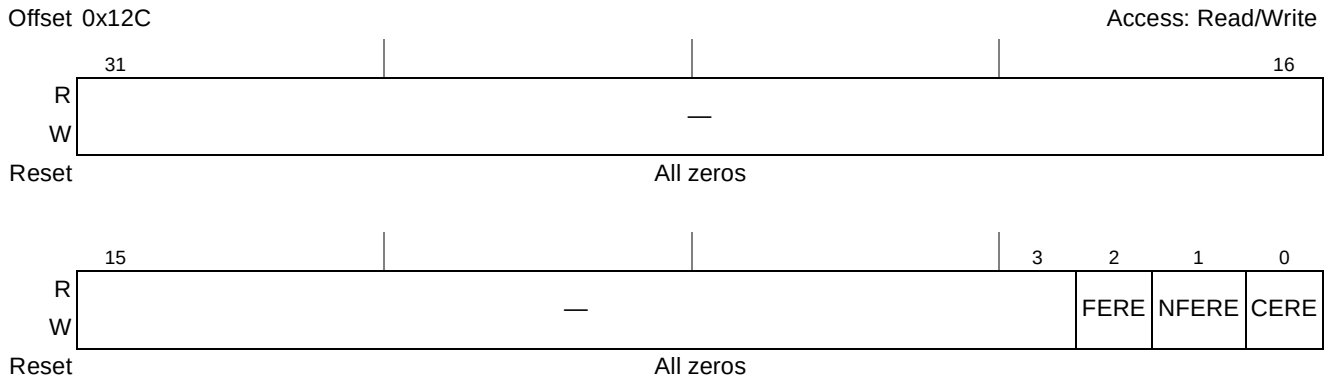


Figure 16-110. PCI Express Root Error Command Register

Table 16-106. PCI Express Root Error Command Register Field Description

Bits	Name	Description
31–3	—	Reserved
2	FERE	Fatal error reporting enable.
1	NFERE	Non-fatal error reporting enable
0	CERE	Correctable error reporting enable

16.3.10.10 PCI Express Root Error Status Register—0x130

The PCI Express root error status register is shown in Figure 16-111.

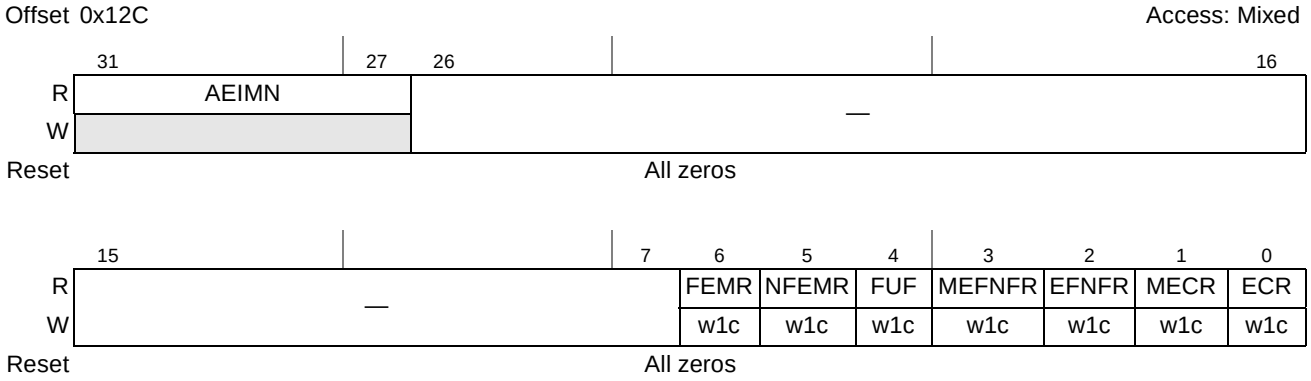


Figure 16-111. PCI Express Root Error Status Register

Table 16-107. PCI Express Root Error Command Status Field Description

Bits	Name	Description
31–27	AEIMN	Advanced error interrupt message number.
26–7	—	Reserved

Table 16-107. PCI Express Root Error Command Status Field Description (continued)

Bits	Name	Description
6	FEMR	Fatal error messages received.
5	NFEMR	Non-fatal error messages received.
4	FUF	First uncorrectable fatal.
3	MEFNFR	Multiple ERR_FATAL/NONFATAL received.
2	EFNFR	ERR_FATAL/NONFATAL received.
1	MECR	Multiple ERR_COR received.
0	ECR	ERR_COR received.

16.3.10.11 PCI Express Correctable Error Source ID Register—0x134

The PCI Express correctable error source ID register is shown in Figure 16-112.

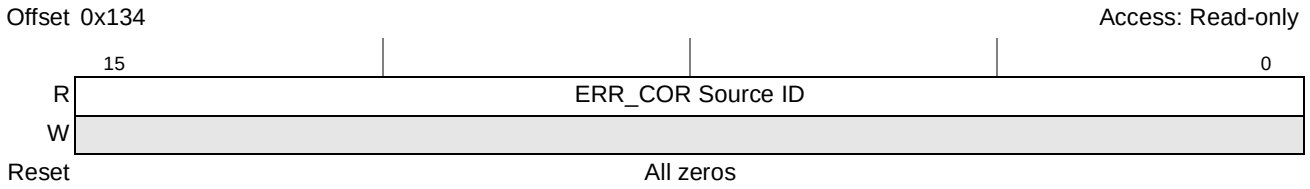


Figure 16-112. PCI Express Correctable Error Source ID Register

Table 16-108. PCI Express Correctable Error Source ID Register Field Description

Bits	Name	Description
15–0	ERR_COR Source ID	Loaded with the Requestor ID indicated in the received ERR_COR Message when the ERR_COR Received register is not already set. Default value of this field is 0.

16.3.10.12 PCI Express Error Source ID Register—0x136

The PCI Express error source ID register is shown in Figure 16-113.

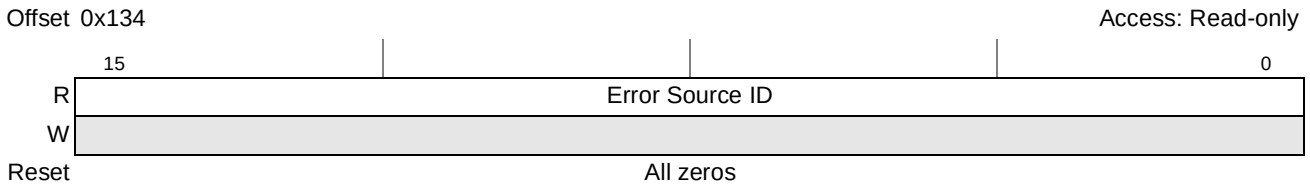


Figure 16-113. PCI Express Correctable Error Source ID Register

Table 16-109. PCI Express Correctable Error Source ID Register Field Description

Bits	Name	Description
15–0	Error Source ID	ERR_FATAL/NONFATAL source ID

16.3.10.13 LTSSM State Status Register—0x404

The PCI Express link training and status state machine (LTSSM) state status register, shown in Figure 16-114, provides detailed information about link training status. This register is useful for debugging link training failures.

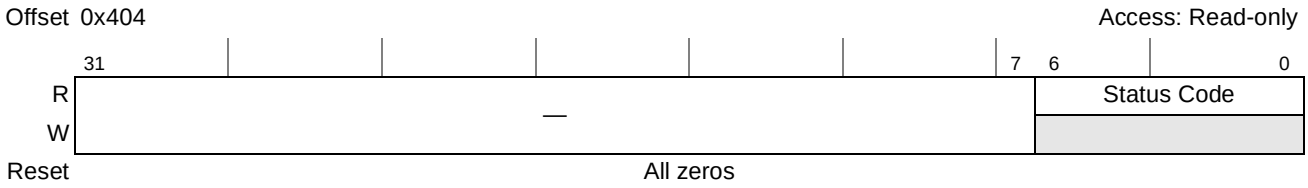


Figure 16-114. PCI Express LTSSM State Status Register (PEX_LTSSM_STAT)

The fields of the PCI Express LTSSM state status register are described in Table 16-110.

Table 16-110. PEX_LTSSM_STAT Field Descriptions

Bits	Name	Description
31–7	—	Reserved
6–0	Status code	Status code. See Table 16-111 for encodings.

Table 16-111 provides the encodings for the status code field of the PEX_LTSSM_STAT register.

Table 16-111. PEX_LTSSM_STAT Status Codes

Status Code (Hex)	LTSSM State Description	Status Code (Hex)	LTSSM State Description
00	Detect quiet	27	TX L0s FTS; RX L0s FTS
01	Detect active (0)	28	L0 to L1 (0)
02	Detect active (1)	29	L0 to L1 (1)
03	Detect active (2)	2A	L1 entry
04	Polling active (0)	2B	L1 idle (0)
05	Polling active (1)	2C	L1 idle (1)
06	Polling config (0)	2D	L0 to L2 (0)
07	Polling config (1)	2E	L0 to L2 (1)
08	Polling compliance	2F	L2 entry
09	Configuration link width start (0)	30	L2 idle (0)
0A	Configuration link width start (1)	31	L2 idle (1)
0B	Configuration link width accept (0)	32	Recovery lock (0)
0C	Configuration link width accept (1)	33	Recovery lock (1)
0D	Configuration lane number wait (0)	34	Recovery lock (2)
0E	Configuration lane number wait (1)	35	Recovery cfg (0)
0F	Configuration lane number wait (2)	36	Recovery cfg (1)

Table 16-111. PEX_LTSSM_STAT Status Codes (continued)

Status Code (Hex)	LTSSM State Description	Status Code (Hex)	LTSSM State Description
10	Configuration lane number wait (3)	37	Recovery idle (0)
11	Configuration lane number accept	38	Recovery idle (1)
12	Configuration complete (0)	39	Recovery to configuration
13	Configuration complete (1)	3A	Recovery cfg to configuration
14	Configuration idle (0)	3F	L0 no training
15	Configuration idle (1)	7F	Detect quiet EI
16	L0	49	Configuration link width start—RC
17	TX L0; RX L0s entry	4A	Configuration link width accept—RC
18	TX L0; RX L0s idle	4B	Configuration lane number wait—RC
19	TX L0; RX L0s fast training sequence (FTS)	4C	Configuration lane number accept—RC
1A	TX L0s entry (0); RX L0	60	Loopback slave active (0)
1B	TX L0s entry (0); RX L0s idle	61	Loopback slave active (1)
1C	TX L0s entry (0); RX L0s FTS	62	Loopback slave exit
1D	TX L0s entry (1); RX L0	68	Hot reset (0)
1E	TX L0s entry (1); RX L0s idle	69	Hot reset (1)
1F	TX L0s entry (1); RX L0s FTS	6A	Hot reset (0)—RC
20	TX L0s idle; RX L0	6B	Hot reset (1)—RC
21	TX L0s idle; RX L0s entry	75	Disabled (0)
22	TX L0s idle; RX L0s idle	71	Disabled (1)
23	TX L0s idle; RX L0s FTS	72	Disabled (2)
24	TX L0s FTS; RX L0	73	Disabled (3)
25	TX L0s FTS; RX L0s entry	74	Disabled (4)
26	TX L0s FTS; RX L0s idle	78	L0 to L1/L2—RC

16.3.10.14 PCI Express Controller Core Clock Ratio Register—0x440

The PCI Express controller clock frequency is determined by the platform clock frequency and the state of the cfg_platform_freq configuration signal at reset, according to the following expression:

$$\text{PCI Express Controller Clock Frequency} = \frac{\text{Platform Clock Frequency}}{(1 + \text{cfg\_platform\_freq})}$$

Therefore, if the platform clock frequency is less than or equal to 400 MHz, then cfg_platform_freq is 0 and the PCI Express controller clock frequency is the same as the platform clock frequency; if the platform clock frequency is greater than or equal to 500 MHz, then cfg_platform_freq is 1 and the PCI Express controller clock frequency is one-half the platform clock frequency.

The PCI Express controller core clock ratio register, shown in Figure 16-115, is used to program the ratio of the actual PCI Express controller clock frequency to the default controller core frequency (333 MHz). This is required only when a PCI Express controller clock frequency other than the default 333 MHz has to be used.

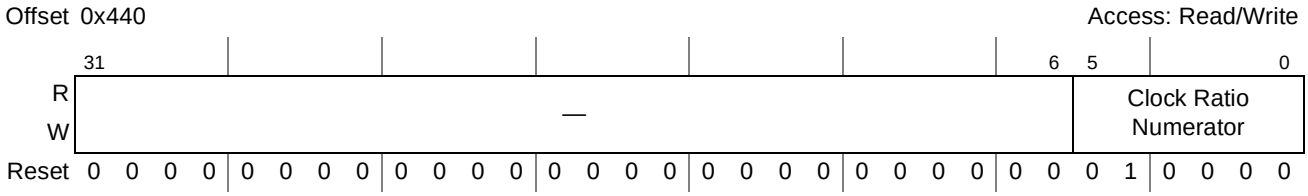


Figure 16-115. PCI Express IP Block Core Clock Ratio Register (PEX_GCLK_RATIO)

The fields of the PCI Express IP block core clock ratio register are described in Table 16-112.

Table 16-112. PEX_GCLK_RATIO Field Descriptions

Bits	Name	Description
31–6	—	Reserved
5–0	Clock Ratio Numerator	The numerator of the ratio of the actual PCI Express controller clock frequency used to the default core clock frequency of 333 MHz. The denominator of the ratio is fixed at 16. The default value of this register is 0x10 (16 decimal), which corresponds to a ratio of 1:1 (or 16/16)

As an example of programming PEX_GCLK_RATIO, consider the case where the actual PCI Express controller clock is 250 MHz, the ratio of the actual clock to the default clock (333 MHz) is 3:4. that is, the default core clock has to be multiplied by the ratio (3/4, which is equivalent to 12/16). So the register has to be programmed with the decimal numerator value 12 or 0x0000_000C.

16.3.10.15 PCI Express Power Management Timer Register—0x450

The PCI Express power management timer register, shown in Figure 16-116, is used to program the time-in values for entering L0s and L1 power management states.

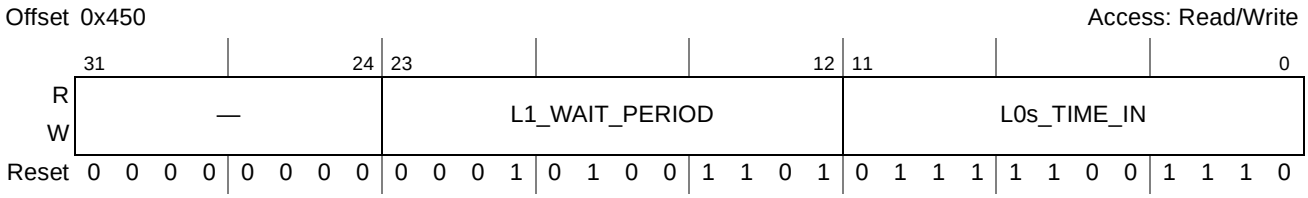


Figure 16-116. PCI Express Power Management Timer Register (PEX_PM_TIMER)

The fields of the PCI Express power management timer register are described in Table 16-113.

Table 16-113. PEX_PM_TIMER Field Descriptions

Bits	Name	Description
31–24	—	Reserved
23–12	L1_WAIT_PERIOD	Wait period (in PCI Express controller core clock cycles) before entering L1 power state after all functions are in a non-D0 power state. The value is calculated as: Time (in µsec) × PCI Express controller core clock frequency (in MHz) The time value must be less than 2 µsec; the default value (0x14D) is 1 µsec for the default clock frequency of 333 MHz.
11–0	L0s_TIME_IN	Time in value (in PCI Express controller core clock cycles) for entering L0s power state. The value is calculated as: Time (in µsec) × PCI Express controller core clock frequency (in MHz) The maximum time value is 7 µsec; the default value (0x7CE) is 6 µsec for the default clock frequency of 333 MHz.

16.3.10.16 PCI Express PME Time-Out Register (EP-Mode Only)—0x454

The PCI Express PME time-out register, shown in Figure 16-117, is used to program the time-out value that the controller uses before re-sending a PME message to the host. If PME is requested by a function and the host does not clear the associated PME_STAT bit even after this time-out has expired, the PME message is sent again to the host by the PCI Express controller. This register is supported only for EP mode.

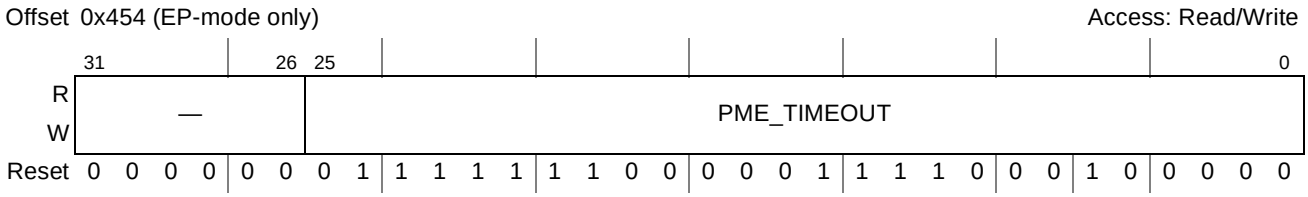


Figure 16-117. PCI Express PME Time-Out Register (PEX_PME_TIMEOUT)

The fields of the PCI Express PME time-out register are described in Table 16-113.

Table 16-114. PEX_PME_TIMEOUT Field Descriptions

Bits	Name	Description
31–26	—	Reserved
25–0	PME_TIMEOUT	The PME time-out value specifies the interval before PME messages are resent by the controller, provided the PME_STAT bit in the PCI Express power management status and control register (offset 0x48) is not cleared by the host. The value for PME_TIMEOUT is specified in terms of PCI Express controller core clock cycles. The value is calculated as: Time (in µsec) × PCI Express controller core clock frequency (in MHz) The minimum time value is 100 msec; the default value (0x1FC1E20) is 100 msec for the default clock frequency of 333 MHz.

16.3.10.17 PCI Express Subsystem Vendor ID Update Register (EP Mode Only)—0x478

The PCI Express subsystem vendor ID update register, shown in Figure 16-118, is used to set the values for the Subsystem ID and Subsystem Vendor ID registers in the Type 0 configuration header.

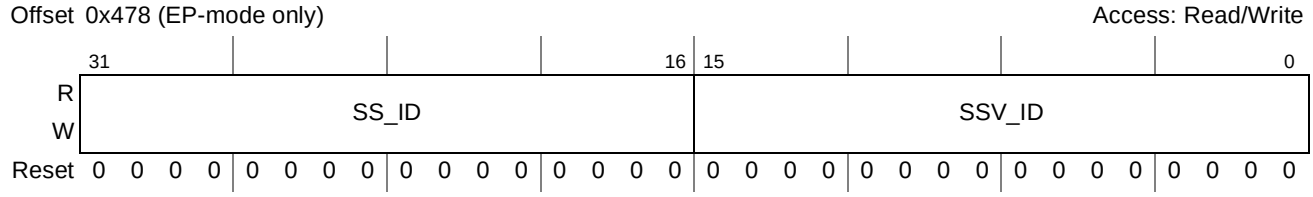


Figure 16-118. PCI Express Subsystem Vendor ID Update Register (PEX_SSVID_UPDATE)

The fields of the PCI Express subsystem vendor ID update register are described in Table 16-115.

Table 16-115. PEX_SSVID_UPDATE Field Descriptions

Bits	Name	Description
31–16	SS_ID	Subsystem ID [15–0] value
15–0	SSV_ID	Subsystem vendor ID [15–0] value

When used as an endpoint, the controller's initialization software programs the desired subsystem ID and subsystem vendor ID values in PEX_SSVID_UPDATE before setting the CFG_READY bit in the PEX_CFG_READY register (see Section 16.3.10.18, "Configuration Ready Register—0x4B0"). That way, when the host begins system enumeration, the correct values are present in the Type 0 configuration header.

16.3.10.18 Configuration Ready Register—0x4B0

The PCI Express configuration ready register, shown in Figure 16-119, is used to indicate configuration complete status to the transaction layer. The transaction layer handles configuration requests from external hosts only after the CFG_READY bit is set. All the configuration requests received from external hosts before the CFG_READY bit is set are completed with configuration request retry status (CRS). The CFG_READY bit in this register should be set after all relevant configuration registers have been programmed. This makes sure the external host reads the correct capability advertisements during enumeration.

Note that the state of PEX_CFG_READY[CFG_READY] is dependent upon the POR configuration setting described in Section 16.5.1, "Boot Mode and Inbound Configuration Transactions."

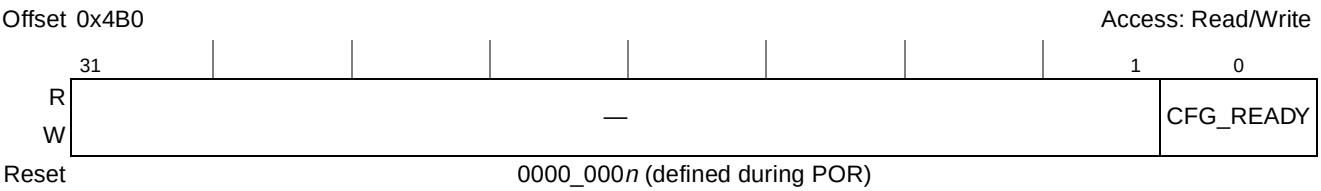


Figure 16-119. PCI Express Configuration Ready Register (PEX_CFG_READY)

The fields of the PCI Express configuration ready register are described in Table 16-116.

Table 16-116. PEX_CFG_READY Field Descriptions

Bits	Name	Description
31–1	—	Reserved
0	CFG_READY	Configuration ready 1 The transaction layer accepts inbound configuration requests. 0 The transaction layer responds to all inbound configuration requests with retry (CRS) Note that the reset state of this bit is determined during POR.

16.3.10.19 PME_To_Ack Timeout Register (RC-Mode Only)—0x590

The PCI Express PME_To_Ack timeout register, shown in Figure 16-120, is used to program the timeout value for a PME_To_Ack message response in terms of PCI Express controller core clock cycles.

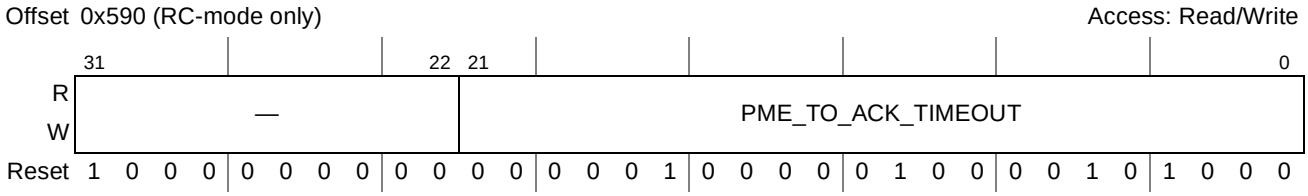


Figure 16-120. PCI Express PME_To_Ack Timeout Register (PEX_PME_TO_ACK_TOR)

The fields of the PCI Express PME_To_Ack timeout register are described in Table 16-117.

Table 16-117. PEX_PME_TO_ACK_TOR Field Descriptions

Bits	Name	Description
31–22	—	Reserved
21–0	PME_TO_ACK_TIMEOUT	After a PME_Turn_Off message is broadcast by the RC, the power management module waits for the duration of the PME_To_Ack timeout interval to receive a PME_To_Ack message from the downstream device. If the Ack message is not received within this interval, the power manager indicates that it is safe to switch off power, since timeout has occurred. The value is calculated as: Time (in µsec) × PCI Express controller core clock frequency (in MHz) The recommended timeout duration is 1 msec to 10 msec to make sure that the downstream devices get enough time to prepare for power-off condition.

16.3.10.20 Secondary Status Interrupt Mask Register (RC-Mode Only)—0x5A0

The PCI Express secondary status interrupt mask register, shown in Figure 16-121, can be used to disable sideband interrupt generation when error bits in the PCI Express secondary status register are set. See Section 16.3.8.3.8, “PCI Express Secondary Status Register—0x1E,” for more information. By default, interrupt generation due to secondary status errors is disabled.

Offset 0x5A0 (RC-mode only) Access: Read/Write

	31–6	5	4	3	2	1	0
R W	—	M_DPE	M_SSE	M_RMA	M_RTA	M_STA	M_MDPE
Reset	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	1	1	1	1	1	1

Figure 16-121. PCI Express PCI Interrupt Mask Register (PEX_SS_INTR_MASK)

The fields of the PCI Express secondary status interrupt mask register are described in Table 16-118.

Table 16-118. PEX_SS_INTR_MASK Field Descriptions

Bits	Name	Description
31–6	—	Reserved
5	M_DPE	Mask detected parity error
4	M_SSE	Mask signaled system error
3	M_RMA	Mask received master abort
2	M_RTA	Mask received target abort
1	M_STA	Mask signaled target abort
0	M_MDPE	Mask master data parity error

16.4 Functional Description

The PCI Express protocol relies on a requestor/completer relationship where one device requests that some desired action be performed by some target device and the target device completes the task and responds. Usually the requests and responses occur through a network of links, but to the requestor and to the completer, the intermediate components are transparent.

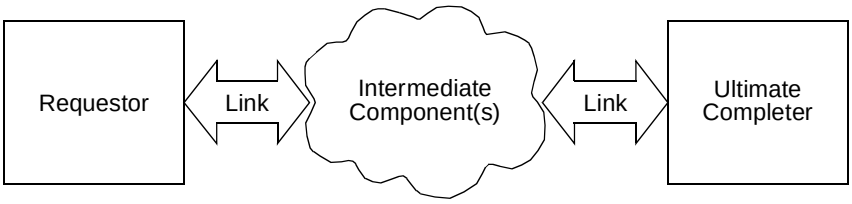


Figure 16-122. Requestor/Completer Relationship

Each PCI Express device is divided into two halves-transmit (TX) and receive (RX), and each of these halves is further divided into three layers—transaction, data link, and physical—as shown in Figure 16-123.

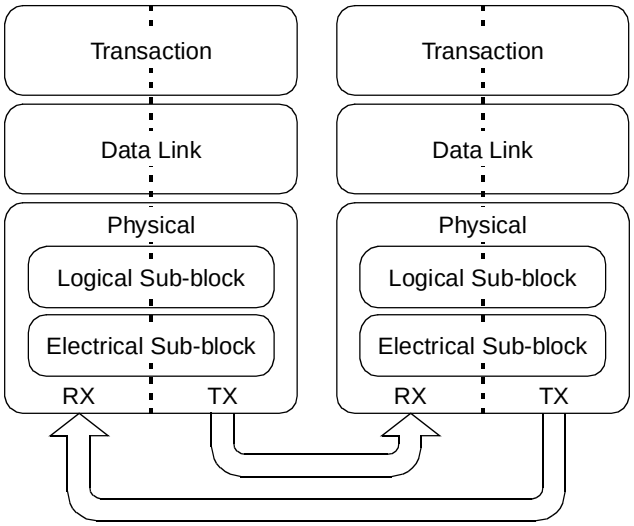


Figure 16-123. PCI Express High-Level Layering

Packets are formed in the transaction layer (TLPs) and data link layer (DLLPs), and each subsequent layer adds the necessary encodings and framing—as shown in Figure 16-124. As packets are received, they are decoded and processed by the same layers but in reverse order, so they may be processed by the layer or by the device application software.

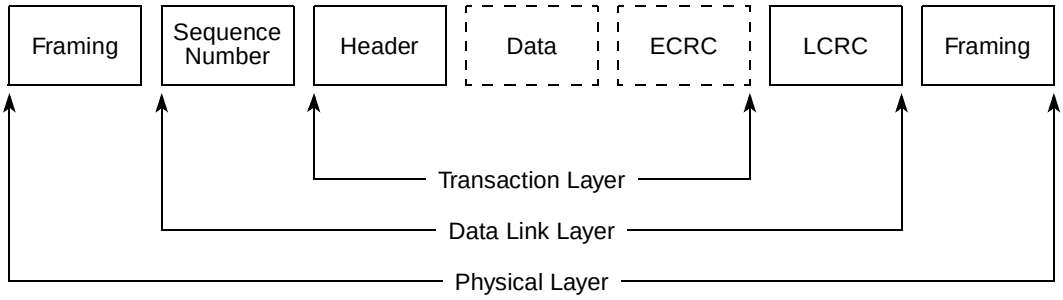


Figure 16-124. PCI Express Packet Flow

16.4.1 Architecture

This section describes implementation details of the PCI Express controller.

16.4.1.1 PCI Express Transactions

Table 16-119 contains the list of transactions that the PCI Express controller supports as an initiator and a target.

Table 16-119. PCI Express Transactions

PCI Express Transaction	Supported as an Initiator	Supported as a Target	Definition
Mrd	Yes	Yes	Memory Read Request
MRdLk	No	No	Memory Read Lock. As a target, CplLk with UR status is returned.
MWr	Yes	Yes	Memory Write Request to memory-mapped PCI-Express space
IORd	Yes (RC only)	No	I/O Read request. As a target, Cpl with UR status is returned.
IOWr	Yes (RC only)	No	I/O Write Request. As a target, Cpl with UR status is returned.
CfgRd0	Yes (RC only	Yes	Configuration Read Type 0
CfgWr0	Yes (RC only)	Yes	Configuration Write Type 0
CfgRd1	Yes (RC only)	No	Configuration Read Type 1. As a target, Cpl with UR status is returned.
CfgWr1	Yes (RC only)	No	Configuration Write Type 1. As a target, Cpl with UR status is returned.
Msg	Yes	Yes	Message Request
MsgD	Yes (RC only)	Yes (EP only)	Message Request with Data payload. Note that Set_Slot_Power_Limit is the only message with data that is supported and then only when the controller is an initiator and in RC mode or a target and in EP mode.
Cpl	Yes	Yes	Completion without Data
CplD	Yes	Yes	Completion with Data
CplLk	No	Yes	Completion for Locked Memory Read without Data. The only time that CplLk is returned with UR status is when the controller receives a MRdLk command.
CplDLk	No	No	Completion for Locked Memory Read with Data

16.4.1.2 Byte Ordering

Whenever data must cross a bridge between two busses, the byte ordering of data on the source and destination buses must be considered. The internal platform bus of this device is inherently big endian and the PCI Express bus interface is inherently little endian.

There are two methods to handle ordering of data as it crosses a bridge—address invariance and data invariance. Address invariance preserves the addressing of bytes within a scalar data element, but not the relative significance of the bytes within that scalar. Conversely, data invariance preserves the relative significance of bytes within a scalar, but not the addressing of the individual bytes that make up a scalar.

This device uses address invariance as its byte ordering policy.

As stated above, address invariance preserves the byte address of each byte on an I/O interface as it is placed in memory or moved into a register. This policy can have the effect of reversing the significance order of bytes (most significant to least significant and vice versa), but it has the benefit of preserving the format of general data structures. Provided that software is aware of the endianness and format of the data structure, it can correctly interpret the data on either side of the bridge.

Figure 16-125 shows the transfer of a 4-byte scalar, 0x4142_4344, from a big endian source across an address invariant bridge to a little endian destination.

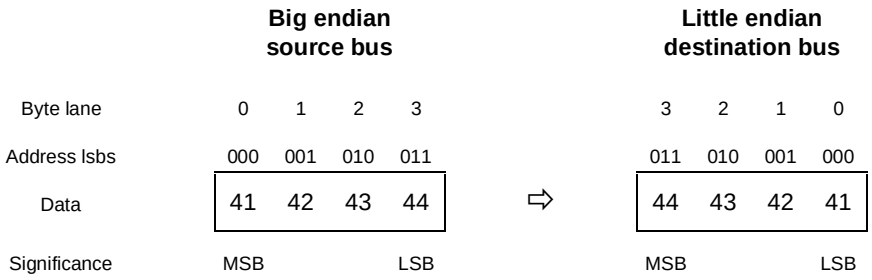


Figure 16-125. Address Invariant Byte Ordering—4 Bytes Outbound

Note that although the significance of the bytes within the scalar have changed, the address of the individual bytes that make up the scalar have not changed. As long as software is aware that the source of the data used a big endian format, the data can be interpreted correctly.

Figure 16-127 shows data flowing the other way, from a little endian source to a big endian destination.

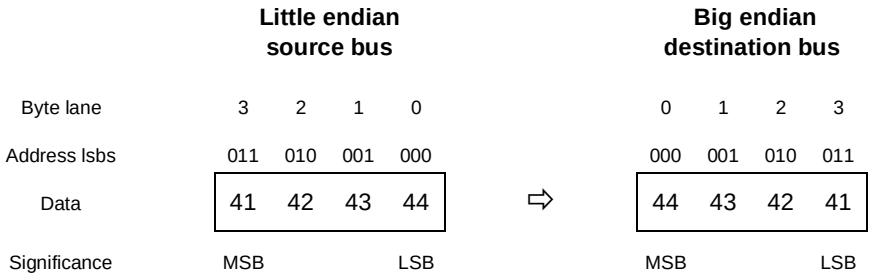


Figure 16-126. Address Invariant Byte Ordering—4 Bytes Inbound

Figure 16-127 shows an outbound transfer of an 8-byte scalar, 0x5455_1617_CDCE_2728, using address invariance.

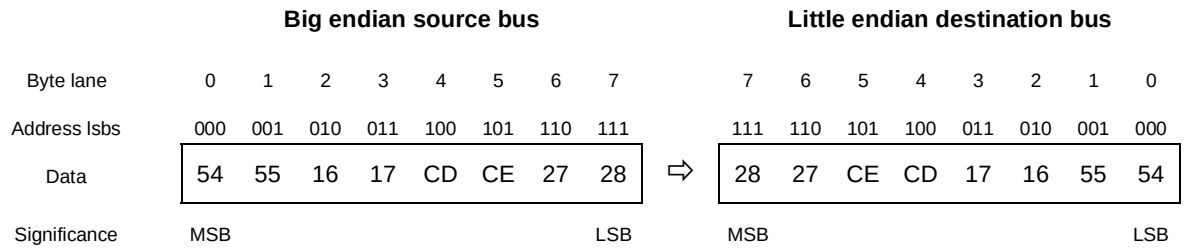


Figure 16-127. Address Invariant Byte Ordering—8 Bytes Outbound

Figure 16-128 shows an inbound transfer of a 2-byte scalar, 0x5837, using address invariance.

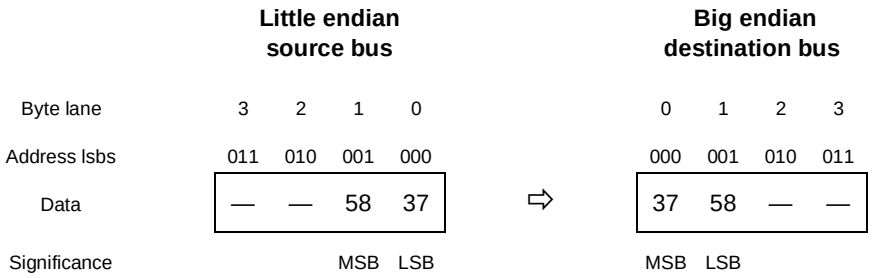


Figure 16-128. Address Invariant Byte Ordering—2 bytes Inbound

Note that in all of these examples, the original addresses of the individual bytes within the scalars (as created by the source) have been preserved.

16.4.1.2.1 Byte Order for Configuration Transactions

All internal memory-mapped registers in the CCSR space use big endian byte ordering. However, the PCI Express specification defines PCI Express configuration registers as little endian. All accesses to the PCI Express configuration port, PEX_CONFIG_DATA, including the those targeting the internal PCI Express configuration registers, use the address invariance policy as shown in Figure 16-129. Therefore, software must access PEX_CONFIG_DATA with little-endian formatted data—either using the **lwbrx**/**stwbrx** instructions or by manipulating the data before writing to and after reading from PEX_CONFIG_DATA.

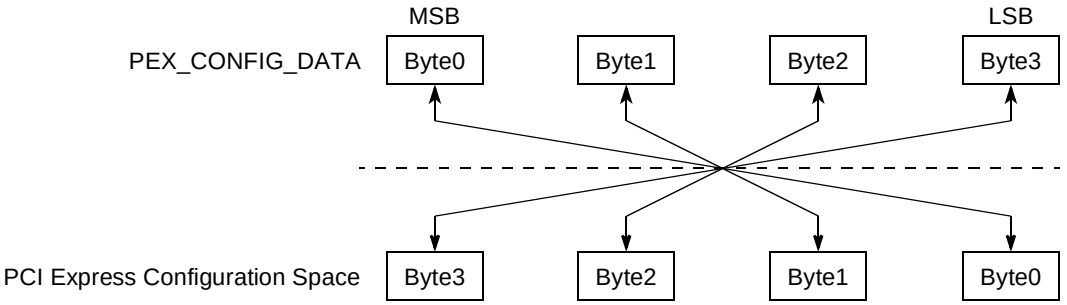


Figure 16-129. PEX_CONFIG_DATA Byte Ordering

16.4.1.3 Lane Reversal

The PCI Express link supports lane reversal. Table 16-120 describes the supported configurations.

Table 16-120. Lane Assignment With and Without Lane Reversal

Link Configuration	Lane 0	Lane 1	Lane 2	Lane 3	Lane 4	Lane 5	Lane 6	Lane 7
x8 link without lane reversal	0	1	2	3	4	5	6	7
x4 link without lane reversal	0	1	2	3	—	—	—	—
x2 link without lane reversal	0	1	—	—	—	—	—	—
x1 link without lane reversal	0	—	—	—	—	—	—	—
x8 link with lane reversal	7	6	5	4	3	2	1	0

Table 16-120. Lane Assignment With and Without Lane Reversal (continued)

Link Configuration	Lane 0	Lane 1	Lane 2	Lane 3	Lane 4	Lane 5	Lane 6	Lane 7
x4 link with lane reversal	—	—	—	—	3	2	1	0
x2 link with lane reversal	—	—	—	—	—	—	1	0
x1 link with lane reversal	—	—	—	—	—	—	—	0

Note: The numbers shown in this table (0–7) are the lane numbers assigned to each lane as a result of link initialization and configuration.
— indicates that the lane is not part of the configured link.

Note that lane reversal is only effective for devices that use the full 8 lanes. That is, if a x4 device is connected to lanes 0–3 and the link training fails without lane reversal, the lane reversal causes the link to attempt connection on lanes 7–4 which would be impossible.

16.4.1.4 Transaction Ordering Rules

In general, transactions are serviced in the order that they are received. However, transactions can be reordered as they are sent due to a stalled condition such as a full internal buffer. The following are the ordering rules for sending the next outstanding request:

- A posted request can and bypasses all other transactions except another posted request.
- A completion can and only bypasses non-posted. It can and bypasses posted requests only if the relaxed ordering (RO) bit is set.
- A non-posted request cannot bypass posted or other non-posted requests, but it can bypass a completion if the relaxed ordering (RO) bit is set.

16.4.1.5 Memory Space Addressing

A PCI Express memory transaction can address a 32- or 64-bit memory space. The FMT[0] field in the PCI Express TLP header for a 32-bit address packet is 0; a 64-bit address packet has a FMT[0] = 1. The PCI Express TLP header for a memory read transaction has TYPE[4:0] = 00000 and FMT[1] = 0. A memory write transaction has TYPE[4:0] = 00000 and FMT[1] = 1. As an initiator, the controller is capable of sending 32- or 64-bit memory packets. Any transaction from the internal platform that (after passing through the translation mechanism) has a translated address greater than 4G is sent as a 64-bit memory packet. Otherwise, a 32-bit memory packet is sent. As a target device, the controller is capable of decoding 32- or 64-bit memory packets. This is done through two 32-bit inbound windows and two 64-bit inbound windows. All inbound addresses are translated to 36-bit internal platform addresses.

16.4.1.6 I/O Space Addressing

The controller does not support I/O transactions as a target. As an initiator, the controller can send I/O transactions in RC mode only. This can be done by programming one of the outbound translation window's attribute to send I/O transactions. All I/O transactions only access 32-bit address I/O space. The PCI Express TLP header for an I/O read transaction has TYPE[4:0] = 00010 and FMT[1] = 0. The PCI Express TLP header for an I/O write transaction has TYPE[4:0] = 00010 and FMT[1] = 1.

16.4.1.7 Configuration Space Addressing

As an initiator, the controller supports both type 0 and type 1 configuration cycles when configured in RC mode. There are two methods of generating a configuration transaction; refer to Section 16.3.7, "PCI Express Configuration Space Access," for more information. A configuration transaction can hit into the controller's internal configuration space, it can be sent out on the PCI Express link, or it can be internally terminated. The PCI Express TLP header for a type 0 configuration read transaction has TYPE[4:0] = 00100 and FMT[1] = 0; the PCI Express TLP header for a type 0 configuration write transaction has TYPE[4:0] = 00100 and FMT[1] = 1. The PCI Express TLP header for a type 1 configuration read transaction has TYPE[4:0] = 00101 and FMT[1] = 0; the PCI Express TLP header for a type 1 configuration write transaction has TYPE[4:0] = 00101 and FMT[1] = 1. Note that all configuration transactions sent on PCI Express require a response regardless whether they are read or a write configuration transactions.

The controller does not generate configuration transactions in EP mode. Only inbound configuration transactions are supported in EP mode.

16.4.1.8 Serialization of Configuration and I/O Writes

Configuration and I/O writes originating from the PCI Express outbound ATMUs are serialized by the controller. The logic after issuing a configuration write or IO write does not issue any new transactions until the outstanding configuration or I/O write is finished. This means that an acknowledgement packet from the link partner in the form of a CpL TLP packet must be seen or the transaction has timed out. If the CpL packet contains a CRS status, then the logic re-issues the configuration write transaction. It keeps retrying the request until either a status other than CRS is returned or the transaction times out. Note that configuration writes originating from the PCI Express configuration access registers (PEX_CONFIG_ADDR/PEX_CONFIG_DATA) are not serialized.

Note that it is possible for outbound configuration read request to be requeued and be placed at the end of the request queue due to CRS condition.

16.4.1.9 Messages

Software message generation is supported in both RC and EP modes.

16.4.1.9.1 Outbound ATMU Message Generation

Software can choose to send a message by programming PEXOWAR*n*[WTT] = 0x5. A message is sent by writing a 4-byte transaction in big-endian format that hits in an outbound window configured to send messages.

Part of the 4-byte data is used to store information such as message code and routing information. Table 16-121 describes the message data format.

Table 16-121. Internal Platform (OCeaN) Message Data Format

Bits	Name	Reset Value	Description
0–15	—	—	Reserved
16–18	Routing	x	Routing mechanism. Contains the message's routing information
19–23	—	—	Reserved
24–31	Code	x	Message code. Contains the actual message type to be sent.

In addition to the outbound ATMU, the PEX PM Command register also provides the capability to send PME_Turn_Off message or PM_PME message by setting bits 31 or 29. See Section 16.3.3.4, "PCI Express Power Management Command Register (PEX_PMCR)," on page 16-18 for more information.

Table 16-122 provides a complete list of supported outbound messages depending on whether RC or EP is configured.

Table 16-122. PCI Express ATMU Outbound Messages

Name	Code[7:0]	Routing[2:0]	RC	EP	Description
Assert_INTA	0010 0000	100	N/A	Yes	Sent upstream by endpoint
Assert_INTB	0010 0001	100	N/A	Yes	Sent upstream by endpoint
Assert_INTC	0010 0010	100	N/A	Yes	Sent upstream by endpoint
Assert_INTD	0010 0011	100	N/A	Yes	Sent upstream by endpoint
Deassert_INTA	0010 0100	100	N/A	Yes	Sent upstream by endpoint
Deassert_INTB	0010 0101	100	N/A	Yes	Sent upstream by endpoint
Deassert_INTC	0010 0110	100	N/A	Yes	Sent upstream by endpoint
Deassert_INTD	0010 0111	100	N/A	Yes	Sent upstream by endpoint
PM_Active_State_Nak	0001 0100	100	Yes	N/A	Terminate at receiver
PM_PME	0001 1000	000	N/A	Yes	Sent Upstream by PME-requesting component
PME_Turn_Off	0001 1001	011	Yes	N/A	Broadcast Downstream
PM_TO_Ack	0001 1011	101	N/A	Yes	Sent Upstream by Endpoint
ERR_COR	0011 0000	000	N/A	Yes	Sent by component when it detects a correctable error
ERR_NONFATAL	0011 0001	000	N/A	Yes	Sent by component when it detects a Non-fatal, uncorrectable error
ERR_FATAL	0011 0011	000	N/A	Yes	Sent by component when it detects a Fatal, uncorrectable error
Unlock	0000 0000	000	No	N/A	Not supported
Set_Slot_Power_Limit	0101 0000	100	Yes	N/A	Set Slot Power Limit in Upstream Port

Table 16-122. PCI Express ATMU Outbound Messages (continued)

Name	Code[7:0]	Routing[2:0]	RC	EP	Description
Vendor_Defined Type 0	0111 1110	—	No	No	Not supported
Vendor_Defined Type 1	0111 1111	—	No	No	Not supported
Attention_Indicator_On	0100 0001	100	Yes	N/A	Hot-plug message
Attention_Indicator_Blink	0100 0011	100	Yes	N/A	Hot-plug message
Attention_Indicator_Off	0100 0000	100	Yes	N/A	Hot-plug message
Power_Indicator_On	0100 0101	100	Yes	N/A	Hot-plug message
Power_Indicator_Blink	0100 0111	100	Yes	N/A	Hot-plug message
Power_Indicator_Off	0100 0100	100	Yes	N/A	Hot-plug message
Attention_Button_Pressed	0100 1000	100	Yes	N/A	Hot-plug message

16.4.1.9.2 Inbound Messages

Table 16-123 provides a complete list of supported inbound messages in RC mode.

Table 16-123. PCI Express RC Inbound Message Handling

Name	Code[7:0]	Routing[2:0]	Action
Assert_INTA	0010 0000	100	Send to PIC
Assert_INTB	0010 0001	100	Send to PIC
Assert_INTC	0010 0010	100	Send to PIC
Assert_INTD	0010 0011	100	Send to PIC
De-assert_INTA	0010 0100	100	Send to PIC
De-assert_INTB	0010 0101	100	Send to PIC
De-assert_INTC	0010 0110	100	Send to PIC
De-assert_INTD	0010 0111	100	Send to PIC
PM_Active_State_Nak	0001 0100	100	No action taken
PM_PME	0001 1000	000	Generate interrupt to PIC if enabled
PME_Turn_Off	0001 1001	011	No action taken
PME_TO_Ack	0001 1011	101	Set PEX_PME_MES_DR[ENL23] bit and generate interrupt to PIC if enabled
ERR_COR	0011 0000	000	Generate interrupt to PIC if enabled
ERR_NONFATAL	0011 0001	000	Generate interrupt to PIC if enabled
ERR_FATAL	0011 0011	000	Generate interrupt to PIC if enabled
Unlock	0000 0000	000	No action taken
Set_Slot_Power_Limit	0101 0000	100	No action taken
Vendor_Defined Type 0	0111 1110	—	No action taken

Table 16-123. PCI Express RC Inbound Message Handling (continued)

Name	Code[7:0]	Routing[2:0]	Action
Vendor_Defined Type 1	0111 1111	—	No action taken
Attention_Indicator_On	0100 0001	100	No action taken
Attention_Indicator_Blink	0100 0011	100	No action taken
Attention_Indicator_Off	0100 0000	100	No action taken
Power_Indicator_On	0100 0101	100	No action taken
Power_Indicator_Blink	0100 0111	100	No action taken
Power_Indicator_Off	0100 0100	100	No action taken
Attention_Button_Pressed	0100 1000	100	Set PEX_PME_MES_DR[ABP] bit and send interrupt if enabled.

Table 16-124 provides a complete list of supported inbound messages in EP mode.

Table 16-124. PCI Express EP Inbound Message Handling

Name	Code[7:0]	Routing[2:0]	Action
Assert_INTA	0010 0000	100	No action taken
Assert_INTB	0010 0001	100	No action taken
Assert_INTC	0010 0010	100	No action taken
Assert_INTD	0010 0011	100	No action taken
Deassert_INTA	0010 0100	100	No action taken
Deassert_INTB	0010 0101	100	No action taken
Deassert_INTC	0010 0110	100	No action taken
Deassert_INTD	0010 0111	100	No action taken
PM_Active_State_Nak	0001 0100	100	No action taken
PM_PME	0001 1000	000	No action taken
PME_Turn_Off	0001 1001	011	Set PEX_PME_MES_DR[PTO] bit. Send interrupt if enabled.
PM_TO_Ack	0001 1011	101	No action taken
ERR_COR	0011 0000	000	No action taken
ERR_NONFATAL	0011 0001	000	No action taken
ERR_FATAL	0011 0011	000	No action taken
Unlock	0000 0000	000	No action taken
Set_Slot_Power_Limit	0101 0000	100	Update power value in PCI Express device capability register in configuration space.
Vendor_Defined Type 0	0111 1110	—	No action taken
Vendor_Defined Type 1	0111 1111	—	No action taken
Attention_Indicator_On	0100 0001	100	Set PEX_PME_MES_DR[AION] bit. Send interrupt if enabled.

Table 16-124. PCI Express EP Inbound Message Handling (continued)

Name	Code[7:0]	Routing[2:0]	Action
Attention_Indicator_Blink	0100 0011	100	Set PEX_PME_MES_DR[AIB] bit. Send interrupt if enabled.
Attention_Indicator_Off	0100 0000	100	Set PEX_PME_MES_DR[AIOF] bit. Send interrupt if enabled.
Power_Indicator_On	0100 0101	100	Set PEX_PME_MES_DR[PION] bit. Send interrupt if enabled.
Power_Indicator_Blink	0100 0111	100	Set PEX_PME_MES_DR[PIB] bit. Send interrupt if enabled.
Power_Indicator_Off	0100 0100	100	Set PEX_PME_MES_DR[PIOF] bit. Send interrupt if enabled.
Attention_Button_Pressed	0100 1000	100	No action taken

16.4.1.10 Error Handling

The PCI Express specification classifies errors as correctable and uncorrectable. Correctable errors result in degraded performance, but uncorrectable errors generally result in functional failures. As shown in Figure 16-130 uncorrectable errors can further be classified as fatal or non-fatal.

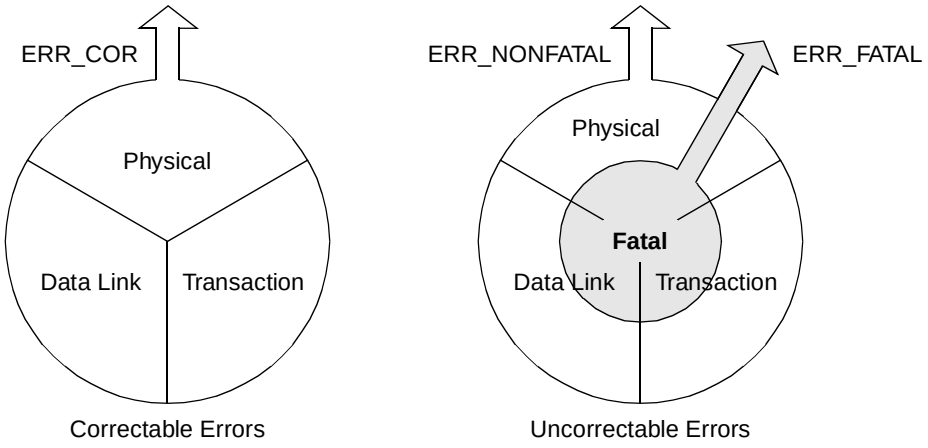


Figure 16-130. PCI Express Error Classification

16.4.1.10.1 PCI Express Error Logging and Signaling

Figure 16-131 shows the PCI Express-defined sequence of operations related to signaling and logging of errors detected by a device. Note that the PCI Express controller on this device supports the advanced error handling capabilities shown within the dotted lines.

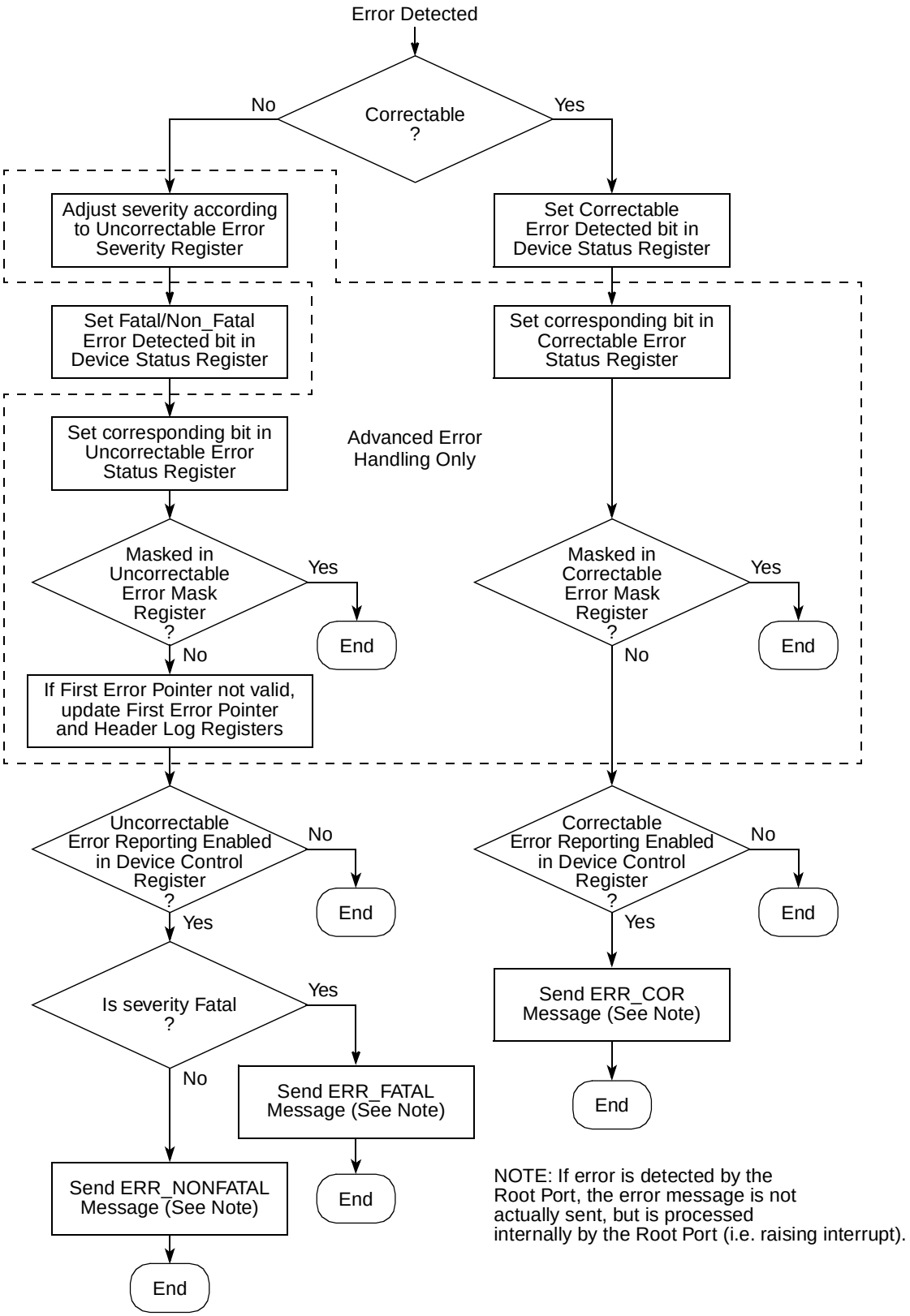


Figure 16-131. PCI Express Device Error Signaling Flowchart

16.4.1.10.2 PCI Express Controller Internal Interrupt Sources

Table 16-125 describes the sources of the PCI Express controller internal interrupt to the PIC and the preconditions for signaling the interrupt.

Table 16-125. PCI Express Internal Controller Interrupt Sources

Status Register Bit	Preconditions
Any bit in PEX_PME_MES_DR set	The corresponding interrupt enable bits must be set in PEX_PME_MES_IER
Any bit in PEX_ERR_DR set	The corresponding interrupt enable bits must be set in PEX_ERR_EN.
PCI Express Root Status Register[16] (PME status) is set	PCI Express Root Control Register [3] (PME interrupt enable) is set
PCI Express Root Error Status Register[6] (fatal error messages received) is set	PCI Express Root Error Command Register [2] (fatal error reporting enable) is set or PCI Express Root Control Register [2] (system error on fatal error enable) is set
PCI Express Root Error Status Register [5] (non-fatal error messages received) is set	PCI Express Root Error Command Register [1] (non-fatal error reporting enable) is set or PCI Express Root Control Register [1] (system error on non-fatal error enable) is set
PCI Express Root Error Status Register[0] (correctable error messages received) is set	PCI Express Root Error Command Register[0] (correctable error reporting enable) is set or PCI Express Root Control Register[0] (system error on correctable error enable) is set.
Any correctable error status bit in PCI Express Correctable Error Status Register is set	The corresponding error mask bit in PCI Express Correctable Error Mask Register is clear and PCI Express Root Error Command Register[0] (correctable error reporting enable) is set
Any fatal uncorrectable error status bit in PCI Express Uncorrectable Error Status Register is set. (The corresponding error is classified as fatal based on the severity setting in PCI Express Uncorrectable Error Severity Register.)	The corresponding error mask bit in PCI Express Uncorrectable Error Mask Register is clear and either PCI Express Device Control Register[2] (fatal error reporting) is set or PCI Express Command Register[8] (SERR) is set.
Any non-fatal uncorrectable error status bit in PCI Express Uncorrectable Error Status Register is set. (The corresponding error is classified as non-fatal based on the severity setting in PCI Express Uncorrectable Error Severity Register.)	The corresponding error mask bit in PCI Express Uncorrectable Error Mask Register is clear and either PCI Express Device Control Register[1] (non-fatal error reporting) is set or PCI Express Command Register[8] (SERR) is set.
PCI Express Secondary Status Register[8] (master data parity error) is set.	PCI Express Secondary Status Interrupt Mask Register[0] (mask master data parity error) is cleared and PCI Express Command Register[6] (parity error response) is set.
PCI Express Secondary Status Register[11] (signaled target abort) is set	PCI Express Secondary Status Interrupt Mask Register[1] (mask signaled target abort) is cleared.
PCI Express Secondary Status Register[12] (received target abort) is set	PCI Express Secondary Status Interrupt Mask Register[2] (mask received target abort) is cleared.

Table 16-125. PCI Express Internal Controller Interrupt Sources (continued)

Status Register Bit	Preconditions
PCI Express Secondary Status Register[13] (received master abort) is set	PCI Express Secondary Status Interrupt Mask Register[3] (mask received master abort) is cleared.
PCI Express Secondary Status Register[14] (signaled system error) is set.	PCI Express Secondary Status Interrupt Mask Register[4] (mask signaled system error) is cleared.
PCI Express Secondary Status Register[15] (detected parity error) is set	PCI Express Secondary Status Interrupt Mask Register[5] (mask detected parity error) is cleared.
PCI Express Slot Status Register[0] (attention button pressed) is set	PCI Express Slot Control Register[0] (attention button pressed enable) is set and PCI Express Slot Control Register[5] (hot plug interrupt enable) is set and either PCI Express PM Control Register[1–0] = 00 (the function power state is D0) or PCI Express PM Control Register[8] (PME enable) is set.
PCI Express Slot Status Register[1] (power fault detected) is set	PCI Express Slot Control Register[1] (power fault detected enable) is set and PCI Express Slot Control Register[5] (hot plug interrupt enable) is set and either PCI Express PM Control Register[1–0] = 00 (the function power state is D0) or PCI Express PM Control Register[8] (PME enable) is set.
PCI Express Slot Status Register[2] (MRL sensor changed) is set	PCI Express Slot Control Register[2] (MRL sensor changed enable) is set and PCI Express Slot Control Register[5] (hot plug interrupt enable) is set and either PCI Express PM Control Register[1–0] = 00 (the function power state is D0) or PCI Express PM Control Register[8] (PME enable) is set.
PCI Express Slot Status Register[3] (presence detect changed) is set	PCI Express Slot Control Register[3] (presence detect changed enable) is set and PCI Express Slot Control Register[5] (hot plug interrupt enable) is set and either PCI Express PM Control Register[1–0] = 00 (the function power state is D0) or PCI Express PM Control Register[8] (PME enable) is set.
PCI Express Slot Status Register[4] (command completed) is set	PCI Express Slot Control Register[4] (command completed interrupt enable) is set and PCI Express Slot Control Register[5] (hot plug interrupt enable) is set.

16.4.1.10.3 Error Conditions

Table 16-126 describes specific error types and the action taken for various transaction types.

Table 16-126. Error Conditions

Transaction Type	Error Type	Action
Inbound response	PEX response time out. This case happens when the internal platform sends a non-posted request that did not get a response back after a specific amount of time specified in the outbound completion timeout register (PEX_OTB_CPL_TOR)	Log error (PEX_ERR_DR[PCT]) and send interrupt to PIC, if enabled.
Inbound response	Unexpected PEX response. This can happen if, after the response times out and the internal queue entry is deallocated, the response comes back.	Log unexpected completion error (PCI Express Uncorrectable Status Register[16]) and send interrupt to PIC, if enabled.
Inbound response	Unsupported request (UR) response status	Depending upon whether the initial internal request was broken up, the error is not sent until all responses come back for all portions of the internal request. Log the error (PEX_ERR_DR[CDNSC] and PCI Express Uncorrectable Status Register[20]) and send interrupt to PIC, if enabled.
Inbound response	Completer abort (CA) response status	Depending upon whether the initial internal request was broken up, the error is not sent until all responses come back for all portions of the internal request. Log the error (PEX_ERR_DR[PCAC, CDNSC] and PCI Express Uncorrectable Status Register[15] and send interrupt to PIC, if enabled.
Inbound response	Poisoned TLP (EP=1)	Depending upon whether the initial internal request was broken up, the error is not sent until all responses come back for all portions of the internal request. Log the error (PCI Express Uncorrectable Status Register[12]) and send interrupt to PIC, if enabled.
Inbound response	ECRC error	Depending upon whether the initial internal request was broken up, the error is not sent until all responses come back for all portions of the internal request. Log the error (PCI Express Uncorrectable Status Register[19]) and send interrupt to PIC, if enabled.
Inbound response	Configuration Request Retry Status (CRS) timeout for a configuration transaction that originates from PEX_CONFIG_ADDR/ PEX_CONFIG_DATA	1.The controller always retries the transaction as soon as possible until a status other than CRS is returned. However, if a CRS status is returned after the configuration retry timeout (PEXCONF_RTY_TOR) timer expires, then the controller aborts the transaction and sends all 1s (0xFFFF_FFFF) data back to requester. 2. Log the error (PEX_ERR_DR[PCT]) and send interrupt to the PIC, if enabled.
Inbound response	UR response for configuration transaction that originates from PEX_CONFIG_ADDR/ PEX_CONFIG_DATA	1. Send back all 1s (0xFFFF_FFFF) data. 2. Log the error (PEX_ERR_DR[CDNSC] and PCI Express Uncorrectable Status Register[20]) and send interrupt to PIC, if enabled.

Table 16-126. Error Conditions (continued)

Transaction Type	Error Type	Action
Inbound response	CA response for Configuration transaction that originates from PEX_CONFIG_ADDR/ PEX_CONFIG_DATA	1. Send back all 1s (0xFFFF_FFFF) data. 2. Log the error (PEX_ERR_DR[PCAC, CDNSC] and PCI Express Uncorrectable Status Register[15]) and send interrupt to PIC, if enabled.
Inbound response	Poisoned TLP (EP=1) response for Configuration transaction that originates from PEX_CONFIG_ADDR/ PEX_CONFIG_DATA	1. Send back all 1s (0xFFFF_FFFF) data. 2. Log the error (PCI Express Uncorrectable Status Register[12]) and send interrupt to PIC, if enabled.
Inbound response	ECRC error response for Configuration transaction that originates from PEX_CONFIG_ADDR/ PEX_CONFIG_DATA	1. Send back all 1s (0xFFFF_FFFF) data. 2. Log the error (PCI Express Uncorrectable Status Register[19]) and send interrupt to PIC, if enabled.
Inbound response	Configuration Request Retry Status (CRS) response for Configuration transaction that originates from ATMU	1.The controller always retries the transaction as soon as possible until a status other than CRS is returned. However, if a CRS status is returned after the configuration retry timeout (PEXCONF_RTY_TOR) timer expires, then the controller aborts the transaction. 2. Log the error (PEX_ERR_DR[CRST]) and send interrupt to the PIC, if enabled.
Inbound response	UR response for Configuration transaction that originates from ATMU	Log the error (PEX_ERR_DR[CDNSC] and PCI Express Uncorrectable Status Register[20]) and send interrupt to PIC, if enabled.
Inbound response	CA response for Configuration transaction that originates from ATMU	Log the error (PEX_ERR_DR[PCAC, CDNSC] and PCI Express Uncorrectable Status Register[15]) and send interrupt to PIC, if enabled.
Inbound response	Malformed TLP response	PCI Express controller does not pass the response back to the core. Therefore, a completion timeout error eventually occurs.
Inbound request	Poisoned TLP (EP=1)	1. If it is a posted transaction, the controller drops it. 2. If it is a non-posted transaction, the controller returns a completion with UR status. 3. Release the proper credits
Inbound request	ECRC error	1. If it is a posted transaction, the controller drops it. 2. If it is a non-posted transaction, the controller returns a completion with UR status. 3. Release the proper credits
Inbound request	PCI Express nullified request	The packet is dropped.
Outbound request	Outbound ATMU crossing	Log the error (PEX_ERR_DR[OAC]). The transaction is not sent out on the link.
Outbound request	Illegal message size	Log the error (PEX_ERR_DR[MIS]). The transaction is not sent out on the link.
Outbound request	Illegal I/O size	Log the error (PEX_ERR_DR[IOIS]). The transaction is not sent out on the link.
Outbound request	Illegal I/O address	Log the error (PEX_ERR_DR[IOIA]). The transaction is not sent out on the link.
Outbound request	Illegal configuration size	Log the error (PEX_ERR_DR[CIS]). The transaction is not sent out on the link.

Table 16-126. Error Conditions (continued)

Transaction Type	Error Type	Action
Outbound response	Internal platform response with error (for example, an ECC error on a DDR read or the transaction maps to unknown address space).	Send poisoned TLP (EP=1) completion(s) for data that are known bad. If the poison data happens in the middle of the packet, the rest of the response packet(s) is also poisoned.

16.4.2 Interrupts

Both INTx and message signaled interrupts (MSI) are supported; however there are differences depending on whether the PCI Express controller is configured as an RC or EP device.

16.4.2.1 EP Interrupt Generation

16.4.2.1.1 Hardware INTx Message Generation

Hardware INTx message generation is not supported in EP mode.

16.4.2.1.2 Hardware MSI Generation

In EP mode, the PCI Express controller can be configured to automatically generate MSI transactions to the root complex when an interrupt event occurs. The PCI Express controller uses *irq_out* (an internal version of the IRQ_OUT signal) to trigger the generation of the MSI. To trigger the MSI, interrupt events must be routed to the to *irq_out* by setting the EP (external pin) bit in the associated Interrupt Destination register in the PIC. Note that the IRQ_OUT signal should not be used for any other function if it is being used to trigger MSI transactions.

The remote root complex is expected set up the MSI capability structure of all endpoints at system initialization by filling the Message Address and Message Data registers with appropriate values and setting the MSIE bit in the MSI Message Control register.

With the PCI Express controller properly configured, when it detects the leading edge of *irq_out* going active, it generates a PCI Express memory write transaction to the address specified in the MSI Message Address register (and MSI Message Upper Address register) with a data payload as specified in the MSI Message Data register (with leading zeros appended).

16.4.2.1.3 Software INTx Message Generation

Software can generate outbound assert or deassert INTx message transactions by using the outbound ATMU mechanism described in Section 16.4.1.9.1, “Outbound ATMU Message Generation.”

16.4.2.1.4 Software MSI Generation

Host software has to set up the MSI capability registers to enable MSI mode, and have the correct values for the MSI address and data register. Then local software has to read the MSI address in the MSI capability register and configure the outbound ATMU window to map the translated address to the MSI address. Software has to determine the number of allocated messages in the MSI capability register and

allocates the appropriate data values to use. A write to the ATMU window containing the MSI address with the appropriate data value generates the desired MSI transaction to the remote RC.

16.4.2.2 RC Handling of INTx Message and MSI Interrupts

16.4.2.2.1 INTx Message Handling

MSIs are the preferred interrupt signaling mechanism for PCI Express. However, in RC mode, the PCI Express controller supports the INTx virtual-wire interrupt signaling mechanism (as described in the PCI Express specification). Whenever the controller receives an inbound INTx (INTA, INTB, INTC, or INTD) asserted or negated message, it asserts or negates an equivalent internal INTx signal (*inta*, *intb*, *intc*, or *intd*) to the PIC.

The internal INTx signals from the PCI Express controller are logically combined with the interrupt request (IRQ*n*) input signals so that they share the same interrupt controlled by the associated EIVPR*n* and EIDR*n* registers in the PIC. Refer to Section 9.4.5, "PCI Express INTx," for more information about handling of PCI Express INTx interrupts and the external interrupt request (IRQ*n*) signals.

If a PCI Express INTx interrupt is being used, then the PIC must be configured so that external interrupts are level-sensitive (EIVPRn[S] = 1).

16.4.2.2.2 MSI Handling

An inbound MSI cycle must hit into the PEXCSRBAR window with the address offset that points to the MSIIR register in the PIC. Note that it is the responsibility of the host software to configure each EP's MSI capability registers such that an MSI cycle generated from the EP device is routed to the MSIIR register in the PIC and for the appropriate interrupt to be generated to the core.

16.4.3 Initial Credit Advertisement

To prevent overflowing of the receiver's buffers and for ordering compliance purposes, the transmitter cannot send transactions unless it has enough flow control (FC) credits to send. Each device maintains an FC credit pool. The FC information is conveyed between the two link partners by DLLPs during link training (initial credit advertisement). The transaction layer performs the FC accounting functions. One FC unit is four DWs (16-bytes) of data.

Table 16-127. Initial Credit Advertisement

Credit Type	Initial Credit Advertisement
PH (Memory Write, Message Write)	4
PD (Memory Write, Message Write)	(256/16)x4=64
NPH (Memory Read, IO Read, Cfg Read, Cfg Write)	8
NPD (IO Write, Cfg Write)	2
CPLH (Memory Read Completion, IO R/W Completion, Cfg R/W Completion)	Infinite
CPLD (Memory Read Completion, IO Read Completion, Cfg Read Completion)	Infinite

16.4.4 Power Management

All device power states are supported with the exception of D3cold with Vaux. In addition, all link power states are supported with the exception of L2 states. Only L0s ASPM mode is supported if enabled by configuring the Link Control register's bits 1–0 in configuration space. Note that there is no power saving in the controller when the device is put into a non-D0 state. The only power saving is the I/O drivers when the controller is put into a non-L0 link state.

Table 16-128. Power Management State Supported

Component D-State	Permissible Interconnect State	Action
D0	L0, L0s	In full operation.
D1	L0, L0s, L1	All outbound traffics are stalled. All inbound traffic is thrown away. The only exceptions are PME messages and configuration transactions. If the device is in RC mode, it is permissible to send a PM_Turn_Off message through the PEX Power Management Command register.
D2	L0, L0s, L1	All outbound traffics are stalled. All inbound traffic is thrown away. The only exceptions are PME messages and configuration transactions. If the device is in RC mode, it is permissible to send a PM_Turn_Off message through the PEX Power Management Command register.
D3hot	L0, L0s, L1, L2/L3 Ready	All outbound traffics are stalled. All inbound traffic is thrown away. The only exceptions are PME messages and configuration transactions. If the device is in RC mode, it is permissible to send a PM_Turn_Off message through the PEX Power Management Command register. Note that if a transition of D3hot->D0 occurs, a reset is performed to the controller's configuration space. In addition, link training restarts.
D3cold	L3	Completely off.

16.4.4.1 L2/L3 Ready Link State

The L2/L3 Ready link state is entered after the EP device is put into a D3hot state followed by a PME_Turn_Off/PME_TO_Ack message handshake protocol. Exiting this state requires a POR reset or a $\overline{\text{WAKE}}$ signal from the EP device. The PCI Express controller (in EP mode) does not support the generation of beacon; therefore, as an alternative, the device can use one of the GPIO signals as an enable to an external tristate buffer to generate a $\overline{\text{WAKE}}$ signal, as shown in Figure 16-132.

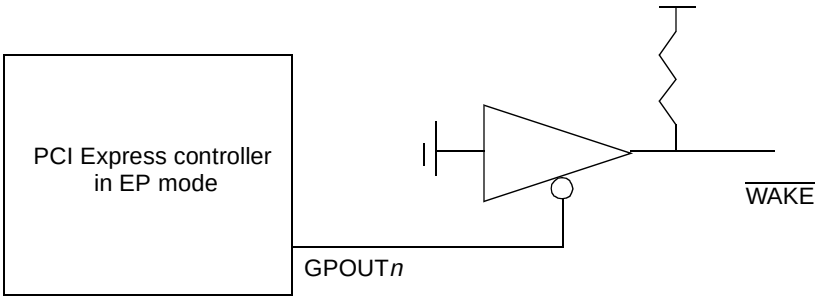


Figure 16-132. $\overline{\text{WAKE}}$ Generation Example

In RC mode, the $\overline{\text{WAKE}}$ signal from the EP device can be connected to one of the external interrupt inputs to service the $\overline{\text{WAKE}}$ request.

16.4.5 Hot Reset

When a hot reset condition occurs, the controller (in both RC and EP mode) initiates a clean-up of all outstanding transactions and returns to an idle state. All configuration register bits that are non-sticky are reset. Link training takes place subsequently. The device is permitted to generate a hot reset condition on the bus when it is configured as an RC device by setting the “Secondary Bus Reset” bit in the Bridge Control Register in the configuration space. As an EP device, it is not permitted to generate a hot reset condition; it can only detected a hot reset condition and initiates the clean-up procedure appropriately.

16.4.6 Link Down

Typically, a link down condition occurs after a hot reset event; however, it is possible for the link to go down unexpectedly without a hot reset event. When this occurs, a link down condition is detected (PEX_PME_MSG_DR[LDD]=1). Link down is treated similarly to a hot reset condition.

Subsequently, while the link is down, all new posted outbound transactions are discarded. All new non-posted ATMU transactions are errored out. Non-posted configuration transactions issued using PEX_CONFIG_ADDR/PEX_CONFIG_DATA toward the link returns 0xFFFF_FFFF (all 1s). As soon as the link is up again, the sending of transaction resumes.

Note that in EP mode, a link down condition causes the controller to reset all non-sticky bits in its PCI Express configuration registers as if it had been hot reset.

16.5 Initialization/Application Information

16.5.1 Boot Mode and Inbound Configuration Transactions

In normal boot mode (cfg_cpu_boot = 1), the core is allowed to boot and configure the device. During this time, the PCI Express interface retries all inbound PCI Express configuration transactions. When the core has configured the device to a state where it can accept inbound PCI Express configuration transactions, the boot code should set the CFG_READY bit in the PEX_CFG_READY register after which inbound PCI Express configuration transactions are accepted. Refer to Section 16.3.10.18, “Configuration Ready Register—0x4B0,” for more information about the CFG_READY bit.

In boot hold-off mode (cfg_cpu_boot = 0), the core is prevented from fetching its first instruction by withholding its internal bus grant. During this time, the PCI Express interface accepts all inbound PCI Express configuration transactions which allows an external host/RC to configure the device. When the external host/RC has configured the device to a state where it can allow the core to fetch code from the boot vector, it sets the PCR[PORT0_EN] bit after which the core is granted the internal bus.

Part IV
Global Functions and Debug

Part IV defines other global blocks of the MPC8641. The following chapters are included:

- Chapter 17, “Global Utilities,” defines the global utilities of the MPC8641. These include power management, I/O device enabling, power-on-reset (POR) configuration monitoring, general-purpose I/O signal use, and multiplexing for the interrupt and local bus chip select signals
- Chapter 18, “Device Performance Monitor,” describes the performance monitor of the MPC8641.
- Chapter 19, “Debug Features and Watchpoint Facilities,” describes the debug features and watchpoint monitor of the MPC8641.

Chapter 17
Global Utilities

This chapter describes the global utilities of the MPC8641. It provides signal descriptions, register descriptions, and a functional description of these utilities.

17.1 Overview

The global utilities block controls power management, I/O device enabling, power-on-reset (POR) configuration monitoring, general-purpose I/O signal configuration, alternate function selection for multiplexed signals, and clock control.

17.2 Global Utilities Features

This section provides an overview of global utilities features.

17.2.1 Power Management and Block Disables

The following features affect the device's overall power consumption:

- Software-controlled power management (core nap, sleep; device sleep)
- Static power management (I/O block disables)

17.2.2 Accessing Current POR Configuration Settings

The POR configuration values of all device parameters sampled from pins at reset are available through memory-mapped registers in the global utilities block.

17.2.3 General-Purpose I/O

The eTSEC1 and eTSEC2 data bus signals can be used as general-purpose I/O signals when not used for their primary function. Section 17.5.2, "General-Purpose I/O Signals," details the use of this supplementary general purpose I/O functionality. Memory-mapped registers provide control and status for the use of these signals. See Section 17.4.1.7.1, "General-Purpose I/O Control Register (GPIOCR)," Section 17.4.1.7.2, "General-Purpose Output Data Register (GPOUTDR)," and Section 17.4.1.7.2, "General-Purpose Output Data Register (GPOUTDR)," for more information. A general purpose input register is loaded with the values of the local bus address/data pins at the negation of $\overline{\text{HRESET}}$.

17.2.4 Interrupt and Local Bus Signal Multiplexing

IRQ[9:11] and $\overline{\text{LCS}}$[5:7] serve multiple functions that can be selected by configuration registers in the global utilities block.

17.2.5 Clock Control

The global utilities block also selects the internal clock signal driven on CLK_OUT.

17.3 External Signal Descriptions

The following sections provide information about signals that serve as global utilities.

17.3.1 Signals Overview

Table 17-1 summarizes the external signals used by the global utilities block

Table 17-1. External Signal Summary

Signal Name	I/O	Description	Reference (Section/Page)
ASLEEP	O	Signals that the device has reached a sleep state.	—
CLK_OUT	O	Clock out. Selected by CLKOCR values.	17.4.1.18/17-27
$\overline{\text{CKSTP\_ IN}}$	I	Checkstop input.	Table 17-2 on page 17-3
$\overline{\text{CKSTP\_OUT}}$	O	Checkstop output.	Table 17-2 on page 17-3
GPOUT[15:0]	O	General-purpose output.	17.4.1.7/17-14
GPIN[15:0]	I	General-purpose input.	17.4.1.7/17-14
$\overline{\text{SMI\_0}}$	I	System Management Interrupt input for core 0	Table 17-2 on page 17-3
$\overline{\text{SMI\_1}}$	I	System Management Interrupt input for core 1	Table 17-2 on page 17-3

17.3.2 Detailed Signal Descriptions

Table 17-2 describes signals in the global utilities block in detail.

Table 17-2. Detailed Signal Descriptions

Signal	I/O	Description	
ASLEEP	O	Asleep. After negation of HRESET, ASLEEP is asserted until the device completes its power-on reset sequence and reaches its ready state.	
		State Meaning	Asserted—Indicates that the device is either still in its power-on reset sequence or it has reached a sleep state after a power-down command is issued by software. Negated—The device is not in sleep mode. (It has either awakened from a power-down state, or has completed the POR sequence.)
		Timing	Assertion—May occur at any time; may be asserted asynchronously to the input clocks. Negation—Negates synchronously with SYSCLK when leaving power-on sequence; otherwise negation is asynchronous.
CKSTP_IN	I	Checkstop in	
		State Meaning	Asserted—Indicates that the e600 core must enter a hard stop condition. All e600 clocks are turned off. CKSTP_OUT is asserted. The rest of 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 device is reset with assertion of HRESET.
CKSTP_OUT	O	Checkstop out	
		State Meaning	Asserted—Indicates that the e600 core of the device is in a checkstop state. The rest of the device logic remains functional. Negated—Indicates normal operation. After CKSTP_OUT has been asserted, it is negated after the next negation (low-to-high transition) of HRESET.
		Timing	Assertion—May occur at any time; may be asserted asynchronously to the input clocks. Negation—Must remain asserted until the device has been reset with a hard reset.
CLK_OUT	O	Clock out. Reflects clock signal selected by CLKOCR (see Section 17.4.1.18, “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].
GPOUT[15:0]	O	General-Purpose Output. See Section 17.4.1.7, “General-Purpose I/O Registers for details.	
GPIN[15:0]	I	General-Purpose Input. See Section 17.4.1.7, “General-Purpose I/O Registers for details.	
SMI_0	I	System Management Interrupt for Core 0. See interrupt section of the *e600 Core Reference Manual* for additional information	
		State Meaning	Asserted—Indicates that the e600 core 0 should take a System Management Interrupt exception if enabled in the MSR. Negated—Indicates that there is no pending System Management exception.
		Timing	Assertion—May occur at any time; may be asserted asynchronously to the input clocks. SMI_0 is level-sensitive. Negation—Must not occur until after the interrupt is taken.

Table 17-2. Detailed Signal Descriptions (continued)

Signal	I/O	Description
$\overline{\text{SMI\_1}}$	I	System Management Interrupt for Core 1. See interrupt section of the *e600 Core Reference Manual* for additional information
	State Meaning	Asserted—Indicates that the e600 core 1 should take a System Management Interrupt exception if enabled in the MSR. Negated—Indicates that there is no pending System Management exception.
	Timing	Assertion—May occur at any time; may be asserted asynchronously to the input clocks. $\overline{\text{SMI\_1}}$ is level-sensitive. Negation—Must not occur until after the interrupt is taken.

17.4 Memory Map/Register Definition

Table 17-3 summarizes the global utilities registers and their addresses.

Table 17-3. Global Utilities Registers

Global Utilities—Block Base Address 0xE_0000				
Offset	**Register**	**Access**	**Reset [1]**	**Section/Page**
Power-On Reset Configuration Values				
0x000	PORPLLSR—POR PLL ratio status register	R	0x00*nn*_*n*0*nn*	17.4.1.1/17-6
0x004	PORBMSR—POR boot mode status register	R	0x*nnnn*_0000	17.4.1.2/17-7
0x008	PORIMPCR—POR I/O impedance control register	Mixed	0x000*n*_007F	17.4.1.3/17-8
0x00C	PORDEVSR—POR I/O device status register	R	see ref***.	17.4.1.4/17-9
0x010	PORDBGMSR—POR debug mode status register	R	see ref.	17.4.1.5/17-12
0x020	PORCIR—POR configuration information register	R	see ref.	17.4.1.6/17-13
Signal Multiplexing and GPIO Controls				
0x030	GPIOCR—GPIO control register	R/W	0x0000_0000	17.4.1.7.1/17-14
0x040	GPOUTDR—General-purpose output data register	R/W	0x0000_0000	17.4.1.7.2/17-15
0x050	GPINDR—General-purpose input data register	R	0x*nnnn*_*nnnn*	17.4.1.7.3/17-15
0x060	PMUXCR—Alternate function signal multiplex control	R/W	0x0000_0000	17.4.1.8/17-16
Device Disables				
0x070	DEVDISR—Device disable control	R/W	0x0000_0000	17.4.1.9/17-17
Power Management Registers				
0x080	POWMGTCSR—Power management status and control register	Mixed	0x0000_0000	17.4.1.10/17-20
Interrupt and Reset Status and Control Registers				
0x090	MCPSUMR—Machine check summary register	w1c	0x0000_0000	17.4.1.11/17-22

Table 17-3. Global Utilities Registers (continued)

Global Utilities—Block Base Address 0xE_0000				
Offset	**Register**	**Access**	**Reset** [1]	**Section/Page**
0x094	RSTRSCR—Reset request status and control register	Mixed	0x0000_0000	17.4.1.12/17-23
Version Registers				
0x0A0	PVR—Processor version register	R	0x8004_*nnnn*	17.4.1.13/17-24
0x0A4	SVR—System version register	R	0x8090_00*nn* (MPC8641) 0x8090_01*nn* (MPC8641D)	17.4.1.14/17-24
Reset Registers				
0x0B0	RSTCR—Reset control register	R/W	0x0000_0000	17.4.1.15/17-25
DDR Clock Control				
0xB20	DDR1CLKDR—DDRC1 Clock Disable Register	R/W	0x0000_0000	17.4.1.16/17-26
0xB28	DDR2CLKDR—DDRC2 Clock Disable Register	R/W	0x0000_0000	17.4.1.17/17-26
Clock and SerDes Control				
0xE00	CLKOCR—Clock out control register	R/W	0x0000_0000	17.4.1.18/17-27
0xF04	SRDS1CR0—SerDes1 control register 0	R/W	0x*nn*00_*nn*00	17.4.1.19/17-28
0xF08	SRDS1CR1—SerDes1 control register 1	R/W	0x0000_0000	17.4.1.20/17-30
0xF40	SRDS2CR0—SerDes2 control register 0	R/W	0x*nn*00_*nn*00	17.4.1.21/17-31
0xF44	SRDS2CR1—SerDes2 control register 1	R/W	0x0000_0000	17.4.1.22/17-32

[1] Bits indicated with *n* are set from configuration signals.

17.4.1 Register Descriptions

This section describes the global utilities registers in detail.

17.4.1.1 POR PLL Status Register (PORPLLSR)

PORPLLSR, shown in Figure 17-1, contains the settings for the PLL ratios as set by the cfg_sys_pll[0:3], cfg_core_pll[0:4], and POR configuration pins. See Section 4.4.3.1, “System PLL Ratio,” and Section 4.4.3.3, “e600 Core PLL Ratio,” for more information.

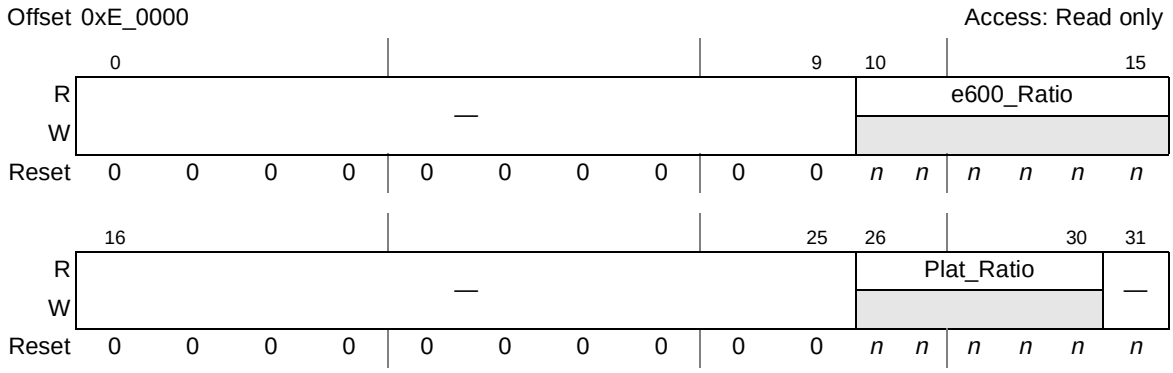


Figure 17-1. POR PLL Status Register (PORPLLSR)

Table 17-4 describes the bit settings of PORPLLSR.

Table 17-4. PORPLLSR Field Descriptions

Bits	Name	Description
0–9	—	Reserved
10–15	e600_Ratio	Clock ratio between the e600 cores and the MPX clock. Patterns not shown are reserved. 010000 2:1 011001 2.5:1 100000 3:1 111001 3.5:1 101000 4:1 011101 4.5:1 *nnnnnn* Any other combinations besides the ones listed are not supported for this device
16–25	—	Reserved
26–30	Plat_Ratio	Clock ratio between the MPX (platform) clock and SYSCLK. Patterns not shown are reserved. 00010 2:1 00011 3:1 00100 4:1 00101 5:1 00110 6:1 01000 8:1 01001 9:1
31	—	Reserved

17.4.1.2 POR Boot Mode Status Register (PORBMSR)

The PORBMSR, shown in Figure 17-2, reports setting of the POR configuration pins that control the boot mode settings (described in Section 4.4.3.6, "Boot ROM Location," Section 4.4.3.10, "CPU Boot Configuration," and Section 4.4.3.11, "Boot Sequencer Configuration") and the default settings of host/agent mode (described in Section 4.4.3.9, "SerDes Host/Agent Configuration").

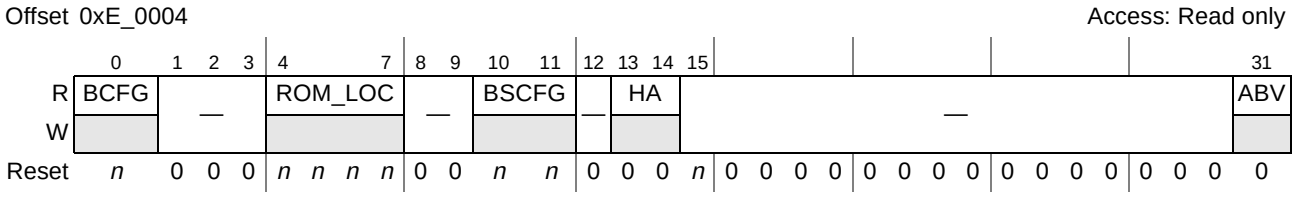


Figure 17-2. POR Boot Mode Status Register (PORBMSR)

Figure 17-5 describes the bit settings of the PORBMSR.

Table 17-5. PORBMSR Field Description

Bits	Name	Description
0	BCFG	CPU boot configuration 0 Core 0 is prevented from booting until configured by an external master. In this case, the external master must also set MCM reg [PORT0_EN]bit when the external master is ready for core 0 to boot. 1 Core 0 is allowed to boot without waiting for configuration by either an external master.
1–3	—	Reserved
4–7	ROM_LOC	Location of boot ROM. This field indicates the location of the boot ROM, as determined by the values on cfg_rom_loc[0:3] at the negation of $\overline{\text{HRESET}}$. Note that the values in this register do not directly map to the values on the cfg_rom_loc[0:3] signals. 0000 PCI Express 1 0001 PCI Express 2 0010 Serial RapidIO 0100 DDR Controller 0 0101 DDR Controller 1 0110 Reserved 1101 Local bus GPCM: 8-bit 1110 Local bus GPCM:16-bit 1111 Local bus GPCM: 32-bit (default)
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	—	Reserved

Table 17-5. PORBMSR Field Description (continued)

Bits	Name	Description
13–14	HA	Host-Agent 00 The controller on SD1 is in Endpoint (PEX) mode The controller on SD2 is in Agent (SRIO) or Endpoint (PEX) mode 01 The controller on SD1 is in Root Complex (PEX) mode The controller on SD2 is in Host (SRIO) or Root Endpoint (PEX) mode 10 The controller on SD1 is in Endpoint (PEX) mode The controller on SD2 is in Agent (SRIO) or Root Complex (PEX) mode 11 Host/Host The controller on SD1 is in Root Complex (PEX) mode The controller on SD2 is in Host (SRIO) or Root Complex (PEX) mode
15–30	—	Reserved
31	ABV	Alternate Boot Vector 0 Alternate Boot Vector mode is enabled. Outbound Window 1 in PEX 1 is automatically enabled to translate addresses matching 0x0_FFF0_xxxx to 0x0_FFFF_xxxx. Used with PORBMSR[ROM_LOC]=PEX1 to steer the e600 boot vectors. 1 Alternate Boot Vector mode is disabled (default)

17.4.1.3 POR I/O Impedance Control Register (PORIMPCR)

PORIMPCR, shown in Figure 17-3, contains the I/O driver impedance select for the local bus interface.

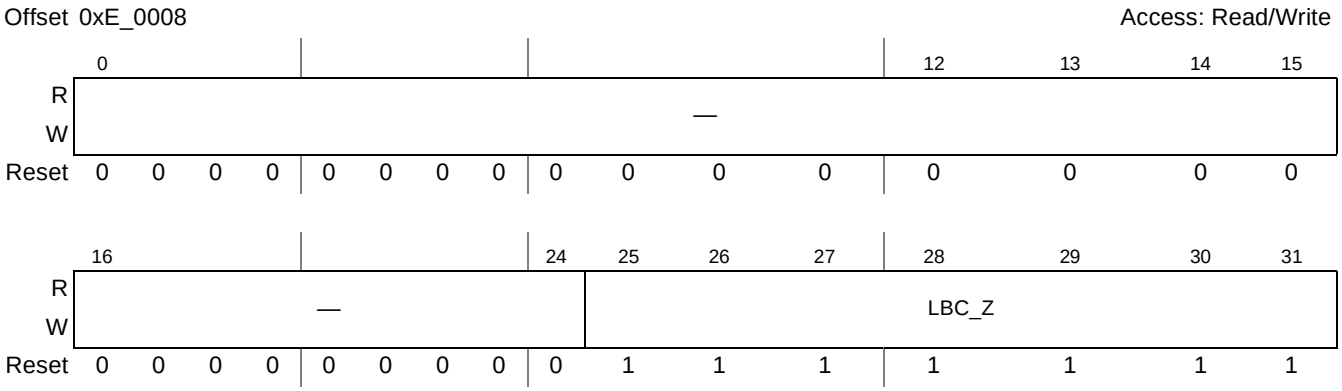


Figure 17-3. POR I/O Impedance Control Register (PORIMPCR)

The I/O impedance of local bus signals (including the local bus clock) is controlled through this register. The *MPC8641 Integrated Processor Hardware Specification* provides exact I/O impedances.

Table 17-6 describes PORIMPCR fields.

Table 17-6. PORIMPCR Field Descriptions

Bits	Name	Description
0–24	—	Reserved
25–31	LBC_Z	I/O impedance for the local bus signals: LALE, LAD[0:31], LDP[0:3], LA[27:31], LCKE, LCS[1:2], LWE[0:3], LGPL1, LGPL2, LGPL3, LGPL4, LGPL5, LCLK[0:2]. All other local bus I/O signals use a fixed high impedance. 111111High impedance All other settingsLow impedance

17.4.1.4 POR Device Status Register (PORDEVSR)

Shown in Figure 17-4, PORDEVSR reports other POR settings for I/O devices as described in Section 4.4.3.13, “eTSECn Width,” Section 4.4.3.14, “eTSECn Protocol,” Section 4.4.3.8, “SerDes Port Selection,” and Section 4.4.3.16, “Serial RapidIO System Size.”

Offset 0xE_000C — Access: Read only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ECW1	ECW2	ECW3	ECW4	ECP1		ECP2		ECP3		ECP4		HS_CFG			
W																
Reset	*n*	0	0	0	0	0	*n*	*n*	*n*	0	0	0	*n*	*n*	*n*	*n*

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SRIOW	—				PLAT_FREQ	RTYPE		CORE 1TE	—				DEV_ID		
W																
Reset	All zeros															

Figure 17-4. POR Device Status Register (PORDEVSR)

Table 17-7 describes the bit settings of PORDEVSR.

Table 17-7. PORDEVSR Field Descriptions

Bits	Name	Description
0	ECW1	eTSEC controller 1 width 0 Reduced interfaces for eTSEC1 (RGMII, RTBI, or 8-bit FIFO on eTSEC1) 1 Full interfaces for eTSEC1 (MII, GMII, TBI, or 16-bit FIFO on eTSEC1)
1	ECW2	eTSEC controller 2 width 0 Reduced interfaces for eTSEC2 (RGMII, RTBI, or 8-bit FIFO on eTSEC2) 1 Full interfaces for eTSEC2 (MII, GMII, TBI, or 8-bit FIFO on eTSEC2) Note: FIFO mode on eTSEC2 must be 8-bits regardless of ECW1
2	ECW3	eTSEC controller 3 width 0 Reduced interfaces for eTSEC3 (RGMII, RTBI, or 8-bit FIFO on eTSEC3) 1 Full interfaces for eTSEC3 (MII, GMII, TBI, or 16-bit FIFO on eTSEC3)

Table 17-7. PORDEVSR Field Descriptions (continued)

Bits	Name	Description
3	ECW4	eTSEC controller 4 width 0 Reduced interfaces for eTSEC4 (RGMII, RTBI, or 8-bit FIFO on eTSEC4) 1 Full interfaces for eTSEC4 (MII, GMII, TBI, or 8-bit FIFO on eTSEC4) Note: FIFO mode on eTSEC4 is always 8-bits regardless of ECW3.
4–5	ECP1	eTSEC1 controller 1 protocol (See Section 4.4.3.13, “eTSECn Width.”) 00 The eTSEC1 controller operates using the 16-bit FIFO protocol (or 8-bit FIFO if configured in reduced mode). 01 The eTSEC1 controller operates using the MII protocol (or RMII if configured in reduced mode). 10 The eTSEC1 controller operates using the GMII protocol (or RGMII if configured in reduced mode). 11 The eTSEC1 controller operates using the TBI protocol (or RTBI if configured in reduced mode). Note: 16-bit FIFO mode on eTSEC1 disables eTSEC2
6–7	ECP2	eTSEC2 controller 2 protocol (See Section 4.4.3.13, “eTSECn Width.”) 00 The eTSEC2 controller operates using the 16-bit FIFO protocol (or 8-bit FIFO if configured in reduced mode). 01 The eTSEC2 controller operates using the MII protocol (or RMII if configured in reduced mode). 10 The eTSEC2 controller operates using the GMII protocol (or RGMII if configured in reduced mode). 11 The eTSEC2 controller operates using the TBI protocol (or RTBI if configured in reduced mode). Note: 16-bit FIFO mode on eTSEC1 disables eTSEC2
8–9	ECP3	eTSEC3 controller 3 protocol (See Section 4.4.3.13, “eTSECn Width.”) 00 The eTSEC3 controller operates using the 16-bit FIFO protocol (or 8-bit FIFO if configured in reduced mode). 01 The eTSEC3 controller operates using the MII protocol (or RMII if configured in reduced mode). 10 The eTSEC3 controller operates using the GMII protocol (or RGMII if configured in reduced mode). 11 The eTSEC3 controller operates using the TBI protocol (or RTBI if configured in reduced mode). Note: 16-bit FIFO mode on eTSEC3 disables eTSEC4
10–11	ECP4	eTSEC4 controller 4 protocol (See Section 4.4.3.13, “eTSECn Width.”) 00 The eTSEC4 controller operates using the 16-bit FIFO protocol (or 8-bit FIFO if configured in reduced mode). 01 The eTSEC4 controller operates using the MII protocol (or RMII if configured in reduced mode). 10 The eTSEC4 controller operates using the GMII protocol (or RGMII if configured in reduced mode). 11 The eTSEC4 controller operates using the TBI protocol (or RTBI if configured in reduced mode). Note: 16-bit FIFO mode on eTSEC3 disables eTSEC4

Table 17-7. PORDEVSR Field Descriptions (continued)

Bits	Name	Description
12–15	HS_CFG	I/O port selection mode (See Section 4.4.3.8, “SerDes Port Selection.”) 0000 Ports disabled SerDes1: Disabled, no reference clock required SerDes2: Disabled, no reference clock required 0001 Reserved 0010 x1/x2/x4/x8 PCI Express SerDes1: x1/x2/x4/x8 PCI Express (100 MHz reference clock to SerDes1) SerDes2: Disabled 0011 x1/x2/x4/x8 PCI Express and x1/x2/x4/x8 PCI Express SerDes1: x1/x2/x4/x8 PCI Express (100 MHz reference clock to SerDes1) SerDes2: x1/x2/x4/x8 PCI Express (100 MHz reference clock to SerDes2) 0100 Reserved 0101 x4 Serial RapidIO 3.125 Gbps and x1/x2/x4/x8 PCI Express SerDes1: x1/x2/x4/x8 PCI Express (100 MHz reference clock to SerDes1) SerDes2: x4 Serial RapidIO, 3.125 Gb interface (125 MHz reference clock to SerDes2) 0110 x4 serial RapidIO 2.5 Gbps and x1/x2/x4/x8 PCI Express SerDes1: x1/x2/x4/x8 PCI Express (100 MHz reference clock to SerDes1) SerDes2: x4 Serial RapidIO, 2.5 Gb interface (100 MHz reference clock to SerDes2) 0111 x4 serial RapidIO 1.25 Gbps and x1/x2/x4/x8 PCI Express SerDes1: x1/x2/x4/x8 PCI Express (100 MHz reference clock to SerDes1) SerDes2: x4 Serial RapidIO, 1.25 Gb interface (100 MHz reference clock to SerDes2) 1000 Reserved 1001 x4 Serial RapidIO 3.125 Gbps SerDes1: Disabled SerDes2: x4 Serial RapidIO, 3.125 Gb interface (125 MHz reference clock to SerDes2) 1010 x4 Serial RapidIO 2.5 Gbps SerDes1: Disabled SerDes2: x4 Serial RapidIO, 2.5 Gb interface (100 MHz reference clock to SerDes2) 1011 x4 Serial RapidIO 1.25 Gbps SerDes1: Disabled SerDes2: x4 Serial RapidIO, 1.25 Gb interface (100 MHz reference clock to SerDes2) 1100 Reserved 1101 Reserved 1110 x1/x2/x4/x8 PCI Express SerDes1: Disabled SerDes2: x1/x2/x4/x8 PCI Express (100 MHz reference clock to SerDes2) 1111 x1/x2/x4/x8 PCI Express and x1/x2/x4/x8 PCI Express (default) SerDes1: x1/x2/x4/x8 PCI Express (100 MHz reference clock to SerDes1) SerDes2: x1/x2/x4/x8 PCI Express (100 MHz reference clock to SerDes2)
16	SRIOTS	Serial RapidIO transport size (See Section 4.4.3.16, “Serial RapidIO System Size.”) 0) Large transport (8-bit and 16-bit device IDs) supported on SRIO 1) Only small transport (8-bit device IDs) supported on SRIO (default)
17–20	—	Reserved
21	PLAT_ FREQ	Platform Frequency (See Section 4.4.3.2, “Platform Frequency”) 0) The MPC8641 platform frequency is = 400 MHz 1) The MPC8641 platform frequency is >= 500 MHz Note: MPC8641 does not support platform frequencies between 400 and 500 MHz

Table 17-7. PORDEVSR Field Descriptions (continued)

Bits	Name	Description
22–23	RTYPE	DRAM Type (See Section 4.4.3.12, “DDR SDRAM Type.”) 00 Reserved 01 DDR1, 2.5V, CKE low at reset 10 Reserved 11 DDR2, 1.8V, CKE low at reset
24	CORE1TE	Core 1 low memory offset mode enable. (See Section 4.4.3.5, “e600 Core 1 Low Memory Offset Mode”) 0)If cfg_core1_lm_offset (TSEC3_TXD[3] at POR) is 1 during reset, then PORDEVSR[CORE1TE] is cleared and core 1 translation is disabled. System address == real address. 1)If cfg_core1_lm_offset (TSEC3_TXD[3] at POR) is 0 during reset, then PORDEVSR[CORE1TE] is set and core 1 translation is enabled. Real address A in range 0 to 256 Mbyte-1 translated to address A + 256 Mbytes. If core 1 translation is enabled, then the system creates a private 256 MB region per core at the low end of the system address space that the other core cannot access. Core 0's private region is from 0 to 256MB-1 in the system address map. Core 1's private region is from 256MB to 512MB-1 in the system address map. The OSes must not permit the cores to generate real addresses from 256MB to 512MB-1. This is a hole in the real address map to redirect the private region accesses of the other core. Requests from core 1 to the system (memory or I/O) are translated from real address (RA) to system address (SA) as follows: If RA < 256MB then SA = RA+256MB Requests from elsewhere in the system (including core 0) are translated from system address (SA) to core 1 real address (RA) for snooping as follows: If SA < 256MB then RA = SA+256MB If 256MB <= SA < 512MB then RA = SA-256MB Core 0 real addresses (requests and snoops) always match the system address. Core 1 real addresses below 256 MB do not match the system address. The OS for core 1 must take this into account when, e.g., programming a DMA engine to access memory in the private region
29–31	DEV_ID	Serial RapidIO device ID (See Section 4.4.3.15, “Serial RapidIO Device ID.”)

17.4.1.5 POR Debug Mode Status Register (PORDBGMSR)

PORDBGMSR, shown in Figure 17-5, holds debug mode settings from the POR configuration pins as described in Section 4.4.3.17, “Memory Debug Configuration,” Section 4.4.3.18, “DDR Debug Configuration,” and Section 4.4.3.18, “DDR Debug Configuration.”

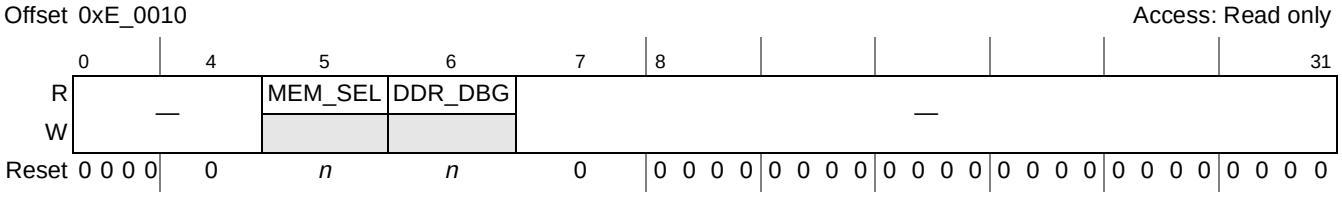


Figure 17-5. POR Debug Mode Status Register (PORDBGMSR)

Table 17-8 describes the bit settings of PORDBGMSR.

Table 17-8. PORDBGMSR Field Descriptions

Bits	Name	Description
0–4	—	Reserved
5	MEM_SEL	Memory select. Indicates which controller is driving D1_MSRCID[0:4] and D1_MDVAL*n*. 0 Local bus controller is driving debug information 1 DDR SDRAM controller 1 is driving debug information
6	DDR_DBG	DDR debug configuration 0 SourceID and data valid information is being driven on ECC pins of both DDR SDRAM interfaces 1 Normal mode. ECC information is being driven on ECC pins of both DDR SDRAM interfaces
7–31	—	Reserved

17.4.1.6 POR Configuration Information Register (PORCIR)

Shown in Figure 17-7, PORCIR stores the value sampled from the local bus address/data signals, LAD[0:31], during POR, as described in Section 4.4.3.19, "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.

NOTE

This read-only register is not for use as general-purpose input/output (GPIO). It is intended to allow system designers to pass (for example) a device-specific version value. This happens one time only, at reset.

True general-purpose I/O is available. See the registers of Section 17.4.1.7, "General-Purpose I/O Registers," for more information.

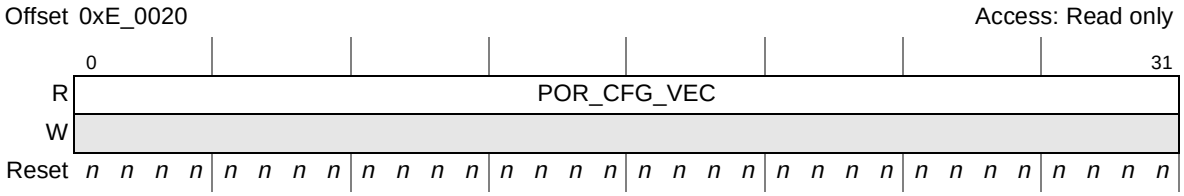


Figure 17-6. POR Configuration Information Register (PORCIR)

Table 17-9 describes the bit settings of PORCIR.

Table 17-9. PORCIR Field Descriptions

Bits	Name	Description
0–31	POR_CFG_VEC	POR configuration vector sampled from local bus address/data signals LAD[0:31]at the negation of $\overline{\text{HRESET}}$. Note that if nothing is driven on these signals during reset, the value of this register is indeterminate.

17.4.1.7 General-Purpose I/O Registers

The device can be configured, using registers described in the next sections, to have 16 general-purpose input and 16 general-purpose output signals; however eTSEC1 and eTSEC2 must be disabled for their signals to be used for general purpose input and output. In brief, this configuration may be achieved as follows:

1. GPIOCR[RXIN] must be set to enable general-purpose input and GPIOCR[TXOUT] must be set to enable general-purpose output. See Section 17.4.1.7.1, "General-Purpose I/O Control Register (GPIOCR)," for a description of GPIOCR.
2. DEVDISR[eTSEC1] and DEVDISR[eTSEC2] must be set to disable eTSEC1 and eTSEC2. See Section 17.4.1.9, "Device Disable Register (DEVDISR)," for more complete information about DEVDISR.

17.4.1.7.1 General-Purpose I/O Control Register (GPIOCR)

Shown in Figure 17-7, GPIOCR contains the enable bits for each group of pins that may be used for general-purpose I/O. These bits have meaning only if the signals are not being used for their primary function. Note that when these signals are enabled as general-purpose I/O signals, they are read and written through GPINDR and GPOUTDR, described in Section 17.4.1.7.3, "General-Purpose Input Data Register (GPINDR)," and Section 17.4.1.7.2, "General-Purpose Output Data Register (GPOUTDR)." Section 17.5.2, "General-Purpose I/O Signals," describes the use of general-purpose I/O signals.

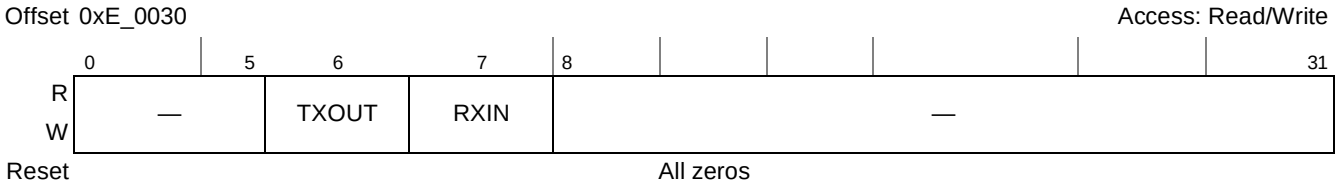


Figure 17-7. General-Purpose I/O Control Register (GPIOCR)

Table 17-10 describes the bit settings of GPIOCR.

Table 17-10. GPIOCR Field Descriptions

Bits	Name	Description
0–5	—	Reserved
6	TXOUT	0 TSEC1_TXD[7:0] and TSEC2_TXD[7:0] are not being used as GPIO signals. 1 Enables TSEC1_TXD[7:0] and TSEC2_TXD[7:0] for use as general-purpose output. Note that both eTSEC1 and eTSEC2 interfaces must be disabled for these signals to be used as general-purpose output. For details on disabling TSEC1 and TSEC2 see Section 17.4.1.9, "Device Disable Register (DEVDISR)."
7	RXIN	0 TSEC1_RXD[7:0] and TSEC2_RXD[7:0] are not being used as GPIO signals. 1 Enables TSEC1_RXD[7:0] and TSEC2_RXD[7:0] for use as general-purpose input. Note that both eTSEC1 and eTSEC2 interfaces must be disabled for these signals to be used as general-purpose output. For details on disabling TSEC1 and TSEC2 see Section 17.4.1.9, "Device Disable Register (DEVDISR)."
8–31	—	Reserved

17.4.1.7.2 General-Purpose Output Data Register (GPOUTDR)

GPOUTDR, shown in Figure 17-8, contains the data driven on TSEC1_TXD[0:7] and/or TSEC2_TXD[0:7] when these signals are configured for use as general-purpose outputs in GPIOCR, as described in Section 17.4.1.7.1, "General-Purpose I/O Control Register (GPIOCR)." Writes to GPOUTDR only affect signals 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 the other. Reads return the data currently being driven on the outputs only if the associated bits are configured as general-purpose output signals rather than their primary function as eTSEC signals.

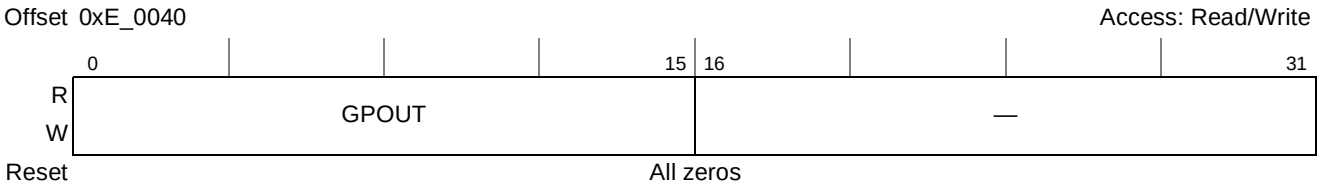


Figure 17-8. General-Purpose Output Data Register (GPOUTDR)

Table 17-11 describes the fields of GPOUTDR.

Table 17-11. GPOUTDR Field Descriptions

Bits	Name	Description
0–15	GPOUT	General-purpose output data. When the corresponding signals are configured to be general-purpose output signals, the values of the bits of GPOUT are driven onto those pins. GPOUT[7:0] corresponds to TSEC1_TXD[7:0] as follows: GPOUT[7] ↔ TSEC1_TXD[7] GPOUT[6] ↔ TSEC1_TXD[6] GPOUT[5] ↔ TSEC1_TXD[5] GPOUT[4] ↔ TSEC1_TXD[4] GPOUT[3] ↔ TSEC1_TXD[3] GPOUT[2] ↔ TSEC1_TXD[2] GPOUT[1] ↔ TSEC1_TXD[1] GPOUT[0] ↔ TSEC1_TXD[0] GPOUT[15:8] corresponds to TSEC2_TXD[7:0] as follows: GPOUT[15] ↔ TSEC2_TXD[7] GPOUT[14] ↔ TSEC2_TXD[6] GPOUT[13] ↔ TSEC2_TXD[5] GPOUT[12] ↔ TSEC2_TXD[4] GPOUT[11] ↔ TSEC2_TXD[3] GPOUT[10] ↔ TSEC2_TXD[2] GPOUT[9] ↔ TSEC2_TXD1] GPOUT[8] ↔ TSEC2_TXD[0]
16–31	—	Reserved, should be cleared

17.4.1.7.3 General-Purpose Input Data Register (GPINDR)

GPINDR, shown in Figure 17-9, contains the data currently sampled on TSEC1_RXD[7:0] and/or TSEC2_RXD[7:0] when these signals are configured for use as general-purpose input in GPIOCR (see Section 17.4.1.7.1, "General-Purpose I/O Control Register (GPIOCR)." GPINDR bits are updated only if the associated bits are configured as general-purpose input signals rather than their primary function as eTSEC signals.

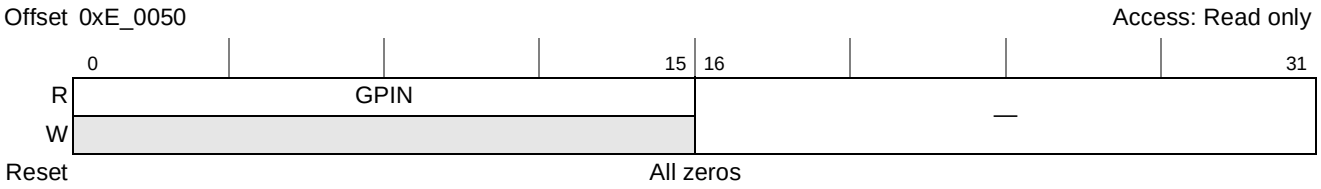


Figure 17-9. General-Purpose Input Data Register (GPINDR)

Table 17-12 describes the fields of GPINDR.

Table 17-12. GPINDR Field Descriptions

Bits	Name	Description
0–15	GPIN	General-purpose input data. When the corresponding signals are configured to be general-purpose input signals, the values sampled on these signals are reflected in GPIN. GPIN[7:0] corresponds to TSEC1_RXD[7:0] as follows: GPIN[7] ↔ TSEC1_RXD[7] GPIN[6] ↔ TSEC1_RXD[6] GPIN[5] ↔ TSEC1_RXD[5] GPIN[4] ↔ TSEC1_RXD[4] GPIN[3] ↔ TSEC1_RXD[3] GPIN[2] ↔ TSEC1_RXD[2] GPIN[1] ↔ TSEC1_RXD[1] GPIN[0] ↔ TSEC1_RXD[0] GPIN[15:8] corresponds to TSEC2_RXD[7:0] as follows: GPIN[15] ↔ TSEC2_RXD[7] GPIN[14] ↔ TSEC2_RXD[6] GPIN[13] ↔ TSEC2_RXD[5] GPIN[12] ↔ TSEC2_RXD[4] GPIN[11] ↔ TSEC2_RXD[3] GPIN[10] ↔ TSEC2_RXD[2] GPIN[9] ↔ TSEC2_RXD1] GPIN[8] ↔ TSEC2_RXD[0]
16–31	—	Reserved

17.4.1.8 Alternate Function Signal Multiplex Control Register (PMUXCR)

Shown in Figure 17-10, PMUXCR contains bits that enable DMA channels 2 and 3 which exist as alternate functions on local bus chip select pins $\overline{\text{LCS}}$[5:7], and interrupt input pins IRQ[9:11], respectively. Specifically, DMA request, acknowledge, and done signals comprise the secondary functions for the associated IRQ and local bus chip select signals.

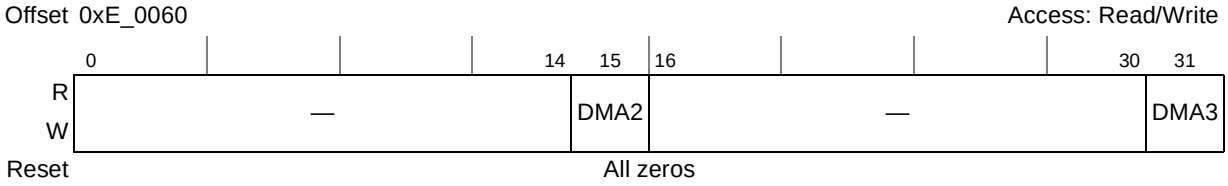


Figure 17-10. Alternate Function Pin Multiplex Control Register (PMUXCR)

Table 17-13 describes the bit settings of PMUXCR.

Table 17-13. PMUXCR Field Descriptions

Bits	Name	Description
0–14	—	Reserved
15	DMA2	Enables DMA channel 2 signals for external control of DMA transfers through this channel. 0 DMA channel 2 is not exposed to pins; the pins retain their primary function as local bus chip selects. Note that in this case, DMA channel 2 is still functional, but cannot perform externally controlled transfers. 1 DMA channel 2 is exposed to pins as follows: $\overline{\text{LCS5}}$ functions as $\overline{\text{DMA\_DREQ2}}$ $\overline{\text{LCS6}}$ functions as $\overline{\text{DMA\_DACK2}}$ $\overline{\text{LCS7}}$ functions as $\overline{\text{DMA\_DDONE2}}$
16–30	—	Reserved
31	DMA3	Enables DMA channel 3 signals for external control of DMA transfers through this channel. 0 DMA channel 3 is not exposed to pins; the pins retain their primary function as interrupt requests. Note that in this case, DMA channel 3 is still functional, but cannot perform externally controlled transfers. 1 DMA channel 3 is exposed to pins as follows: IRQ9 functions as $\overline{\text{DMA\_DREQ3}}$ IRQ10 functions as $\overline{\text{DMA\_DACK3}}$ IRQ11 functions as $\overline{\text{DMA\_DDONE3}}$

17.4.1.9 Device Disable Register (DEVDISR)

DEVDISR, shown in Figure 17-11, contains disable bits for various functional blocks. All functional blocks are enabled after reset; unneeded blocks can be disabled to reduce power consumption or allow their signals to be used as general-purpose I/O signals. See Section 17.5.1.2, “Shutting Down Unused Blocks,” and Section 17.4.1.7.1, “General-Purpose I/O Control Register (GPIOCR),” for more information.

When a block is disabled using the DEVDISR register, its clocks are shut down to save power. Blocks disabled by DEVDISR must not be re-enabled without a hard reset. The timebase disable (TB) control in DEVDISR may be set and cleared without requiring a hard-reset. However, because changing other bits in DEVDISR could cause errant behavior, it is recommended that DEVDISR be accessed using a read-modify-write operation when changing TB.

Note that a block can be enabled in DEVDISR and disabled functionally. The SerDes blocks are enabled or disabled based on the SerDes port selection (cfg_io_ports[0:2]) configuration inputs. See Section 4.4.3, “Power-On Reset Configuration,” for additional details.

Offset 0xE_0070 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PCIE1	PCIE2	—	—	LBC	—							SRIO	RMSG	DDR1	DDR2
W																
Reset	All zeros															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	e600_0	e600_1	TB	—	—	DMA	—		eTSEC1	eTSEC2	eTSEC3	eTSEC4	—	I2C	DUART	—
W																
Reset	All zeros															

Figure 17-11. Device Disable Register (DEVDISR)

Table 17-14 describes DEVDISR fields.

Table 17-14. DEVDISR Field Descriptions

Bits	Name	Description
0	PCIE1	Powers down the PCI Express controller 1 0 PCI Express controller 1 is enabled (clocks are on). 1 PCI Express controller 1 is disabled (clocks are shut down to minimize power).
1	PCIE2	Powers down the PCI Express controller 2 0 PCI Express controller 2 is enabled (clocks are on). 1 PCI Express controller 2 is disabled (clocks are shut down to minimize power).
2–3	—	Reserved
4	LBC	Local Bus controller 0 Local Bus controller is enabled (clocks are on). 1 Local Bus controller is disabled (clocks are shut down to minimize power).
5–11	—	Reserved
12	SRIO	Serial RapidIO controller disable 0 Serial RapidIO controller is enabled (clocks are on). 1 Serial RapidIO controller is disabled (clocks are shut down to minimize power).
13	RMSG	RapidIO Message Unit disable (Step 1) 0 RapidIO Message Unit is enabled (clocks are on). 1 RapidIO Message Unit is disabled (clocks are shut down to minimize power).
14	DDR1	DDR SDRAM controller 1 disable 0 DDR SDRAM controller 1 is enabled (clocks are on). 1 DDR SDRAM controller 1 is disabled (clocks are shut down to minimize power).
15	DDR2	DDR SDRAM controller 2 disable 0 DDR SDRAM controller 2 is enabled (clocks are on). 1 DDR SDRAM controller 2 is disabled (clocks are shut down to minimize power).
16	e600_0	e600 core 0 disable Note that prior to setting this bit, software must first place the core into a sleep state. See Section 17.5.1.3, “Software-Controlled e600 Core Power-Down States.”) 0 e600 core 0 is enabled 1 e600 core 0 is disabled. The core in the stopped state, in which it does not respond to interrupts. Instruction fetching is stopped, snooping is disabled, PLL is disabled, and clocks are shut down to all functional units of the core.

Table 17-14. DEVDISR Field Descriptions (continued)

Bits	Name	Description
17	e600_1	e600 core 1 disable Note that prior to setting this bit, software must first place the core into a sleep state. See Section 17.5.1.3, "Software-Controlled e600 Core Power-Down States.") 0 e600 core 1 is enabled 1 e600 core 1 is disabled. The core in the stopped state, in which it does not respond to interrupts. Instruction fetching is stopped, snooping is disabled, PLL is disabled, and clocks are shut down to all functional units of the core.
18	TB	Time base (timer facilities) of the e600 cores disable 0 If DEVDISR[e600_0]=0, the timer facilities for e600 core 0 are enabled. If DEVDISR[e600_1]=0, the timer facilities for e600 core 1 are enabled 1 The timer facilities for e600 core 0 and core 1 are disabled
19–20	—	Reserved
21	DMA	DMA controller disabled 0 DMA controller is enabled (clocks are on). 1 DMA controller is disabled (clocks are shut down to minimize power).
22–23	—	Reserved
24	eTSEC1	Three-speed Ethernet controller 1 disable 0 The eTSEC1 controller is enabled (clocks are on). 1 The eTSEC1 controller is disabled (clocks are shut down to minimize power). If DEVDISR[eTSEC2] is also set, RxD and TxD pins may be used for general-purpose I/O.
25	eTSEC2	Three-speed Ethernet controller 2 disable 0 The eTSEC2 controller is enabled (clocks are on). 1 The eTSEC2 controller is disabled (clocks are shut down to minimize power). If DEVDISR[eTSEC1] is also set, RxD and TxD pins may be used for general-purpose I/O.
26	eTSEC3	Three-speed Ethernet controller 3 disable 0 The eTSEC3 controller is enabled (clocks are on). 1 The eTSEC3 controller is disabled (clocks are shut down to minimize power).
27	eTSEC4	Three-speed Ethernet controller 4 disable 0 The eTSEC4 controller is enabled (clocks are on) 1 The eTSEC4 controller is disabled (clocks are shut down to minimize power).
28	—	Reserved
29	I2C	I^2C controllers disabled (Step 2) 0 The I^2C controllers are enabled (clocks are on) 1 The I^2C controllers are disabled (clocks are shut down to minimize power).
30	DUART	Dual UART controller disabled (Step 2) 0 Dual UART controllers are enabled (clocks are on). 1 The Dual UART controllers are disabled (clocks are shut down to minimize power).
31	—	Reserved

17.4.1.9.1 Using DEVDISR[TB] to Synchronize the Timebases of Multiple Cores

The time bases of multiple cores can be synchronized using the following procedure:

1. The primary core disables the time base units of all cores by clearing DEVDISR[TB] using a read-modify-write operation.

2. The primary core signals via a messaging interrupt (or some other messaging mechanism) that the subordinate core should initialize/clear its timebase registers (TBU/TBL).
3. The primary core could poll the message status register to see if the interrupt has been accepted, or there could be a message interrupt sent back by the subordinate core in response, or there could be a flag in memory to indicate that all time bases have been initialized/cleared.
4. The primary core then initializes/clears its time base registers.
5. The master then re-enables all time bases simultaneously by setting DEVDISR[TB] using a read-modify-write operation.

The time bases should then be fully synchronized and deterministic.

17.4.1.10 Power Management Control and Status Register (POWMGTCSR)

Shown in Figure 17-12, POWMGTCSR contains bits for placing the MPC8641 into low power states and for controlling when it wakes up. It also contains power management status bits. See Section 17.5.1.6.2, "Interrupts and Power Management Controlled by POWMGTCSR," for more information.

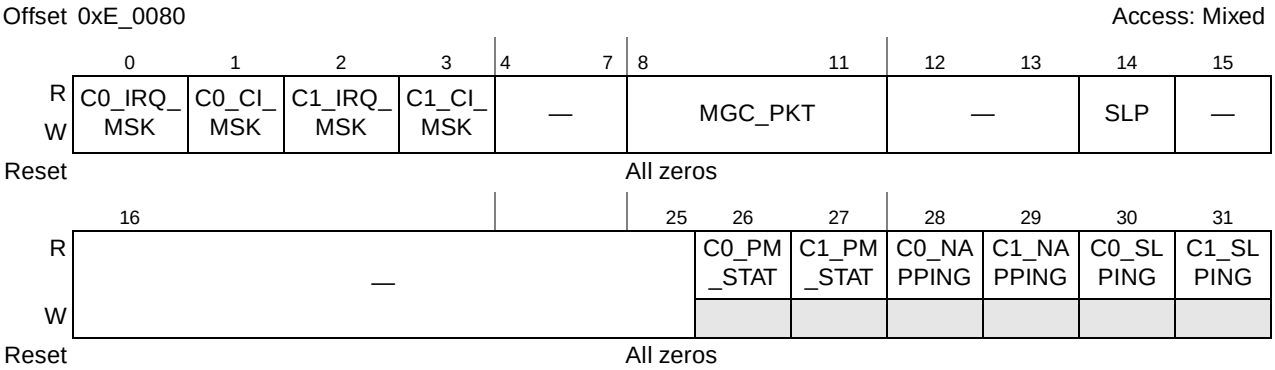


Figure 17-12. Power Management Control & Status Register (POWMGTCSR)

Table 17-15 describes the fields in PORMGTCSR.

Table 17-15. POWMGTCSR Field Descriptions

Bits	Name	Description
0	C0_IRQ_MSK	Core 0 Interrupt input mask 0 Interrupts cause core 0 to wake up from a low-power state. 1 Interrupts to core 0 are masked as a wake-up condition. Core 0 remains in a low-power state despite the presence of an interrupt request.
1	C0_CI_MSK	Core 0 Critical interrupt input mask 0 Critical interrupts cause core 0 to wake up from a low power state. 1 Critical interrupts to core 0 are masked as a wake-up condition. Core 0 remains in a low-power state despite the presence of a critical interrupt (machine check).
2	C1_IRQ_MSK	Core 1 Interrupt input mask 0 Interrupts cause core 1 to wake up from a low-power state. 1 Interrupts to core 1 are masked as a wake-up condition. Core 1 remains in a low-power state despite the presence of an interrupt request.

Table 17-15. POWMGTCSR Field Descriptions (continued)

Bits	Name	Description
3	C1_CI_MSK	Core 1 Critical interrupt input mask 0 Critical interrupts cause core 1 to wake up from a low power state. 1 Critical interrupts to core 1 are masked as a wake-up condition. Core 1 remains in a low-power state despite the presence of a critical interrupt (machine check).
4–7	—	Reserved
8–11	MGC_PKT	Magic packet wake-up Enables magic packet wake-up for each eTSEC
12-13	—	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 e600 cores and the system logic.
15–25	—	Reserved
26	C0_PM_STAT	Core 0 Power Management status 0 e600 core 0 is not in doze, nap, or sleep mode. 1 e600 core 0 is in doze, nap, or sleep mode because HID0[NAP] or HID0[SLP], and MSR[POW] are set. The core has halted instruction fetching and all functional units (except timer facilities in doze and nap mode) are quiesced. Note: if this bit is set and the associated core napping or sleeping bit is not set, then the core is in dozing state. For e600, dozing is a transient state. See Section 17.5.1, “Power Management for details.
27	C1_PM_STAT	Core 1 Power Management status 0 e600 core 1 is not in doze, nap, or sleep mode. 1 e600 core 1 is in doze, nap, or sleep mode because HID0[NAP] or HID0[SLP], and MSR[POW] are set. The core has halted instruction fetching and all functional units (except timer facilities in doze and nap mode) are quiesced. Note: if this bit is set and the associated core napping or sleeping bit is not set, then the core is in dozing state. For e600, dozing is a transient state. See Section 17.5.1, “Power Management for details.
28	C0_NAPPING	Core 0 Nap status 0 e600 core 0 is not in nap mode. 1 e600 core 0 is in nap mode because HID0[NAP] and MSR[POW] are set and the platform has no pending snoop transactions. The core has halted instruction fetching, snooping to the L1 and L2 caches is disabled and all of the core’s functional units except the timer facilities are shut down.
29	C1_NAPPING	Core 1 Nap status 0 e600 core 1 is not in nap mode. 1 e600 core 1 is in nap mode because HID0[NAP] and MSR[POW] are set and the platform has no pending snoop transactions. The core has halted instruction fetching, snooping to the L1 and L2 caches is disabled and all of the core’s functional units except the timer facilities are shut down.

Table 17-15. POWMGTCSR Field Descriptions (continued)

Bits	Name	Description
30	C0_SLPING	Core 0 Sleep status 0 e600 core 0 is not in sleep mode. 1 e600 core 0 is in sleep mode because HID0[SLEEP] and MSR[POW] are set. The core has halted fetching, snooping to the L1 and L2 caches is disabled and all of the core's functional units, including the timer facilities, are shut down
31	C1_SLPING	Core 1 Sleep status 0 e600 core 1 is not in sleep mode. 1 e600 core 1 is in sleep mode because HID0[SLEEP] and MSR[POW] are set. The core has halted fetching, snooping to the L1 and L2 caches is disabled and all of the core's functional units, including the timer facilities, are shut down

17.4.1.11 Machine Check Summary Register (MCPSUMR)

Shown in Figure 17-16, MCPSUMR contains bits summarizing some of the sources of a pending machine check interrupt. All MCPSUMR bits function as write-1-to-clear.

NOTE

Register fields designated as write-1-to-clear are cleared only by writing ones to them. Writing zeros to them has no effect.

Offset 0xE_0090 Access: w1c[1]

	0	27	28	29	30	31
R	—		SRESET_0	SRESET_1	MCP_0	MCP_1
W			w1c	w1c	w1c	w1c
Reset	All zeros					

Figure 17-13. Machine Check Summary Register (MCPSUMR)

1 Note that writing a one to any of bits MPCSUMR[28-31] will clear all the bits in MCPSUMR. Writing a one to any bits MCPSUMR[0-27] will be ignored.

Table 17-16 describes the bit settings of MCPSUMR.

Table 17-16. MCPSUMR Field Descriptions

Bits	Name	Description
0–24	—	Reserved
28	SRESET_0	Soft reset 0 0 Soft reset exception was not caused by $\overline{\text{SRESET\_0}}$ assertion. 1 Reset exception was caused by the assertion of the $\overline{\text{SRESET\_0}}$ input signal.
29	SRESET_1	Soft reset 1 0 Soft reset exception was not caused by $\overline{\text{SRESET\_1}}$ assertion. 1 Reset exception was caused by the assertion of the $\overline{\text{SRESET\_1}}$ input signal.

Table 17-16. MCPSUMR Field Descriptions (continued)

Bits	Name	Description
30	MCP_0	Machine check 0 0 Machine check exception was not caused by $\overline{\text{MCP\_0}}$ assertion. 1 Machine check exception was caused by the assertion of the $\overline{\text{MCP\_0}}$ input signal.
31	MCP_1	Machine check 1 0 Machine check exception was not caused by $\overline{\text{MCP\_1}}$ assertion. 1 Machine check exception was caused by the assertion of the $\overline{\text{MCP\_1}}$ input signal.

17.4.1.12 Reset Request Status and Control Register (RSTRSCR)

Shown in Table 17-17, the RSTRSCR contains a mask bit for serial RapidIO reset requests and status for boot sequencer, serial RapidIO, and software reset request bits. If any of the status bits are set and not masked, the MPC8641 will assert $\overline{\text{HRESET\_REQ}}$. This usually results from the detection of an unrecoverable error condition.

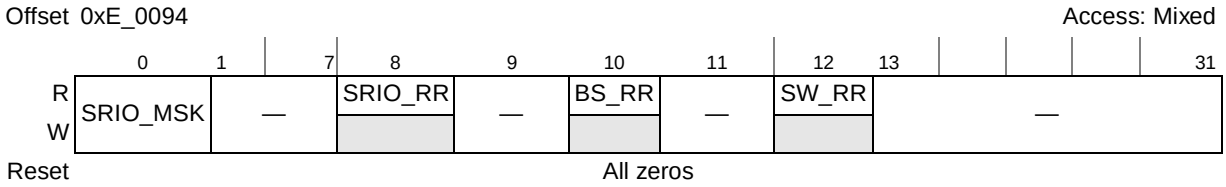


Figure 17-14. Reset Request Status and Control Register (RSTRSCR)

Table 17-17 describes the field settings of RSTRSCR.

Table 17-17. RSTRSCR Field Descriptions

Bits	Name	Description
0	SRIO_MSK	Serial RapidIO (SRIO) reset request mask 0 Mask is not enabled. Reset request from SRIO will cause device $\overline{\text{HRESET}}$ request. 1 Mask is enabled, reset request from SRIO will be ignored
2–7	—	Reserved
8	SRIO_RR	Serial RapidIO (SRIO) reset request 0 Normal operation 1 Reset requested by SRIO block
9	—	Reserved
10	BS_RR	Boot sequencer reset request 0 Normal operation 1 Reset requested by boot sequencer (due to error)
11	—	Reserved
12	SW_RR	Software settable reset request 0 Normal operation 1 Reset requested by software (RSTCR[HRESET_REQ]=1)
13–31	—	Reserved

17.4.1.13 Processor Version Register (PVR)

Shown in Figure 17-15, the PVR contains the e600 processor version number. It is a memory-mapped copy of the PVR from the e600 core (and is therefore accessible to external devices). See Section 6.1.4.1, “Processor Version Register (PVR),” for information on the e600 core PVR register.

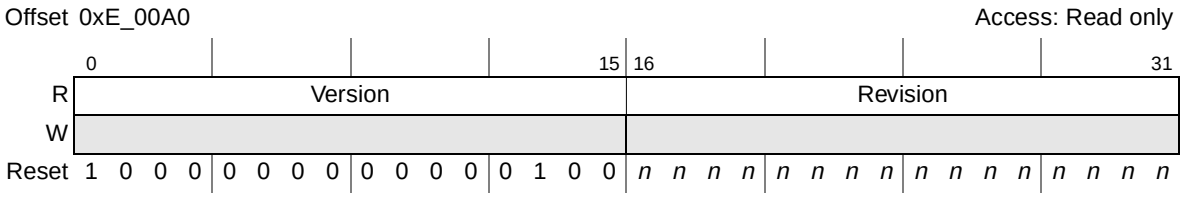


Figure 17-15. Processor Version Register (PVR)

Table 17-18 describes the fields of PVR.

Table 17-18. PVR Field Descriptions

Bits	Name	Description
0–15	Version	A 16-bit number that identifies the version of the processor core. Different version numbers indicate major differences between processors, such as which optional facilities and instructions are supported.
16–31	Revision	A 16-bit number that distinguishes between implementations of the version. Different revision numbers indicate minor differences between processors having the same version number, such as clock rate and engineering change level.

17.4.1.14 System Version Register (SVR)

Shown in Figure 17-16, the SVR contains the system version number for the MPC8641 implementation. This value can also be read though the SVR SPR of the e600 core. See Section 6.1.4.2, “System Version Register (SVR),” for information on the e600 core SVR register.

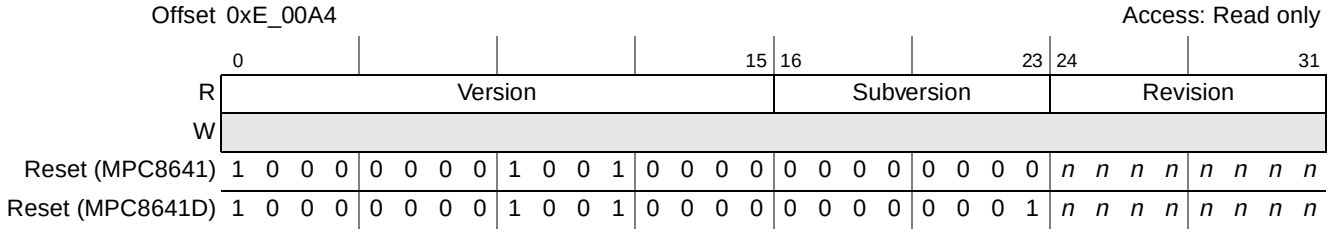


Figure 17-16. System Version Register (SVR)

Table 17-19 describes the fields of SVR.

Table 17-19. SVR Field Descriptions

Bits	Name	Description
0-15	Version	A 16-bit number that identifies the version of the device. Different version numbers indicate major differences between devices, such as which optional facilities and instructions are supported.
16-23	Subversion	An 8-bit number that further differentiates the device. Different subversion numbers indicate the number of cores for the MPC8641 family.
24–31	Revision	System revision for the MPC8641 system logic.

Table 17-20 shows the SVR settings for the MPC8641 and MPC8641D. The revision level is updated for each silicon revision.

Table 17-20. SVR Settings

Device Name	System Version Number		
	Version Number	Subversion Number	Starting Revision Level
MPC8641	0x8090	0x00	0x21
MPC8641D	0x8090	0x01	0x21

17.4.1.15 Reset Control Register (RSTCR)

Shown in Figure 17-17, the RSTCR contains the reset control bits. Through this register, software can request the assertion of device $\overline{\text{HRESET\_REQ}}$.

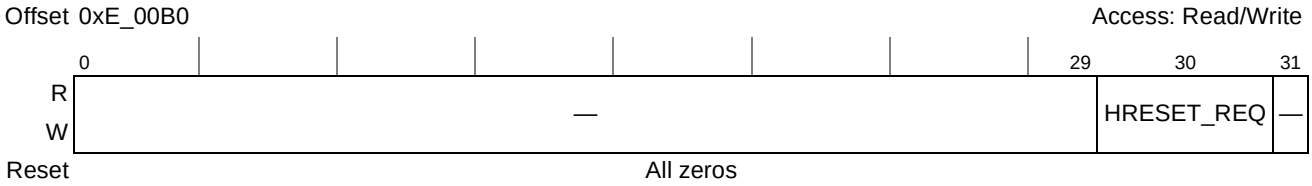


Figure 17-17. Reset Control Register (RSTCR)

Table 17-21 describes the bit settings of RSTCR.

Table 17-21. RSTCR Field Descriptions

Bits	Name	Description
0–29	—	Reserved
30	HRESET_REQ	Hardware reset request 0 No hardware reset request 1 Request hardware reset
31	—	Reserved

17.4.1.16 DDRC1 Clock Disable Register (DDR1CLKDR)

Shown in Figure 17-18, the DDR1CLKDR contains bits that allow disabling the clocks of the DDR SDRAM controller on port 1. This allows power savings by turning off unused DDR clock outputs.

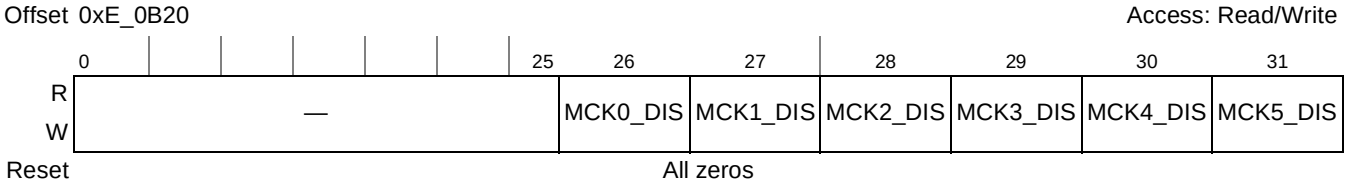


Figure 17-18. DDRC1 Clock Disable Register (DDR1CLKDR)

Table 17-22 describes the bit settings of DDR1CLKDR.

Table 17-22. DDR1CLKDR Field Descriptions

Bits	Name	Description
0–25	—	Reserved
26	MCK0_DIS	DDR clock 0 disable 0 D1_MCK0 is enabled. 1 D1_MCK0 is disabled.
27	MCK1_DIS	DDR clock 1 disable 0 D1_MCK1 is enabled. 1 D1_MCK1 is disabled.
28	MCK2_DIS	DDR clock 2 disable 0 D1_MCK2 is enabled. 1 D1_MCK2 is disabled.
29	MCK3_DIS	DDR clock 3 disable 0 D1_MCK3 is enabled. 1 D1_MCK3 is disabled.
30	MCK4_DIS	DDR clock 4 disable 0 D1_MCK4 is enabled. 1 D1_MCK4 is disabled.
31	MCK5_DIS	DDR clock 5 disable 0 D1_MCK5 is enabled. 1 D1_MCK5 is disabled.

17.4.1.17 DDRC2 Clock Disable Register (DDR2CLKDR)

Shown in Figure 17-19, the DDR2CLKDR contains bits that allow disabling the clocks of the DDR SDRAM controller on port 2. This allows power savings by turning off unused DDR clock outputs.

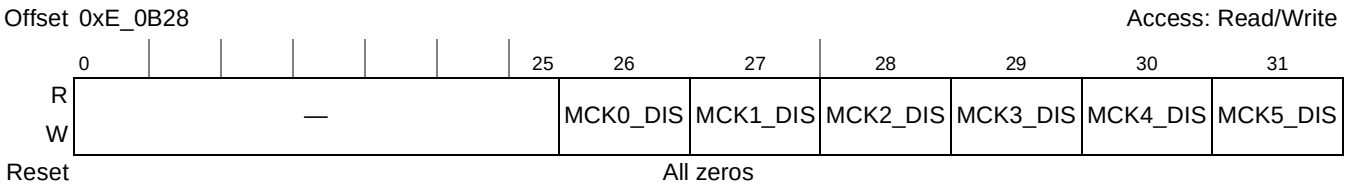


Figure 17-19. DDRC2 Clock Disable Register (DDR2CLKDR)

Table 17-23 describes the bit settings of DDRCLKDR.

Table 17-23. DDR2CLKDR Field Descriptions

Bits	Name	Description
0–25	—	Reserved
26	MCK0_DIS	DDR clock 0 disable 0 D2_MCK0 is enabled. 1 D2_MCK0 is disabled.
27	MCK1_DIS	DDR clock 1 disable 0 D2_MCK1 is enabled. 1 D2_MCK1 is disabled.
28	MCK2_DIS	DDR clock 2 disable 0 D2_MCK2 is enabled. 1 D2_MCK2 is disabled.
29	MCK3_DIS	DDR clock 3 disable 0 D2_MCK3 is enabled. 1 D2_MCK3 is disabled.
30	MCK4_DIS	DDR clock 4 disable 0 D2_MCK4 is enabled. 1 D2_MCK4 is disabled.
31	MCK5_DIS	DDR clock 5 disable 0 D2_MCK5 is enabled. 1 D2_MCK5 is disabled.

17.4.1.18 Clock Out Control Register (CLKOCR)

Shown in Figure 17-20, the CLKOCR contains control bits that select the clock sources to be placed on the clock out (CLK_OUT) signal.

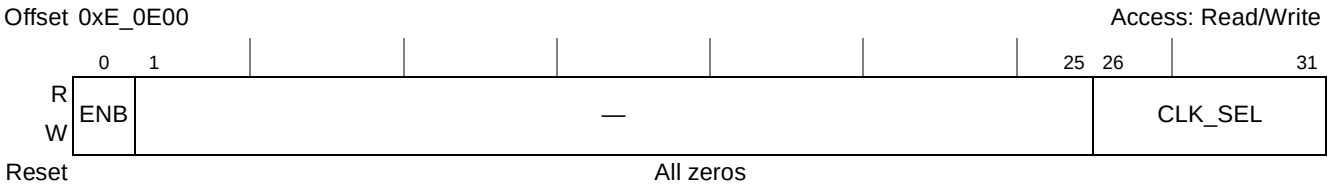


Figure 17-20. Clock Out Control Register (CLKOCR)

Table 17-24 describes the bit settings of CLKOCR.

Table 17-24. 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]

Table 17-24. CLKOCR Field Descriptions (continued)

Bits	Name	Description
1–25	—	Reserved
26–31	CLK_SEL	Clock out select 1*nnnnn* Reserved 01*nn*0*n* Reserved 01*nn*1*n* Reserved 000000 MPX (platform) clock 000001 MPX (platform) clock divided by 2 000010 SYSCLK (echoes SYSCLK input) 000011 SYSCLK divided by 2 (demonstrates platform PLL lock) 000100 Reserved 000101 Reserved 000110 Reserved 000111 Reserved 001000 Reserved 001001 Reserved 001010 Reserved 001011 Reserved 001100 Reserved 001101 Reserved 001110 Reserved 001111 Reserved

17.4.1.19 SerDes 1 Control Register 0 (SRDS1CR0)

Shown in Figure 17-21, the SRDS1CR0 contains control bits for the SerDes on high speed interface port 1.

Offset 0xE_0F04 Access: Read/Write

	0–3	4–7	8	9	10	11	12	13	14	15
R W	—									
Reset	*n* *n* *n* *n*	*n* *n* *n* *n*	0	0	0	0	0	0	0	0

	16–19	20–23	24	25	26	27	28	29	30	31
R W	XMITEQAD	XMITEQEH	—							
Reset	*n* *n* *n* *n*	*n* *n* *n* *n*	0	0	0	0	0	0	0	0

Figure 17-21. SerDes 1 Control Register 0 (SRDS1CR0)

Table 17-25 describes the bit settings of SRDS1CR0.

Table 17-25. SRDS1CR0 Field Descriptions

Bits	Name	Description
0–15	—	Reserved. Software must preserve the values of these bits.
16–19	XMITEQAD	Transmit equalization selection bus for SD1 lanes 0-3 (a-d) Default value: 1100 PCI Express Bit 0 is amplitude select 0 5/6 Vdd-diff-pk=pk 1 Vdd-diff-pk=pk Bits 1:3 are equalization amplitude: 000 No Equalization 001 1.09x relative amplitude 010 1.2x relative amplitude 011 1.33x relative amplitude 100 1.5x relative amplitude 101 1.71x relative amplitude 110 2.0x relative amplitude
20–23	XMITEQEH	Transmit equalization selection bus for SD1 lanes 4-7 (e-h) Default value: 1100 PCI Express Bit 0 is amplitude select 0 5/6 Vdd-diff-pk=pk 1 Vdd-diff-pk=pk Bits 1:3 are defined: 000 No Equalization 001 1.09x relative amplitude 010 1.2x relative amplitude 011 1.33x relative amplitude 100 1.5x relative amplitude 101 1.71x relative amplitude 110 2.0x relative amplitude
24–31	—	Reserved. Software must preserve the values of these bits.

17.4.1.20 SerDes 1 Control Register 1 (SRDS1CR1)

Shown in Figure 17-22, the SRDS1CR1 contains control bits for the SerDes on high speed interface port 1.

Offset 0xE_0F08 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9–15
R	PD0	PD1	PD2	PD3	PD4	PD5	PD6	PD7	PDTXCLK	—
W										
Reset	All zeros									

	16–20	21–24	25–31
R	—	LBSELTYPE	—
W			
Reset	All zeros		

Figure 17-22. SerDes 1 Control Register 1 (SRDS1CR1)

Table 17-26 describes the bit settings of SRDS1CR1.

Table 17-26. SRDS1CR1 Field Descriptions

Bits	Name	Description
0	PD0	Lane 0 power down 0 Normal 1 The serial links are disabled and power usage is minimized in all internal cells.
1	PD1	Lane 1 power down 0 Normal 1 The serial links are disabled and power usage is minimized in all internal cells.
2	PD2	Lane 2 power down 0 Normal 1 The serial links are disabled and power usage is minimized in all internal cells.
3	PD3	Lane 3 power down 0 Normal 1 The serial links are disabled and power usage is minimized in all internal cells.
4	PD4	Lane 4 power down 0 Normal 1 The serial links are disabled and power usage is minimized in all internal cells.
5	PD5	Lane 5 power down 0 Normal 1 The serial links are disabled and power usage is minimized in all internal cells.
6	PD6	Lane 6 power down 0 Normal 1 The serial links are disabled and power usage is minimized in all internal cells.
7	PD7	Lane 7 power down 0 Normal 1 The serial links are disabled and power usage is minimized in all internal cells.
8	PDTXCLK	Power down Txclock 0 Normal 1 Tx clock is disabled

Table 17-26. SRDS1CR1 Field Descriptions (continued)

Bits	Name	Description
9–31	—	Reserved
21–24	LBSELTYPE	Select type loop-back 0000 Application mode (normal operating mode) 0001 Digital loopback mode All other settings reserved.

17.4.1.21 SerDes 2 Control Register 0 (SRDS2CR0)

Shown in Figure 17-21, the SRDS2CR0 contains control bits for the SerDes on high speed port 2.

Offset 0xE_0F40 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	–															
Reset	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	XMITEQAD				XMITEQEH				–							
Reset	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	0	0	0	0	0	0	0	0

Figure 17-23. SerDes 2 Control Register 0 (SRDS2CR0)

Table 17-25 describes the bit settings of SRDS2CR0.

Table 17-27. SRDS2CR0 Field Descriptions

Bits	Name	Description
0-15	—	Reserved. Software must preserve the values of these bits.
16–19	XMITEQAD	Transmit equalization selection bus for SD2 lanes 0-3 (a-d) Default value: 1100 PCI Express Bit 0 is amplitude select: 0 5/6 Vdd-diff-pk=pk 1 Vdd-diff-pk=pk Bits 1:3 are equalization amplitude: 000 No Equalization 001 1.09x relative amplitude 010 1.2x relative amplitude 011 1.33x relative amplitude 100 1.5x relative amplitude 101 1.71x relative amplitude 110 2.0x relative amplitude

Table 17-27. SRDS2CR0 Field Descriptions (continued)

Bits	Name	Description
20–23	XMITEQEH	Transmit equalization selection bus for SD2 lanes 4-7 (e-h) Default value: 1100 PCI Express 0011 Serial RapidIO Bit 0 is amplitude select: 0 5/6 Vdd-diff-pk=pk 1 Vdd-diff-pk=pk Bits 1:3 are equalization amplitude: 000 No Equalization 001 1.09x relative amplitude 010 1.2x relative amplitude 011 1.33x relative amplitude 100 1.5x relative amplitude 101 1.71x relative amplitude 110 2.0x relative amplitude
24–31	—	Reserved. Software must preserve the values of these bits.

17.4.1.22 SerDes 2 Control Register 1 (SRDS2CR1)

Shown in Figure 17-24, the SRDS2CR1 contains control bits for the SerDes on high speed interface port 2.

Offset 0xE_0F44 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9–15
R W	PD0	PD1	PD2	PD3	PD4	PD5	PD6	PD7	PDTX CLK	—
Reset	All zeros									

	16–20	21–24	25–31
R W	—	LBSELTYPE	—
Reset	All zeros		

Figure 17-24. SerDes 2 Control Register 1 (SRDS2CR1)

Table 17-28 describes the bit settings of SRDS2CR1.

Table 17-28. SRDS2CR1 Field Descriptions

Bits	Name	Description
0	PD0	Lane 0 power down 0 Normal 1 The serial links are disabled and power usage is minimized in all internal cells.
1	PD1	Lane 1 power down 0 Normal 1 The serial links are disabled and power usage is minimized in all internal cells.

Table 17-28. SRDS2CR1 Field Descriptions (continued)

Bits	Name	Description
2	PD2	Lane 2 power down 0 Normal 1 The serial links are disabled and power usage is minimized in all internal cells.
3	PD3	Lane 3 power down 0 Normal 1 The serial links are disabled and power usage is minimized in all internal cells.
4	PD4	Lane 4 power down 0 Normal 1 The serial links are disabled and power usage is minimized in all internal cells.
5	PD5	Lane 5 power down 0 Normal 1 The serial links are disabled and power usage is minimized in all internal cells.
6	PD6	Lane 6 power down 0 Normal 1 The serial links are disabled and power usage is minimized in all internal cells.
7	PD7	Lane 7 power down 0 Normal 1 The serial links are disabled and power usage is minimized in all internal cells.
8	PDTXCLK	Power down Tx clock 0 Normal 1 Transmit clock is disabled
9–31	—	Reserved
21–24	LBSELTYPE	Select type loop-back 0000 Application mode (normal operating mode) 0001 Digital loopback mode All other settings reserved.

17.5 Functional Description

This section describes the global utilities from a functional perspective.

17.5.1 Power Management

The MPC8641 has features to minimize power consumption at several levels. Software can shut down clocks to individual blocks when they are not needed through a memory-mapped register (DEVDISR). Additionally, software running on the e600 core can access the core's SPRs to put the device into 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 a sleep state.

Note that the software that writes to either DEVDISR or POWMGTCR can be running either on the e600 core or on an external master that can write to the MPC8641 memory-mapped registers through the Serial RapidIO or PCI Express interfaces.

These features are described in further detail in this section.

17.5.1.1 Relationship Between Core and Device Power Management States

The MPC8641 has one low-power device state: sleep. The e600 cores have three low-power states independent of device state: doze, nap and sleep. The mapping of core power management states is shown in Figure 17-25 showing state transitions and platform controls from the perspective of the e600 core. The platform $\overline{\text{qack}}$ signal is based on pending snoops, either from one of the cores or from a coherent I/O transaction. Note that in a dual-core MPC8641, each core can independently enter any low-power state.

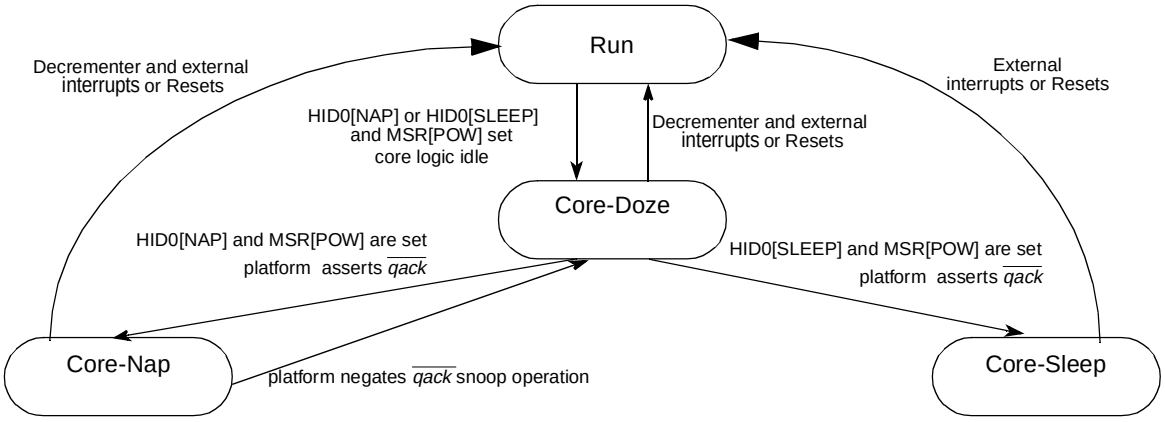


Figure 17-25. e600 Core Power Management State Diagram

Table 17-29 lists basic characteristics of the low-power modes and the full on mode, along with the corresponding states of the MPC8641. Note that there are many other variables that control the state transitions between MPC8641 power management states.

Table 17-29. MPC8641 Power Management Modes—Basic Description

MPC8641 Mode	e600 Core Mode	Description	Core Responds To		Core Clocks
			Snoop	Interrupts	
Run	Full On	All units operating normally.	Yes	Yes	On
	Doze	Core is attempting to enter Nap or Sleep state. Core stops dispatching new instructions (core is halted). Platform automatically transitions the core from Doze to Nap or Sleep state when core interface is quiesced.	Yes	Yes	On
	Nap	Core is stopped with clocks off except to time base and decrementer. Platform automatically transitions the core from Nap to Doze state in order to service a snoop.	No	Yes	Dec/Time Base clocks on & others off
	Sleep	Core is stopped with clocks off. Clocks powered down to all blocks (including core time base and decrementer)	No	Yes	PLL on & internal clocks off
Sleep	Sleep	Both cores must be attempting to enter Sleep state before POWMGTCSR[SLP] is set.	No	No	PLL on & internal clocks off

17.5.1.2 Shutting Down Unused Blocks

As described in Section 17.4.1.9, "Device Disable Register (DEVDISR)," DEVDISR provides a way to shut down certain functional blocks within the MPC8641 when they are not needed in a particular system. DEVDISR can be written by the e600 core or by an external master. Powering down a block in this way turns off all clocks to that block.

With the exception of the timebase disable (TB), the controls in DEVDISR were designed with the expectation that, once initialized by software, they would be modified only by a hard system reset ($\overline{\text{HRESET}}$). It is recommended that this register be written only during system initialization. The results of re-enabling previously disabled blocks (by clearing the corresponding DEVDISR field) without a hard reset are boundedly undefined. DEVDISR[TB] may be set and cleared without requiring a hard-reset. However, because changing other bits in DEVDISR could cause errant behavior, it is recommended that DEVDISR be accessed using a read-modify-write operation when changing TB.

NOTE

Functional blocks disabled using DEVDISR cannot respond to configuration accesses. Any access to configuration, control, and status registers of a disabled block is a programming error.

17.5.1.3 Software-Controlled e600 Core Power-Down States

The e600 software can attempt to place the core in nap or sleep power-down states by writing to HID0 in the core. The device controls shutting off the clocks to the e600 core based on pending coherency transactions (snoops).

17.5.1.3.1 Doze Mode

In e600, doze mode is an intermediate state between full on or nap or sleep state. The e600 core remains in doze state until all pending snoops in the device are serviced.

In doze mode, the e600 core suspends instruction execution, significantly reducing the power consumption of the core. Snooping of the L1 and L2 caches is still supported and thus the data in the data cache is kept coherent. Interrupts directed to the core as described in Section 9.1.3, "Interrupts to the Processor Core," are monitored by the device and cause the MPC8641 to use the defined handshake mechanism to exit the core from doze mode to allow the core to recognize and process the interrupt. Note that on any interrupt or exception, the e600 core clears the bit associated with POW in SRR1[13]. So the core does not automatically return to the previous low power mode when it returns from the interrupt.

The e600 core's timer facilities are still enabled during doze mode, and core time base interrupts can be generated. All device logic external to the core remains fully operational in doze mode.

17.5.1.3.2 Nap Mode

In nap mode all clocks internal to the e600 core are turned off except for its timer facilities clock (the core time base and decrementer). The L1 and L2 caches do not respond to snoops in nap mode, however a pending snoop will cause the device to transition the e600 core from nap to doze state, in which the snoop can be serviced. Once the snoop processing is complete, the device transitions the e600 core back into nap state.

Similar to doze mode, interrupts occurring in nap mode cause the device to wake up the e600 core in order to service the interrupt. Note that the e600 does not automatically return to the previous low power mode because the e600 clears the bit associated with MSR[POW] in SRR1[13] upon taking an exception.

All device logic external to the e600 core remains fully operational in nap mode.

17.5.1.3.3 Core Sleep Mode

In core sleep mode all clocks internal to the e600 core are turned off including its timer facilities clock (the core time base and decrementer). The L1 and L2 caches do not respond to snoops in sleep mode, so software must flush the caches before entering sleep mode.

Similar to doze and nap modes, interrupts occurring in sleep mode cause the device to wake up the e600 core in order to service the interrupt. Note that the e600 does not automatically return to the previous low power mode because the e600 clears the bit associated with MSR[POW] in SRR1[13] upon taking an exception.

All device logic external to the e600 core remains fully operational in core sleep mode.

17.5.1.4 Software-Controlled Device Power-Down State

The e600 software or an external device can attempt to place the device in sleep power-down states by writing to the POWMGTCSR. Before entering device sleep mode, software must place the core or cores in core sleep mode. For e600 software placing the device in sleep power-down state, the suggested code sequence is:

```
mtspr setting HID0[SLEEP] to 1
cache flushes etc.
store setting POWMGTCSR[SLP] to 1
sync
mtspr setting MSR[POW] to 1
```

17.5.1.4.1 Device Sleep Mode

In device sleep mode, the e600 cores are in core sleep mode and all I/O interfaces in the device logic are shut down. Only the clocks to the MPC8641 PIC are still running so that an external interrupt can wake up the device. After the core and I/O interfaces have shut down, ASLEEP is asserted and READY is negated.

NOTE

Only external interrupts can wake the MPC8641 from sleep mode. Internal interrupt sources depend on an active clock for their operation and these are disabled in sleep mode.

17.5.1.5 Power Management Control Fields

The e600 core provides the following fields to signal power management requests to the MPC8641 device logic.

- MSR[POW]—Enables core power management modes selected by HID0[NAP,SLEEP].
- HID0[NAP]—Signals the MPC8641 to initiate nap mode.

- HID0[SLEEP]—Signals the MPC8641 to initiate core sleep mode.

These register fields and their functional relationship are shown in Figure 17-25. The *e600 Reference Manual* has details on accessing these power management control bits.

Once the e600 cores are in core sleep mode, an external master can initiate device power management requests by setting or SLP bit in the memory-mapped power management control and status register (POWMGTCSR). Maintaining cache coherency requires significant preparation by the core before entering sleep mode. For this reason only the core can initiate core sleep mode.

17.5.1.6 Interrupts and Power Management

17.5.1.6.1 Interrupts and Power Management Controlled by MSR and HID0

When an interrupt is asserted to the CPU, the core saves portions of the MSR to SRR1, and restores those values on return from the routine. However, MSR[POW], which gates the HID0[NAP] and HID0[SLEEP] bits, is always cleared when saved to SRR1 and remains cleared when it is restored; hence the corresponding power management status bits POWMGTCSR[Cx_NAPING,Cx_SLPING]) negate when the interrupt begins processing in the core. They do not return to their previous state when the core executes an **rfi** instruction.

17.5.1.6.2 Interrupts and Power Management Controlled by POWMGTCSR

The IRQ_MSK and CI_MSK fields of the POWMGTCSR register prevent interrupts or critical interrupts from waking the device from a low power state.

Note that interrupts caused by the system management ($\overline{\text{SMI}}$) and machine check ($\overline{\text{MCP}}$) signals are not masked by the IRQ_MSK and CI_MSK fields. See Section 17.4.1.10, “Power Management Control and Status Register (POWMGTCSR),” for detailed information about the bits of POWMGTCSR.

Note also that unmasked interrupts that occur while the device is in the process of going into the sleep state (before sleep is completely attained) can also cause the device to return the core to full power operation.

17.5.1.7 Requirements for Reaching and Recovering from Device Sleep State

In order to successfully reach the sleep state, I/O traffic to the device must be stopped. The logic that controls the power down sequence waits for all I/O interfaces to become idle. In some applications this may happen eventually without actively shutting down interfaces, but most likely, software will have to take steps to shut down the eTSEC, PCI Express, and Serial RapidIO interfaces before issuing the command (writing to POWMGTCSR[SLP]) to put the device into sleep state.

The Serial RapidIO and PCI Express interfaces will begin retrying inbound transactions before entering a power down state. Upon exiting sleep, the Serial RapidIO interface begins its training sequence to re-establish the link. The PCI Express 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 Express 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.

17.5.2 General-Purpose I/O Signals

Several groups of signals can optionally be used as general-purpose I/O signals when not being used for their primary function. The general-purpose I/O functionality of these signals can be enabled through configuration registers in the global utilities block. These signals are the following:

- TSEC1_RXD[0:7], TSEC1_TXD[0:7], TSEC2_RXD[0:7] and TSEC2_TXD[0:7]. The eTSEC1 and eTSEC2 pins are fixed as either inputs or outputs based on the direction of the signal's primary function. The TSEC1_TXD and TSEC2_TXD pins are always outputs, so these signals may only be used as outputs when configured as general-purpose I/O. Similarly, the TSEC1_RXD and TSEC2_RXD pins are used as inputs when configured as general-purpose I/O.

  The eTSEC1 and eTSEC2 pins are available when the eTSEC1 and eTSEC2 block are disabled. The TxD signals can then be enabled as general-purpose outputs and the RxD pins can be enabled as general-purpose inputs.

  Note that both eTSEC1 and eTSEC2 must be disabled when being used for GPIO; one cannot be used for ethernet while the other is being used for GPIO.

When configured as general-purpose I/O signals, software can read inputs by reading the associated GPIO data register. Output values can be set by writing the to the associated GPIO data register. For details regarding the control and status of the general-purpose I/O signals, see Section 17.4.1.7.1, "General-Purpose I/O Control Register (GPIOCR)."

17.5.3 Interrupt and Local Bus Signal Multiplexing

The MPC8641 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 $\overline{\text{DMA\_DDONE3}}$.

Chapter 18
Device Performance Monitor

This chapter describes the performance monitor facility, which can be used to monitor and optimize performance. The e600 core implements a separate performance monitor for strictly core-related behavior, such as instruction timing and L1 cache operations. This is described in the *PowerPC e600 Core Reference Manual* (Freescale Document Order No. E600CORERM).

Section 18.4.7, "Performance Monitor Events," briefly describes the events that can be monitored. Refer to the individual chapters for a better understanding of these events.

18.1 Introduction

The MPC8641 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 e600 core, it differs in that it is implemented using memory-mapped registers and it counts events outside the e600 core, for example, PCI, DDR, and L2 cache events.

Performance monitor counters (PMC0–PMC9) 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–PMC9 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 MPC8641's behavior in any system or software environment.
- The performance monitor facility can be used to aid system developers when bringing up and debugging systems.
- System performance can be increased by monitoring memory hierarchy behavior. This can help to optimize algorithms used to schedule or partition tasks and to refine the data structures and distribution used by each task.

18.1.1 Overview

Figure 18-1 is a high-level block diagram of the performance monitor, which consists of a global control register (PMGC0), one 64-bit counter (PMC0), nine 32-bit counters, and two control registers per counter (18 total control registers). The global control register PMGC0 affects all counters and takes priority over local control registers. The local control registers are divided into two groups, as follows:

- Local control A registers control counter freezing, overflow condition enable, event selection, and burstiness. Local control register PMLCA0, which controls counter PMC0, does not contain event selection because PMC0 counts only cycles.
- Local control B registers control the start and stop triggering, contain the counters' threshold values, and the value of the threshold multiplier. Local control register PMLCB0, which controls PMC0, does not contain threshold information because PMC0 only counts cycles.

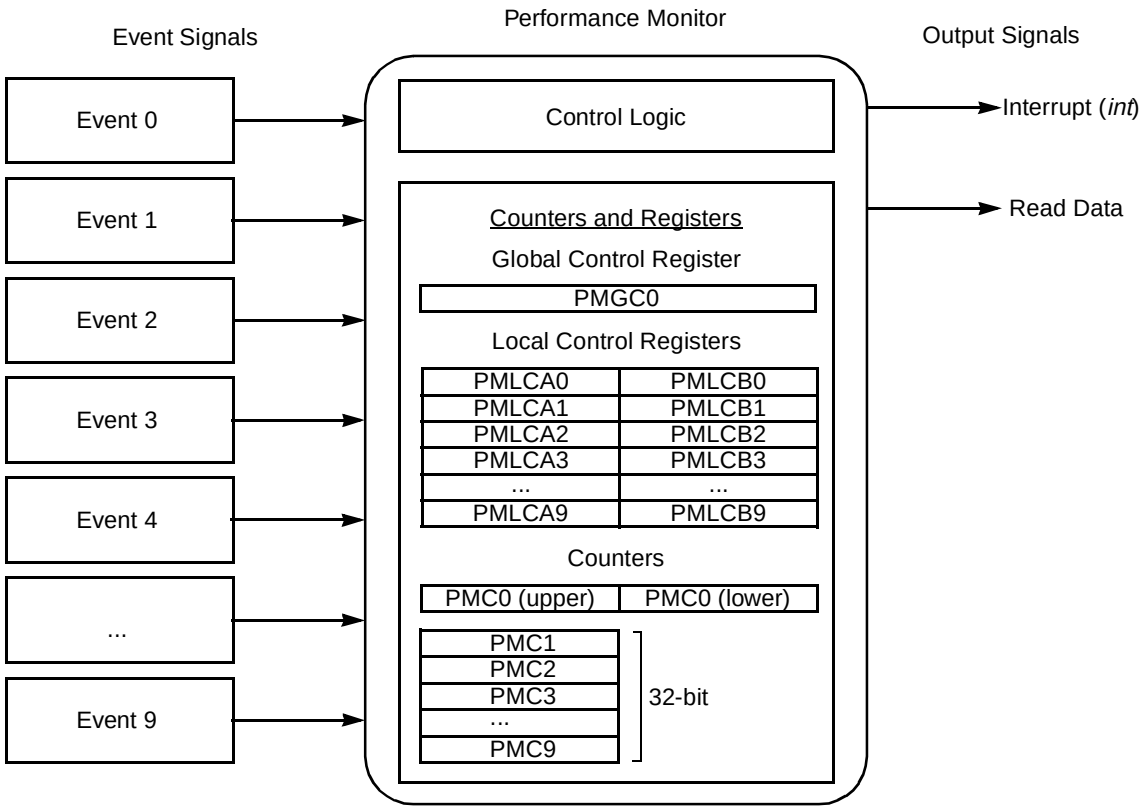


Figure 18-1. Performance Monitor Block Diagram

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 nine counters. Counter-specific events can be counted only on the counter where the event is defined.

The performance monitor can generate an interrupt on overflow. Several control registers specify how a performance monitor interrupt is signalled. The PMCs can also be programmed to freeze when an interrupt is signalled.

18.1.2 Features

The 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
- Nine 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 nine 32-bit counters
- Ability to count up to 576 counter-specific events
- Triggering and chaining capability
- Duration threshold counting
- Burstiness feature that permits counting of burst events with a programmable time between bursts
- Ability to generate an interrupt on overflow

18.2 Signal Descriptions

The performance monitor does not have any signals that are driven externally (off-chip) but it does assert the internal interrupt (*int*) signal on a performance monitor interrupt condition.

18.3 Memory Map and Register Definition

The performance monitor registers reside in the CCSR space starting at block base address 0xE_1000. Undefined 4-byte address spaces within offset 0x000–0xFFF are reserved. This section describes the registers implemented to support the performance monitor facilities. Table 18-1 lists the performance monitor registers. These registers can be read or written only with 32-bit accesses.

18.3.1 Register Summary

The performance monitor uses nine counter registers and a group of local control registers that are used to specify the method of counting. Two local control registers are associated with each counter in addition to a global control register that applies to all counters.

Table 18-1. Control Register Memory Map

Performance Monitor—Block Base Address 0xE_1000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0x000	PMGC0—Performance monitor global control register	R/W	0x0000_0000	18.3.2.1/18-5
0x010	PMLCA0—Performance monitor local control register A0	R/W	0x0000_0000	18.3.2.2/18-5
0x014	PMLCB0—Performance monitor local control register B0	R/W	0x0000_0000	18.3.2.2/18-5
0x018	PMC0 (lower)—Performance monitor counter 0 lower	R/W	0x0000_0000	18.3.3.1/18-9
0x01C	PMC0 (upper)—Performance monitor counter 0 upper	R/W	0x0000_0000	18.3.3.1/18-9
0x020	PMLCA1—Performance monitor local control register A1	R/W	0x0000_0000	18.3.2.2/18-5
0x024	PMLCB1—Performance monitor local control register B1	R/W	0x0000_0000	18.3.2.2/18-5
0x028	PMC1—Performance monitor counter 1	R/W	0x0000_0000	18.3.3.1/18-9
0x030	PMLCA2—Performance monitor local control register A2	R/W	0x0000_0000	18.3.2.2/18-5
0x034	PMLCB2—Performance monitor local control register B 2	R/W	0x0000_0000	18.3.2.2/18-5

Table 18-1. Control Register Memory Map (continued)

Performance Monitor—Block Base Address 0xE_1000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0x038	PMC2—Performance monitor counter 2	R/W	0x0000_0000	18.3.3.1/18-9
0x040	PMLCA3—Performance monitor local control register A3	R/W	0x0000_0000	18.3.2.2/18-5
0x044	PMLCB3—Performance monitor local control register B3	R/W	0x0000_0000	18.3.2.2/18-5
0x048	PMC3—Performance monitor counter 3	R/W	0x0000_0000	18.3.3.1/18-9
0x050	PMLCA4—Performance monitor local control register A4	R/W	0x0000_0000	18.3.2.2/18-5
0x054	PMLCB4—Performance monitor local control register B4	R/W	0x0000_0000	18.3.2.2/18-5
0x058	PMC4—Performance monitor counter 4	R/W	0x0000_0000	18.3.3.1/18-9
0x060	PMLCA5—Performance monitor local control register A5	R/W	0x0000_0000	18.3.2.2/18-5
0x064	PMLCB5—Performance monitor local control register B 5	R/W	0x0000_0000	18.3.2.2/18-5
0x068	PMC5—Performance monitor counter 5	R/W	0x0000_0000	18.3.3.1/18-9
0x070	PMLCA6—Performance monitor local control register A6	R/W	0x0000_0000	18.3.3.1/18-9
0x074	PMLCB6—Performance monitor local control register B6	R/W	0x0000_0000	18.3.2.2/18-5
0x078	PMC6—Performance monitor counter 6	R/W	0x0000_0000	18.3.3.1/18-9
0x080	PMLCA7—Performance monitor local control register A7	R/W	0x0000_0000	18.3.2.2/18-5
0x084	PMLCB7—Performance monitor local control register B7	R/W	0x0000_0000	18.3.2.2/18-5
0x088	PMC7—Performance monitor counter 7	R/W	0x0000_0000	18.3.3.1/18-9
0x090	PMLCA8—Performance monitor local control register A8	R/W	0x0000_0000	18.3.2.2/18-5
0x094	PMLCB8—Performance monitor local control register B8	R/W	0x0000_0000	18.3.2.2/18-5
0x098	PMC8—Performance monitor counter 8	R/W	0x0000_0000	18.3.3.1/18-9
0x0A0	PMLCA9—Performance monitor local control register A9	R/W	0x0000_0000	18.3.2.2/18-5
0x0A4	PMLCB9—Performance monitor local control register B9	R/W	0x0000_0000	18.3.2.2/18-5
0x0A8	PMC9—Performance monitor counter 9	R/W	0x0000_0000	18.3.3.1/18-9

In addition to these registers, the interrupt control provides four pairs of mask registers that can be used to monitor message, interprocessor, timer, and external interrupts. See Section 9.3.4, “Performance Monitor Mask Registers (PMMRs).”

18.3.2 Control Registers

This section describes the performance monitor control registers in detail.

18.3.2.1 Performance Monitor Global Control Register (PMGC0)

The performance monitor global control register (PMGC0), shown in Figure 18-2, is a 32-bit register used to control all PMCs.

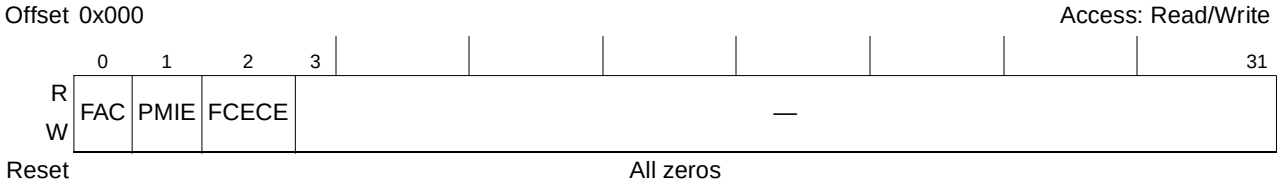


Figure 18-2. Performance Monitor Global Control Register (PMGC0)

Table 18-2 describes PMGC0 fields.

Table 18-2. PMGC0 Field Descriptions

Bits	Name	Description
0	FAC	Freeze all counters. 0 PMCs are incremented (if permitted by other PMGC0/PMLC bits). 1 PMCs are not incremented. Set by hardware when an interrupt is signalled and FCECE =1.
1	PMIE	Performance monitor interrupt enable. Interrupts are caused by PMC overflows. 0 Interrupts are disabled. 1 Interrupts are enabled and occur when an enabled condition or event occurs.
2	FCECE (DISCOUNT)	Freeze counters on enabled condition or event. An enabled condition or event is defined as: The msb = 1 in PMC*n* and PMLCA*n*[CE] = 1. The use of the trigger and freeze counter conditions depends on the enabled condition. 0 PMCs can be incremented (if permitted by other control bits). 1 PMCs can be incremented (if permitted by other control bits) only until an enabled condition or event occurs, at which time PMGC0[FAC] is set. It is up to software to clear FAC.
3–31	—	Reserved

18.3.2.2 Performance Monitor Local Control Registers (PMLCA*n*, PMLCB*n*)

The performance monitor local control registers (PMLCA*n* and PMLCB*n*) are used to control the operation of the PMCs. The performance monitor local control A and B registers are paired 32-bit control registers that are associated with an individual counter to specify how the counter is used and what event is monitored on that counter. Figure 18-3 shows the performance monitor local control A0 register (PMLCA0).

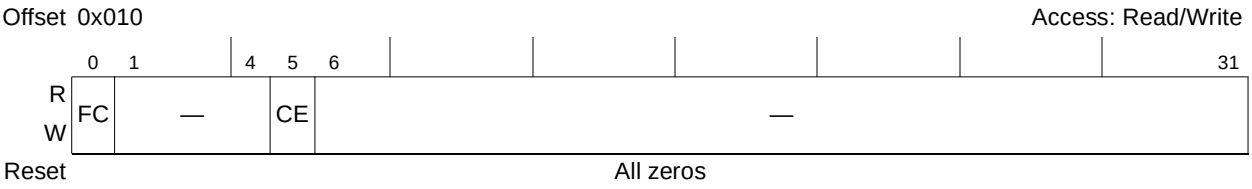


Figure 18-3. Performance Monitor Local Control Register A0 (PMLCA0)

Table 18-3 describes PMLCA0 fields.

Table 18-3. PMLCA0 Field Descriptions

Bits	Name	Description
0	FC	Freeze counter. Basic counter enable. 0 The PMCs are enabled and incremented (if permitted by other SPM control bits). 1 The PMCs are disabled–they do not increment.
1–4	—	Reserved
5	CE	Condition enable. Controls counter overflow condition. Should be cleared when PMC0 is used as a trigger or is selected for chaining. 0 Overflow conditions for PMC0 cannot occur (PMC0 cannot cause interrupts or freeze counters) 1 Overflow conditions occur when PMC0[msb] is set.
6–31	—	Reserved

Figure 18-4 shows the performance monitor local control registers A1–A8.

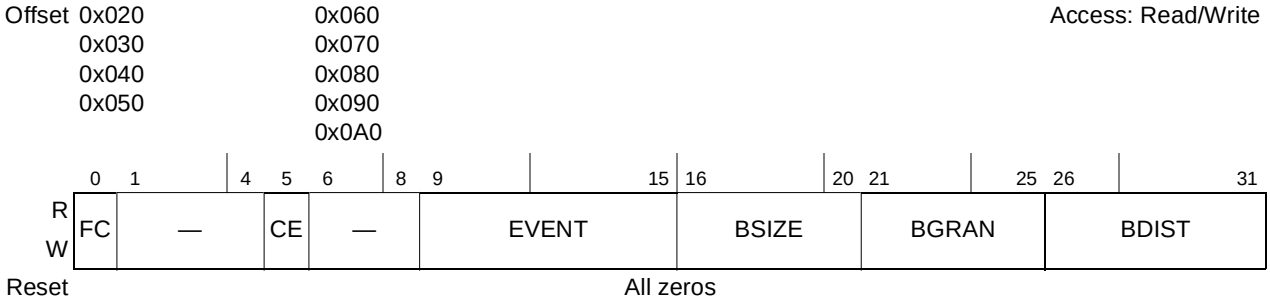


Figure 18-4. Performance Monitor Local Control A Registers (PMLCA1–PMLCA9)

Table 18-4 describes PMLCA*n* fields.

Table 18-4. PMLCA1–PMLCA9 Field Descriptions

Bits	Name	Description
0	FC	Freeze counter 0 The PMCs are incremented (if permitted by other PMC control bits). 1 The PMCs are not incremented (if permitted by other PMC control bits).
1–4	—	Reserved
5	CE	Condition enable 0 Overflow conditions for PMC*n* cannot occur (PMC*n* cannot cause interrupts or freeze counters). Should be cleared when PMC*n* is used as a trigger or is selected for chaining. 1 Overflow conditions occur when PMC*n*[msb] is set.
6–8	—	Reserved
9–15	EVENT	Event selector. Up to 128 events selectable. See Table for definition of events.
16–20	BSIZE	Burst size. Fewest event occurrences that constitute a burst, that is, a rapid sequence of events followed by a relatively long pause. A value less than two implies regular event counting. Any non-threshold, regular event may be counted in a bursty fashion. See Section 18.4.6, “Burstiness Counting,” for more information.

Table 18-4. PMLCA1–PMLCA9 Field Descriptions (continued)

Bits	Name	Description
21–25	BGRAN	Burst granularity. The maximum number of clock cycles between events that are considered part of a single burst. See Section 18.4.6, “Burstiness Counting.”
26–31	BDIST	Burst distance (used with TBMULT). The number of clock cycles between bursts. Must be set to a value greater than BSIZE for proper burstiness counting behavior. 00_0000 Regular counting

Figure 18-5 shows the performance monitor local control B0 register (PMLCB0).

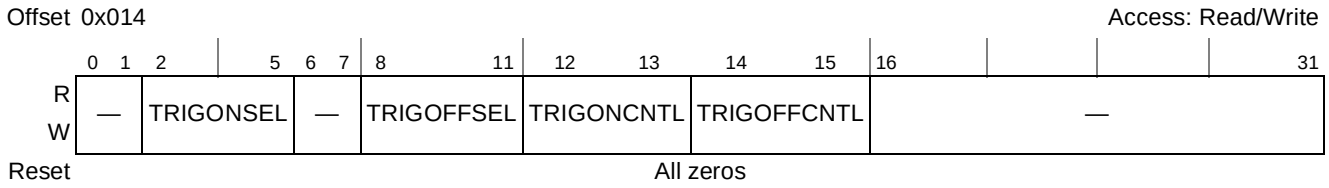


Figure 18-5. Performance Monitor Local Control Register B0 (PMLCB0)

Table 18-5 describes PMLCB0 fields.

Table 18-5. PMLCB0 Field Descriptions

Bits	Name	Description
0–1	—	Reserved
2–5	TRIGONSEL	Trigger-on select. The number of the counter that starts event counting. When the specified counter’s TRIGONCNTL event overflows, the current counter begins counting. No triggering occurs if the value is self-referential, that is, when set to the current counter number.
6–7	—	Reserved
8–11	TRIGOFFSEL	Trigger-off select. The number of the counter that stops event counting. When the specified counter’s TRIGONCNTL event overflows, the current counter stops counting. No triggering occurs if the value is self-referential, that is, when set to the current counter number.
12–13	TRIGONCNTL	Trigger-on control. Indicates the condition under which triggering to start counting occurs 00 Trigger off (no triggering to start) 01 Trigger on change 10 Trigger on overflow 11 Reserved
14–15	TRIGOFFCNTL	Trigger-off control. Indicates the condition under which triggering to stop occurs 00 Trigger off (no triggering to stop) 01 Trigger on change 10 Trigger on overflow 11 Reserved
16–31	—	Reserved

Figure 18-6 shows performance monitor local control registers B1–B9.

Offset 0x024 0x034 0x044 0x054 0x064 0x074 0x084 0x094 0x0A4 Access: Read/Write

	0 1	2 5	6 7	8 11	12 13	14 15	16 20	21 23	24 25 26	31
R W	—	TRIGONSEL	—	TRIGOFFSEL	TRIGONCNTL	TRIGOFFCNTL	—	TBMULT	—	THRESHOLD
Reset	All zeros									

Figure 18-6. Performance Monitor Local Control Register B (PMLCB1–PMLCB9)

Table 18-5 describes PMLCB*n* fields.

Table 18-6. PMLCB*n* Field Descriptions

Bits	Name	Description
0–1	—	Reserved
2–5	TRIGONSEL	Trigger-on select. Set this field equal to the number of the counter that should trigger event counting to start. When the specified counter's TRIGONCNTL event overflows, the current counter begins counting. No triggering occurs when TRIGONSEL = current counter.
6–7	—	Reserved
8–11	TRIGOFFSEL	Trigger-off select. Set this field equal to the number of the counter that should trigger event counting to stop. When the specified counter's TRIGONCNTL event overflows, the current counter stops counting. No triggering occurs when TRIGOFFSEL = current counter.
12–13	TRIGONCNTL	Trigger-on control. Indicates the condition under which triggering to start counting occurs 00 Trigger off (no triggering to start) 01 Trigger on change 10 Trigger on overflow 11 Reserved
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

Table 18-6. PMLCB*n* Field Descriptions (continued)

Bits	Name	Description
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 BD read latencies for which the duration exceeds the threshold, software can obtain the distribution of BD read latencies for a given program by monitoring the program repeatedly using a different threshold value each time.

18.3.3 Counter Registers

This section describes the PMCs in detail.

NOTE

Because accessing a PMC manually has priority over incrementing it due to event counting, reading or writing a PMC while it is counting may affect the count. Likewise, accessing a performance monitor control register while its target counter is counting may also affect the count.

18.3.3.1 Performance Monitor Counters (PMC0–PMC9)

PMC0–PMC9 are used to count events selected by the performance monitor local control registers. PMC0, shown in Figure 18-7, is associated with two 32-bit registers that form a 64-bit counter designated to count clock cycles. PMC0 upper represents the upper 32 bits of counter 0, and PMC0 lower represents the lower 32 bits.

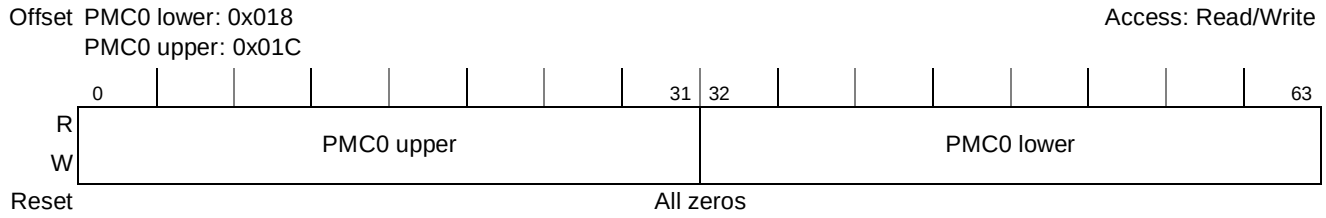


Figure 18-7. Performance Monitor Counter Register 0 (PMC0)

Table 18-7 describes PMC0 fields.

Table 18-7. PMC0 Field Descriptions

Bits	Name	Description
0–63	PMC0	Event count. Counts only clock cycles

PMC1–PMC8, shown in Figure 18-8, are 32-bit counters that can monitor 64 unique events in addition to the 64 reference events that can be counted on all of these registers.

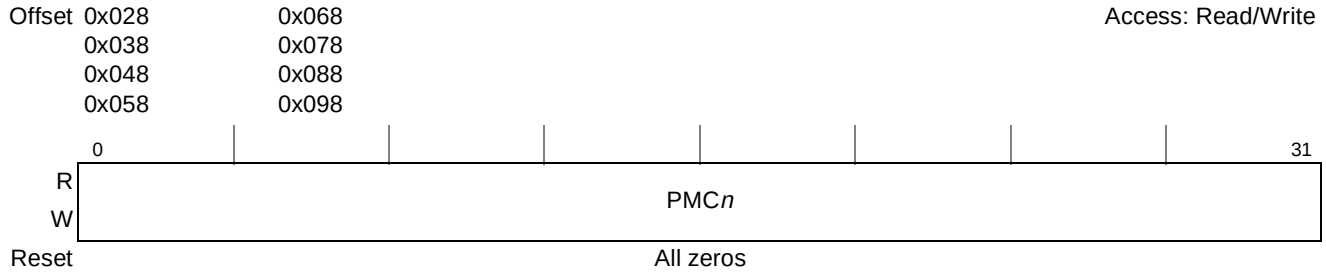


Figure 18-8. Performance Monitor Counter Register (PMC1–PMC8)

Table 18-8 describes PMC*n* fields.

Table 18-8. PMC[1–9] Field Descriptions

Bits	Name	Description
0–31	PMC*n*	Event count. An overflow is indicated when msb = 1. Manually setting the msb can cause an immediate interrupt.

18.4 Functional Description

This section describes the use of some features of the performance monitor.

18.4.1 Performance Monitor Interrupt

PMCs can generate an interrupt on an overflow when the msb of a counter changes from 0 to 1. For the interrupt to be signalled, the condition enable bit (PMLCA*n*[CE]) and performance monitor interrupt enable bit (PMGC0[PMIE]) must be set. When an interrupt is signalled and the freeze-counters-on-enabled-condition-or-event bit (PMGC0[FCECE]) is set, PMGC0[FAC] is set by hardware and all of the registers are frozen. Software can clear the interrupt condition by resetting the performance monitor and clearing the most significant bit of the counter that generated the overflow.

18.4.2 Event Counting

Using the control registers described in Section 18.3.2, “Control Registers,” the nine PMCs can count the occurrences of specific events. The 64-bit PMC0 is designated to count only clock cycles. However, to provide flexibility, a total of 64 reference events can be counted on any of the 32-bit PMCs (PMC1–PMC9). Additionally, up to 64 unique events can be counted on each 32-bit counter.

The performance monitor must be reset before event counting sequences. The performance monitor can be reset by first freezing one or more counters and then clearing the freeze condition to allow the counters to count according to the settings in the performance monitor registers.þCounters can be frozen individually by setting PMLCAn[FC] bits, or simultaneously by setting PMGC0[FAC]. Simply clearing these freeze bits will then allow the performance monitor to begin counting based on the register settings.þ

Note that using PMLCAn[FC] to reset the performance monitor resets only the specified counter. Performance monitor registers can be configured through reads or writes while the counters are frozen as long as freeze bits are not cleared by the register accesses.

18.4.3 Threshold Events

The threshold feature allows characterization of events that can take a variable number of clock cycles to occur. Threshold events are counted only if the latency is greater than the threshold value specified in PMLCB*n*[THRESHOLD].

For duration threshold event sequences, the PMC increments only when the duration of the event is equal to or greater than the threshold value. The threshold value is scaled by a multiple specified in PMLCB*n*[TBMULT].

A threshold event requires two signals: The first indicates when a threshold event sequence begins, and the second indicates when it ends. An internal counter determines when the threshold count is exceeded and when the PMC can increment. This internal counter decrements during a threshold event sequence until it reaches the value of one. A new sequence cannot begin until the current one completes. Additional threshold start signals are ignored during a sequence until a threshold stop signal occurs. If both a start and stop signal are asserted during the same cycle in a current sequence, the stop terminates the current sequence and the start signals the beginning of a new one. However, if both signals are asserted during the same cycle while not in a current event sequence, both signals are ignored. Figure 18-9 is a timing diagram for duration threshold event counting.

An illegal condition exists if the threshold value obtained from PMLCB*n*[THRESHOLD] and PMLCB*n*[TBMULT] is less than two. Under these conditions the intent of threshold counting is ambiguous.

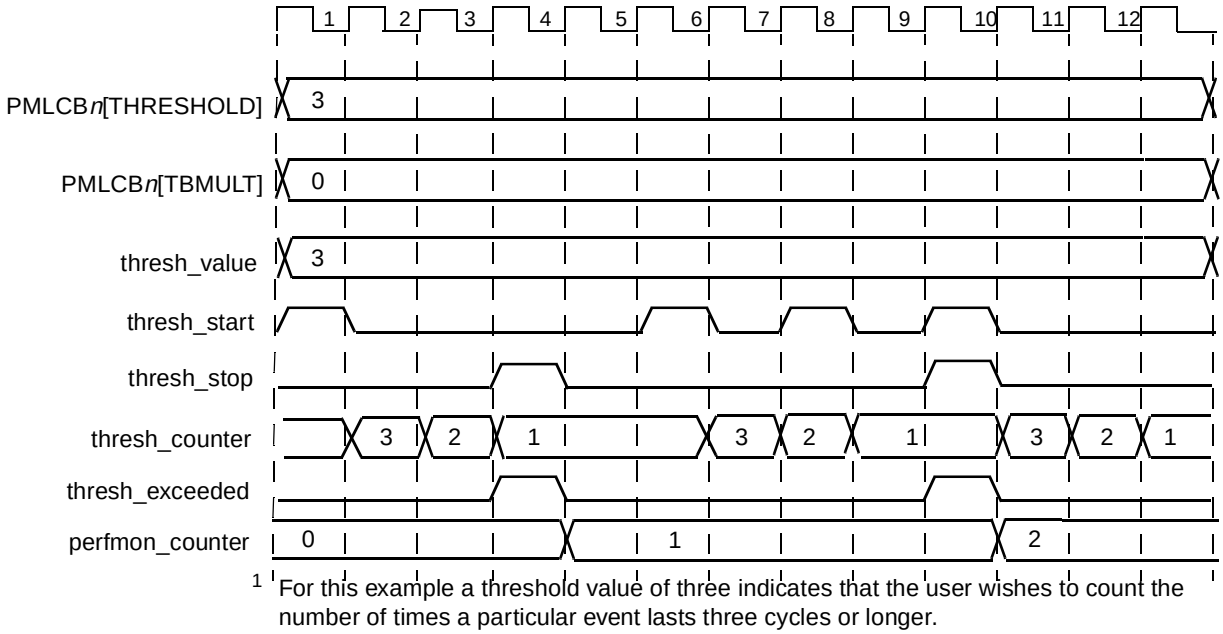


Figure 18-9. Duration Threshold Event Sequence Timing Diagram

The second type of threshold event is the quantity threshold event. For these types of threshold event sequences the performance monitor counter is only incremented when the specified threshold event exceeds the threshold value. These events do not use the multiplier register field (PMLCB*n*[TBMULT]) like the duration threshold events. This type of threshold event is generally used to monitor the usage of buffers and queues. For example, the usage of a specific queue could be characterized by measuring the amount of time the queue is completely full or partially full. For this example the threshold field would be used to specify how many entries are required to be valid in the queue for that event to be counted.

18.4.4 Chaining

By configuring one counter to increment each time another counter overflows, several counters can be chained together to provide event counts larger than 32 bits. Each counter in a chain adds 32 bits to the maximum count. The register chaining sequence is not arbitrary and is specified indirectly by selecting the register overflow event to be counted. Selecting an event has the effect of selecting a source register because all available chaining events, as shown in Table , are dedicated to specific registers.

Note that the chaining overflow event occurs when the counter reaches its maximum value and wraps, not when the register's msb is set. For this overflow to occur, PMLCA*n*[CE] should be cleared to avoid signalling an interrupt when the counter's most significant bit is set. Note that several cycles may be required for the chained counters to reflect the true count because of the internal delay between when an overflow occurs and a counter increments.

18.4.5 Triggering

Triggering allows one counter to start or stop counting on the change of another counter or on the overflow of another counter. More specifically, if PMC1 is set to start or stop counting as a result of a change or overflow in counter PMC2, then counter PMC2 must be identified in the local control register of counter PMC1. This is done by appropriately setting the trigger-on select bit or trigger-off select bit (PMLCB1[TRIGOFFSEL] or PMLCB1[TRIGONSEL]). Additionally, the condition that triggers the counter must be selected by configuring the corresponding control bits (PMLCB1[TRIGONCNTL] or PMLCB1[TRIGOFFCNTL]). Assuming the counter is enabled by other control register settings, the counter increments (or freezes) when its specified event occurs after the trigger-on (or off) condition occurs.

When trigger on and trigger off are both selected, the trigger-off condition is ignored until the trigger-on condition has occurred. Furthermore, when a trigger-off condition occurs, the counter state is preserved; it is not restarted by subsequent trigger-on conditions.

Triggering is disabled when the counter's trigger-select bits specify itself as the trigger source. Similarly, triggering is disabled when the trigger control bits are cleared.

18.4.6 Burstiness Counting

The burstiness counting feature makes it easier to characterize events that occur in rapid succession followed by a relatively long pause. As shown in Table 18-9, event bursts are defined by size, granularity, and distance.

Table 18-9. Burst Definition

Parameter	Description	Register Field
Size	The minimum number of events constituting a burst	PMLCA*n* [BSIZE]
Granularity	The maximum time between individual events counted as members of the same burst	PMLCA*n* [BGRAN]
Distance	The minimum time between bursts	PMLCA*n* [BDIST] x PMLCB*n*[TBMULT]

Figure 18-10 shows the relationships between size, granularity, and distance. Burstiness counting can be performed for all events except threshold events.

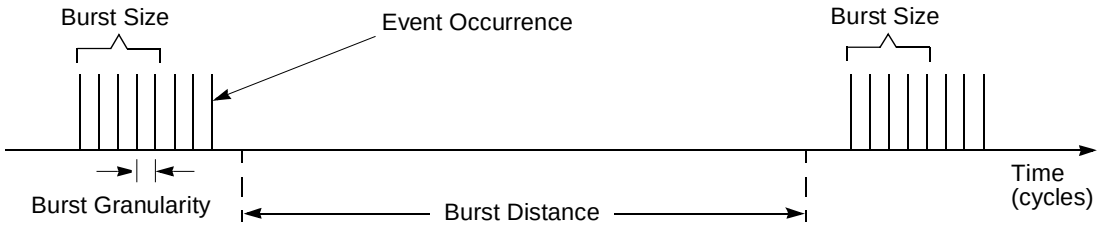


Figure 18-10. Burst Size, Distance, Granularity, and Burstiness Counting

The burstiness size field (PMLCA*n*[BSIZE]) specifies the minimum number of event occurrences that constitute a burst. A burst is identified when the number of event occurrences equals or exceeds PMLCA*n*[BSIZE]. Furthermore, these individual event occurrences must be separated by no more clock cycles than the value in the burstiness granularity field (PMLCA*n*[BGRAN]). Note that, although a burst is identified when the minimum number of events occurs, it is not counted until the burst sequence has ended. A burst sequence ends when the specified burstiness granularity is exceeded, at which point the last valid event has occurred for that sequence.

PMLCA*n*[BGRAN] specifies the maximum number of cycles between individual events for them to qualify as members of the same burst sequence.

The burstiness distance field (PMLCA*n*[BDIST]) and threshold/burstiness multiplier field (PMLCB*n*[TBMULT]) specify the acceptable number of cycles between the end of a burst sequence and the beginning of a new sequence for a group of event occurrences to be counted as an individual burst. The product of the burstiness distance field and the threshold/burstiness multiplier field determine the burstiness distance value used to determine when another burst sequence can begin. Note that the burst distance count begins when a new burst sequence ends and the PMC is incremented. No new burst sequence may begin until the burst distance count has reached zero. After the burst distance count reaches zero, it holds the zero value indicating that a new burst sequence can be counted. The burst distance count begins again when a new burst sequence is identified and counted.

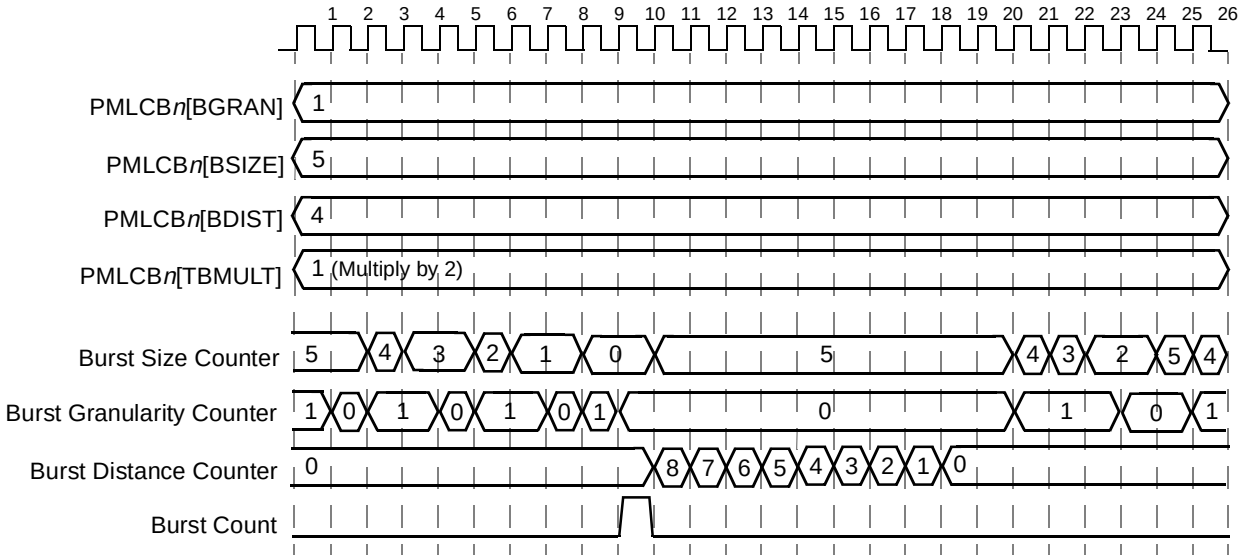
Burstiness counting is disabled when the definition of a burst is ambiguous, that is, when the burst size field is less than two, or the burst distance is zero. When burstiness counting is disabled, regular counting is allowed.

Figure 18-10 shows that the burst distance is measured from the end of one burst sequence and that a new burst sequence may not begin until the burst distance count expires.

Three internal counters track the different values required for burstiness counting.

- Burstiness size is monitored by a counter. It is loaded with the value specified in the local control register when the burst granularity counter and the burst distance counters reach zero, and no new event is occurring. It always decrements when the following conditions occur: its value is not already zero, an event occurs, and the burst distance count equals zero.
- Burstiness granularity is monitored by a counter that is loaded with the specified value in the local control register on the rising edge of an event occurrence whenever the burst distance count equals zero. The granularity counter is decremented (if it has not already reached zero) when an event is not occurring and burst distance count equals zero.
- Burstiness distance is measured by a counter that is loaded with the product of PMLCB*n*[BDIST] and PMLCB*n*[TBMULT] when a burst sequence has been identified and counted. This counter is decremented when burstiness counting is enabled (and the counter has not already reached zero).

A burst is counted at the end of a burst sequence when the three burst parameter counters are all equal to zero. Figure 18-11 shows a burstiness counting example.



1 For this example: count bursts of 5 event occurrences, burst granularity of 1, and acceptable distance between bursts of 8 (product of TBMULT and BDIST).

Figure 18-11. Burstiness Counting Timing Diagram

18.4.7 Performance Monitor Events

Table 18-10 lists performance monitor events specified in PMLCA1–PMLC9.

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[0–9]:#—Counter-specific events. C8 indicates an event assigned to PMC8. Thus C8:62 means PMC8 is assigned event 62 (PIC interrupt wait cycles).

Note that with counter-specific events, an offset of 64 must be used when programming the field, because counter-specific events occupy the bottom 64 values of the 7-bit event field where events are numbered. For example, to specify counter-specific event 0, the event field must be programmed to 64.

Counter events not specified in Table 18-10 are reserved.

Table 18-10. Performance Monitor Events Assignment

Event	Event Assignment	Description
General Events		
Nothing	Ref:0	Register counter holds current value
System Cycles	C0	Counts every platform cycle
Chaining Events		
PMC0 carry-out	Ref:1	Counts the number of times PMC0[0] transitioned from 1 to 0.
PMC1 carry-out	Ref:2	Counts the number of times PMC1[0] transitioned from 1 to 0. This event is reserved for PMC1.
PMC2 carry-out	Ref:3	Counts the number of times PMC2[0] transitioned from 1 to 0. This event is reserved for PMC2.
PMC3 carry-out	Ref:4	Counts the number of times PMC3[0] transitioned from 1 to 0. This event is reserved for PMC3.
PMC4 carry-out	Ref:5	Counts the number of times PMC4[0] transitioned from 1 to 0. This event is reserved for PMC4.
PMC5 carry-out	Ref:6	Counts the number of times PMC5[0] transitioned from 1 to 0. This event is reserved for PMC5.
PMC6 carry-out	Ref:7	Counts the number of times PMC6[0] transitioned from 1 to 0. This event is reserved for PMC6.
PMC7 carry-out	Ref:8	Counts the number of times PMC7[0] transitioned from 1 to 0. This event is reserved for PMC7.
PMC8 carry-out	Ref:9	Counts the number of times PMC8[0] transitioned from 1 to 0. This event is reserved for PMC8.
PMC9 carry-out	Ref:10	Counts the number of times PMC9[0] transitioned from 1 to 0. This event is reserved for PMC9.
DDR Memory Controller 1 Events		
Total number of cycles a read or write is returning/sending data from/to DRAM.	Ref:11	This event will assert once for each beat of data that is transferred from or to the DRAM.
Number of pipelined reads that missed in the Row Open Table.	C1:57	This event will assert when a read that missed in the Row Open Table is issued while there is still a previous outstanding read.
Number of pipelined reads or writes that missed in the Row Open Table.	C2:0	This event will assert when a read or write that missed in the Row Open Table is issued while there is still a previous outstanding read or write.

Table 18-10. Performance Monitor Events Assignment (continued)

Event	Event Assignment	Description
Number of non-pipelined reads that missed in the Row Open Table.	C3:60	This event will assert when a read that missed in the Row Open Table is issued while there are not any outstanding reads.
Number of non-pipelined reads or writes that missed in the Row Open Table.	C4:0	This event will assert when a read or write that missed in the Row Open Table is issued while there are not any outstanding reads or writes.
Number of pipelined reads that hit in the Row Open Table.	C5:56	This event will assert when a read that hit in the Row Open Table is issued while there is still a previous outstanding read.
Number of pipelined reads or writes that hit in the Row Open Table.	C6:0	This event will assert when a read or write that hit in the Row Open Table is issued while there is still a previous outstanding read or write.
Number of non-pipelined reads that hit in the Row Open Table.	C7:57	This event will assert when a read that hit in the Row Open Table is issued while there are not any outstanding reads.
Number of non-pipelined reads or writes that hit in the Row Open Table.	C8:0	This event will assert when a read or write that hit in the Row Open Table is issued while there are not any outstanding reads or writes.
Number of forced page closings not caused by a refresh.	C1:0	This event will assert if a precharge has been issued to the DRAM for any reason other than a refresh. The three possible situations are as follows: 1) A new transaction needs to be issued to an already active bank and sub-bank, which has a different row open; 2) A new transaction needs to be issued, but the Row Open Table is full and there is not a bank/sub-bank match between the current transaction and the Row Open Table; 3) The 'BSTOPRE' interval has expired for an already open row.
Total number of Row Open Table misses.	C2:1	This event will assert for each transaction that misses in the Row Open Table.
Total number of Row Open Table hits.	C3:0	This event will assert for each transaction that hits in the Row Open Table.
Total number of force page closings.	C4:1	This event will assert for all forced page closings plus page closings due to refreshes.
Number of read-modify-write transactions due to ECC.	C5:0	If ECC is enabled and a transaction requires byte enables, a read-modify-write sequence will be issued on the DRAM interface.
Number of forced page closings due to collision with bank and sub-bank.	Ref:12	This event will assert if a new transaction needs to be issued to an already active bank and sub-bank which has a different row open.
Total number of reads or writes from core.	Ref:13	Total number of reads or writes from Core 0. Memory select errors must be enabled (ERR_DISABLE[MSED] = 0) for accurate counting of this event.
Total number of reads or writes from Ethernet	C3:1	Total number of reads or writes from Ethernets 1-4

Table 18-10. Performance Monitor Events Assignment (continued)

Event	Event Assignment	Description
Total number of reads or writes from high speed interfaces	C4:3	Total number of reads or writes from PEX/SRIO
Total number of reads or writes from DMA	C5:2	Total number of reads or writes from DMA
Total number of Row Open Table hits for reads or writes from core	Ref:14	Total number of Row Open Table hits for reads or writes from core
Total number of Row Open Table hits for reads or writes from Ethernet	C6:1	Total number of Row Open Table hits for reads or writes from Ethernet 1-4
Total number of Row Open Table hits for reads or writes from PCI	C6:2	Total number of Row Open Table hits for reads or writes from PCI 1-2
Total number of Row Open Table hits for reads or writes from high speed interfaces	C7:1	Total number of Row Open Table hits for reads or writes from PEX/SRIO
Total number of Row Open Table hits for reads or writes from DMA	C8:2	Total number of Row Open Table hits for reads or writes from DMA
Total number of cycles a read is returning data from DRAM.	Ref:19	This event will assert once for each beat of data that is returned to the memory controller on the DRAM interface
DDR Memory Controller 2 Events		
Total number of cycles a read or write is returning/sending data from/to DRAM.	Ref:22	This event will assert once for each beat of data that is transferred from or to the DRAM.
Number of pipelined reads that missed in the Row Open Table.	C2:2	This event will assert when a read that missed in the Row Open Table is issued while there is still a previous outstanding read.
Number of pipelined reads or writes that missed in the Row Open Table.	C3:62	This event will assert when a read or write that missed in the Row Open Table is issued while there is still a previous outstanding read or write.
Number of non-pipelined reads that missed in the Row Open Table.	C4:37	This event will assert when a read that missed in the Row Open Table is issued while there are not any outstanding reads.
Number of non-pipelined reads or writes that missed in the Row Open Table.	C5:40	This event will assert when a read or write that missed in the Row Open Table is issued while there are not any outstanding reads or writes.
Number of pipelined reads that hit in the Row Open Table.	C6:11	This event will assert when a read that hit in the Row Open Table is issued while there is still a previous outstanding read.

Table 18-10. Performance Monitor Events Assignment (continued)

Event	Event Assignment	Description
Number of pipelined reads or writes that hit in the Row Open Table.	C7:53	This event will assert when a read or write that hit in the Row Open Table is issued while there is still a previous outstanding read or write.
Number of non-pipelined reads that hit in the Row Open Table.	C8:27	This event will assert when a read that hit in the Row Open Table is issued while there are not any outstanding reads.
Number of non-pipelined reads or writes that hit in the Row Open Table.	C1:9	This event will assert when a read or write that hit in the Row Open Table is issued while there are not any outstanding reads or writes.
Number of forced page closings not caused by a refresh.	C2:13	This event will assert if a precharge has been issued to the DRAM for any reason other than a refresh. The three possible situations are: 1) A new transaction needs to be issued to an already active bank and sub-bank, which has a different row open; 2) A new transaction needs to be issued, but the Row Open Table is full and there is not a bank/sub-bank match between the current transaction and the Row Open Table; 3) The 'BSTOPRE' interval has expired for an already open row.
Total number of Row Open Table misses.	C3:13	This event will assert for each transaction that misses in the Row Open Table.
Total number of Row Open Table hits.	C4:38	This event will assert for each transaction that hits in the Row Open Table.
Total number of force page closings.	C5:8	This event will assert for all forced page closings plus page closings due to refreshes.
Number of read-modify-write transactions due to ECC.	C3:14	If ECC is enabled and a transaction requires byte enables, a read-modify-write sequence will be issued on the DRAM interface.
Number of forced page closings due to collision with bank and sub-bank.	Ref:23	This event will assert if a new transaction needs to be issued to an already active bank and sub-bank which has a different row open.
Total number of reads or writes from core.	Ref:24	Total number of reads or writes from Core 0. Memory select errors must be enabled (ERR_DISABLE[MSED] = 0) for accurate counting of this event.
Total number of reads or writes from Ethernet	C3:15	Total number of reads or writes from Ethernets 1-4
Total number of reads or writes from high speed interfaces	C5:15	Total number of reads or writes from PEX/SRIO
Total number of reads or writes from DMA	C6:37	Total number of reads or writes from DMA
Total number of Row Open Table hits for reads or writes from core	Ref:25	Total number of Row Open Table hits for reads or writes from core
Total number of Row Open Table hits for reads or writes from Ethernet	C6:32	Total number of Row Open Table hits for reads or writes from Ethernet 1-4

Table 18-10. Performance Monitor Events Assignment (continued)

Event	Event Assignment	Description
Total number of Row Open Table hits for reads or writes from PCI	C7:27	Total number of Row Open Table hits for reads or writes from PCI 1-2
Total number of Row Open Table hits for reads or writes from high speed interfaces	C8:29	Total number of Row Open Table hits for reads or writes from PEX/SRIO
Total number of Row Open Table hits for reads or writes from DMA	C7:30	Total number of Row Open Table hits for reads or writes from DMA
Total number of cycles a read is returning data from DRAM.	Ref:61	This event will assert once for each beat of data that is returned to the memory controller on the DRAM interface
DMA Events		
Channel 0 read request	C1:2	DMA channel 0 read request active in the system
Channel 1 read request	C2:5	DMA channel 1 read request active in the system
Channel 2 read request	C3:4	DMA channel 2 read request active in the system
Channel 3 read request	C4:6	DMA channel 3 read request active in the system
Channel 0 write request	C1:3	DMA channel 0 write request active in the system
Channel 1 write request	C2:6	DMA channel 1 write request active in the system
Channel 2 write request	C3:5	DMA channel 2 write request active in the system
Channel 3 write request	C4:7	DMA channel 3 write request active in the system
Channel 0 descriptor request	C5:41	DMA channel 0 descriptor request active in the system
Channel 1 descriptor request	C6:44	DMA channel 1 descriptor request active in the system
Channel 2 descriptor request	C7:41	DMA channel 2 descriptor request active in the system
Channel 3 descriptor request	C8:41	DMA channel 3 descriptor request active in the system
Channel 0 read DW or less	C1:4 and C5:53	DMA channel 0 read doubleword valid
Channel 1 read DW or less	C2:7 and C6:58	DMA channel 1 read doubleword valid
Channel 2 read DW or less	C3:6 and C7:54	DMA channel 2 read doubleword valid
Channel 3 read DW or less	C4:8 and C8:52	DMA channel 3 read doubleword valid
Channel 0 write DW or less	C1:5	DMA channel 0 write doubleword valid
Channel 1 write DW or less	C2:8	DMA channel 1 write doubleword valid
Channel 2 write DW or less	C3:7	DMA channel 2 write doubleword valid
Channel 3 write DW or less	C4:9	DMA channel 3 write doubleword valid

Table 18-10. Performance Monitor Events Assignment (continued)

Event	Event Assignment	Description
DUART Events		
UART0 Baud Rate	C1:63	uart0 real baud rate
UART1 Baud Rate	C5:63	uart1 real baud rate
Debug Events		
Ext. Event	C3:61	Number of cycles trig_in pin is asserted
Number of watchpoint monitor hits	C2:61	Number of watchpoint monitor hits
Number of trace buffer hits	C1:58	Number of trace buffer hits
MCM Events		
MCM request wait core 0	C8:13	Asserted for every cycle Core 0 request occurs
MCM request wait core 1	C9:14	Asserted for every cycle Core 1 request occurs
MCM dispatch	Ref:15	MCM dispatch—includes addr only's from Core. Note: all MCM dispatch events are for committed dispatches
MCM dispatch from core 0	C1:16	MCM dispatch from core 0—includes addr only's from Core
MCM dispatch from TSEC1	C3:19	MCM dispatch from TSEC1
MCM dispatch from TSEC2	C4:21	MCM dispatch from TSEC2
MCM dispatch from TSEC3	C8:16	MCM dispatch from TSEC3
MCM dispatch from TSEC4	C9:16	MCM dispatch from TSEC4
MCM dispatch from RIO1	C5:17	MCM dispatch from RIO
MCM dispatch from PEX1	C7:16	MCM dispatch from PEX 1
MCM dispatch from PEX2	C8:17	MCM dispatch from PEX 2
MCM dispatch from DMA	C7:14	MCM dispatch from DMA
MCM dispatch from core 1	C9:17	MCM dispatch from core 1
MCM dispatch from RIO message unit, door bell, or port write	C9:18	MCM dispatch from RIO message unit, door bell, or port write
MCM dispatch from other	C8:14	MCM dispatch from other
MCM transaction interleaving	C1:15	Non-address only transactions whose target address hit mem interleave window
MCM dispatch write	C1:17	MCM dispatch write
MCM dispatch read	C4:23	MCM dispatch read
MCM dispatch read atomic clr, set, dec, inc	C9:23	MCM dispatch read clear atomics
MCM Core 0 read from DDR 0	Ref:17	Core 0 read to DDR port 1

Table 18-10. Performance Monitor Events Assignment (continued)

Event	Event Assignment	Description
MCM Core 1 read from DDR 0	C2:24	Core 1 read to DDR port 1
MCM Core 0 read from DDR 1	C6:23	Core 0 read to DDR port 2
MCM Core 1 read from DDR 1	C4:30	Core 1 read to DDR port 2
TSEC 1 Events		
DMA write data beats	C3:45	DMA write data beats
DMA read data beats	C4:46	DMA read data beats
DMA Write Request	C5:42	DMA Write Request
DMA Read Request	C6:45	DMA Read Request
Number of dropped frames	C9:24	Number of dropped frames
TxBD read lifetime (Duration Threshold)	Ref:34	TxBD read lifetime
RxBD read lifetime (Duration Threshold)	Ref:35	RxBD read lifetime
TxBD write lifetime (Duration Threshold)	Ref:36	TxBD write lifetime
RxBD write lifetime (Duration Threshold)	Ref:37	RxBD write lifetime
Read data lifetime (Duration Threshold)	Ref:38	Read data lifetime
Rx IP packets checked for checksum	C9:28	Rx IP packets checked for checksum
TX IP packet with checksum	C1:41	TX IP packet with checksum
TX TCP/UDP packet with checksum	C2:49	TX TCP/UDP packet with checksum
TCP/UDP packets checked for c.s.	C3:50	TCP/UDP packets checked for c.s.
IP or TCP/UDP Rx checksum error	C4:51	IP or TCP/UDP Rx checksum error
Number of rejected frames by filer	C5:47	Number of rejected frames by filer
Number of rejected frames due to filer error	C6:50	Number of rejected frames due to filer error
Number of cycles Rx FIFO > 1/4 full	C5:46	Number of cycles Rx FIFO > 1/4 full

Table 18-10. Performance Monitor Events Assignment (continued)

Event	Event Assignment	Description
Number of cycles Rx FIFO > 1/2 full	C6:49	Number of cycles Rx FIFO > 1/2 full
Number of cycles Rx FIFO > 3/4 full	C7:46	Number of cycles Rx FIFO > 3/4 full
Number of cycles Rx FIFO = full	C8:46	Number of cycles Rx FIFO = full
TSEC 2 Events		
DMA write data beats	C5:43	DMA write data beats
DMA read data beats	C6:46	DMA read data beats
DMA Write Request	C7:43	DMA Write Request
DMA Read Request	C8:43	DMA Read Request
Number of dropped frames	C2:46	Number of dropped frames
TxBD read lifetime (Duration Threshold)	Ref:39	TxBD read lifetime
RxBD read lifetime (Duration Threshold)	Ref:40	RxBD read lifetime
TxBD write lifetime (Duration Threshold)	Ref:41	TxBD write lifetime
RxBD write lifetime (Duration Threshold)	Ref:42	RxBD write lifetime
Read data lifetime (Duration Threshold)	Ref:43	Read data lifetime
Rx IP packets checked for checksum	C3:51	Rx IP packets checked for checksum
TX IP packet with checksum	C4:52	TX IP packet with checksum
TX TCP/UDP packet with checksum	C5:37	TX TCP/UDP packet with checksum
TCP/UDP packets checked for c.s.	C6:51	TCP/UDP packets checked for c.s.
IP or TCP/UDP Rx checksum error	C7:47	IP or TCP/UDP Rx checksum error
Number of rejected frames by filer	C8:47	Number of rejected frames by filer
Number of rejected frames due to filer error	C9:29	Number of rejected frames due to filer error
Number of cycles Rx FIFO > 1/4 full	C7:49	Number of cycles Rx FIFO > 1/4 full

Table 18-10. Performance Monitor Events Assignment (continued)

Event	Event Assignment	Description
Number of cycles Rx FIFO > 1/2 full	C8:49	Number of cycles Rx FIFO > 1/2 full
Number of cycles Rx FIFO > 3/4 full	C9:32	Number of cycles Rx FIFO > 3/4 full
Number of cycles Rx FIFO = full	C1:44	Number of cycles Rx FIFO = full
TSEC 3 Events		
DMA write data beats	C7:44	DMA write data beats
DMA read data beats	C8:44	DMA read data beats
DMA Write Request	C9:26	DMA Write Request
DMA Read Request	C1:39	DMA Read Request
Number of dropped frames	C4:48	Number of dropped frames
TxBD read lifetime (Duration Threshold)	Ref:44	TxBD read lifetime
RxBD read lifetime (Duration Threshold)	Ref:45	RxBD read lifetime
TxBD write lifetime (Duration Threshold)	Ref:46	TxBD write lifetime
RxBD write lifetime (Duration Threshold)	Ref:47	RxBD write lifetime
Read data lifetime (Duration Threshold)	Ref:48	Read data lifetime
Rx IP packets checked for checksum	C6:52	Rx IP packets checked for checksum
TX IP packet with checksum	C7:48	TX IP packet with checksum
TX TCP/UDP packet with checksum	C8:48	TX TCP/UDP packet with checksum
TCP/UDP packets checked for c.s.	C9:30	TCP/UDP packets checked for c.s.
IP or TCP/UDP Rx checksum error	C1:42	IP or TCP/UDP Rx checksum error
Number of rejected frames by filer	C2:50	Number of rejected frames by filer
Number of rejected frames due to filer error	C3:52	Number of rejected frames due to filer error
Number of cycles Rx FIFO > 1/4 full	C9:33	Number of cycles Rx FIFO > 1/4 full

Table 18-10. Performance Monitor Events Assignment (continued)

Event	Event Assignment	Description
Number of cycles Rx FIFO > 1/2 full	C1:45	Number of cycles Rx FIFO > 1/2 full
Number of cycles Rx FIFO > 3/4 full	C2:52	Number of cycles Rx FIFO > 3/4 full
Number of cycles Rx FIFO = full	C3:54	Number of cycles Rx FIFO = full
TSEC 4 Events		
DMA write data beats	C9:27	DMA write data beats
DMA read data beats	C1:40	DMA read data beats
DMA Write Request	C2:48	DMA Write Request
DMA Read Request	C3:48	DMA Read Request
Number of dropped frames	C6:40	Number of dropped frames
TxBD read lifetime (Duration Threshold)	Ref:49	TxBD read lifetime
RxBD read lifetime (Duration Threshold)	Ref:50	RxBD read lifetime
TxBD write lifetime (Duration Threshold)	Ref:51	TxBD write lifetime
RxBD write lifetime (Duration Threshold)	Ref:52	RxBD write lifetime
Read data lifetime (Duration Threshold)	Ref:53	Read data lifetime
Rx IP packets checked for checksum	C9:31	Rx IP packets checked for checksum
TX IP packet with checksum	C1:43	TX IP packet with checksum
TX TCP/UDP packet with checksum	C2:51	TX TCP/UDP packet with checksum
TCP/UDP packets checked for c.s.	C3:53	TCP/UDP packets checked for c.s.
IP or TCP/UDP Rx checksum error	C4:53	IP or TCP/UDP Rx checksum error
Number of rejected frames by filer	C5:38	Number of rejected frames by filer
Number of rejected frames due to filer error	C6:42	Number of rejected frames due to filer error
Number of cycles Rx FIFO > 1/4 full	C8:40	Number of cycles Rx FIFO > 1/4 full

Table 18-10. Performance Monitor Events Assignment (continued)

Event	Event Assignment	Description
Number of cycles Rx FIFO > 1/2 full	C9:34	Number of cycles Rx FIFO > 1/2 full
Number of cycles Rx FIFO > 3/4 full	C1:46	Number of cycles Rx FIFO > 3/4 full
Number of cycles Rx FIFO = full	C2:53	Number of cycles Rx FIFO = full
Interrupt Controller Events		
PIC total interrupt count	Ref:26	Total number of interrupts serviced
PIC interrupt active cycles	C8:62	Number of cycles there is an active interrupt
PIC interrupt select 0 (Duration Threshold)	C1:56	Select 0: interrupt count over threshold (Please note that a request will be acknowledged only if it is not masked, and it's priority is greater than 0)
PIC interrupt select 1 (Duration Threshold)	C3:59	Select 1: interrupt count over threshold
PIC interrupt select 2 (Duration Threshold)	C5:55	Select 2: interrupt count over threshold
PIC interrupt select 3 (Duration Threshold)	C6:60	Select 3: interrupt count over threshold
PIC interrupt service cycles	C2:19	Number of cycles there is an interrupt currently being serviced
LBIU Events		
Number of accesses hitting bank (chip-select) 1	C1:51	Number of accesses hitting bank (chip-select) 1
Number of accesses hitting bank (chip-select) 2	C2:56	Number of accesses hitting bank (chip-select) 2
Number of accesses hitting bank (chip-select) 3	C3:55	Number of accesses hitting bank (chip-select) 3
Number of accesses hitting bank (chip-select) 4	C4:54	Number of accesses hitting bank (chip-select) 4
Number of accesses hitting bank (chip-select) 5	C5:48	Number of accesses hitting bank (chip-select) 5
Number of accesses hitting bank (chip-select) 6	C6:53	Number of accesses hitting bank (chip-select) 6
Number of accesses hitting bank (chip-select) 7	C7:50	Number of accesses hitting bank (chip-select) 7
Number of accesses hitting bank (chip-select) 8	C8:50	Number of accesses hitting bank (chip-select) 8
Number of atomic lock time-outs for port 2	C6:54	Number of atomic lock time-outs for port 2
Number of cycles a read is taking in GPCM	C1:53	Number of cycles a read is taking in GPCM

Table 18-10. Performance Monitor Events Assignment (continued)

Event	Event Assignment	Description
Number of cycles a read is taking in UPM	C2:58	Number of cycles a read is taking in UPM
Number of cycles a read is taking in SDRAM	C3:57	Number of cycles a read is taking in SDRAM
Number of cycles a write is taking in GPCM	C4:56	Number of cycles a write is taking in GPCM
Number of cycles a write is taking in UPM	C5:50	Number of cycles a write is taking in UPM
Number of cycles a write is taking in SDRAM	C6:55	Number of cycles a write is taking in SDRAM
Number of SDRAM bank misses	C7:51	Number of SDRAM bank misses
Number of SDRAM page misses	C8:51	Number of SDRAM page misses
PCI Express port 1 Events		
Inbound G2PI read	C8:55	A single pulse to indicate an inbound G2PI read has occurred.
Inbound G2PI write	C9:37	A single pulse to indicate an inbound G2PI write has occurred.
Inbound G2PI data	C5:60	A level signal to indicate the amount of data transferred if any for inbound G2PI request. Active for every beat of G2PI data.
Outbound G2PI read	C6:62	A single pulse to indicate an outbound G2PI read has occurred.
Outbound G2PI write	C7:61	A single pulse to indicate an outbound G2PI write has occurred.
Outbound G2PI data	C8:60	A level signal to indicate the amount of data transferred if any for outbound G2PI request. Active for every beat of G2PI data.
Inbound Static Queue 0 start (Duration Threshold)	Ref:54	Lifetime of ISQ entry 0 or 6.
Outbound Static Queue 0 start (Duration Threshold)	Ref:55	Lifetime of OSQ entry 0.
PCI Express Port 2 Events		
Inbound G2PI read	C9:43	A single pulse to indicate an inbound G2PI read has occurred.
Inbound G2PI write	C5:52	A single pulse to indicate an inbound G2PI write has occurred.
Inbound G2PI data	C6:57	A level signal to indicate the amount of data transferred if any for inbound G2PI request. Active for every beat of G2PI data.
Outbound G2PI read	C7:62	A single pulse to indicate an outbound G2PI read has occurred.
Outbound G2PI write	C8:63	A single pulse to indicate an outbound G2PI write has occurred.
Outbound G2PI data	C1:54	A level signal to indicate the amount of data transferred if any for outbound G2PI request. Active for every beat of G2PI data.
Inbound Static Queue 0 start (Duration Threshold)	Ref:29	Lifetime of ISQ entry 0 or 6.

Table 18-10. Performance Monitor Events Assignment (continued)

Event	Event Assignment	Description
Outbound Static Queue 0 start (Duration Threshold)	Ref:30	Lifetime of OSQ entry 0.
PCI Express Port 1 PIPE Interface Events		
PEX Symbol Gain of 1 when SKP detected	C2:54	Number of times the Elastic Buffer gains 1 PEX Symbol before receiving a SKP ordered-set
PEX Symbol Gain of 2 when SKP detected	C3:31	Number of times the Elastic Buffer gains 2 PEX Symbols before receiving a SKP ordered-set
PEX Symbol Gain of 3 when SKP detected	C4:59	Number of times the Elastic Buffer gains 3 PEX Symbols before receiving a SKP ordered-set
PEX Symbol Loss of 1 when SKP detected	C5:54	Number of times the Elastic Buffer loss 1 PEX Symbol before receiving a SKP ordered-set
PEX Symbol Loss of 2 when SKP detected	C6:59	Number of times the Elastic Buffer loss 2 PEX Symbols before receiving a SKP ordered-set
PEX Symbol Loss of 3 when SKP detected	C7:55	Number of times the Elastic Buffer loss 3 PEX Symbols before receiving a SKP ordered-set
PEX Symbol Disparity Error	C8:53	Number of times a receive PEX Symbol Disparity Error was detected
PEX Symbol Decode Error	C9:35	Number of times an unrecognizable PEX Symbol (Decode Error) was received and detected
PCI Express Port 2 PIPE Interface Events		
PEX Symbol Gain of 1 when SKP detected	C3:37	Number of times the Elastic Buffer gains 1 PEX Symbol before receiving a SKP ordered-set
PEX Symbol Gain of 2 when SKP detected	C4:43	Number of times the Elastic Buffer gains 2 PEX Symbols before receiving a SKP ordered-set
PEX Symbol Gain of 3 when SKP detected	C5:33	Number of times the Elastic Buffer gains 3 PEX Symbols before receiving a SKP ordered-set
PEX Symbol Loss of 1 when SKP detected	C6:41	Number of times the Elastic Buffer loss 1 PEX Symbol before receiving a SKP ordered-set
PEX Symbol Loss of 2 when SKP detected	C7:52	Number of times the Elastic Buffer loss 2 PEX Symbols before receiving a SKP ordered-set
PEX Symbol Loss of 3 when SKP detected	C8:34	Number of times the Elastic Buffer loss 3 PEX Symbols before receiving a SKP ordered-set
PEX Symbol Disparity Error	C9:10	Number of times a receive PEX Symbol Disparity Error was detected
PEX Symbol Decode Error	C2:60	Number of times an unrecognizable PEX Symbol (Decode Error) was received and detected
Serial RapidIO Events		
Inbound packet accepted	C1:60	Packet accepted on RIO
Inbound priority 0 packet accepted	C2:62	Packet accepted from RIO of priority 0

Table 18-10. Performance Monitor Events Assignment (continued)

Event	Event Assignment	Description
Inbound non-idles received	C4:62	Non idles received. This can be used to determine the RIO link utilization. This is actually 1/2 of the actual count (please see section 4.14.1 for more information).
Inbound packet retry occurred	C5:19	Packet retry occurred, any cause
Inbound buffer is full	C6:8	Clock cycle occurred in which the inbound buffer is full to any priority (measured from when EOP received to EOP transferred on OCN). Event asserted for as many clock cycles as this is true.
Inbound buffer is full to priority 0	C7:59	Clock cycle occurred in which the inbound buffer is full to priority 0 (measured from when EOP received to EOP transferred on OCN). Event asserted for as many clock cycles as this is true.
Outbound non-idles transmitted	Ref:59	Non idles transmitted. This can be used to determine the RIO link utilization. This is actually 1/2 of the actual count (please see section 4.14.1 for more information).
Outbound packet sent to RapidIO interface	C2:63	Outbound packet sent to RapidIO interface
Outbound packet was misaligned	C4:12	Outbound packet was misaligned
Outbound packet was retried	C5:20	Outbound packet was retried
Outbound packet was reordered	C6:9	Outbound packet was reordered
Outbound packet sent to RIO of priority 0	C7:60	Outbound packet sent to RIO of priority 0
Outbound buffer is full	C8:59	Clock cycle occurred in which the outbound buffer is full to any priority. Event asserted for as many clock cycles as this is true.
Outbound buffer is full to priority 0	C9:39	Clock cycle occurred in which the outbound buffer is full to priority 0. Event asserted for as many clock cycles as this is true.
RapidIO Message Unit Events		
Inbound packet received	Ref:60	Packet received of any priority
Inbound priority 0 packet received	C5:13	Packet received of priority 0
Inbound buffer is full	C6:21	Clock cycle occurred in which the inbound buffer is full to any priority (measured from when SOP received to buffer release transferred on OCN). Event asserted for as many clock cycles as this is true.
Inbound buffer is full for priority 0	C7:8	Clock cycle occurred in which the inbound buffer is full to priority 0 (measured from when SOP received to buffer release transferred on OCN). Event asserted for as many clock cycles as this is true.

18.4.8 Performance Monitor Examples

Table 18-12 contains sample register settings for the following supported modes:

- Simple event performance monitoring example

- Triggering event performance monitoring example
- Threshold event performance monitoring example
- Burstiness event performance monitoring example

The settings in Table 18-11 are identical for all four examples.

Table 18-11. PMGC0 and PMLCA*n* Settings

Field	Setting	Reason
PMGC0[FAC]	0	Counters must not be frozen.
PMGC0[PMIE]	1	Performance monitor interrupts are enabled
PMGC0[FCECE]	1	Counters should be frozen when an interrupt is signalled.
PMLCA*n*[FC]	0	Counters cannot be frozen for counting.
PMLCA*n*[CE]	1	Overflow condition enable is required to allow interrupt signalling.

For simple event counting, a non-threshold event is selected in PMLCA*n*[EVENT] and all other features are disabled by clearing all register fields except for CE.

For the triggering example any event can be selected in PMLCA*n*[EVENT]. All other features are disabled by clearing these register fields except for CE to allow interrupt signalling. If PMLCB*n*[TRIGONSEL] is 3 and PMLCB*n*[TRIGOFFSEL] is 5, the counter begins and ends counting based on the conditions in counters three and five. Furthermore, if PMLCB*n*[TRIGONCNTL] is 1, the counter begins counting when PMC3 changes value. According to the setting in PMLCB*n*[TRIGOFFCNTL], the counter ends counting when PMC5 overflows. Also, although the register settings for PMC5 is not shown, PMLCA*n*[CE] for this counter must be cleared so that interrupt signalling is not enabled and the counter does not freeze when it overflows.

For threshold counting, a threshold event must be specified in PMLCA*n*[EVENT]. For this example, the duration threshold value is scaled by two because PMLCB*n*[TBMULT] is one. All other features are disabled by clearing the appropriate fields.

Any non-threshold event can use the burstiness feature. For burstiness counting, values for PMLCA*n*[BSIZE,BGRAN,BDIST] and PMLCB*n*[TBMULT] must be specified.

Table 18-12. Register Settings for Counting Examples

Register	Register Field	Simple Event	Triggering	Threshold	Burstiness
PMGC0	FAC	0	0	0	0
	PMIE	1	1	1	1
	FCECE	1	1	1	1

Table 18-12. Register Settings for Counting Examples (continued)

Register	Register Field	Simple Event	Triggering	Threshold	Burstiness
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

The performance monitor must be reset before event counting sequences. The performance monitor can be reset by first freezing one or more counters and then clearing the freeze condition to allow the counters to count according to the settings in the performance monitor registers. Counters can be frozen individually by setting PMLCAn[FC] bits, or simultaneously by setting PMGC0[FAC]. Simply clearing these freeze bits will then allow the performance monitor to begin counting based on the register settings.þ

Note that using PMLCAn[FC] to reset the performance monitor resets only the specified counter. Performance monitor registers can be configured through reads or writes while the counters are frozen as long as freeze bits are not cleared by the register accesses.

Chapter 19
Debug Features and Watchpoint Facilities

This chapter describes all customer-visible debug modes of the integrated device. The debug features on the device pertain to these interfaces: the local bus controller (LBC), and the DDR SDRAM interface. In addition to the external interfaces, the device provides triggering capabilities based on user-programmable events. The watchpoint and trace buffer also provide some visibility to internal buses. This chapter also describes context ID registers, useful for software debug, and describes the JTAG access port signals that comply with the IEEE 1149.1 boundary-scan specification.

19.1 Introduction

As shown in the block diagram of Figure 19-1, the device provides the following debug features (listed with references to sections of this chapter that describe them):

- DDR SDRAM interface debug (Section 19.4.2, “DDR SDRAM Interface Debug”)
- Local bus controller (LBC) debug (Section 19.4.3, “Local Bus Interface Debug”)
- Watchpoint monitor and trace buffer debug (Section 19.4.4, “Watchpoint Monitor” and Section 19.4.5, “Trace Buffer”)

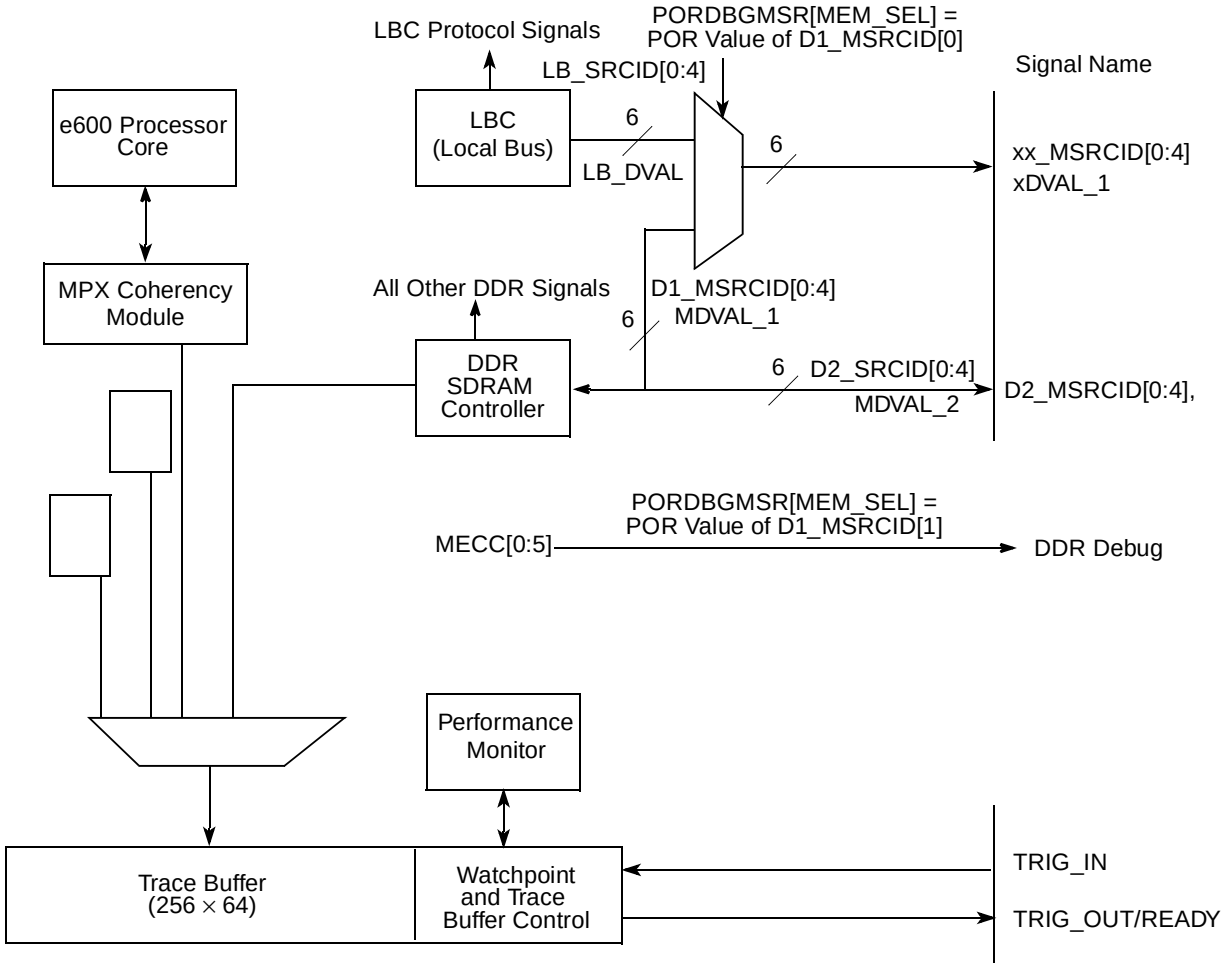


Figure 19-1. Debug and Watchpoint Monitor Block Diagram

19.1.1 Overview

As shown in Figure 19-1, debug information is provided through the following interfaces: LBC, and DDR SDRAM controllers. Limited visibility, through a 256 x 64 trace buffer, is also provided for the processor core interface. This visibility into internal device operation is useful for debugging application software through inverse assembly and reconstruction of the fetch stream.

The combination of a source ID (MSRCID[0:4]) and a data-valid signal (MDVAL) indicates that meaningful debug information is visible on either the local bus or DDR SDRAM interfaces. A logic analyzer can be programmed to capture data based on the values of MSRCID[0:4] and MDVAL. Note that there are two MSRCID/MDVAL pairs. One for local bus or DDR port 1 (selected with POR config). The other for DDR port 2.

Other system debugging is supported by the programmable triggering of the watchpoint monitor and trace buffer. Both can be triggered from one of the following three sources:

- Each other
- A performance monitor event
- An external source (through TRIG_IN).

The watchpoint monitor can be configured to assert TRIG_OUT when a programmed event occurs. The two context ID registers, described in Section 19.3.3, "Context ID Registers," are useful for software debug.

19.1.2 Features

The principle features of the debug modes and the watchpoint monitor are as follows:

- LBC and DDR interface source ID and data-valid indicators
  - LBC or DDR SDRAM port 1 source ID can be selected to be driven onto D1_MSRCID[0:4]
  - DDR SDRAM port 2 source ID can be selected to be driven onto D2_MSRCID[0:4]
  - Source ID and data-valid indicators can be selected to be driven onto the error correcting code (ECC) pins of the DDR interface
- Watchpoint monitor that supports
  - Two-level triggering
  - Programmable external trigger (TRIG_OUT)
  - Interlocked with performance monitor to use its large number of counters
- Trace buffer features that supports
  - Two-level triggering
  - Programmable external trigger (TRIG_OUT)
  - Interlocked with performance monitor to use its large number of counters
  - 256-entry trace buffer, 64 bits each
  - Programmable trace start and stop
  - Can function as a second watchpoint monitor
- Context ID registers that can be programmed to trigger events

19.1.3 Modes of Operation

The LBC, and DDR SDRAM interfaces all have debug modes, which are controlled by values on configuration inputs during the power-on reset (POR) sequence, as shown in Table 19-3.The DDR controller can also drive debug information on either MSRCID[0:4] or MECC[0:5]. See Section 19.4.1, "Source and Target ID," for additional information about the source ID information driven on the debug signals in these modes.

Note that both the watchpoint monitor and trace buffer also operate in a variety of modes.

Table 19-1. POR Configuration Settings and Debug Modes

Configuration Signal	POR Value	Effect	Reference
MSRCID0	0	Local bus SDRAM information appears on MSRCID[0:4] and MDVAL.	19.1.3.1/19-4
	1	Default value (internal pull-up resistor). DDR SDRAM information appears on MSRCID[0:4] and MDVAL.	
MSRCID1	0	MECC[0:4] operate in debug mode and provide memory debug source ID and MECC5 provides data-valid information.	19.1.3.2/19-4
	1	Default value (internal pull-up resistor). MECC[0:4] operate in normal mode and provide DDR SDRAM error correcting code information.	

19.1.3.1 Local Bus (LBC) Debug Mode

The LBC and the DDR SDRAM controller can drive debug information (source ID and data-valid indicator) onto MSRCID[0:4] and MDVAL As shown in Table 19-1, the MSRCID0 value during POR controls multiplexing. If MSRCID0 is low when sampled during POR, the local bus SDRAM information appears on MSRCID[0:4] and MDVAL; otherwise, the DDR SDRAM debug information is presented.

19.1.3.2 DDR SDRAM Interface Debug Modes

MSRCID1 is sampled during POR to multiplex either ECC or debug information on the ECC pins of the DDR SDRAM interface. As shown in Table 19-1, if MSRCID1 is low during POR, the ECC pins operate in debug mode and provide memory debug source ID and data-valid information. MSRCID1 must be pulled low during POR to use the ECC pins in debug mode. If MSRCID1 is unconnected, an internal pull-up resistor ensures the ECC pins always source DDR SDRAM error correcting code information as their default power-on reset configuration.

NOTE

If the DDR ECC pins are in debug mode (configured for debug during POR), ECC checking is disabled in the memory controller. In this case, MECC[0:4] do not provide ECC information and must not be connected to SDRAM devices.

19.1.3.3 Watchpoint Monitor Modes

The watchpoint monitor supports the following operating modes:

- Immediate trigger arming (one-level triggering)—The watchpoint monitor triggers as soon as the first trigger event occurs.
- Wait for trigger arming (two-level triggering)—The watchpoint monitor waits for a specific event before enabling (arming) the trigger logic. The monitor does not respond to trigger events until after the arming event occurs. This function is similar to two-level triggering on a logic analyzer.
- Assert TRIG_OUT on hit—The debug block can be programmed to assert the TRIG_OUT signal when a programmed watchpoint monitor event occurs. This signal can be used to trigger a logic analyzer.

19.1.3.4 Trace Buffer Modes

The trace buffer supports the following operating modes:

- Immediate trigger arming (one-level triggering)—The trace buffer triggers as soon as the first trigger event occurs.
- Wait for trigger arming (two-level triggering)—The trace buffer waits for a specific event before enabling (arming) the trigger logic. The trace buffer does not respond to trigger events until after the arming event occurs. This function is similar to two-level triggering on a logic analyzer.
-
- Specific interface selection—The trace buffer can be programmed to trace one of several internal interfaces.
- Specific event selection—The trace buffer can be programmed to trace on the occurrence of one or several concurrent events.
- Specific trace selection—To facilitate trace data filtering, the trace buffer can be configured to capture data under the following conditions:
  - On every cycle in which a valid transaction is present on the selected interface
  - Only when a watchpoint monitor event occurs
  - Only when the programmed trace event is detected
- Programmable trace stop—The trace buffer may be programmed to stop tracing when a programmed stop-tracing event occurs or when the 256-entry buffer is full.

19.2 External Signal Description

This section provides information about all the external signals associated with the various debug functions.

As shown in Table 19-1, the device 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 device provides a JTAG test access port (TAP) that complies with the IEEE 1149.1 boundary-scan specification. This section also describes JTAG TAP signals.

19.2.1 Overview

All the signals associated with device debug features are summarized in Table 19-2, listed with a reference to the page number of the section with more information. The detailed descriptions are contained in Table 19-2. Some signals (the MECC bus for example) are additionally described in other chapters, but are described here also for completeness, with emphasis on their debugging utility.

Table 19-2. Debug, Watchpoint and Test Signal Summary

Name	Description	Functional Block	Function	Reset Value	I/O	Page #
D1_MDVAL	Memory data-valid	Debug	Selectable data-valid signal from either DDR SDRAM port 1 or LBC.	1	O	19-7
D1_MSRCID [0:1]	Memory source ID	Debug	Selectable transaction source ID from either DDR SDRAM port 1 or local bus controller.	Reset_cfg	O	19-8
D1_MSRCID [2:4]				111	O	19-8
D2_MDVAL	Memory data-valid	Debug	Data-valid signal from DDR SDRAM port 2.	1	O	19-7
D2_MSRCID [0:1]	Memory source ID	Debug	Transaction source ID from DDR SDRAM port 2.	Reset_cfg	O	19-8
D2_MSRCID [2:4]				111	O	19-8
D*n*_MECC [0:7]	DDR error correcting code	DDR SDRAM	In debug mode, the high-order six bits carry debug information (transaction source ID and data-valid indication).	0x08	O[1]	19-7
TRIG_IN	Trigger in	Debug	Trigger for various function in the watchpoint monitor and trace buffer.	1	I	19-8
TRIG_OUT	Trigger out	Debug	Can be used externally for triggering a logic analyzer. Additionally, it can be used for observing system ready indication. Functions are multiplexed onto this signal depending on TOSR[SEL] (see Table 19-29).	1	O	19-8
TCK	Test clock	Debug	Clock for JTAG testing. Internally pulled up.	1	I	19-9
TDI	Test data input	Debug	Serial input for instructions and data to the JTAG test subsystem. Internally pulled up.	1	I	19-9
TDO	Test data output	Debug	Serial data output for the JTAG test subsystem. High impedance except when scanning out data.	Hi Z	O	19-9
TMS	Test mode select	Debug	Carries commands to the TAP controller for boundary scan operations. Internally pulled up.	1	I	19-9
$\overline{\text{TRST}}$	Test reset	Debug	Resets the TAP controller asynchronously.	—	I	19-9
$\overline{\text{LSSD\_MODE}}$	Test	Test	Factory Test. Refer to the *MPC8641 and MPC8641D PowerPC™ Integrated Processor Hardware Specifications* for proper treatment.		I	19-9
L1_TSTCLK	Test	Test	Factory Test. Refer to the *MPC8641 and MPC8641D PowerPC™ Integrated Processor Hardware Specifications* for proper treatment.		I	19-9
L2_TSTCLK	Test	Test	Factory Test. Refer to the *MPC8641 and MPC8641D PowerPC™ Integrated Processor Hardware Specifications* for proper treatment.		I	19-9

1 While these signals are normally bidirectional, when sourcing debug information they are output only.

19.2.2 Detailed Signal Descriptions

This section describes the details of the debug, watchpoint monitor, and JTAG test signals

19.2.2.1 Debug Signals—Details

Table 19-3 describes all signals associated with device debug modes.

Table 19-3. Debug Signals—Detailed Signal Descriptions

Signal	I/O	Description	
D1_MDVAL	O	Selectable 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 port 1 interface is selected to source information on D1_MDVAL, this signal is valid for every cycle that data is driven or received on the DDR SDRAM port 1 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.
D2_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. This signal is valid for every cycle that data is driven or received on the DDR SDRAM port 2 interface.
		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.
D*n*_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 8.5.11, “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.

Table 19-3. Debug Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
D1_MSRCID[0:4]	O	Memory source ID. Attribute signals associated with the memory interface that indicate the source ID for a transaction on an SDRAM interface. The SDRAM interface, DDR or local bus, to which the debug information applies is specified during POR with MSRCID0 as shown in Table 19-1. Two of these signals serve as reset configuration input signals.	
		State Meaning	Asserted/Negated—In debug mode, always driven with the value of the source ID. The source ID has a value of 0x1F for cycles other than RAS and CAS. The encodings shown in Table 19-30 provide detailed information about a memory transaction.
		Timing	Driven every cycle in debug mode. Similar timing to MA.
D2_MSRCID[0:4]	O	Memory source ID. Attribute signals associated with the memory interface that indicate the source ID for a transaction on an SDRAM interface. The SDRAM interface, DDR or local bus, to which the debug information applies is specified during POR with MSRCID0 as shown in Table 19-1. Two of these signals serve as reset configuration input signals.	
		State Meaning	Asserted/Negated—In debug mode, always driven with the value of the source ID. The source ID has a value of 0x1F for cycles other than RAS and CAS. The encodings shown in Table 19-30 provide detailed information about a memory transaction.
		Timing	Driven every cycle in debug mode. Similar timing to MA.

19.2.2.2 Watchpoint Monitor Trigger Signals—Details

Table 19-4 shows detailed descriptions of the watchpoint monitor and trace buffer signals.

Table 19-4. Watchpoint and Trigger Signals—Detailed Signal Descriptions

Signal	I/O	Description	
TRIG_IN	I	Trigger in. Can be used to trigger the watchpoint and trace buffers. Note this is an active-high (rising-edge triggered) signal.	
		State Meaning	Asserted—Indicates that a programmed/armed external event has been detected. Assertion may be used internally to trigger trace buffers and watchpoint mechanisms.
		Timing	Assertion/Negation—The device interprets TRIG_IN as asserted on detection of the rising edge. It may occur at any time. Must remain asserted for at least 3 system clocks to be recognized internally.
TRIG_OUT	O	Trigger out. Function determined by TOSR[SEL]. When TOSR[SEL] is non-zero, it can be used for triggering external devices, like a logic analyzer, with either the watchpoint monitor, the trace buffer, or the performance monitor as trigger sources. When TOSR[SEL] is cleared, TRIG_OUT is multiplexed with READY, which indicates the operational readiness of the device (running or in low-power or debug modes). See Chapter 4, “Reset, Clocking, and Initialization,” and Chapter 17, “Global Utilities,” for more details about reset, low-power, and debug states.	
		State Meaning	Asserted—When TOSR[SEL] is all zeros, serves as the READY signal, indicating that the device is not in a low-power or debug mode and that it has emerged from reset. SEL ≠ 0 indicates that a programmed trigger event has occurred. Negation—No final watchpoint match condition
		Timing	Assertion may occur at any time. Remains asserted for at least 3 system clocks

19.2.2.3 Test Signals—Details

Table 19-5 shows detailed descriptions of the JTAG test signals.

Table 19-5. JTAG Test and Other Signals—Detailed Signal Descriptions

Signal	I/O	Description	
TCK	I	JTAG test clock.	
		State Meaning	Asserted/Negated—Should be driven by a free-running clock signal with a 30–70% duty cycle. Input signals to the TAP are clocked in on the rising edge. Changes to the TAP output signals occur on the falling edge. The test logic allows TCK to be stopped. An unterminated input appears as a high signal level to the test logic due to an internal pull-up resistor.
		Timing	See IEEE 1149.1 standard for more details.
TDI	I	JTAG test data input.	
		State Meaning	Asserted/Negated—The value present on the rising edge of TCK is clocked into the selected JTAG test instruction or data register. An unterminated input appears as a high signal level to the test logic due to an internal pull-up resistor.
		Timing	See IEEE 1149.1 standard for more details.
TDO	O	JTAG test data output.	
		State Meaning	Asserted/Negated—The contents of the selected internal instruction or data register are shifted out on this signal on the falling edge of TCK. Remains in a high-impedance state except when scanning data.
		Timing	See IEEE 1149.1 standard for more details.
TMS	I	JTAG test mode select.	
		State Meaning	Asserted/Negated—Decoded by the internal JTAG TAP controller to distinguish the primary operation of the test support circuitry. An unterminated input appears as a high signal level to the test logic due to an internal pull-up resistor.
		Timing	See IEEE 1149.1 standard for more details.
$\overline{\text{TRST}}$	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 device. 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 *MPC8641 and MPC8641D PowerPC™ Integrated Processor Hardware Specifications* for proper treatment.	
L1_TSTCLK	I	Used for factory test. Refer to the *MPC8641 and MPC8641D PowerPC™ Integrated Processor Hardware Specifications* for proper treatment.	
L2_TSTCLK	I	Used for factory test. Refer to the *MPC8641 and MPC8641D PowerPC™ Integrated Processor Hardware Specifications* for proper treatment.	

19.3 Memory Map/Register Definition

Table 19-6 shows the memory-mapped debug and watchpoint registers of the device. Undefined 4-byte address spaces within offset 0x000–0xFFF are reserved.

Table 19-6. Debug and Watchpoint Monitor Memory Map

Watchpoint Monitor and Trace Buffer—Block Base Address 0xE_2000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
Watchpoint Monitor Registers				
0x000	WMCR0—Watchpoint monitor control register 0	R/W	0x0000_0000	19.3.1.1/19-11
0x004	WMCR1—Watchpoint monitor control register 1	R/W	0x0000_0000	19.3.1.1/19-11
0x008	Watchpoint Monitor Address High Register (WMAHR)	R/W	0x0000_0000	19.3.1.2/19-13
0x00C	WMAR—Watchpoint monitor address register	R/W	0x0000_0000	19.3.1.3/19-13
0x010	Watchpoint Monitor Address Mask High Register (WMAMHR)	R/W	0x0000_0000	19.3.1.4/19-14
0x014	WMAMR—Watchpoint monitor address mask register	R/W	0x0000_0000	19.3.1.5/19-14
0x018	WMTMR—Watchpoint monitor transaction mask register	R/W	0x0000_0000	19.3.1.6/19-15
0x01C	WMSR—Watchpoint monitor status register	R/W	0x0000_0000	19.3.1.7/19-16
Trace Buffer Registers				
0x040	TBCR0—Trace buffer control register 0	R/W	0x0000_0000	19.3.2.1/19-17
0x044	TBCR1—Trace buffer control register 1	R/W	0x0000_0000	19.3.2.1/19-17
0x048	Trace Buffer Address High Register (TBAHR)	R/W	0x0000_0000	19.3.2.2/19-19
0x04C	TBAR—Trace buffer address register	R/W	0x0000_0000	19.3.2.3/19-20
0x050	Trace Buffer Address Mask High Register (TBAMHR)	R/W	0x0000_0000	19.3.2.4/19-20
0x054	TBAMR—Trace buffer address mask register	R/W	0x0000_0000	19.3.2.5/19-21
0x058	TBTMR—Trace buffer transaction mask register	R/W	0x0000_0000	19.3.2.6/19-21
0x05C	TBSR—Trace buffer status register	R/W	0x0000_0000	19.3.2.7/19-22
0x060	TBACR—Trace buffer access control register	R/W	0x0000_0000	19.3.2.8/19-23
0x064	TBADHR—Trace buffer access data high register	R/W	0x0000_0000	19.3.2.9/19-23
0x068	TBADR—Trace buffer access data register	R/W	0x0000_0000	19.3.2.10/19-24
Context ID Registers				
0x0A0	PCIDR—Programmed context ID register	R/W	0x0000_0000	19.3.3.1/19-25
0x0A4	CCIDR—Current context ID register	R/W	0x0000_0000	19.3.3.2/19-25
Other Registers				
0x0B0	TOSR—Trigger output source register	R/W	0x0000_0000	19.3.4.1/19-26

19.3.1 Watchpoint Monitor Register Descriptions

The following sections describe the control registers for the watchpoint monitor facility.

19.3.1.1 Watchpoint Monitor Control Registers 0–1 (WMCR0, WMCR1)

The watchpoint monitor control registers (WMCR0, WMCR1) shown in Figure 19-2 and Figure 19-3 control the specification of watchpoint monitor events.

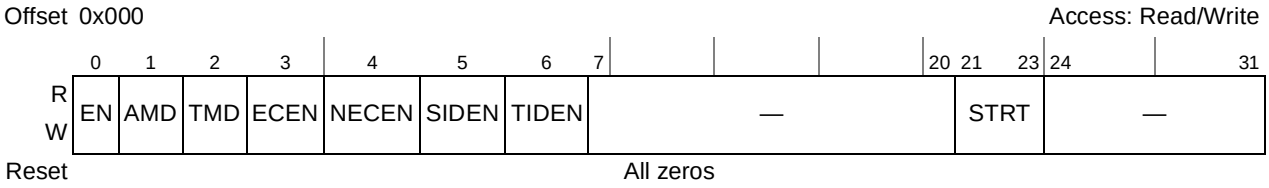


Figure 19-2. Watchpoint Monitor Control Register 0 (WMCR0)

Table 19-7 describes WMCR0 fields.

Table 19-7. WMCR0 Field Descriptions

Bits	Name	Description
0	EN	Enable 0 Watchpoint monitor events are not flagged. 1 A watchpoint monitor event is flagged.
1	AMD	Address match disable. Qualifies address match as a watchpoint event criterion. 0 Address matching is used to recognize a watchpoint event. 1 Address matching does not affect watchpoint event detection.
2	TMD	Transaction match disable. Qualifies transaction type match (as defined in WMCR1[IFSEL] and WMTMR) as a watchpoint event criterion. 0 A transaction type match is used to recognize watchpoint events. 1 A transaction type match does not affect watchpoint event detection.
3	ECEN	Equal context enable. Qualifies the matching of current context with programmed context as a watchpoint event criterion, as written in the context registers described in Section 19.3.3, “Context ID Registers.” 0 Current context match does not affect watchpoint event detection. 1 Watchpoint events are qualified by comparing current context with the programmed context event value. **Note:** ECEN and NECEN must not be enabled in the same run. If both are set, watchpoint events are inhibited (never occur).
4	NECEN	Not equal context enable. Qualifies the matching of current context with programmed context as a watchpoint event criterion, as written in the context registers described in Section 19.3.3, “Context ID Registers.” 0 The failure of a current context match does not affect watchpoint event detection 1 Watchpoint events are qualified with NOT getting a current context compare with the programmed context event value. **Note:** ECEN and NECEN must not be enabled in the same run. If both are set, watchpoint events are inhibited (never occur).

Table 19-7. WMCR0 Field Descriptions (continued)

Bits	Name	Description
5	SIDEN	Source ID enable 0 Source ID does not affect watchpoint event detection. 1 Watchpoint events are qualified by comparison with the programmed WMCR1(SID) value.
6	TIDEN	Target ID enable 0 Target ID does not affect watchpoint event detection. 1 Watchpoint events are qualified by comparison with the programmed WMCR1(TID) value.
7–20	—	Reserved
21–23	STRT	Start condition. Specifies the event that arms the watchpoint monitor to start looking for the programmed event. 000 No event. Armed immediately 001 Trace buffer event is detected 010 Performance monitor signals overflow 011 TRIG_IN transitions from 0 to 1 100 TRIG_IN transitions from 1 to 0 101 Current context ID equals programmed context ID 110 Current context ID is not equal to programmed context ID 111 Reserved
24–31	—	Reserved

Figure 19-3 shows the WMCR1.

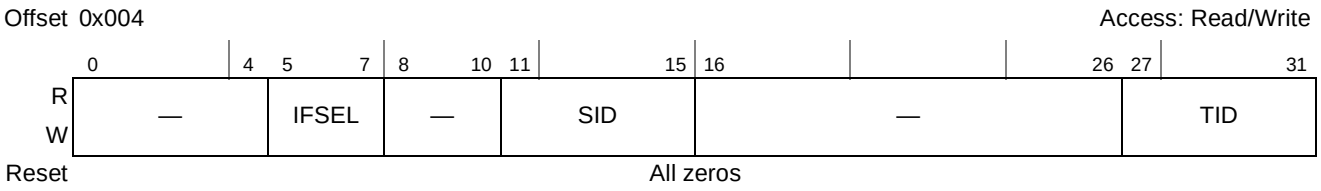


Figure 19-3. Watchpoint Monitor Control Register 1 (WMCR1)

Table 19-8 describes the WMCR1 fields.

Table 19-8. WMCR1 Field Descriptions

Bits	Name	Description
0–4	—	Reserved
5–7	IFSEL	Interface selection. Selects the address, transaction type (as defined in WMTMR), and other attributes to be used for comparison 000 Select coherency module (MCM) dispatch interface 001 Select internal DDR SDRAM interface (TOSR[0] dictates which DDR SDRAM interface) 010–011 Reserved 100 Select internal PCI Express outbound interface (TOSR[1] dictates which PEX interface) 101–111 Reserved
8–10	—	Reserved
11–15	SID	Source ID. Specifies the source ID associated with WMCR0[SIDEN]. For a definition of the source ID, see Table 19-30.

Table 19-8. WMCR1 Field Descriptions (continued)

Bits	Name	Description
16–26	—	Reserved
27–31	TID	Target ID. Specifies the target ID associated with WMCR0[TIDEN]. For a definition of the target ID see Table 19-30.

19.3.1.2 Watchpoint Monitor Address High Register (WMAHR)

The watchpoint monitor address high register (WMAHR) shown in Figure 19-4 contains the high-order address bits of the address to match against if WMCR[AMD] is clear.

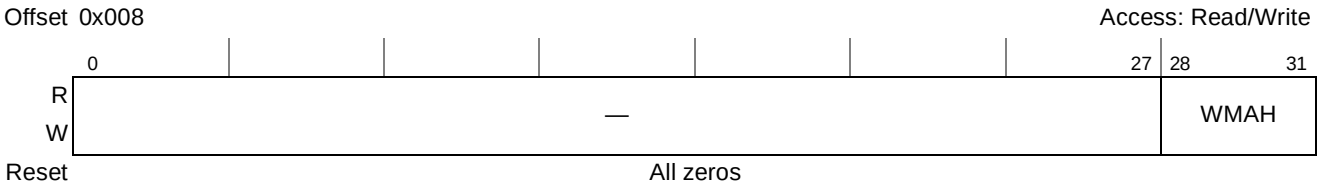


Figure 19-4. Watchpoint Monitor Address High Register (WMAHR)

Table 19-9 describes the WMAHR fields.

Table 19-9. WMAHR Field Descriptions

Bit	Name	Description
0–27	—	Reserved
28–31	WMAH	Watchpoint monitor address high. High-order bits of the match address. A value of 0 masks the address comparison for the corresponding address bit. These correspond to bits 0:3 of the real or platform address.

19.3.1.3 Watchpoint Monitor Address Register (WMAR)

The watchpoint monitor address register (WMAR) shown in Figure 19-5 contains the low-order address bits of the address to match against if WMCR[AMD] is clear. Note that this address may be further qualified with the bits described in Section 19.3.1.5, “Watchpoint Monitor Address Mask Register (WMAMR).”

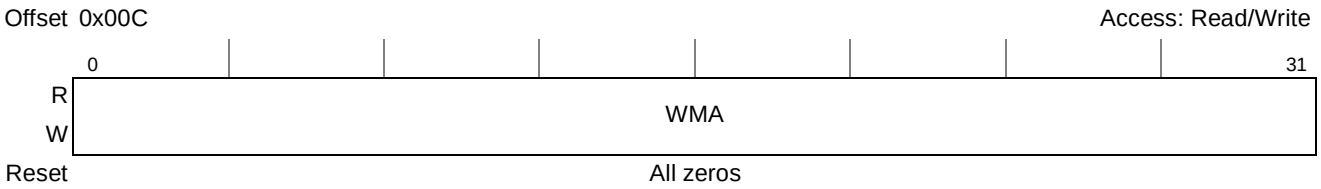


Figure 19-5. Watchpoint Monitor Address Register (WMAR)

Table 19-10 describes the WMAR fields.

Table 19-10. WMAR Field Descriptions

Bits	Name	Description
0–31	WMA	Watchpoint monitor address. Low-order bits of the match address. These correspond to bits 4:35 of the real or platform address.

19.3.1.4 Watchpoint Monitor Address Mask High Register (WMAMHR)

The watchpoint monitor address mask high register (WMAMHR) shown in Figure 19-6 contains the mask for the high-order address bits in the WMAHR. Permits user to mask address bits from the comparison.

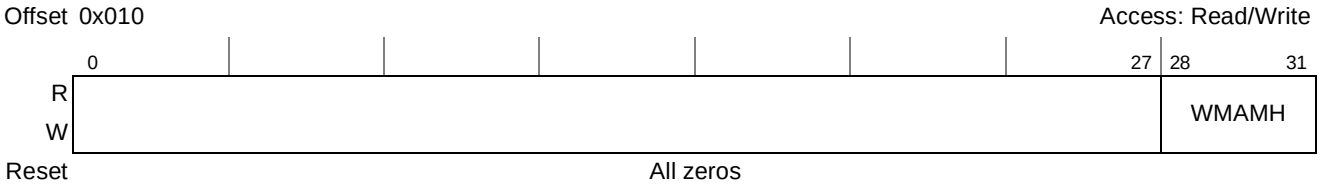


Figure 19-6. Watchpoint Monitor Address Mask High Register (WMAMHR)

Table 19-11 describes the WMAMHR fields.

Table 19-11. WMAMHR Field Descriptions

Bit	Name	Description
0–27	—	Reserved
28–31	WMAMH	Watchpoint monitor address mask high. High-order bits of the match address mask.

19.3.1.5 Watchpoint Monitor Address Mask Register (WMAMR)

The watchpoint monitor address mask register (WMAMR) shown in Figure 19-7 contains the low-order address bits of the mask for the address in the WMAR.

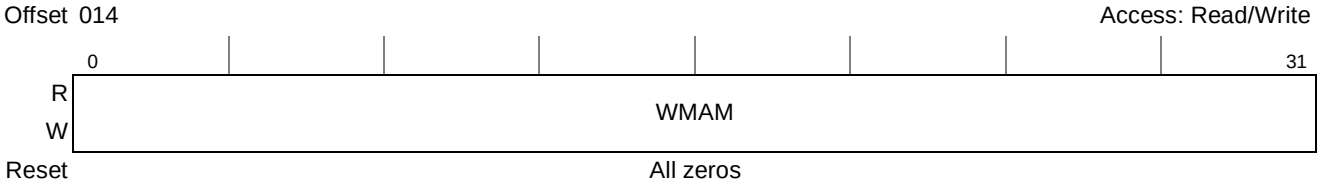


Figure 19-7. Watchpoint Monitor Address Mask Register (WMAMR)

Table 19-12 describes the WMAMR fields.

Table 19-12. WMAMR Field Descriptions

Bits	Name	Description
0–31	WMAM	Watchpoint monitor address mask. Low-order bits of the match address mask. A value of zero masks the address comparison for the corresponding address bit.

19.3.1.6 Watchpoint Monitor Transaction Mask Register (WMTMR)

The watchpoint monitor transaction mask register (WMTMR), shown in Figure 19-8, specifies which transaction types to monitor. WMTMR allows users to qualify watchpoint events specifically with any combination of transaction types. As shown in Table 19-14, each bit represents as many as four separate transaction types; one for each interface. Setting a bit enables watchpoint monitoring for the corresponding transaction types.

Because the supported transaction types vary by interface, the type designated by a WMTMR field also depends on the interface specified by WMCR1[IFSEL]. Table 19-14 lists transaction types associated with each WMTMR bit by interface.

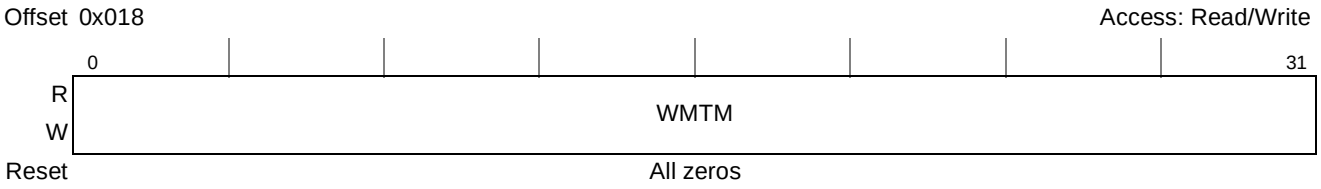


Figure 19-8. Watchpoint Monitor Transaction Mask Register (WMTMR)

Table 19-13 describes the WMTMR fields.

Table 19-13. WMTMR Field Descriptions

Bits	Name	Description
0–31	WMTM	Watchpoint monitor transaction mask. Each bit corresponds to a transaction type as defined in Table 19-14. The transaction associated with any particular bit may be different depending on the interface being monitored. A value of 1 for a given mask bit enables the matching of the transaction associated with that bit. These bits are meaningful only when WMCR0[TMD]=0.

The following table defines the transactions associated with each transaction mask bit for the different interfaces supported by the watchpoint monitor.

Table 19-14. Transaction Types By Interface

Bit	Coherency Module Dispatch	DDR Controller	PCI Outbound Request
0	Write with local processor snoop	Write	Memory write
1	Write with no local processor snoop	—	I/O write
2	Reserved	Write with allocate	—
3	Reserved	Write with allocate and lock	—
4–7	Reserved		
8	Read with local processor snoop	Read	Memory Read
9	Read with no local processor snoop	—	I/O Read
10	Reserved	Read with unlock	—
11–15	Reserved		

Table 19-14. Transaction Types By Interface (continued)

Bit	Coherency Module Dispatch	DDR Controller	PCI Outbound Request
16	Atomic clear	ATOMIC clear	—
17	Atomic set	ATOMIC set	—
18	Atomic decrement	ATOMIC decrement	—
19	Atomic increment	ATOMIC increment	—
20–31	Reserved		

19.3.1.7 Watchpoint Monitor Status Register (WMSR)

The watchpoint monitor status register (WMSR) shown in Figure 19-9 indicates the state of the watchpoint monitor.

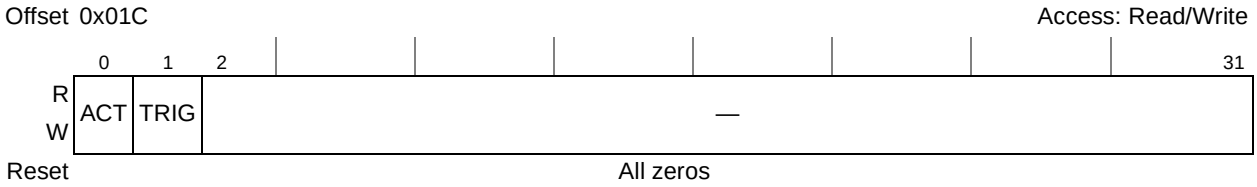


Figure 19-9. Watchpoint Monitor Status Register (WMSR)

Table 19-15 describes the WMSR fields.

Table 19-15. WMSR Field Descriptions

Bits	Name	Description
0	ACT	Active 0 The start triggering event has not occurred; watchpoint monitor is not armed. 1 The start triggering event has occurred; watchpoint monitor is armed.
1	TRIG	Triggered 0 The programmed event in WMCR0 has not yet been triggered. 1 The programmed event in WMCR0 has been triggered at least once.
2–31	—	Reserved

19.3.2 Trace Buffer Register Descriptions

The following sections describes the trace buffer registers.

19.3.2.1 Trace Buffer Control Registers (TBCR0, TBCR1)

The trace buffer control registers (TBCR0, TBCR1), shown in Figure 19-10 and Figure 19-11, specify trace buffer events.

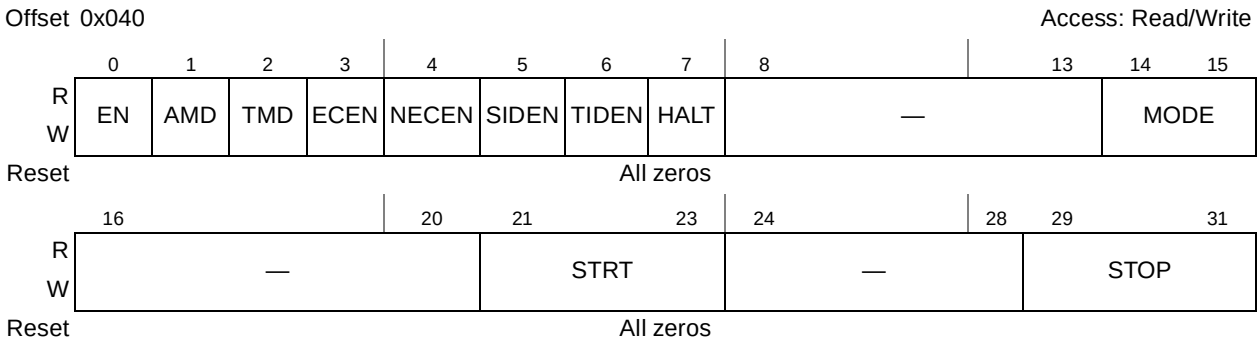


Figure 19-10. Trace Buffer Control Register 0 (TBCR0)

Table 19-16 describes the TBCR0 fields.

Table 19-16. TBCR0 Field Descriptions

Bits	Name	Description
0	EN	Enable 0 The trace buffer facility is disabled. 1 The trace buffer facility is enabled.
1	AMD	Address match disable 0 The address match is used to qualify a trace buffer event. 1 The address match is ignored when detecting a trace buffer event.
2	TMD	Transaction match disable 0 The transaction type match is used to qualify a trace buffer event. 1 The transaction type match is ignored when detecting a trace buffer event.
3	ECEN	Equal context enable. Qualifies the matching of current context with programmed context as a trace buffer event criterion, as written in the context registers described in Section 19.3.3, "Context ID Registers." 0 Current context match does not affect trace buffer event detection 1 Trace buffer events are qualified by comparing current context with the programmed context event value. **Note:** ECEN and NECEN must not be enabled in the same run. If both are set, watchpoint events are inhibited (never occur).

Table 19-16. TBCR0 Field Descriptions (continued)

Bits	Name	Description
4	NECEN	Not equal context enable. Qualifies the matching of current context with programmed context as a trace buffer event criterion, as written in the context registers described in Section 19.3.3, "Context ID Registers." 0 The failure of a current context match does not affect trace buffer event detection 1 trace buffer events are qualified with NOT getting a current context compare with the programmed context event value. **Note:** ECEN and NECEN must not be enabled in the same run. If both are set, watchpoint events are inhibited (never occur).
5	SIDEN	Source ID enable 0 Trace buffer events ignore the programmed source ID value. 1 Trace buffer events are qualified by comparison with the programmed SID event value.
6	TIDEN	Target ID enable 0 Trace buffer events ignore the programmed TID event value. 1 Trace buffer events are qualified by comparison with the programmed TID event value. This comparison only applies when the MCM is selected for tracing (TBCR1[IFSEL] is all zeros).
7	HALT	Halt causes the trace buffer to stop tracing immediately.
8–13	—	Reserved
14–15	MODE	Trace mode. Specifies one of two trace modes. 00 Trace every valid transaction 01 Reserved 10 Trace only cycles in which a trace event is detected. Note that if EN and other TBCR0 fields are not properly programmed to specify a traceable event, tracing occurs for every valid address. 11 Reserved
16–20	—	Reserved
21–23	STRT	Start condition. Specifies the event that arms the trace buffer to start looking for the programmed event 000 No event. Armed immediately 001 Watchpoint monitor event is detected 010 Trace buffer event is detected 011 Performance monitor signals overflow 100 TRIG_IN transitions from 0 to 1 101 TRIG_IN transitions from 1 to 0 110 Current context ID equals programmed context ID 111 Current context ID does not equal programed context ID
24–28	—	Reserved
29–31	STOP	Trace stop mode. Specifies the event that stops the updating of the trace buffer after it has been started. Trace buffer only stops after it has been triggered at least once. 000 Buffer is full 001 Watchpoint monitor event is detected 010 Trace buffer event is detected 011 Performance monitor signals overflow 100 TRIG_IN transitions from 0 to 1 101 TRIG_IN transitions from 1 to 0 110 Current context ID equals programmed context ID 111 Current context ID does not equal programed context ID

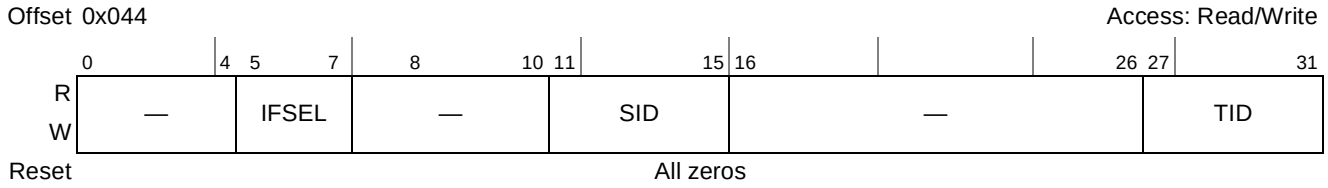


Figure 19-11. Trace Buffer Control Register 1 (TBCR1)

Table 19-17 describes the TBCR1 fields.

Table 19-17. 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 coherency module (MCM) dispatch interface 001 Selects internal DDR SDRAM interface (TOSR[0] dictates which DDR SDRAM interface) 010–011 Reserved 100 Selects internal PEX outbound interface(TOSR[1] dictates which PEX interface) 101–111 Reserved
8–10	—	Reserved
11–15	SID	Source ID. Specifies the source ID associated with TBCR0[SIDEN]. The source ID is defined in Table 19-30.
16–26	—	Reserved
27–31	TID	Target ID. Specifies the target ID associated with TBCR0[TIDEN]. The target ID is defined in Table 19-30.

19.3.2.2 Trace Buffer Address High Register (TBAHR)

The trace buffer address high register (TBAHR) shown in Figure 19-12 contains the high-order address bits of the address to match against if TBCR[AMD] is clear.

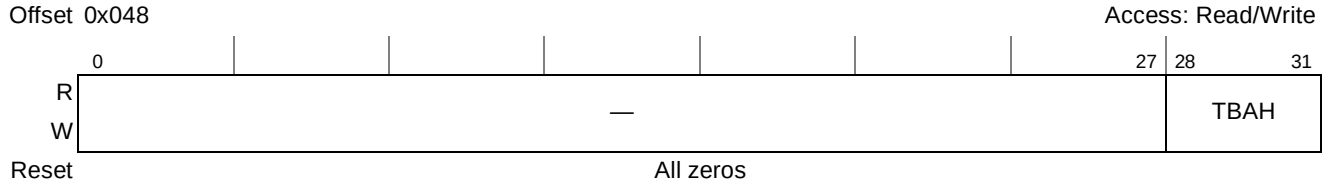


Figure 19-12. Trace Buffer Address High Register (TBAHR)

Table 19-18 describes the TBAHR fields.

Table 19-18. TBAHR Field Descriptions

Bit	Name	Description
0–27	—	Reserved
28–31	TBAH	Trace buffer address high. High-order bits of the match address.

19.3.2.3 Trace Buffer Address Register (TBAR)

The trace buffer address register (TBAR) shown in Figure 19-13 contains the low-order address bits of the address to match against (if TBCR[AMD] is zero). This address may be further qualified by the mask bits defined in Section 19.3.2.5, "Trace Buffer Address Mask Register (TBAMR)."

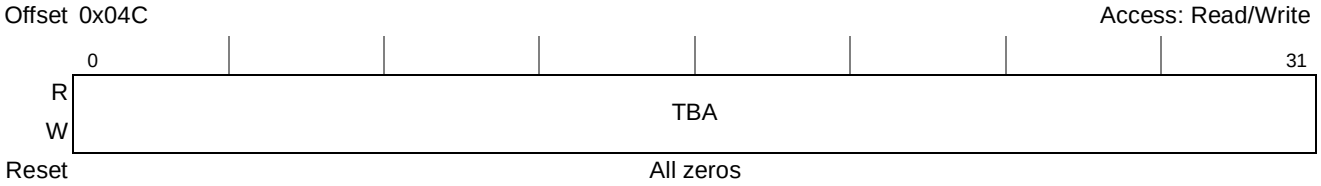


Figure 19-13. Trace Buffer Address Register (TBAR)

Table 19-19 describes the TBAR field.

Table 19-19. TBAR Field Descriptions

Bits	Name	Description
0–31	TBA	Trace buffer address. Low-order bits of the match address.

19.3.2.4 Trace Buffer Address Mask High Register (TBAMHR)

The trace buffer address mask high register (TBAMHR) shown in Figure 19-14 contains the mask for the high-order address bits in the TBAHR. Allows excluding address bits from the comparison.

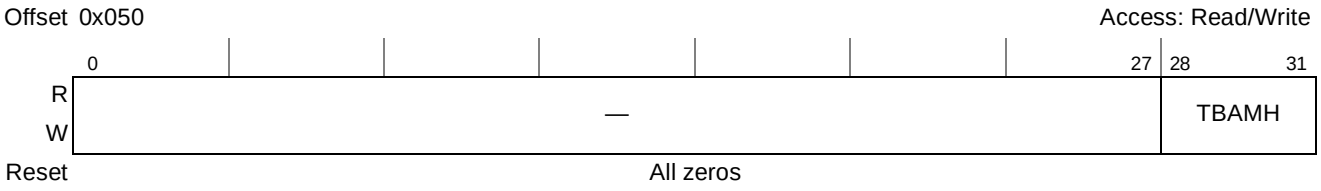


Figure 19-14. Trace Buffer Address High Register (TBAMHR)

Table 19-20 describes the TBAMHR fields.

Table 19-20. TBAMHR Field Descriptions

Bits	Name	Description
0–27	—	Reserved
28–31	TBAMH	Trace buffer address mask high. High-order bits of the match address mask.

19.3.2.5 Trace Buffer Address Mask Register (TBAMR)

The trace buffer address mask register (TBAMR) shown in Figure 19-15 contains a mask for the TBAR, which allows excluding address bits from the comparison.

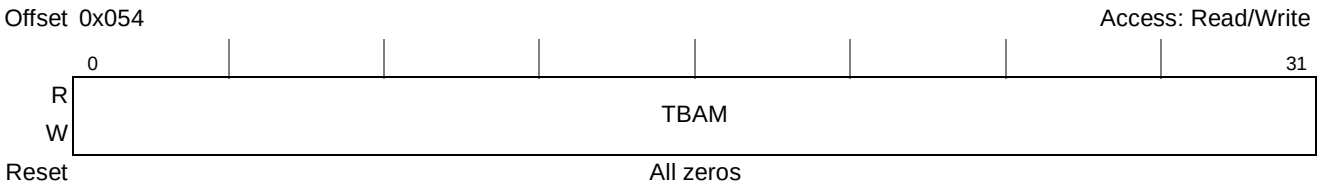


Figure 19-15. Trace Buffer Address Mask Register (TBAMR)

Table 19-21 describes the TBAMR field.

Table 19-21. TBAMR Field Descriptions

Bits	Name	Description
0–31	TBAM	Trace buffer address mask. Low-order bits of the match address mask. A value of zero masks the address comparison for the corresponding address bit.

19.3.2.6 Trace Buffer Transaction Mask Register (TBTMR)

The trace buffer transaction mask register (TBTMR) shown in Figure 19-16 specifies which transaction types to monitor. Each bit in the TBTMR represents a transaction type on the selected interface. The transaction associated with any particular bit depends on the interface being monitored as specified by TBCR1[IFSEL]. Note that the transactions used for defining trace buffer events are the same as those defined for watchpoint monitor events. Thus, Table 19-14 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 TBCR[TMD] is clear).

Different interfaces support different transaction types, and the same bit may represent different transaction types depending on the interface.

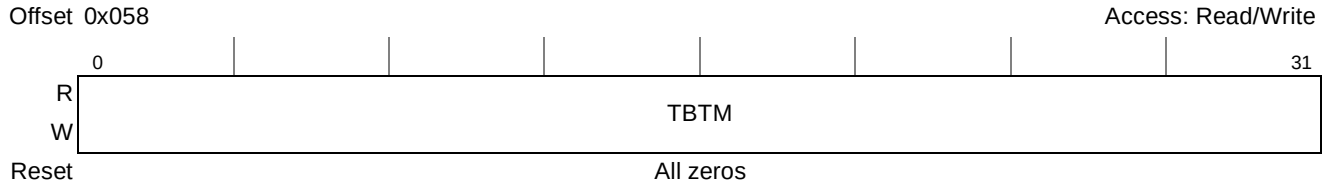


Figure 19-16. Trace Buffer Transaction Mask Register (TBTMR)

Table 19-22 describes the TBTMR field.

Table 19-22. TBTMR Field Descriptions

Bits	Name	Description
0–31	TBTM	Trace buffer transaction mask. Each bit corresponds to a transaction type as defined in Table 19-14. The transaction associated with a bit depends on the interface being monitored. A value of 1 for a given mask bit enables the matching of the transaction associated with that bit. These bits are meaningful only when TBCR0[TMD]=0.

19.3.2.7 Trace Buffer Status Register (TBSR)

The trace buffer status register (TBSR) shown in Figure 19-17 indicates the operational state of the trace buffer.

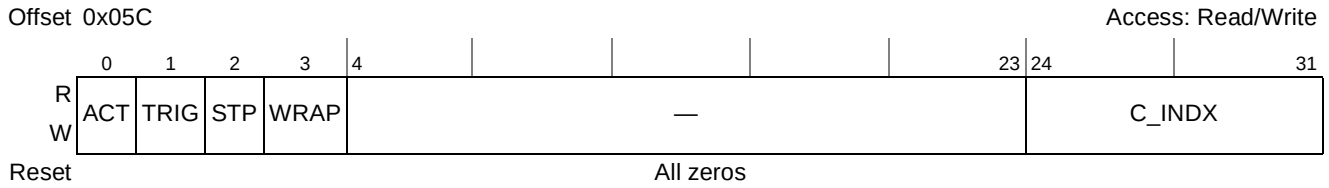


Figure 19-17. Trace Buffer Status Register (TBSR)

Table 19-23 describes the TBSR fields.

Table 19-23. 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

Table 19-23. TBSR Field Descriptions (continued)

Bits	Name	Description
4–23	—	Reserved
24–31	C_INDX	Current index. Represents the current value of the write pointer at the time TBSR was read. This value may be written by software to initialize the write pointer; however, software is not allowed to write the write pointer while the trace buffer is active. Writes are ignored while the trace buffer is active. It is recommended to write the status register before enabling the trace buffer in order to zero out any bits that might have been set during a prior run and to initialize the write pointer to zero.

19.3.2.8 Trace Buffer Access Control Register (TBACR)

The trace buffer access control register (TBACR), shown in Figure 19-18, enables software to read or write the trace buffer. Each entry is 64 bits; therefore, it takes one write of TBACR and two reads of the access data register (TBADR and TBADHR) to read one 256-entry array entry. Similarly, it takes one write of TBACR and two writes of TBADR and TBADHR to write one array entry. Software can access any entry by writing the appropriate index into TBACR[INDX]. To read or write the buffer sequentially, starting with entry 0, the index must start with a value of 0 and increment every time a new entry is accessed.

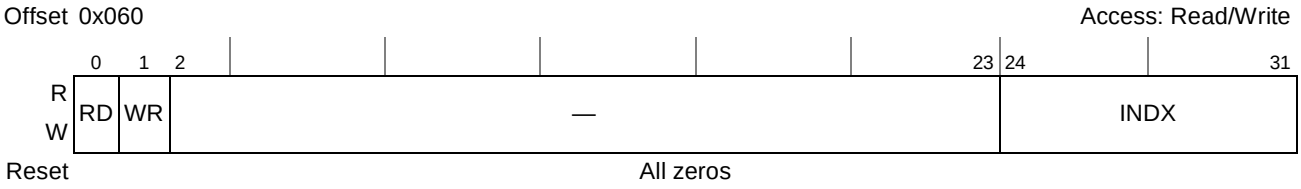


Figure 19-18. Trace Buffer Access Control Register (TBACR)

Table 19-24 describes the TBACR fields.

Table 19-24. TBACR Field Descriptions

Bits	Name	Description
0	RD	Read command. When set, a trace buffer read is performed using the value of TBACR[INDX]. This bit is automatically cleared when the read is performed.
1	WR	Write command. When set, a trace buffer write is performed using the value of TBACR[INDX]. This bit is automatically cleared when the write is performed. A write occurs only if the trace buffer is not active: write requests are ignored while the buffer is active.
2–23	—	Reserved
24–31	INDX	Buffer index to read from or write into (0–255). Used in conjunction with TBACR[RD] and TBACR[WR].

19.3.2.9 Trace Buffer Access Data High Register (TBADHR)

The trace buffer access data high register (TBADHR), shown in Figure 19-19, contains the high-order 32 bits of the data read from the trace buffer during a software-initiated read command (TBACR[RD]), or the write data to be written into the trace buffer during a software-initiated write command (TBACR[WR]).

TBACR must be configured to perform a read before this register contains valid data. This register must be initialized by software before configuring the TBACR to perform a write command.

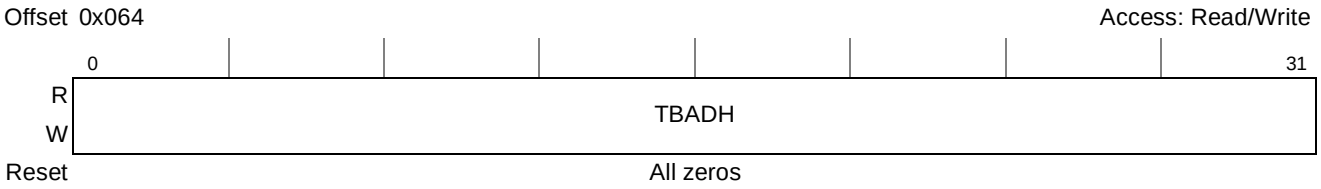


Figure 19-19. Trace Buffer Read High Register (TBADHR)

Table 19-25 describes TBADHR.

Table 19-25. TBADHR Field Descriptions

Bits	Name	Description
0–31	TBADH	Trace buffer access data high. The higher 32 bits of the data read from or to be written into the trace buffer, depending on whether the array is accessed with a read or a write.

19.3.2.10 Trace Buffer Access Data Register (TBADR)

The trace buffer access data register (TBADR), shown in Figure 19-20, contains the low-order 32 bits of the data read from the trace buffer during a software-initiated read command (TBACR[RD]) or the write data to be written into the trace buffer during a software-initiated write command (TBACR[WR]). TBACR must be configured to perform a read before this register contains valid data. This register must be initialized by software before configuring the TBACR to perform a write command.

Figure 19-20. Trace Buffer Access Data Register (TBADR)

Table 19-26 describes the TBADR field.

Table 19-26. TBADR Field Descriptions

Bits	Name	Description
0–31	TBAD	Trace buffer access data. Corresponds to the lower 32 bits of the data read from the trace buffer or to be written into the trace buffer, depending on whether software is accessing the array with a read or a write.

19.3.3 Context ID Registers

This section describes the context ID registers. The current context ID register (CCIDR) and programmed context ID registers (PCIDR) are set by software and facilitate debugging complex software.

19.3.3.1 Programmed Context ID Register (PCIDR)

The programmed context ID register (PCIDR), shown in Figure 19-21, contains the user-programmed context ID. This register can be configured to trigger watchpoint events when its value matches the current context ID register (CCIDR), as controlled by WMCR0[ECEN] and WMCR0[NECEN]. See Section 19.3.1.1, “Watchpoint Monitor Control Registers 0–1 (WMCR0, WMCR1),” for more information.

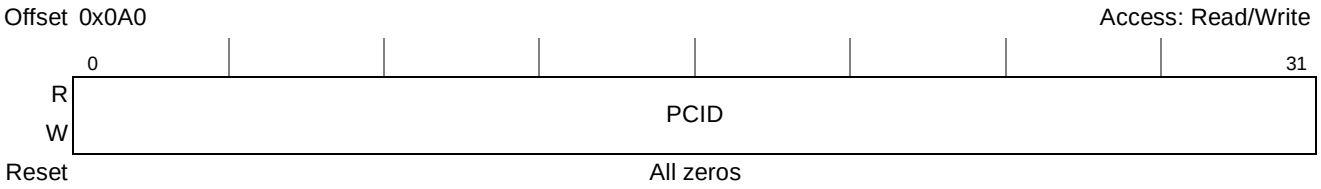


Figure 19-21. Programmed Context ID Register (PCIDR)

Table 19-27 describes the PCIDR field.

Table 19-27. PCIDR Field Descriptions

Bits	Name	Description
0–31	PCID	Programmed context ID. Contains the user-programmed context ID. Compared with current context ID for context-sensitive event triggering

19.3.3.2 Current Context ID Register (CCIDR)

The current context ID register (CCIDR) shown in Figure 19-22 contains the current context ID. This register is written by software after a context switch and can be used to trigger events when compared with the programmed context ID register (PCIDR).

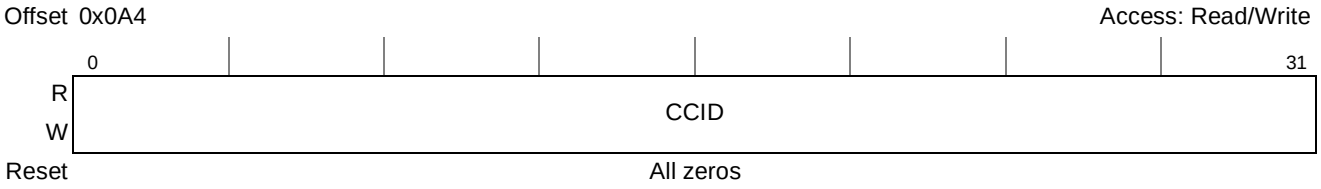


Figure 19-22. Current Context ID Register (CCIDR)

Table 19-28 describes the CCIDR field.

Table 19-28. CCIDR Field Descriptions

Bits	Name	Description
0–31	CCID	Current context ID. Set by user software. Typically loaded immediately following a context switch. Compared with user-programmed context ID for context-sensitive event triggering

19.3.4 Trigger Out Function

TRIG_OUT provides a convenient mechanism for triggering external system monitors and diagnostic equipment such as logic analyzers. Note that READY is multiplexed with TRIG_OUT. See the last paragraph of Section 4.4.2, “Power-On Reset Sequence,” for more information about READY functionality.

When the trace buffer hit is selected by TOSR[SEL], TRIG_OUT is only meaningful if the trace buffer control register 0 (TBCR0) is properly configured to hit on a traceable event. The same holds true for the watchpoint monitor when the watchpoint monitor is selected by TOSR[SEL].

19.3.4.1 Trigger Out Source Register (TOSR)

The trigger out source register (TOSR) shown in Figure 19-23 specifies the source for TRIG_OUT. The three event-trigger sources are the following:

- The watchpoint monitor
- The trace buffer
- The performance monitor

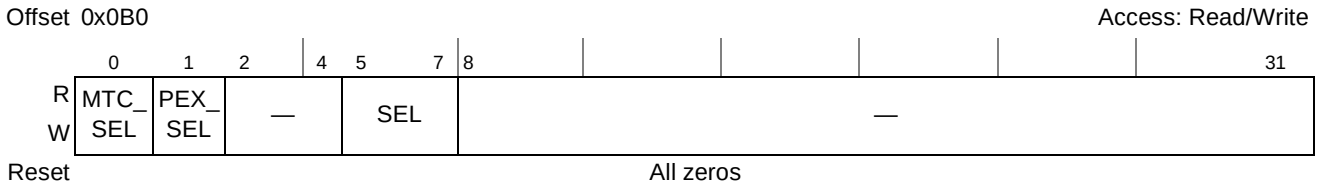


Figure 19-23. Trigger Out Source Register (TOSR)

Table 19-29 describes the TOSR fields.

Table 19-29. TOSR Field Descriptions

Bits	Name	Description
0	MTC_SEL	MTC Select. Selects which MTC on the DDR I/F 0 MTC1 1 MTC2
1	PEX_SEL	PEX Select. Selects which PEX on the PEX I/F 0 PEX1 1 PEX2
2–4	—	Reserved
5–7	SEL	Select. Selects the source for TRIG_OUT 000 READY signal. Multiplexed with TRIG_OUT. Basic device state indicator. READY asserts whenever the device is not in reset or not asleep. See Chapter 4, “Reset, Clocking, and Initialization,” for more details about the reset sequence, and Chapter 17, “Global Utilities,” for more information about power management states. 001 Selects the watchpoint monitor hit indication 010 Selects the trace buffer hit indication 011 Selects the performance monitor overflow indication
8–31	—	Reserved

19.4 Functional Description

The debug features on the device use the LBC interfaces, and the DDR SDRAM interfaces.

19.4.1 Source and Target ID

Debug information that is common to all the interfaces is the source ID (SID). The transaction source ID provides enough information to determine which block or port originated a transaction including the distinction between instruction and data fetches from the processor core. Table 19-30 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 coherency module (MCM) dispatch bus
- The watchpoint monitor (WMCR1[IFSEL] = 000)
- The trace buffer (TBCR1[IFSEL] = 000)

Table 19-30. Source and Target ID Values

Source/Target ID	Source/Target	Source/Target ID	Source/Target
00000	PCI Express 1	10000	Core 0 (instruction/data)
00001	PCI Express 2/Serial RapidIO (as a source)	10001	Reserved
00010	Reserved	10010	Core 1 (instruction/data)
00011	Reserved	10011	Reserved
00100	Local bus controller	10100	Reserved
00101	Reserved	10101	DMA
00110	Reserved	10110	DDR port 2
00111	Reserved	10111	System access port (SAP)
01000	Configuration space	11000	eTSEC1
01001	Reserved	11001	eTSEC2
01010	Boot sequencer	11010	eTSEC3
01011	Interleaved DDR memory controllers 1 & 2	11011	eTSEC4
01100	Serial RapidIO (as a target)	11100	RapidIO message unit
01101	Reserved	11101	RapidIO doorbell unit
01110	Reserved	11110	RapidIO port-write unit
01111	DDR port 1	11111	Reserved

19.4.2 DDR SDRAM Interface Debug

The DDR interface has two debug modes distinguished by which pins drive the debug information. In one mode, debug information (source ID, data valid) is multiplexed onto the ECC pins; the other mode uses the debug pins.

19.4.2.1 Debug Information on Debug Pins

If MSRCID0 is high when sampled during POR, the debug information from the DDR SDRAM interface is driven on MSRCID[0:4] and MDVAL. This POR value is captured in PORDBGMSR[MEM_SEL] as described in Section 17.4.1.5, “POR Debug Mode Status Register (PORDBGMSR).” In this mode, the source ID appears on MSRCID[0:4] during a RAS or CAS cycle. During any other cycle, the value of MSRCID[0:4] is all ones, which indicates idle cycles on the address/command interface. Similarly, MDVAL is asserted during valid data cycles on the DDR interface.

19.4.2.2 Debug Information on ECC Pins

If MSRCID1 is low when sampled during POR, debug information from the DDR SDRAM interface is selected to appear on MECC[0:5] as shown in Figure 19-1. In this mode, the ID value of the source port, (the source ID), appears on MECC[0:4] during a RAS or CAS cycle. During any other cycle the value of MECC[0:4] is all ones. A data-valid signal (DVAL) is driven on MECC5 during valid DDR SDRAM data cycles.

NOTE

In this mode, MECC[0:5] must be disconnected from all SDRAM devices to prevent contention on those lines.

19.4.3 Local Bus Interface Debug

If MSRCID0 is low when sampled during POR, the LBC is selected as the source for the debug information appearing on MSRCID[0:4] and MDVAL. For more information on this mode, see Section 12.1.3.2, “Source ID Debug Mode.”

19.4.4 Watchpoint Monitor

The watchpoint monitor (WM) can be programmed to arm and trigger on many different events including any of the following:

- External event (through TRIG_IN)
- A trace buffer event
- A performance monitor overflow event
- A comparison of the current and programmed context ID registers.

A watchpoint event can be used in the following ways:

- Trigger a logic analyzer (using TRIG_OUT)
- Arm or trigger the trace buffer

- Trigger a performance monitor event.

The large counters available in the performance monitor block and the interlock between it and the watchpoint monitor support sophisticated debug scenarios.

A WM trigger event may be composed of several events programmed in the watchpoint monitor control registers (WMCR0–WMCR1). Because the watchpoint monitor is disabled by default during POR, these registers must be initialized to make use of this debug feature. Note that the WM address mask register (WMAMR) and the type mask register (WMTMR) are cleared during POR. This means that the watchpoint monitor's default behavior following a power-on reset is to trigger on any address and no transaction type. The reset value of WMCR0[TMD] is 0 which means transaction matching is enabled but since no transaction is selected (WMTMR=0), a match will never occur. Either the transaction matching must be disabled by setting WMCR0[TMD] to a value of 1, or valid transactions must be selected by setting one or more of the WMTMR bits to a value of 1.

19.4.4.1 Watchpoint Monitor Performance Monitor Events

The WM can produce a performance monitor (PM) event with every trigger. This is accomplished by configuring the performance monitor to count WM events. For more information on this configuration see the events named 'Number of watchpoint monitor hits' and 'Number of trace buffer hits' in Table .

Multi-level triggers can be created using the watchpoint monitor, the performance monitor, and the trace buffer combined. For example, the WM can be programmed to trigger on events that also increment a PM counter (the performance monitor must also be programmed to respond to this event), the output of which (perfmon_overflow) could trigger the start of tracing in the trace buffer.

19.4.5 Trace Buffer

The trace buffer is a 256×64 array that can capture information about the internal processing of transactions to selected interfaces. The trace buffer controls are a superset of those for the watchpoint monitor. Close inspection of the trace buffer control registers (TBCR*n*) and the WM control registers (WMCR*n*) shows that trace buffer controls not needed for the WM are marked reserved in WMCR*n*. This permits using the trace buffer as a second watchpoint monitor by simply ignoring the trace options.

The trace buffer provides great flexibility about when to start tracing, when to stop tracing, and what to trace. The trace mode field, TBCR0[MODE], indicates when to trace: on every valid cycle, on a watchpoint monitor event, or when all the programmed events in the TBCR are met. This permits a user to program the trace condition in the watchpoint monitor and to program a start or stop condition in the trace buffer control register. The user can also program the TBCR with the conditions in which to stop tracing: on an event, or when the buffer is full. TBCR0[IFSEL] specifies which interface transactions are being captured.

The trace buffer can be programmed to trace the dispatch bus from any of the following:

- Coherency module (MCM)
- Outbound host interface to the PCI controller
- Host interface to the DDR controller

Transactions come into the MCM, 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.

19.4.5.1 Traced Data Formats (as a Function of TBCR1[IFSEL])

Figure 19-24 shows the trace buffer entry format for an MCM dispatch (CMD) transaction that is specified when TBCR1[IFSEL] = 000.

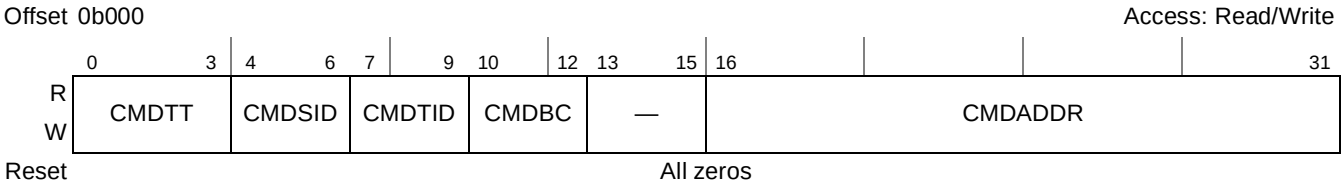


Figure 19-24. Coherency Module Dispatch (CMD) Trace Buffer Entry

Table 19-31 describes the fields of CMD trace buffer entries.

Table 19-31. CMD Trace Buffer Entry Field Descriptions (TBCR1[IFSEL] = 000)

Bits	Name	Function
0–4	CMDTT	Transaction type. Specifies the transaction type as shown in Table 19-14. For example, a value of zero indicates a write with local processor snoop condition.
5–9	CMDSID	Source ID. Identifies the source of the transaction as shown in Table 19-30. For example, a value of 010101 indicates that DMA is the transaction source.
10–13	CMDTID	Target ID. Identifies the target of the transaction as shown in Table 19-30. For example, a value of 010101 indicates that DMA is the transaction target.
14–18	CMDBC	Byte count. Range: 32 to 1 where a value of 0 indicates 32 bytes. 00000 = 32 bytes 00001 = 1 byte 00010 = 2 bytes . . . 11110 = 30 bytes 11111 = 31 bytes
19–31	—	Reserved
32–63	CMDADDR	Address bits 0–31

Figure 19-25 shows the trace buffer entry format for the DDR SDRAM interface, TBCR1[IFSEL] = 001.

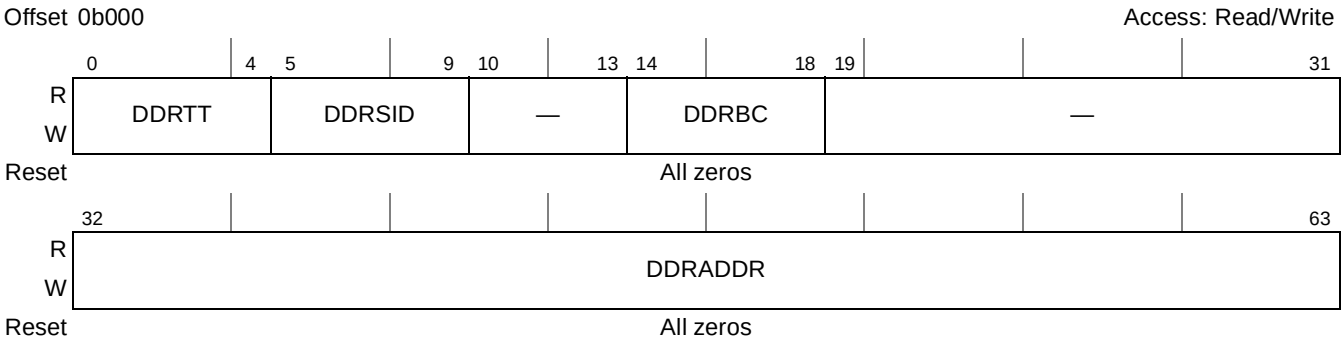


Figure 19-25. DDR Trace Buffer Entry

Table 19-32 describes the fields of DDR SDRAM trace buffer entries when TBCR1[IFSEL] = 001.

Table 19-32. DDR Trace Buffer Entry Field Descriptions (TBCR1[TRSEL] = 001)

Bits	Name	Function
0–4	DDRTT	Transaction type. Specifies the transaction type as shown in Table 19-14. For example, a value of all zeros maps to write.
5–9	DDRSID	Source ID. Specifies the source of the transaction as shown in Table 19-30. For example, a value of 010101 indicates that DMA is the transaction source, and so on.
10–13	—	Reserved
14–18	DDRBC	Byte count
19–31	—	Reserved
32–63	DDRADDR	Address bits 0–31

Figure 19-26 shows the PCI Express trace buffer entry format when TBCR1[IFSEL] = 100.

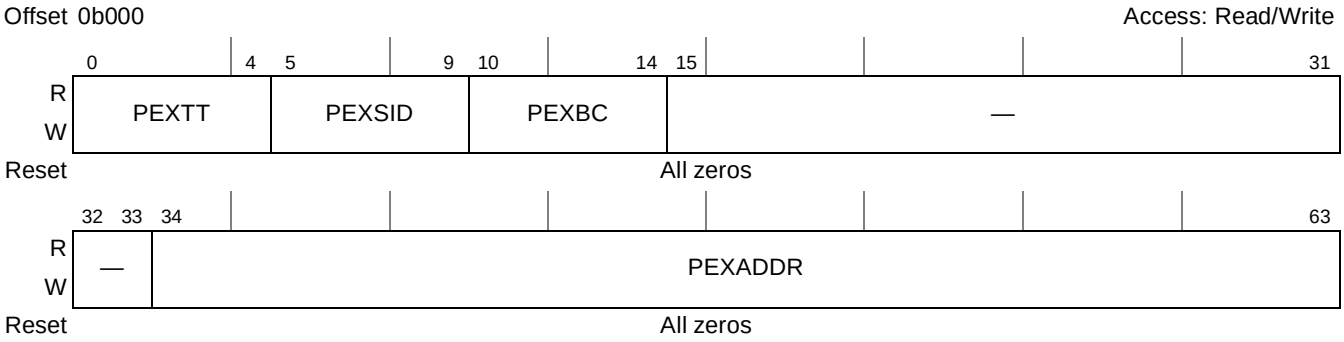


Figure 19-26. PCI Express Trace Buffer Entry

Table 19-33 describes the fields of PCI Express trace buffer entries when TBCR1[IFSEL] = 100.

Table 19-33. PCI Express Trace Buffer Entry Field Descriptions (TBCR1[IFSEL] = 100)

Bits	Name	Function
0–4	PEXTT	Transaction type. Specifies the transaction type as shown in Table 19-14. For example, a value of all zeros maps to write.
5–9	PEXSID	Source ID. Identifies the source of the transaction as shown in Table 19-30. For example, a value of 010101 identifies DMA as the transaction source. For responses, this corresponds to Requestor's ID's bus number bits 3–7.
10–14	PCIBC	Byte count. The size of the transaction. 00000 4 bytes 00001 8 bytes 00010 12 bytes ... 11111 256 bytes

Table 19-33. PCI Express Trace Buffer Entry Field Descriptions (TBCR1[IFSEL] = 100)

Bits	Name	Function
15–33	—	Reserved
34–63	PEXADDR	Address bits 37–2

19.5 Initialization

Configuring the appropriate control register must be the last step in the initialization sequence for either the watchpoint or trace buffer. That is, all required registers except the corresponding control register must be configured before any control register bits that enable watchpoint or trace events are set.

Appendix A
Complete List of Configuration, Control, and Status Registers

A.1 General Utilities

The general utilities registers are the functional block-specific registers that occupy the first 256 Kbytes of CCSR space (0x0_0000–0x3_FFFF). Each functional block is allocated a 4-Kbyte address range for its registers within the general utilities space.

A.1.1 Local Configuration Control

Table A-1. Local Configuration Control Registers

Local Configuration Control—Block Base Address 0x0_0000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0x000	CCSRBAR—Configuration, control, and status registers base address register	R/W	0x000F_F700	4.3.1.1.3/4-5
0x008	ALTCBAR—Alternate configuration base address register	R/W	0x0000_0000	4.3.1.2.1/4-6
0x010	ALTCAR—Alternate configuration attribute register	R/W	0x0000_0000	4.3.1.2.2/4-6
0x020	BPTR—Boot page translation register	R/W	0x0000_0000	4.3.1.3.1/4-8

A.1.2 Local Access Windows

Table A-2. Local Access Window Registers

Local Access Windows—Block Base Address 0x0_0000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0xC08	LAWBAR0—Local access window 0 base address register	R/W	0x0000_0000	2.2.1.1/2-4
0xC10	LAWAR0—Local access window 0 attribute register	R/W	0x0000_0000	2.2.1.1/2-4
0xC28	LAWBAR1—Local access window 1 base address register	R/W	0x0000_0000	2.2.1.1/2-4
0xC30	LAWAR1—Local access window 1 attribute register	R/W	0x0000_0000	2.2.1.1/2-4
0xC48	LAWBAR2—Local access window 2 base address register	R/W	0x0000_0000	2.2.1.1/2-4
0xC50	LAWAR2—Local access window 2 attribute register	R/W	0x0000_0000	2.2.1.1/2-4
0xC68	LAWBAR3—Local access window 3 base address register	R/W	0x0000_0000	2.2.1.1/2-4
0xC70	LAWAR3—Local access window 3 attribute register	R/W	0x0000_0000	2.2.1.1/2-4

Table A-2. Local Access Window Registers (continued)

Local Access Windows—Block Base Address 0x0_0000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0xC88	LAWBAR4—Local access window 4 base address register	R/W	0x0000_0000	2.2.1.1/2-4
0xC90	LAWAR4—Local access window 4 attribute register	R/W	0x0000_0000	2.2.1.1/2-4
0xCA8	LAWBAR5—Local access window 5 base address register	R/W	0x0000_0000	2.2.1.1/2-4
0xCB0	LAWAR5—Local access window 5 attribute register	R/W	0x0000_0000	2.2.1.1/2-4
0xCC8	LAWBAR6—Local access window 6 base address register	R/W	0x0000_0000	2.2.1.1/2-4
0xCD0	LAWAR6—Local access window 6 attribute register	R/W	0x0000_0000	2.2.1.1/2-4
0xCE8	LAWBAR7—Local access window 7 base address register	R/W	0x0000_0000	2.2.1.1/2-4
0xCF0	LAWAR7—Local access window 7 attribute register	R/W	0x0000_0000	2.2.1.1/2-4
0xD08	LAWBAR8—Local access window 8 base address register	R/W	0x0000_0000	2.2.1.1/2-4
0xD10	LAWAR8—Local access window 8 attribute register	R/W	0x0000_0000	2.2.1.1/2-4
0xD28	LAWBAR9—Local access window 9 base address register	R/W	0x0000_0000	2.2.1.1/2-4
0xD30	LAWAR9—Local access window 9 attribute register	R/W	0x0000_0000	2.2.1.1/2-4

A.1.3 MPX Coherency Module (MCM)

Table A-3. MCM Registers

MCM—Block Base Address 0x0_1000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0x000	MCM MPX address bus configuration register (ABCR)	R/W	0x0000_0003	7.4.1.1/7-5
0x008	MCM MPX data bus configuration register (DBCR)	R/W	0x0000_0003	7.4.1.2/7-6
0x010	MCM MPX port configuration register (PCR)	R/W	0x0*00_0000	7.4.1.3/7-6
0xBF8	MCM IP block revision register 1	R	0x0001_0000	—
0xBFC	MCM IP block revision register 2	R	0x0000_0000	—
0xE00	MCM error detect register (EDR)	w1c	0x0000_0000	7.4.1.4/7-7
0xE08	MCM error enable register (EER)	R/W	0x0000_0000	7.4.1.5/7-9
0xE0C	MCM error attributes capture register (EATR)	R	0x0000_0000	7.4.1.6/7-9
0xE10	MCM error low address capture register (ELADR)	R	0x0000_0000	7.4.1.7/7-11
0xE14	MCM error high address capture register (EHADR)	R	0x0000_0000	7.4.1.8/7-11

A.1.4 DDR Memory Controller 1

Table A-4. DDR Memory Controller 1 Registers

DDR Memory Controller 1—Block Base Address 0x0_2000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0x000	CS0_BNDS—Chip select 0 memory bounds	R/W	0x0000_0000	8.4.1.1/8-12
0x008	CS1_BNDS—Chip select 1 memory bounds	R/W	0x0000_0000	8.4.1.1/8-12
0x010	CS2_BNDS—Chip select 2 memory bounds	R/W	0x0000_0000	8.4.1.1/8-12
0x018	CS3_BNDS—Chip select 3 memory bounds	R/W	0x0000_0000	8.4.1.1/8-12
0x080	CS0_CONFIG—Chip select 0 configuration	R/W	0x0000_0000	8.4.1.2/8-13
0x084	CS1_CONFIG—Chip select 1 configuration	R/W	0x0000_0000	8.4.1.2/8-13
0x088	CS2_CONFIG—Chip select 2 configuration	R/W	0x0000_0000	8.4.1.2/8-13
0x08C	CS3_CONFIG—Chip select 3 configuration	R/W	0x0000_0000	8.4.1.2/8-13
0x100	TIMING_CFG_3—DDR SDRAM timing configuration 3	R/W	0x0000_0000	8.4.1.3/8-15
0x104	TIMING_CFG_0—DDR SDRAM timing configuration 0	R/W	0x0011_0105	8.4.1.4/8-16
0x108	TIMING_CFG_1—DDR SDRAM timing configuration 1	R/W	0x0000_0000	8.4.1.5/8-18
0x10C	TIMING_CFG_2—DDR SDRAM timing configuration 2	R/W	0x0000_0000	8.4.1.6/8-20
0x110	DDR_SDRAM_CFG—DDR SDRAM control configuration	R/W	0x0200_0000	8.4.1.7/8-22
0x114	DDR_SDRAM_CFG_2—DDR SDRAM control configuration 2	R/W	0x0000_0000	8.4.1.8/8-24
0x118	DDR_SDRAM_MODE—DDR SDRAM mode configuration	R/W	0x0000_0000	8.4.1.9/8-26
0x11C	DDR_SDRAM_MODE_2—DDR SDRAM mode configuration 2	R/W	0x0000_0000	8.4.1.10/8-27
0x120	DDR_SDRAM_MD_CNTL—DDR SDRAM mode control	R/W	0x0000_0000	8.4.1.11/8-27
0x124	DDR_SDRAM_INTERVAL—DDR SDRAM interval configuration	R/W	0x0000_0000	8.4.1.12/8-30
0x128	DDR_DATA_INIT—DDR SDRAM data initialization	R/W	0x0000_0000	8.4.1.13/8-30
0x130	DDR_SDRAM_CLK_CNTL—DDR SDRAM clock control	R/W	0x0200_0000	8.4.1.14/8-31
0x140–0x144	Reserved	—	—	—
0x148	DDR_INIT_ADDR—DDR training initialization address	R/W	0x0000_0000	8.4.1.15/8-31
0x14C	DDR_INIT_EXT_ADDRESS—DDR training initialization extended address	R/W	0x0000_0000	8.4.1.16/8-32
0x150–0xB1F	Reserved	—	—	—
0xB20	DDRDSR_1—DDR Debug Status Register 1	R	0x0000_0000	8.4.1.17/8-33
0xB24	DDRDSR_2—DDR Debug Status Register 2	R	0x0000_0000	8.4.1.18/8-33
0xB28	DDRCDR_1—DDR Control Driver Register 1	R/W	0x0000_0000	8.4.1.19/8-34
0xB2C	DDRCDR_2—DDR Control Driver Register 2	R/W	0x0000_0000	8.4.1.20/8-35
0xB30–0xBF7	Reserved	—	—	—
0xBF8	DDR_IP_REV1—DDR IP block revision 1	R	0x*nnnn_nnnn*[1]	8.4.1.21/8-36

Table A-4. DDR Memory Controller 1 Registers (continued)

DDR Memory Controller 1—Block Base Address 0x0_2000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0xBFC	DDR_IP_REV2—DDR IP block revision 2	R	0x00*nn*_00*nn*[1]	8.4.1.22/8-37
0xE00	DATA_ERR_INJECT_HI—Memory data path error injection mask high	R/W	0x0000_0000	8.4.1.23/8-37
0xE04	DATA_ERR_INJECT_LO—Memory data path error injection mask low	R/W	0x0000_0000	8.4.1.24/8-38
0xE08	ECC_ERR_INJECT—Memory data path error injection mask ECC	R/W	0x0000_0000	8.4.1.25/8-38
0xE20	CAPTURE_DATA_HI—Memory data path read capture high	R/W	0x0000_0000	8.4.1.26/8-39
0xE24	CAPTURE_DATA_LO—Memory data path read capture low	R/W	0x0000_0000	8.4.1.27/8-39
0xE28	CAPTURE_ECC—Memory data path read capture ECC	R/W	0x0000_0000	8.4.1.28/8-40
0xE40	ERR_DETECT—Memory error detect	w1c	0x0000_0000	8.4.1.29/8-40
0xE44	ERR_DISABLE—Memory error disable	R/W	0x0000_0000	8.4.1.30/8-41
0xE48	ERR_INT_EN—Memory error interrupt enable	R/W	0x0000_0000	8.4.1.31/8-42
0xE4C	CAPTURE_ATTRIBUTES—Memory error attributes capture	R/W	0x0000_0000	8.4.1.32/8-43
0xE50	CAPTURE_ADDRESS—Memory error address capture	R/W	0x0000_0000	8.4.1.33/8-44
0xE54	CAPTURE_EXT_ADDRESS—Memory error extended address capture	R/W	0x0000_0000	8.4.1.34/8-44
0xE58	ERR_SBE—Single-Bit ECC memory error management	R/W	0x0000_0000	8.4.1.35/8-45

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

A.1.5 I²C Controllers

Table A-5. I²C Controller 1 & 2 Registers

I²C Controller 1—Block Base Address 0x0_3000 I²C Controller 2—Block Base Address 0x0_3100				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
I²C1 Registers				
0x000	I2CADR—I²C address register	R/W	0x00	10.3.1.1/10-5
0x004	I2CFDR—I²C frequency divider register	R/W	0x00	10.3.1.2/10-6
0x008	I2CCR—I²C control register	Mixed	0x00	10.3.1.3/10-7
0x00C	I2CSR—I²C status register	Mixed	0x81	10.3.1.4/10-9
0x010	I2CDR—I²C data register	R/W	0x00	10.3.1.5/10-10
0x014	I2CDFSRR—I²C digital filter sampling rate register	R/W	0x10	10.3.1.6/10-11
I²C2 Registers				
0x100–0x114	I²C2 Registers[1]			

[1] I^2C2 has the same memory-mapped registers that are described for I^2C1 from 0x000 to 0x014, except the offsets range from 0x100 to 0x114.

A.1.6 DUART

Table A-6. DUART Registers

DUART—Block Base Address 0x0_4000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
UART0 Registers				
0x500	URBR—ULCR[DLAB] = 0 UART0 receiver buffer register	R	0x00	11.3.1.1/11-5
0x500	UTHR—ULCR[DLAB] = 0 UART0 transmitter holding register	W	0x00	11.3.1.2/11-5
0x500	UDLB—ULCR[DLAB] = 1 UART0 divisor least significant byte register	R/W	0x00	11.3.1.3/11-6
0x501	UIER—ULCR[DLAB] = 0 UART0 interrupt enable register	R/W	0x00	11.3.1.4/11-8
0x501	UDMB—ULCR[DLAB] = 1 UART0 divisor most significant byte register	R/W	0x00	11.3.1.3/11-6
0x502	UIIR—ULCR[DLAB] = 0 UART0 interrupt ID register	R	0x01	11.3.1.5/11-8
0x502	UFCR—ULCR[DLAB] = 0 UART0 FIFO control register	W	0x00	11.3.1.6/11-10
0x502	UAFR—ULCR[DLAB] = 1 UART0 alternate function register	R/W	0x00	11.3.1.12/11-16
0x503	ULCR—ULCR[DLAB] = x UART0 line control register	R/W	0x00	11.3.1.7/11-11
0x504	UMCR—ULCR[DLAB] = x UART0 modem control register	R/W	0x00	11.3.1.8/11-13
0x505	ULSR—ULCR[DLAB] = x UART0 line status register	R	0x60	11.3.1.9/11-14
0x506	UMSR—ULCR[DLAB] = x UART0 modem status register	R	0x00	11.3.1.10/11-15
0x507	USCR—ULCR[DLAB] = x UART0 scratch register	R/W	0x00	11.3.1.11/11-16
0x510	UDSR—ULCR[DLAB] = x UART0 DMA status register	R	0x01	11.3.1.13/11-17
UART1 Registers				
0x600–0x610	UART1 Registers[1]			

[1] UART1 has the same memory-mapped registers that are described for UART0 from 0x500 to 0x510, except the offsets range from 0x600 to 0x610.

A.1.7 Local Bus Controller

Table A-7. Local Bus Controller Registers

Local Bus Controller—Block Base Address 0x0_5000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0x000	BR0—Base register 0	R/W	0x0000_*nn*01[1]	12.3.1.1/12-10
0x008	BR1—Base register 1		0x0000_0000	
0x010	BR2—Base register 2			
0x018	BR3—Base register 3			
0x020	BR4—Base register 4			
0x028	BR5—Base register 5			
0x030	BR6—Base register 6			
0x038	BR7—Base register 7			
0x004	OR0—Options register 0	R/W	0x0000_0FF7	12.3.1.2/12-12
0x00C	OR1—Options register 1		0x0000_0000	
0x014	OR2—Options register 2			
0x01C	OR3—Options register 3			
0x024	OR4—Options register 4			
0x02C	OR5—Options register 5			
0x034	OR6—Options register 6			
0x03C	OR7—Options register 7			
0x068	MAR—UPM address register	R/W	0x0000_0000	12.3.1.3/12-17
0x070	MAMR—UPMA mode register	R/W	0x0000_0000	12.3.1.4/12-17
0x074	MBMR—UPMB mode register	R/W	0x0000_0000	12.3.1.4/12-17
0x078	MCMR—UPMC mode register	R/W	0x0000_0000	12.3.1.4/12-17
0x084	MRTPR—Memory refresh timer prescaler register	R/W	0x0000_0000	12.3.1.5/12-20
0x088	MDR—UPM data register	R/W	0x0000_0000	12.3.1.6/12-20
0x094	LSDMR—SDRAM mode register	R/W	0x0000_0000	12.3.1.7/12-21
0x0A0	LURT—UPM refresh timer	R/W	0x0000_0000	12.3.1.8/12-23
0x0A4	LSRT—SDRAM refresh timer	R/W	0x0000_0000	12.3.1.9/12-23
0x0B0	LTESR—Transfer error status register	w1c	0x0000_0000	12.3.1.10/12-24
0x0B4	LTEDR—Transfer error disable register	R/W	0x0000_0000	12.3.1.11/12-25
0x0B8	LTEIR—Transfer error interrupt register	R/W	0x0000_0000	12.3.1.12/12-26
0x0BC	LTEATR—Transfer error attributes register	R/W	0x0000_0000	12.3.1.13/12-27
0x0C0	LTEAR—Transfer error address register	R/W	0x0000_0000	12.3.1.14/12-28

Table A-7. Local Bus Controller Registers (continued)

Local Bus Controller—Block Base Address 0x0_5000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0x0D0	LBCR—Configuration register	R/W	0x0000_0000	12.3.1.15/12-29
0x0D4	LCRR—Clock ratio register	R/W	0x8000_0008	12.3.1.16/12-30

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

A.1.8 DDR Memory Controller 2

Table A-8. DDR Memory Controller 2 Registers

DDR Memory Controller 2—Block Base Address 0x0_6000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0x000	CS0_BNDS—Chip select 0 memory bounds	R/W	0x0000_0000	8.4.1.1/8-12
0x008	CS1_BNDS—Chip select 1 memory bounds	R/W	0x0000_0000	8.4.1.1/8-12
0x010	CS2_BNDS—Chip select 2 memory bounds	R/W	0x0000_0000	8.4.1.1/8-12
0x018	CS3_BNDS—Chip select 3 memory bounds	R/W	0x0000_0000	8.4.1.1/8-12
0x080	CS0_CONFIG—Chip select 0 configuration	R/W	0x0000_0000	8.4.1.2/8-13
0x084	CS1_CONFIG—Chip select 1 configuration	R/W	0x0000_0000	8.4.1.2/8-13
0x088	CS2_CONFIG—Chip select 2 configuration	R/W	0x0000_0000	8.4.1.2/8-13
0x08C	CS3_CONFIG—Chip select 3 configuration	R/W	0x0000_0000	8.4.1.2/8-13
0x100	TIMING_CFG_3—DDR SDRAM timing configuration 3	R/W	0x0000_0000	8.4.1.3/8-15
0x104	TIMING_CFG_0—DDR SDRAM timing configuration 0	R/W	0x0011_0105	8.4.1.4/8-16
0x108	TIMING_CFG_1—DDR SDRAM timing configuration 1	R/W	0x0000_0000	8.4.1.5/8-18
0x10C	TIMING_CFG_2—DDR SDRAM timing configuration 2	R/W	0x0000_0000	8.4.1.6/8-20
0x110	DDR_SDRAM_CFG—DDR SDRAM control configuration	R/W	0x0200_0000	8.4.1.7/8-22
0x114	DDR_SDRAM_CFG_2—DDR SDRAM control configuration 2	R/W	0x0000_0000	8.4.1.8/8-24
0x118	DDR_SDRAM_MODE—DDR SDRAM mode configuration	R/W	0x0000_0000	8.4.1.9/8-26
0x11C	DDR_SDRAM_MODE_2—DDR SDRAM mode configuration 2	R/W	0x0000_0000	8.4.1.10/8-27
0x120	DDR_SDRAM_MD_CNTL—DDR SDRAM mode control	R/W	0x0000_0000	8.4.1.11/8-27
0x124	DDR_SDRAM_INTERVAL—DDR SDRAM interval configuration	R/W	0x0000_0000	8.4.1.12/8-30
0x128	DDR_DATA_INIT—DDR SDRAM data initialization	R/W	0x0000_0000	8.4.1.13/8-30
0x130	DDR_SDRAM_CLK_CNTL—DDR SDRAM clock control	R/W	0x0200_0000	8.4.1.14/8-31
0x140–0x144	Reserved	—	—	—
0x148	DDR_INIT_ADDR—DDR training initialization address	R/W	0x0000_0000	8.4.1.15/8-31
0x14C	DDR_INIT_EXT_ADDRESS—DDR training initialization extended address	R/W	0x0000_0000	8.4.1.16/8-32

Table A-8. DDR Memory Controller 2 Registers (continued)

DDR Memory Controller 2—Block Base Address 0x0_6000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0x150–0xB1F	Reserved	—	—	—
0xB20	DDRDSR_1—DDR Debug Status Register 1	R	0x0000_0000	8.4.1.17/8-33
0xB24	DDRDSR_2—DDR Debug Status Register 2	R	0x0000_0000	8.4.1.18/8-33
0xB28	DDRCDR_1—DDR Control Driver Register 1	R/W	0x0000_0000	8.4.1.19/8-34
0xB2C	DDRCDR_2—DDR Control Driver Register 2	R/W	0x0000_0000	8.4.1.20/8-35
0xB30–0xBF7	Reserved	—	—	—
0xBF8	DDR_IP_REV1—DDR IP block revision 1	R	0x*nnnn_nnnn*[1]	8.4.1.21/8-36
0xBFC	DDR_IP_REV2—DDR IP block revision 2	R	0x00*nn*_00*nn*[1]	8.4.1.22/8-37
0xE00	DATA_ERR_INJECT_HI—Memory data path error injection mask high	R/W	0x0000_0000	8.4.1.23/8-37
0xE04	DATA_ERR_INJECT_LO—Memory data path error injection mask low	R/W	0x0000_0000	8.4.1.24/8-38
0xE08	ECC_ERR_INJECT—Memory data path error injection mask ECC	R/W	0x0000_0000	8.4.1.25/8-38
0xE20	CAPTURE_DATA_HI—Memory data path read capture high	R/W	0x0000_0000	8.4.1.26/8-39
0xE24	CAPTURE_DATA_LO—Memory data path read capture low	R/W	0x0000_0000	8.4.1.27/8-39
0xE28	CAPTURE_ECC—Memory data path read capture ECC	R/W	0x0000_0000	8.4.1.28/8-40
0xE40	ERR_DETECT—Memory error detect	w1c	0x0000_0000	8.4.1.29/8-40
0xE44	ERR_DISABLE—Memory error disable	R/W	0x0000_0000	8.4.1.30/8-41
0xE48	ERR_INT_EN—Memory error interrupt enable	R/W	0x0000_0000	8.4.1.31/8-42
0xE4C	CAPTURE_ATTRIBUTES—Memory error attributes capture	R/W	0x0000_0000	8.4.1.32/8-43
0xE50	CAPTURE_ADDRESS—Memory error address capture	R/W	0x0000_0000	8.4.1.33/8-44
0xE54	CAPTURE_EXT_ADDRESS—Memory error extended address capture	R/W	0x0000_0000	8.4.1.34/8-44
0xE58	ERR_SBE—Single-Bit ECC memory error management	R/W	0x0000_0000	8.4.1.35/8-45

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

A.1.9 PCI Express Controllers

Table A-9. PCI Express Controller 1 & 2 Registers

PCI Express Controller 1 —Block Base Address 0x0_8000 PCI Express Controller 2—Block Base Address 0x0_9000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
PCI Express Controller 1 Memory-Mapped Registers				
PCI Express Configuration Access Registers				
0x000	PEX_CONFIG_ADDR—PCI Express configuration address register	R/W	0x0000_0000	16.3.2.1/16-9

Table A-9. PCI Express Controller 1 & 2 Registers (continued)

PCI Express Controller 1 —Block Base Address 0x0_8000 PCI Express Controller 2—Block Base Address 0x0_9000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0x004	PEX_CONFIG_DATA—PCI Express configuration data register	R/W	0x0000_0000	16.3.2.2/16-10
0x008	Reserved	—	—	—
0x00C	PEX_OTB_CPL_TOR—PCI Express outbound completion timeout register	R/W	0x0010_FFFF	16.3.2.3/16-11
0x010	PEX_CONF_RTY_TOR—PCI Express configuration retry timeout register	R/W	0x0400_FFFF	16.3.2.4/16-11
0x014	PEX_CONFIG—PCI Express configuration register	R/W	0x0000_0000	16.3.2.5/16-12
0x018–0x01C	Reserved	—	—	—
PCI Express Power Management Event & Message Registers				
0x020	PEX_PME_MES_DR—PCI Express PME & message detect register	w1c	0x0000_0000	16.3.3.1/16-13
0x024	PEX_PME_MES_DISR—PCI Express PME & message disable register	R/W	0x0000_0000	16.3.3.2/16-15
0x028	PEX_PME_MES_IER—PCI Express PME & message interrupt enable register	R/W	0x0000_0000	16.3.3.3/16-16
0x02C	PEX_PMCR—PCI Express power management command register	R/W	0x0000_0000	16.3.3.4/16-18
0x030–0xBF4	Reserved	—	—	—
PCI Express IP Block Revision Registers				
0xBF8	IP block revision register 1 (PEX_IP_BLK_REV1)	R	0x0208_0100	16.3.4.1/16-18
0xBFC	IP block revision register 2 (PEX_IP_BLK_REV2)	R	0x0000_0000	16.3.4.2/16-19
PCI Express ATMU Registers				
Outbound Window 0 (Default)				
0xC00	PEXOTAR0—PCI Express outbound translation address register 0 (default)	R/W	0x0000_0000	16.3.5.1.1/16-20
0xC04	PEXOTEAR0—PCI Express outbound translation extended address register 0 (default)	R/W	0x0000_0000	16.3.5.1.2/16-21
0xC08–0xC0C	Reserved	—	—	—
0xC10	PEXOWAR0—PCI Express outbound window attributes register 0 (default)	Mixed	0x8004_4023	16.3.5.1.4/16-22
0xC14–0xC1C	Reserved	—	—	—
Outbound Window 1				
0xC20	PEXOTAR1—PCI Express outbound translation address register 1	R/W	0x0000_0000	16.3.5.1.1/16-20

Table A-9. PCI Express Controller 1 & 2 Registers (continued)

PCI Express Controller 1 —Block Base Address 0x0_8000 PCI Express Controller 2—Block Base Address 0x0_9000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0xC24	PEXOTEAR1—PCI Express outbound translation extended address register 1	R/W	0x0000_0000	16.3.5.1.2/16-21
0xC28	PEXOWBAR1—PCI Express outbound window base address register 1	R/W	0x0000_0000	16.3.5.1.3/16-21
0xC2C	Reserved	—	—	—
0xC30	PEXOWAR1—PCI Express outbound window attributes register 1	R/W	0x0004_4023	16.3.5.1.4/16-22
0xC34–0xC3C	Reserved	—	—	—
Outbound Window 2				
0xC40	PEXOTAR2—PCI Express outbound translation address register 2	R/W	0x0000_0000	16.3.5.1.1/16-20
0xC44	PEXOTEAR2—PCI Express outbound translation extended address register 2	R/W	0x0000_0000	16.3.5.1.2/16-21
0xC48	PEXOWBAR2—PCI Express outbound window base address register 2	R/W	0x0000_0000	16.3.5.1.3/16-21
0xC4C	Reserved	—	—	—
0xC50	PEXOWAR2—PCI Express outbound window attributes register 2	R/W	0x0004_4023	16.3.5.1.4/16-22
0xC54–0xC5C	Reserved	—	—	—
Outbound Window 3				
0xC60	PEXOTAR3—PCI Express outbound translation address register 3	R/W	0x0000_0000[1]	16.3.5.1.1/16-20
0xC64	PEXOTEAR3—PCI Express outbound translation extended address register 3	R/W	0x0000_0000	16.3.5.1.2/16-21
0xC68	PEXOWBAR3—PCI Express outbound window base address register 3	R/W	0x0000_0000[2]	16.3.5.1.3/16-21
0xC6C	Reserved	—	—	—
0xC70	PEXOWAR3—PCI Express outbound window attributes register 3	R/W	0x0004_4023[3]	16.3.5.1.4/16-22
0xC74–0xC7C	Reserved	—	—	—
Outbound Window 4				
0xC80	PEXOTAR4—PCI Express outbound translation address register 4	R/W	0x0000_0000	16.3.5.1.1/16-20
0xC84	PEXOTEAR4—PCI Express outbound translation extended address register 4	R/W	0x0000_0000	16.3.5.1.2/16-21
0xC88	PEXOWBAR4—PCI Express outbound window base address register 4	R/W	0x0000_0000	16.3.5.1.3/16-21

Table A-9. PCI Express Controller 1 & 2 Registers (continued)

PCI Express Controller 1 —Block Base Address 0x0_8000 PCI Express Controller 2—Block Base Address 0x0_9000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0xC8C	Reserved	—	—	—
0xC90	PEXOWAR4—PCI Express outbound window attributes register 4	R/W	0x0004_4023	16.3.5.1.4/16-22
0xC94–0xC9C	Reserved	—	—	—
0xD14–0xD9C	Reserved	—	—	—
Inbound Window 3				
0xDA0	PEXITAR3—PCI Express inbound translation address register 3	R/W	0x0000_0000	16.3.5.2.3/16-25
0xDA4	Reserved	—	—	—
0xDA8	PEXIWBAR3—PCI Express inbound window base address register 3	R/W	0x0000_0000	16.3.5.2.4/16-26
0xDAC	PEXIWBEAR3—PCI Express inbound window base extended address register 3	R/W	0x0000_0000	16.3.5.2.5/16-27
0xDB0	PEXIWAR3—PCI Express inbound window attributes register 3	R/W	0x20F4_4023	16.3.5.2.6/16-27
0xDB4–0xDBC	Reserved	—	—	—
Inbound Window 2				
0xDC0	PEXITAR2—PCI Express inbound translation address register 2	R/W	0x0000_0000	16.3.5.2.3/16-25
0xDC4	Reserved	—	—	—
0xDC8	PEXIWBAR2—PCI Express inbound window base address register 2	R/W	0x0000_0000	16.3.5.2.4/16-26
0xDCC	PEXIWBEAR2—PCI Express inbound window base extended address register 2	R/W	0x0000_0000	16.3.5.2.5/16-27
0xDD0	PEXIWAR2—PCI Express inbound window attributes register 2	R/W	0x20F4_4023	16.3.5.2.6/16-27
0xDD4–0xDDC	Reserved	—	—	—
Inbound Window 1				
0xDE0	PEXITAR1—PCI Express inbound translation address register 1	R/W	0x0000_0000	16.3.5.2.3/16-25
0xDE4	Reserved	—	—	—
0xDE8	PEXIWBAR1—PCI Express inbound window base address register 1	R/W	0x0000_0000	16.3.5.2.4/16-26
0xDEC	Reserved	—	—	—
0xDF0	PEXIWAR1—PCI Express inbound window attributes register 1	R/W	0x20F4_4023	16.3.5.2.6/16-27

Table A-9. PCI Express Controller 1 & 2 Registers (continued)

PCI Express Controller 1 —Block Base Address 0x0_8000 PCI Express Controller 2—Block Base Address 0x0_9000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0xDF4–0xDFC	Reserved	—	—	—
PCI Express Error Management Registers				
0xE00	PEX_ERR_DR—PCI Express error detect register	w1c	0x0000_0000	16.3.6.1/16-29
0xE04	Reserved	—	—	—
0xE08	PEX_ERR_EN—PCI Express error interrupt enable register	R/W	0x0000_0000	16.3.6.2/16-32
0xE0C	Reserved	—	—	—
0xE10	PEX_ERR_DISR—PCI Express error disable register	R/W	0x0000_0000	16.3.6.3/16-34
0xE14–0xE1C	Reserved	—	—	—
0xE20	PEX_ERR_CAP_STAT—PCI Express error capture status register	Mixed	0x0000_0000	16.3.6.4/16-35
0xE24	Reserved	—	—	—
0xE28	PEX_ERR_CAP_R0—PCI Express error capture register 0	R/W	0x0000_0000	16.3.6.5/16-36
0xE2C	PEX_ERR_CAP_R1—PCI Express error capture register 1	R/W	0x0000_0000	16.3.6.6/16-38
0xE30	PEX_ERR_CAP_R2—PCI Express error capture register 2	R/W	0x0000_0000	16.3.6.7/16-39
0xE34	PEX_ERR_CAP_R3—PCI Express error capture register 3	R/W	0x0000_0000	16.3.6.8/16-41
0xE38–0xFFC	Reserved	—	—	—
PCI Express Controller 2 Memory-Mapped Registers				
0x000–0xFFC	PCI Express Controller 2 registers **Note:** All registers defined for PCI Express Controller 1 are also defined for PCI Express Controller 2; the offsets of PCI Express Controller 2 registers are the same except they have a different block base address.			

1 If the device is configured to use the alternate boot vector (cfg_boot_vec = 0), the reset value for PCI controller 1 PEXOTAR3 is 0x000F_FFF0.

2 If the device is configured to use the alternate boot vector (cfg_boot_vec = 0), the reset value for PCI controller 1 PEXOWBAR3 is 0x000F_FF00.

3 If the device is configured to use the alternate boot vector (cfg_boot_vec = 0), the reset value for PCI controller 1 PEXOWAR3 is 0x8004_400F.

A.1.10 DMA Controller

Table A-10. DMA Controller Registers

DMA Controller —Block Base Address 0x2_1000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0x100	MR0—DMA 0 mode register	R/W	0x0000_0000	14.3.1.1/14-10
0x104	SR0—DMA 0 status register	Mixed	0x0000_0000	14.3.1.2/14-12

Table A-10. DMA Controller Registers (continued)

DMA Controller —Block Base Address 0x2_1000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0x108	ECLNDAR0—DMA 0 current link descriptor extended address register	R/W	0x0000_0000	14.3.1.3/14-14
0x10C	CLNDAR0—DMA 0 current link descriptor address register	R/W	0x0000_0000	14.3.1.3/14-14
0x110	SATR0—DMA 0 source attributes register	R/W	0x0000_0000	14.3.1.4/14-16
0x114	SAR0—DMA 0 source address register	R/W	0x0000_0000	14.3.1.5/14-17
0x118	DATR0—DMA 0 destination attributes register	R/W	0x0000_0000	14.3.1.6/14-19
0x11C	DAR0—DMA 0 destination address register	R/W	0x0000_0000	14.3.1.7/14-20
0x120	BCR0—DMA 0 byte count register	R/W	0x0000_0000	14.3.1.8/14-22
0x124	ENLNDAR0—DMA 0 next link descriptor extended address register	R/W	0x0000_0000	14.3.1.9/14-22
0x128	NLNDAR0—DMA 0 next link descriptor address register	R/W	0x0000_0000	14.3.1.9/14-22
0x130	ECLSDAR0—DMA 0 current list descriptor extended address register	R/W	0x0000_0000	14.3.1.10/14-23
0x134	CLSDAR0—DMA 0 current list descriptor address register	R/W	0x0000_0000	14.3.1.10/14-23
0x138	ENLSDAR0—DMA 0 next list descriptor extended address register	R/W	0x0000_0000	14.3.1.11/14-25
0x13C	NLSDAR0—DMA 0 next list descriptor address register	Mixed	0x0000_0000	14.3.1.11/14-25
0x140	SSR0—DMA 0 source stride register	R/W	0x0000_0000	14.3.1.12/14-26
0x144	DSR0—DMA 0 destination stride register	R/W	0x0000_0000	14.3.1.13/14-26
0x148–0x17C	Reserved	—	—	—
0x180	MR1—DMA 1 mode register	R/W	0x0000_0000	14.3.1.1/14-10
0x184	SR1—DMA 1 status register	Mixed	0x0000_0000	14.3.1.2/14-12
0x188	ECLNDAR1—DMA 1 current link descriptor extended address register	R/W	0x0000_0000	14.3.1.3/14-14
0x18C	CLNDAR1—DMA 1 current link descriptor address register	R/W	0x0000_0000	14.3.1.3/14-14
0x190	SATR1—DMA 1 source attributes register	R/W	0x0000_0000	14.3.1.4/14-16
0x194	SAR1—DMA 1 source address register	R/W	0x0000_0000	14.3.1.5/14-17
0x198	DATR1—DMA 1 destination attributes register	R/W	0x0000_0000	14.3.1.6/14-19
0x19C	DAR1—DMA 1 destination address register	R/W	0x0000_0000	14.3.1.7/14-20
0x1A0	BCR1—DMA 1 byte count register	R/W	0x0000_0000	14.3.1.8/14-22
0x1A4	ENLNDAR1—DMA 1 next link descriptor extended address register	R/W	0x0000_0000	14.3.1.9/14-22
0x1A8	NLNDAR1—DMA 1 next link descriptor address register	R/W	0x0000_0000	14.3.1.9/14-22
0x1B0	ECLSDAR1—DMA 1 current list descriptor extended address register	R/W	0x0000_0000	14.3.1.10/14-23

Table A-10. DMA Controller Registers (continued)

DMA Controller —Block Base Address 0x2_1000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0x1B4	CLSDAR1—DMA 1 current list descriptor address register	R/W	0x0000_0000	14.3.1.10/14-23
0x1B8	ENLSDAR1—DMA 1 next list descriptor extended address register	R/W	0x0000_0000	14.3.1.11/14-25
0x1BC	NLSDAR1—DMA 1 next list descriptor address register	R/W	0x0000_0000	14.3.1.11/14-25
0x1C0	SSR1—DMA 1 source stride register	R/W	0x0000_0000	14.3.1.12/14-26
0x1C4	DSR1—DMA 1 destination stride register	R/W	0x0000_0000	14.3.1.13/14-26
0x1C8–0x1FC	Reserved	—	—	—
0x200	MR2—DMA 2 mode register	R/W	0x0000_0000	14.3.1.1/14-10
0x204	SR2—DMA 2 status register	Mixed	0x0000_0000	14.3.1.2/14-12
0x208	ECLNDAR2—DMA 2 current link descriptor extended address register	R/W	0x0000_0000	14.3.1.3/14-14
0x20C	CLNDAR2—DMA 2 current link descriptor address register	R/W	0x0000_0000	14.3.1.3/14-14
0x210	SATR2—DMA 2 source attributes register	R/W	0x0000_0000	14.3.1.4/14-16
0x214	SAR2—DMA 2 source address register	R/W	0x0000_0000	14.3.1.5/14-17
0x218	DATR2—DMA 2 destination attributes register	R/W	0x0000_0000	14.3.1.6/14-19
0x21C	DAR2—DMA 2 destination address register	R/W	0x0000_0000	14.3.1.7/14-20
0x220	BCR2—DMA 2 byte count register	R/W	0x0000_0000	14.3.1.8/14-22
0x224	ENLNDAR2—DMA 2 next link descriptor extended address register	R/W	0x0000_0000	14.3.1.9/14-22
0x228	NLNDAR2—DMA 2 next link descriptor address register	R/W	0x0000_0000	14.3.1.9/14-22
0x230	ECLSDAR2—DMA 2 current list descriptor extended address register	R/W	0x0000_0000	14.3.1.10/14-23
0x234	CLSDAR2—DMA 2 current list descriptor address register	R/W	0x0000_0000	14.3.1.10/14-23
0x238	ENLSDAR2—DMA 2 next list descriptor extended address register	R/W	0x0000_0000	14.3.1.11/14-25
0x23C	NLSDAR2—DMA 2 next list descriptor address register	R/W	0x0000_0000	14.3.1.11/14-25
0x240	SSR2—DMA 2 source stride register	R/W	0x0000_0000	14.3.1.12/14-26
0x244	DSR2—DMA 2 destination stride register	R/W	0x0000_0000	14.3.1.13/14-26
0x248–0x27C	Reserved	—	—	—
0x280	MR3—DMA 3 mode register	R/W	0x0000_0000	14.3.1.1/14-10
0x284	SR3—DMA 3 status register	Mixed	0x0000_0000	14.3.1.2/14-12
0x288	ECLNDAR3—DMA 3 current link descriptor extended address register	R/W	0x0000_0000	14.3.1.3/14-14
0x28C	CLNDAR3—DMA 3 current link descriptor address register	R/W	0x0000_0000	14.3.1.3/14-14

Table A-10. DMA Controller Registers (continued)

DMA Controller —Block Base Address 0x2_1000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0x290	SATR3—DMA 3 source attributes register	R/W	0x0000_0000	14.3.1.4/14-16
0x294	SAR3—DMA 3 source address register	R/W	0x0000_0000	14.3.1.5/14-17
0x298	DATR3—DMA 3 destination attributes register	R/W	0x0000_0000	14.3.1.6/14-19
0x29C	DAR3—DMA 3 destination address register	R/W	0x0000_0000	14.3.1.7/14-20
0x2A0	BCR3—DMA 3 byte count register	R/W	0x0000_0000	14.3.1.8/14-22
0x2A4	ENLNDAR3—DMA 3 next link descriptor extended address register	R/W	0x0000_0000	14.3.1.9/14-22
0x2A8	NLNDAR3—DMA 3 next link descriptor address register	R/W	0x0000_0000	14.3.1.9/14-22
0x2B0	ECLSDAR3—DMA 3 current list descriptor extended address register	R/W	0x0000_0000	14.3.1.10/14-23
0x2B4	CLSDAR3—DMA 3 current list descriptor address register	R/W	0x0000_0000	14.3.1.10/14-23
0x2B8	ENLSDAR3—DMA 3 next list descriptor extended address register	R/W	0x0000_0000	14.3.1.11/14-25
0x2BC	NLSDAR3—DMA 3 next list descriptor address register	R/W	0x0000_0000	14.3.1.11/14-25
0x2C0	SSR3—DMA 3 source stride register	R/W	0x0000_0000	14.3.1.12/14-26
0x2C4	DSR3—DMA 3 destination stride register	R/W	0x0000_0000	14.3.1.13/14-26
0x2C8–0x2FC	Reserved	—	—	—
0x300	DGSR—DMA general status register	R	0x0000_0000	14.3.1.14/14-27

A.1.11 eTSEC Controllers

Table A-11. eTSEC Controllers 1, 2, 3, & 4 Registers

eTSEC1—Block Base Address 0x2_4000 eTSEC2—Block Base Address 0x2_5000 eTSEC3—Block Base Address 0x2_6000 eTSEC4—Block Base Address 0x2_7000				
Offset	**Register**[1]	**Access** [2]	**Reset**	**Section/Page**
eTSEC1				
eTSEC General Control and Status Registers				
0x000	TSEC_ID*—Controller ID register	R	0x0124_0000	13.5.3.1.1/13-23
0x004	TSEC_ID2*—Controller ID register	R	0x0030_00F0	13.5.3.1.2/13-24
0x008–0x00C	Reserved	—	—	—
0x010	IEVENT—Interrupt event register	w1c	0x0000_0000	13.5.3.1.3/13-25
0x014	IMASK—Interrupt mask register	R/W	0x0000_0000	13.5.3.1.4/13-28

Table A-11. eTSEC Controllers 1, 2, 3, & 4 Registers (continued)

eTSEC1—Block Base Address 0x2_4000 eTSEC2—Block Base Address 0x2_5000 eTSEC3—Block Base Address 0x2_6000 eTSEC4—Block Base Address 0x2_7000				
Offset	**Register[1]**	**Access [2]**	**Reset**	**Section/Page**
0x018	EDIS—Error disabled register	R/W	0x0000_0000	13.5.3.1.5/13-30
0x01C	Reserved	—	—	—
0x020	ECNTRL—Ethernet control register	R/W	0x0000_0000	13.5.3.1.6/13-32
0x024	Reserved	—	—	—
0x028	PTV—Pause time value register	R/W	0x0000_0000	13.5.3.1.7/13-34
0x02C	DMACTRL—DMA control register	R/W	0x0000_0000	13.5.3.1.8/13-35
0x030	TBIPA—TBI PHY address register	R/W	0x0000_0000	13.5.3.1.9/13-36
0x034– 0x0FC	Reserved	—	—	—
eTSEC Transmit Control and Status Registers				
0x100	TCTRL—Transmit control register	R/W	0x0000_0000	13.5.3.2.1/13-37
0x104	TSTAT—Transmit status register	w1c	0x0000_0000	13.5.3.2.2/13-39
0x108	DFVLAN*—Default VLAN control word	R/W	0x8100_0000	13.5.3.2.3/13-43
0x10C	Reserved	—	—	—
0x110	TXIC—Transmit interrupt coalescing register	R/W	0x0000_0000	13.5.3.2.4/13-44
0x114	TQUEUE*—Transmit queue control register	R/W	0x0000_8000	13.5.3.2.5/13-45
0x118– 0x13C	Reserved	—	—	—
0x140	TR03WT*—TxBD Rings 0–3 round-robin weightings	R/W	0x0000_0000	13.5.3.2.6/13-46
0x144	TR47WT*—TxBD Rings 4–7 round-robin weightings	R/W	0x0000_0000	13.5.3.2.7/13-46
0x148– 0x17C	Reserved	—	—	—
0x180	TBDBPH*—Tx data buffer pointer high bits	R/W	0x0000_0000	13.5.3.2.8/13-47
0x184	TBPTR0—TxBD pointer for ring 0	R/W	0x0000_0000	13.5.3.2.9/13-47
0x188	Reserved	—	—	—
0x18C	TBPTR1*—TxBD pointer for ring 1	R/W	0x0000_0000	13.5.3.2.9/13-47
0x190	Reserved	—	—	—
0x194	TBPTR2*—TxBD pointer for ring 2	R/W	0x0000_0000	13.5.3.2.9/13-47
0x198	Reserved	—	—	—
0x19C	TBPTR3*—TxBD pointer for ring 3	R/W	0x0000_0000	13.5.3.2.9/13-47
0x1A0	Reserved	—	—	—
0x1A4	TBPTR4*—TxBD pointer for ring 4	R/W	0x0000_0000	13.5.3.2.9/13-47
0x1A8	Reserved	—	—	—

Table A-11. eTSEC Controllers 1, 2, 3, & 4 Registers (continued)

eTSEC1—Block Base Address 0x2_4000 eTSEC2—Block Base Address 0x2_5000 eTSEC3—Block Base Address 0x2_6000 eTSEC4—Block Base Address 0x2_7000				
Offset	**Register[1]**	**Access [2]**	**Reset**	**Section/Page**
0x1AC	TBPTR5*—TxBD pointer for ring 5	R/W	0x0000_0000	13.5.3.2.9/13-47
0x1B0	Reserved	—	—	—
0x1B4	TBPTR6*—TxBD pointer for ring 6	R/W	0x0000_0000	13.5.3.2.9/13-47
0x1B8	Reserved	—	—	—
0x1BC	TBPTR7*—TxBD pointer for ring 7	R/W	0x0000_0000	13.5.3.2.9/13-47
0x1C0– 0x1FC	Reserved	—	—	—
0x200	TBASEH*—TxBD base address high bits	R/W	0x0000_0000	13.5.3.2.10/13-48
0x204	TBASE0—TxBD base address of ring 0	R/W	0x0000_0000	13.5.3.2.11/13-49
0x208	Reserved	—	—	—
0x20C	TBASE1*—TxBD base address of ring 1	R/W	0x0000_0000	13.5.3.2.11/13-49
0x210	Reserved	—	—	—
0x214	TBASE2*—TxBD base address of ring 2	R/W	0x0000_0000	13.5.3.2.11/13-49
0x218	Reserved	—	—	—
0x21C	TBASE3*—TxBD base address of ring 3	R/W	0x0000_0000	13.5.3.2.11/13-49
0x220	Reserved	—	—	—
0x224	TBASE4*—TxBD base address of ring 4	R/W	0x0000_0000	13.5.3.2.11/13-49
0x228	Reserved	—	—	—
0x22C	TBASE5*—TxBD base address of ring 5	R/W	0x0000_0000	13.5.3.2.11/13-49
0x230	Reserved	—	—	—
0x234	TBASE6*—TxBD base address of ring 6	R/W	0x0000_0000	13.5.3.2.11/13-49
0x238	Reserved	—	—	—
0x23C	TBASE7*—TxBD base address of ring 7	R/W	0x0000_0000	13.5.3.2.11/13-49
0x240– 0x2FC	Reserved	—	—	—
eTSEC Receive Control and Status Registers				
0x300	RCTRL—Receive control register	R/W	0x0000_0000	13.5.3.3.1/13-49
0x304	RSTAT—Receive status register	w1c	0x0000_0000	13.5.3.3.2/13-51
0x308– 0x30C	Reserved	—	—	—
0x310	RXIC—Receive interrupt coalescing register	R/W	0x0000_0000	13.5.3.3.3/13-54
0x314	RQUEUE*—Receive queue control register.	R/W	0x0080_0080	13.5.3.3.4/13-55

Table A-11. eTSEC Controllers 1, 2, 3, & 4 Registers (continued)

eTSEC1—Block Base Address 0x2_4000 eTSEC2—Block Base Address 0x2_5000 eTSEC3—Block Base Address 0x2_6000 eTSEC4—Block Base Address 0x2_7000				
Offset	**Register[1]**	**Access [2]**	**Reset**	**Section/Page**
0x318– 0x32C	Reserved	—	—	—
0x330	RBIFX*—Receive bit field extract control register	R/W	0x0000_0000	13.5.3.3.5/13-55
0x334	RQFAR*—Receive queue filing table address register	R/W	0x0000_0000	13.5.3.3.6/13-57
0x338	RQFCR*—Receive queue filing table control register	R/W	0x0000_0000	13.5.3.3.7/13-58
0x33C	RQFPR*—Receive queue filing table property register	R/W	0x0000_0000	13.5.3.3.8/13-59
0x340	MRBLR—Maximum receive buffer length register	R/W	0x0000_0000	13.5.3.3.9/13-62
0x344– 0x37C	Reserved	—	—	—
0x380	RBDBPH*—Rx data buffer pointer high bits	R/W	0x0000_0000	13.5.3.3.10/13-63
0x384	RBPTR0—RxBD pointer for ring 0	R/W	0x0000_0000	13.5.3.3.11/13-63
0x388	Reserved	—	—	—
0x38C	RBPTR1*—RxBD pointer for ring 1	R/W	0x0000_0000	13.5.3.3.11/13-63
0x390	Reserved	—	—	—
0x394	RBPTR2*—RxBD pointer for ring 2	R/W	0x0000_0000	13.5.3.3.11/13-63
0x398	Reserved	—	—	—
0x39C	RBPTR3*—RxBD pointer for ring 3	R/W	0x0000_0000	13.5.3.3.11/13-63
0x3A0	Reserved	—	—	—
0x3A4	RBPTR4*—RxBD pointer for ring 4	R/W	0x0000_0000	13.5.3.3.11/13-63
0x3A8	Reserved	—	—	—
0x3AC	RBPTR5*—RxBD pointer for ring 5	R/W	0x0000_0000	13.5.3.3.11/13-63
0x3B0	Reserved	—	—	—
0x3B4	RBPTR6*—RxBD pointer for ring 6	R/W	0x0000_0000	13.5.3.3.11/13-63
0x3B8	Reserved	—	—	—
0x3BC	RBPTR7*—RxBD pointer for ring 7	R/W	0x0000_0000	13.5.3.3.11/13-63
0x3C0– 0x3FC	Reserved	—	—	—
0x400	RBASEH*—RxBD base address high bits	R/W	0x0000_0000	13.5.3.3.12/13-64
0x404	RBASE0—RxBD base address of ring 0	R/W	0x0000_0000	13.5.3.3.13/13-65
0x408	Reserved	—	—	—
0x40C	RBASE1*—RxBD base address of ring 1	R/W	0x0000_0000	13.5.3.3.13/13-65
0x410	Reserved	—	—	—
0x414	RBASE2*—RxBD base address of ring 2	R/W	0x0000_0000	13.5.3.3.13/13-65

Table A-11. eTSEC Controllers 1, 2, 3, & 4 Registers (continued)

eTSEC1—Block Base Address 0x2_4000 eTSEC2—Block Base Address 0x2_5000 eTSEC3—Block Base Address 0x2_6000 eTSEC4—Block Base Address 0x2_7000				
Offset	**Register[1]**	**Access [2]**	**Reset**	**Section/Page**
0x418	Reserved	—	—	—
0x41C	RBASE3*—RxBD base address of ring 3	R/W	0x0000_0000	13.5.3.3.13/13-65
0x420	Reserved	—	—	—
0x424	RBASE4*—RxBD base address of ring 4	R/W	0x0000_0000	13.5.3.3.13/13-65
0x428	Reserved	—	—	—
0x42C	RBASE5*—RxBD base address of ring 5	R/W	0x0000_0000	13.5.3.3.13/13-65
0x430	Reserved	—	—	—
0x434	RBASE6*—RxBD base address of ring 6	R/W	0x0000_0000	13.5.3.3.13/13-65
0x438	Reserved	—	—	—
0x43C	RBASE7*—RxBD base address of ring 7	R/W	0x0000_0000	13.5.3.3.13/13-65
0x440– 0x4FC	Reserved	—	—	—
eTSEC MAC Registers				
0x500	MACCFG1—MAC configuration register 1	R/W	0x0000_0000	13.5.3.5.1/13-68
0x504	MACCFG2—MAC configuration register 2	R/W	0x0000_7000	13.5.3.5.2/13-69
0x508	IPGIFG—Inter-packet/inter-frame gap register	R/W	0x4060_5060	13.5.3.5.3/13-71
0x50C	HAFDUP—Half-duplex control	R/W	0x00A1_F037	13.5.3.5.4/13-72
0x510	MAXFRM—Maximum frame length	R/W	0x0000_0600	13.5.3.5.5/13-73
0x514– 0x51C	Reserved	—	—	—
0x520	MIIMCFG—MII management configuration	R/W	0x0000_0007	13.5.3.5.6/13-73
0x524	MIIMCOM—MII management command	R/W	0x0000_0000	13.5.3.5.7/13-74
0x528	MIIMADD—MII management address	R/W	0x0000_0000	13.5.3.5.8/13-75
0x52C	MIIMCON—MII management control	WO	0x0000_0000	13.5.3.5.9/13-76
0x530	MIIMSTAT—MII management status	R	0x0000_0000	13.5.3.5.10/13-76
0x534	MIIMIND—MII management indicator	R	0x0000_0000	13.5.3.5.11/13-77
0x538	Reserved	—	—	—
0x53C	IFSTAT—Interface status	R	0x0000_0000	13.5.3.5.12/13-77
0x540	MACSTNADDR1—MAC station address register 1	R/W	0x0000_0000	13.5.3.5.13/13-78
0x544	MACSTNADDR2—MAC station address register 2	R/W	0x0000_0000	13.5.3.5.14/13-79

Table A-11. eTSEC Controllers 1, 2, 3, & 4 Registers (continued)

eTSEC1—Block Base Address 0x2_4000 eTSEC2—Block Base Address 0x2_5000 eTSEC3—Block Base Address 0x2_6000 eTSEC4—Block Base Address 0x2_7000				
Offset	**Register[1]**	**Access [2]**	**Reset**	**Section/Page**
0x548	MAC01ADDR1*—MAC exact match address 1, part 1	R/W	0x0000_0000	13.5.3.5.15/13-79 13.5.3.5.16/13-80
0x54C	MAC01ADDR2*—MAC exact match address 1, part 2	R/W	0x0000_0000	
0x550	MAC02ADDR1*—MAC exact match address 2, part 1	R/W	0x0000_0000	
0x554	MAC02ADDR2*—MAC exact match address 2, part 2	R/W	0x0000_0000	
0x558	MAC03ADDR1*—MAC exact match address 3, part 1	R/W	0x0000_0000	
0x55C	MAC03ADDR2*—MAC exact match address 3, part 2	R/W	0x0000_0000	
0x560	MAC04ADDR1*—MAC exact match address 4, part 1	R/W	0x0000_0000	
0x564	MAC04ADDR2*—MAC exact match address 4, part 2	R/W	0x0000_0000	
0x568	MAC05ADDR1*—MAC exact match address 5, part 1	R/W	0x0000_0000	
0x56C	MAC05ADDR2*—MAC exact match address 5, part 2	R/W	0x0000_0000	
0x570	MAC06ADDR1*—MAC exact match address 6, part 1	R/W	0x0000_0000	
0x574	MAC06ADDR2*—MAC exact match address 6, part 2	R/W	0x0000_0000	
0x578	MAC07ADDR1*—MAC exact match address 7, part 1	R/W	0x0000_0000	
0x57C	MAC07ADDR2*—MAC exact match address 7, part 2	R/W	0x0000_0000	
0x580	MAC08ADDR1*—MAC exact match address 8, part 1	R/W	0x0000_0000	
0x584	MAC08ADDR2*—MAC exact match address 8, part 2	R/W	0x0000_0000	
0x588	MAC09ADDR1*—MAC exact match address 9, part 1	R/W	0x0000_0000	
0x58C	MAC09ADDR2*—MAC exact match address 9, part 2	R/W	0x0000_0000	
0x590	MAC10ADDR1*—MAC exact match address 10, part 1	R/W	0x0000_0000	
0x594	MAC10ADDR2*—MAC exact match address 10, part 2	R/W	0x0000_0000	13.5.3.5.15/13-79 13.5.3.5.16/13-80
0x598	MAC11ADDR1*—MAC exact match address 11, part 1	R/W	0x0000_0000	
0x59C	MAC11ADDR2*—MAC exact match address 11, part 2	R/W	0x0000_0000	
0x5A0	MAC12ADDR1*—MAC exact match address 12, part 1	R/W	0x0000_0000	
0x5A4	MAC12ADDR2*—MAC exact match address 12, part 2	R/W	0x0000_0000	
0x5A8	MAC13ADDR1*—MAC exact match address 13, part 1	R/W	0x0000_0000	
0x5AC	MAC13ADDR2*—MAC exact match address 13, part 2	R/W	0x0000_0000	
0x5B0	MAC14ADDR1*—MAC exact match address 14, part 1	R/W	0x0000_0000	
0x5B4	MAC14ADDR2*—MAC exact match address 14, part 2	R/W	0x0000_0000	
0x5B8	MAC15ADDR1*—MAC exact match address 15, part 1	R/W	0x0000_0000	
0x5BC	MAC15ADDR2*—MAC exact match address 15, part 2	R/W	0x0000_0000	
0x5C0– 0x67C	Reserved	—	—	—
eTSEC Transmit and Receive Counters				
0x680	TR64—Transmit and receive 64-byte frame counter	R/W	0x0000_0000	13.5.3.6.1/13-81

Table A-11. eTSEC Controllers 1, 2, 3, & 4 Registers (continued)

eTSEC1—Block Base Address 0x2_4000 eTSEC2—Block Base Address 0x2_5000 eTSEC3—Block Base Address 0x2_6000 eTSEC4—Block Base Address 0x2_7000				
Offset	**Register[1]**	**Access [2]**	**Reset**	**Section/Page**
0x684	TR127—Transmit and receive 65- to 127-byte frame counter	R/W	0x0000_0000	13.5.3.6.2/13-81
0x688	TR255—Transmit and receive 128- to 255-byte frame counter	R/W	0x0000_0000	13.5.3.6.3/13-82
0x68C	TR511—Transmit and receive 256- to 511-byte frame counter	R/W	0x0000_0000	13.5.3.6.4/13-82
0x690	TR1K—Transmit and receive 512- to 1023-byte frame counter	R/W	0x0000_0000	13.5.3.6.5/13-83
0x694	TRMAX—Transmit and receive 1024- to 1518-byte frame counter	R/W	0x0000_0000	13.5.3.6.6/13-83
0x698	TRMGV—Transmit and receive 1519- to 1522-byte good VLAN frame count	R/W	0x0000_0000	13.5.3.6.7/13-84
eTSEC Receive Counters				
0x69C	RBYT—Receive byte counter	R/W	0x0000_0000	13.5.3.6.8/13-84
0x6A0	RPKT—Receive packet counter	R/W	0x0000_0000	13.5.3.6.9/13-85
0x6A4	RFCS—Receive FCS error counter	R/W	0x0000_0000	13.5.3.6.10/13-85
0x6A8	RMCA—Receive multicast packet counter	R/W	0x0000_0000	13.5.3.6.11/13-86
0x6AC	RBCA—Receive broadcast packet counter	R/W	0x0000_0000	13.5.3.6.12/13-86
0x6B0	RXCF—Receive control frame packet counter	R/W	0x0000_0000	13.5.3.6.13/13-87
0x6B4	RXPF—Receive PAUSE frame packet counter	R/W	0x0000_0000	13.5.3.6.14/13-87
0x6B8	RXUO—Receive unknown OP code counter	R/W	0x0000_0000	13.5.3.6.15/13-88
0x6BC	RALN—Receive alignment error counter	R/W	0x0000_0000	13.5.3.6.16/13-88
0x6C0	RFLR—Receive frame length error counter	R/W	0x0000_0000	13.5.3.6.17/13-89
0x6C4	RCDE—Receive code error counter	R/W	0x0000_0000	13.5.3.6.18/13-89
0x6C8	RCSE—Receive carrier sense error counter	R/W	0x0000_0000	13.5.3.6.19/13-90
0x6CC	RUND—Receive undersize packet counter	R/W	0x0000_0000	13.5.3.6.20/13-90
0x6D0	ROVR—Receive oversize packet counter	R/W	0x0000_0000	13.5.3.6.21/13-91
0x6D4	RFRG—Receive fragments counter	R/W	0x0000_0000	13.5.3.6.22/13-91
0x6D8	RJBR—Receive jabber counter	R/W	0x0000_0000	13.5.3.6.23/13-92
0x6DC	RDRP—Receive drop counter	R/W	0x0000_0000	13.5.3.6.24/13-92
eTSEC Transmit Counters				
0x6E0	TBYT—Transmit byte counter	R/W	0x0000_0000	13.5.3.6.25/13-93
0x6E4	TPKT—Transmit packet counter	R/W	0x0000_0000	13.5.3.6.26/13-93
0x6E8	TMCA—Transmit multicast packet counter	R/W	0x0000_0000	13.5.3.6.27/13-94
0x6EC	TBCA—Transmit broadcast packet counter	R/W	0x0000_0000	13.5.3.6.28/13-94
0x6F0	TXPF—Transmit PAUSE control frame counter	R/W	0x0000_0000	13.5.3.6.29/13-95

Table A-11. eTSEC Controllers 1, 2, 3, & 4 Registers (continued)

eTSEC1—Block Base Address 0x2_4000 eTSEC2—Block Base Address 0x2_5000 eTSEC3—Block Base Address 0x2_6000 eTSEC4—Block Base Address 0x2_7000				
Offset	**Register[1]**	**Access [2]**	**Reset**	**Section/Page**
0x6F4	TDFR—Transmit deferral packet counter	R/W	0x0000_0000	13.5.3.6.30/13-95
0x6F8	TEDF—Transmit excessive deferral packet counter	R/W	0x0000_0000	13.5.3.6.31/13-96
0x6FC	TSCL—Transmit single collision packet counter	R/W	0x0000_0000	13.5.3.6.32/13-96
0x700	TMCL—Transmit multiple collision packet counter	R/W	0x0000_0000	13.5.3.6.33/13-97
0x704	TLCL—Transmit late collision packet counter	R/W	0x0000_0000	13.5.3.6.34/13-97
0x708	TXCL—Transmit excessive collision packet counter	R/W	0x0000_0000	13.5.3.6.35/13-98
0x70C	TNCL—Transmit total collision counter	R/W	0x0000_0000	13.5.3.6.36/13-98
0x710	Reserved	—	—	—
0x714	TDRP—Transmit drop frame counter	R/W	0x0000_0000	13.5.3.6.37/13-99
0x718	TJBR—Transmit jabber frame counter	R/W	0x0000_0000	13.5.3.6.38/13-99
0x71C	TFCS—Transmit FCS error counter	R/W	0x0000_0000	13.5.3.6.39/13-100
0x720	TXCF—Transmit control frame counter	R/W	0x0000_0000	13.5.3.6.40/13-100
0x724	TOVR—Transmit oversize frame counter	R/W	0x0000_0000	13.5.3.6.41/13-101
0x728	TUND—Transmit undersize frame counter	R/W	0x0000_0000	13.5.3.6.42/13-101
0x72C	TFRG—Transmit fragments frame counter	R/W	0x0000_0000	13.5.3.6.43/13-102
eTSEC Counter Control and TOE Statistics Registers				
0x730	CAR1—Carry register one register[3]	R	0x0000_0000	13.5.3.6.44/13-102
0x734	CAR2—Carry register two register [3]	R	0x0000_0000	13.5.3.6.45/13-104
0x738	CAM1—Carry register one mask register	R/W	0xFE03_FFFF	13.5.3.6.46/13-105
0x73C	CAM2—Carry register two mask register	R/W	0x000F_FFFD	13.5.3.6.47/13-106
0x740	RREJ*—Receive filer rejected packet counter	R/W	0x0000_0000	13.5.3.6.48/13-107
0x744– 0x7FC	Reserved	—	—	—
Hash Function Registers				
0x800	IGADDR0—Individual/group address register 0	R/W	0x0000_0000	13.5.3.7.1/13-108
0x804	IGADDR1—Individual/group address register 1	R/W	0x0000_0000	
0x808	IGADDR2—Individual/group address register 2	R/W	0x0000_0000	
0x80C	IGADDR3—Individual/group address register 3	R/W	0x0000_0000	
0x810	IGADDR4—Individual/group address register 4	R/W	0x0000_0000	
0x814	IGADDR5—Individual/group address register 5	R/W	0x0000_0000	
0x818	IGADDR6—Individual/group address register 6	R/W	0x0000_0000	
0x81C	IGADDR7—Individual/group address register 7	R/W	0x0000_0000	

Table A-11. eTSEC Controllers 1, 2, 3, & 4 Registers (continued)

eTSEC1—Block Base Address 0x2_4000 eTSEC2—Block Base Address 0x2_5000 eTSEC3—Block Base Address 0x2_6000 eTSEC4—Block Base Address 0x2_7000				
Offset	**Register[1]**	**Access [2]**	**Reset**	**Section/Page**
0x820– 0x87C	Reserved	—	—	—
0x880	GADDR0—Group address register 0	R/W	0x0000_0000	13.5.3.7.2/13-109
0x884	GADDR1—Group address register 1	R/W	0x0000_0000	
0x888	GADDR2—Group address register 2	R/W	0x0000_0000	
0x88C	GADDR3—Group address register 3	R/W	0x0000_0000	
0x890	GADDR4—Group address register 4	R/W	0x0000_0000	
0x894	GADDR5—Group address register 5	R/W	0x0000_0000	
0x898	GADDR6—Group address register 6	R/W	0x0000_0000	
0x89C	GADDR7—Group address register 7	R/W	0x0000_0000	
0x8A0– 0x9FC	Reserved	—	—	—
eTSEC FIFO Control Registers				
0xA00	FIFOCFG*—FIFO interface configuration register	R/W	0x0000_00C0	13.5.3.8.1/13-109
0xA04– 0xAFC	Reserved	—	—	—
eTSEC DMA Attribute Registers				
0xB00– 0xBF4	Reserved	—	—	—
0xBF8	ATTR—Attribute register	R/W	0x0000_0000	13.5.3.9.1/13-111
eTSEC Lossless Flow Control Registers				
0xC00	RQPRM0*—Receive Queue Parameters register 0	R/W	0x0000_0000	13.5.3.10.1/13-112
0xC04	RQPRM1*—Receive Queue Parameters register 1	R/W	0x0000_0000	
0xC08	RQPRM2*—Receive Queue Parameters register 2	R/W	0x0000_0000	
0xC0C	RQPRM3*—Receive Queue Parameters register 3	R/W	0x0000_0000	
0xC10	RQPRM4*—Receive Queue Parameters register 4	R/W	0x0000_0000	
0xC14	RQPRM5*—Receive Queue Parameters register 5	R/W	0x0000_0000	
0xC18	RQPRM6*—Receive Queue Parameters register 6	R/W	0x0000_0000	
0xC1C	RQPRM7*—Receive Queue Parameters register 7	R/W	0x0000_0000	
0xC20– 0xC40	Reserved	—	—	—
0xC44	RFBPTR0*—Last Free RxBD pointer for ring 0	R/W	0x0000_0000	13.5.3.10.2/13-113
0xC48	Reserved	—	—	—

Table A-11. eTSEC Controllers 1, 2, 3, & 4 Registers (continued)

eTSEC1—Block Base Address 0x2_4000 eTSEC2—Block Base Address 0x2_5000 eTSEC3—Block Base Address 0x2_6000 eTSEC4—Block Base Address 0x2_7000				
Offset	**Register**[1]	**Access** [2]	**Reset**	**Section/Page**
0xC4C	RFBPTR1*—Last Free RxBD pointer for ring 1	R/W	0x0000_0000	13.5.3.10.2/13-113
0xC50	Reserved	—	—	—
0xC54	RFBPTR2*—Last Free RxBD pointer for ring 2	R/W	0x0000_0000	13.5.3.10.2/13-113
0xC58	Reserved	—	—	—
0xC5C	RFBPTR3*—Last Free RxBD pointer for ring 3	R/W	0x0000_0000	13.5.3.10.2/13-113
0xC60	Reserved	—	—	—
0xC64	RFBPTR4*—Last Free RxBD pointer for ring 4	R/W	0x0000_0000	13.5.3.10.2/13-113
0xC68	Reserved	—	—	—
0xC6C	RFBPTR5*—Last Free RxBD pointer for ring 5	R/W	0x0000_0000	13.5.3.10.2/13-113
0xC70	Reserved	—	—	—
0xC74	RFBPTR6*—Last Free RxBD pointer for ring 6	R/W	0x0000_0000	13.5.3.10.2/13-113
0xC78	Reserved	—	—	—
0xC7C	RFBPTR7*—Last Free RxBD pointer for ring 7	R/W	0x0000_0000	13.5.3.10.2/13-113
eTSEC Future Expansion Space				
0xC00–0xD94	Reserved	—	—	—
eTSEC2				
0x000–0xFF	eTSEC2 Registers **Note:** eTSEC2 has the same memory-mapped registers that are described for eTSEC1 but with a different block base address (0x2_5000).			
eTSEC3				
0x000–0xFFF	eTSEC3 Registers **Note:** eTSEC3 has the same memory-mapped registers that are described for eTSEC1 but with a different block base address (0x2_6000).			
eTSEC4				
0x000–0xFFF	eTSEC4 Registers **Note:** eTSEC4 has the same memory-mapped registers that are described for eTSEC1 but with a different block base address (0x2_7000).			

1 Registers denoted * are new to the enhanced TSEC and not supported by PowerQUICC III TSECs.

2 Key: R = read only, WO = write only, R/W = read and write, LH = latches high, SC = self-clearing.

3 Cleared on read.

A.2 Programmable Interrupt Controller (PIC)

The programmable interrupt controller (PIC) follows the OpenPIC programming model which requires a larger register address space than the 4 Kbytes allocated to other blocks within the general utilities space. For this reason, the PIC is allocated the second 256 Kbytes of CCSR space (0x4_0000–0x6_FFFF)

A.2.1 PIC—Global Registers

Table A-12. PIC Global Registers

PIC Global Registers—Block Base Address 0x4_0000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0x0000	BRR1—Block revision register 1	R	0x0040_0300	9.3.1.1/9-18
0x0010	BRR2—Block revision register 2	R	0x0000_0001	9.3.1.2/9-19
0x0020–0x0030	Reserved	—	—	—
0x0040	IPIDR0—Interprocessor interrupt 0 (IPI 0) dispatch register	W	0x0000_0000	9.3.8.1/9-49
0x0050	IPIDR1—IPI 1 dispatch register			
0x0060	IPIDR2—IPI 2 dispatch register			
0x0070	IPIDR3—IPI 3 dispatch register			
0x0080	CTPR—Current task priority register	R/W	0x0000_000F	9.3.8.2/9-49
0x0090	WHOAMI—Who am I register	R	n/a	9.3.8.3/9-50
0x00A0	IACK—Interrupt acknowledge register	R	0x0000_0000	9.3.8.4/9-51
0x00B0	EOI—End of interrupt register	W	0x0000_0000	9.3.8.5/9-51
0x00C0–0x0FF0	Reserved	—	—	—
0x1000	FRR—Feature reporting register	R	0x0057_01020 x0067_0002	9.3.1.3/9-19
0x1010	Reserved	—	—	—
0x1020	GCR—Global configuration register	R/W	0x0000_0000	9.3.1.4/9-20
0x1030	Reserved	—	—	—
0x1040–0x1070	Vendor reserved	—	—	—
0x1080	VIR—Vendor identification register	R	0x0000_0000	9.3.1.5/9-21
0x1090	PIR—Processor core initialization register	R/W	0x0000_0000	9.3.1.6/9-21
0x1098	PRR—Processor reset register	R/W	0x0000_0000	9.3.1.7/9-22
0x10A0	IPIVPR0—IPI 0 vector/priority register	R/W	0x8000_0000	9.3.1.8/9-23
0x10B0	IPIVPR1—IPI 1 vector/priority register			
0x10C0	IPIVPR2—IPI 2 vector/priority register			
0x10D0	IPIVPR3—IPI 3 vector/priority register			
0x10E0	SVR—Spurious vector register	R/W	0x0000_FFFF	9.3.1.9/9-23
Global Timer Group A Registers				

Table A-12. PIC Global Registers (continued)

PIC Global Registers—Block Base Address 0x4_0000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0x10F0	TFRRA—Timer frequency reporting register (Group A)	R/W	0x0000_0000	9.3.2.1/9-24
0x1100	GTCCRA0—Global timer 0 current count register (Group A)	R	0x0000_0000	9.3.2.2/9-25
0x1110	GTBCRA0—Global timer 0 base count register (Group A)	R/W	0x8000_0000	9.3.2.3/9-25
0x1120	GTVPRA0—Global timer 0 vector/priority register (Group A)	R/W	0x8000_0000	9.3.2.4/9-26
0x1130	GTDRA0—Global timer 0 destination register (Group A)	R/W	0x0000_0001	9.3.2.5/9-27
0x1140	GTCCRA1—Global timer 1 current count register (Group A)	R	0x0000_0000	9.3.2.2/9-25
0x1150	GTBCRA1—Global timer 1 base count register (Group A)	R/W	0x8000_0000	9.3.2.3/9-25
0x1160	GTVPRA1—Global timer 1 vector/priority register (Group A)	R/W	0x8000_0000	9.3.2.4/9-26
0x1170	GTDRA1—Global timer 1 destination register (Group A)	R/W	0x0000_0001	9.3.2.5/9-27
0x1180	GTCCRA2—Global timer 2 current count register (Group A)	R	0x0000_0000	9.3.2.2/9-25
0x1190	GTBCRA2—Global timer 2 base count register (Group A)	R/W	0x8000_0000	9.3.2.3/9-25
0x11A0	GTVPRA2—Global timer 2 vector/priority register (Group A)	R/W	0x8000_0000	9.3.2.4/9-26
0x11B0	GTDRA2—Global timer 2 destination register (Group A)	R/W	0x0000_0001	9.3.2.5/9-27
0x11C0	GTCCRA3—Global timer 3 current count register (Group A)	R	0x0000_0000	9.3.2.2/9-25
0x11D0	GTBCRA3—Global timer 3 base count register (Group A)	R/W	0x8000_0000	9.3.2.3/9-25
0x11E0	GTVPRA3—Global timer 3 vector/priority register (Group A)	R/W	0x8000_0000	9.3.2.4/9-26
0x11F0	GTDRA3—Global timer 3 destination register (Group A)	R/W	0x0000_0001	9.3.2.5/9-27
0x1200–0x12F0	Reserved	—	—	—
0x1300	TCRA—Timer control register (Group A)	R/W	0x0000_0000	9.3.2.6/9-27
0x1308	ERQSR—External interrupt summary register	R	0x0000_0000	9.3.3.1/9-29
0x1310	IRQSR0—IRQ_OUT summary register 0	R	0x0000_0000	9.3.3.2/9-30
0x1320	IRQSR1—IRQ_OUT summary register 1	R	0x0000_0000	9.3.3.3/9-31
0x1324	IRQSR2—IRQ_OUT summary register 2	R	0x0000_0000	9.3.3.4/9-31
0x1330	CISR0—Critical interrupt summary register 0	R	0x0000_0000	9.3.3.5/9-32
0x1340	CISR1—Critical interrupt summary register 1	R	0x0000_0000	9.3.3.6/9-32
0x1344	CISR2—Critical interrupt summary register 2	R	0x0000_0000	9.3.3.7/9-33
0x1350	PM0MR0—Performance monitor 0 mask register 0	R/W	0xFFFF_FFFF	9.3.4.1/9-33
0x1360	PM0MR1—Performance monitor 0 mask register 1	R/W	0xFFFF_FFFF	9.3.4.2/9-34
0x1364	PM0MR2—Performance monitor 0 mask register 2	R/W	0xFFFF_0000	9.3.4.2/9-34
0x1370	PM1MR0—Performance monitor 1 mask register 0	R/W	0xFFFF_FFFF	9.3.4.1/9-33
0x1380	PM1MR1—Performance monitor 1 mask register 1	R/W	0xFFFF_FFFF	9.3.4.2/9-34
0x1384	PM1MR2—Performance monitor 1 mask register 2	R/W	0xFFFF_0000	9.3.4.2/9-34
0x1390	PM2MR0—Performance monitor 2 mask register 0	R/W	0xFFFF_FFFF	9.3.4.1/9-33
0x13A0	PM2MR1—Performance monitor 2 mask register 1	R/W	0xFFFF_FFFF	9.3.4.2/9-34
0x13A4	PM2MR2—Performance monitor 2 mask register 2	R/W	0xFFFF_0000	9.3.4.2/9-34

Table A-12. PIC Global Registers (continued)

PIC Global Registers—Block Base Address 0x4_0000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0x13B0	PM3MR0—Performance monitor 3 mask register 0	R/W	0xFFFF_FFFF	9.3.4.1/9-33
0x13C0	PM3MR1—Performance monitor 3 mask register 1	R/W	0xFFFF_FFFF	9.3.4.2/9-34
0x13C4	PM3MR2—Performance monitor 3 mask register 2	R/W	0xFFFF_0000	9.3.4.2/9-34
0x13D0–0x13F0	Reserved	—	—	—
0x1400	MSGR0—Message register 0	R/W	0x0000_0000	9.3.5.1/9-35
0x1410	MSGR1—Message register 1	R/W	0x0000_0000	9.3.5.1/9-35
0x1420	MSGR2—Message register 2	R/W	0x0000_0000	9.3.5.1/9-35
0x1430	MSGR3—Message register 3	R/W	0x0000_0000	9.3.5.1/9-35
0x1440–0x14F0	Reserved	—	—	—
0x1500	MER—Message enable register	R/W	0x0000_0000	9.3.5.2/9-36
0x1510	MSR—Message status register	R/W	0x0000_0000	9.3.5.3/9-36
0x1520–0x15F0	Reserved	—	—	—
0x1600	MSIR0—Shared message signaled interrupt register 0	RC	0x0000_0000	9.3.6.1/9-37
0x1610	MSIR1—Shared message signaled interrupt register 1	RC	0x0000_0000	9.3.6.1/9-37
0x1620	MSIR2—Shared message signaled interrupt register 2	RC	0x0000_0000	9.3.6.1/9-37
0x1630	MSIR3—Shared message signaled interrupt register 3	RC	0x0000_0000	9.3.6.1/9-37
0x1640	MSIR4—Shared message signaled interrupt register 4	RC	0x0000_0000	9.3.6.1/9-37
0x1650	MSIR5—Shared message signaled interrupt register 5	RC	0x0000_0000	9.3.6.1/9-37
0x1660	MSIR6—Shared message signaled interrupt register 6	RC	0x0000_0000	9.3.6.1/9-37
0x1670	MSIR7—Shared message signaled interrupt register 7	RC	0x0000_0000	9.3.6.1/9-37
0x1680–0x1700	Reserved	—	—	—
0x1720	MSISR—Shared message signaled interrupt status register	R	0x0000_0000	9.3.6.2/9-38
0x1740	MSIIR—Shared message signaled interrupt index register	W	0x0000_0000	9.3.6.3/9-38
0x1750–0x20E0	Reserved	—	—	—
Global Timer Group B Registers				
0x20F0	TFRRB—Timer frequency reporting register group B	R/W	0x0000_0000	9.3.2.1/9-24
0x2100	GTCCRB0—Global timer current count register group B 0	R	0x0000_0000	9.3.2.2/9-25
0x2110	GTBCRB0—Global timer base count register group B 0	R/W	0x8000_0000	9.3.2.3/9-25
0x2120	GTVPRB0—Global timer vector/priority register group B 0	R/W	0x8000_0000	9.3.2.4/9-26
0x2130	GTDRB0—Global timer destination register group B 0	R/W	0x0000_0001	9.3.2.5/9-27
0x2140	GTCCRB1—Global timer current count register group B 1	R	0x0000_0000	9.3.2.2/9-25
0x2150	GTBCRB1—Global timer base count register group B 1	R/W	0x8000_0000	9.3.2.3/9-25

Table A-12. PIC Global Registers (continued)

PIC Global Registers—Block Base Address 0x4_0000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0x2160	GTVPRB1—Global timer vector/priority register group B 1	R/W	0x8000_0000	9.3.2.4/9-26
0x2170	GTDRB1—Global timer destination register group B 1	R/W	0x0000_0001	9.3.2.5/9-27
0x2180	GTCCRB2—Global timer current count register group B 2	R	0x0000_0000	9.3.2.2/9-25
0x2190	GTBCRB2—Global timer base count register group B 2	R/W	0x8000_0000	9.3.2.3/9-25
0x21A0	GTVPRB2—Global timer vector/priority register group B 2	R/W	0x8000_0000	9.3.2.4/9-26
0x21B0	GTDRB2—Global timer destination register group B 2	R/W	0x0000_0001	9.3.2.5/9-27
0x21C0	GTCCRB3—Global timer current count register group B 3	R	0x0000_0000	9.3.2.2/9-25
0x21D0	GTBCRB3—Global timer base count register group B 3	R/W	0x8000_0000	9.3.2.3/9-25
0x21E0	GTVPRB3—Global timer vector/priority register group B 3	R/W	0x8000_0000	9.3.2.4/9-26
0x21F0	GTDRB3—Global timer destination register group B 3	R/W	0x0000_0001	9.3.2.5/9-27
0x2200–0x22F0	Reserved	—	—	—
0x2300	TCRB—Timer control register (Group B)	R/W	0x0000_0000	9.3.2.6/9-27
0x2310–0xFFF0	Reserved	—	—	—

A.2.2 PIC—Interrupt Source Registers

Table A-13. PIC Interrupt Source Registers

PIC Interrupt Source Registers—Block Base Address 0x5_0000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0x0000	EIVPR0—External interrupt 0 (IRQ0) vector/priority register or PEX1-INTA vector/priority register	R/W	0x8000_0000	9.3.7.1/9-42
0x0010	EIDR0—External interrupt 0 (IRQ0) destination register or PEX1-INTA destination register	R/W	0x0000_0001	9.3.7.2/9-43
0x0020	EIVPR1—External interrupt 1 (IRQ1) vector/priority register or PEX1-INTB vector/priority register	R/W	0x8000_0000	9.3.7.1/9-42
0x0030	EIDR1—External interrupt 1 (IRQ1) destination register or PEX1-INTB destination register	R/W	0x0000_0001	9.3.7.2/9-43
0x0040	EIVPR2—External interrupt 2 (IRQ2) vector/priority register or PEX1-INTC vector/priority register	R/W	0x8000_0000	9.3.7.1/9-42
0x0050	EIDR2—External interrupt 2 (IRQ2) destination register or PEX1-INTC destination register	R/W	0x0000_0001	9.3.7.2/9-43
0x0060	EIVPR3—External interrupt 3 (IRQ3) vector/priority register or PEX1-INTD vector/priority register	R/W	0x8000_0000	9.3.7.1/9-42
0x0070	EIDR3—External interrupt 3 (IRQ3) destination register or PEX1-INTD destination register	R/W	0x0000_0001	9.3.7.2/9-43
0x0080	EIVPR4—External interrupt 4 (IRQ4) vector/priority register or PEX2-INTA vector/priority register	R/W	0x8000_0000	9.3.7.1/9-42

Table A-13. PIC Interrupt Source Registers (continued)

PIC Interrupt Source Registers—Block Base Address 0x5_0000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0x0090	EIDR4—External interrupt 4 (IRQ4) destination register or PEX2-INTA destination register	R/W	0x0000_0001	9.3.7.2/9-43
0x00A0	EIVPR5—External interrupt 5 (IRQ5) vector/priority register or PEX2-INTB vector/priority register	R/W	0x8000_0000	9.3.7.1/9-42
0x00B0	EIDR5—External interrupt 5 (IRQ5) destination register or PEX2-INTB destination register	R/W	0x0000_0001	9.3.7.2/9-43
0x00C0	EIVPR6—External interrupt 6 (IRQ6) vector/priority register or PEX2-INTC vector/priority register	R/W	0x8000_0000	9.3.7.1/9-42
0x00D0	EIDR6—External interrupt 6 (IRQ6) destination register or PEX2-INTC destination register	R/W	0x0000_0001	9.3.7.2/9-43
0x00E0	EIVPR7—External interrupt 7 (IRQ7) vector/priority register or PEX2-INTD vector/priority register	R/W	0x8000_0000	9.3.7.1/9-42
0x00F0	EIDR7—External interrupt 7 (IRQ7) destination register or PEX2-INTD destination register	R/W	0x0000_0001	9.3.7.2/9-43
0x0100	EIVPR8—External interrupt 8 (IRQ8) vector/priority register	R/W	0x8000_0000	9.3.7.1/9-42
0x0110	EIDR8—External interrupt 8 (IRQ8) destination register	R/W	0x0000_0001	9.3.7.2/9-43
0x0120	EIVPR9—External interrupt 9 (IRQ9) vector/priority register	R/W	0x8000_0000	9.3.7.1/9-42
0x0130	EIDR9—External interrupt 9 (IRQ9) destination register	R/W	0x0000_0001	9.3.7.2/9-43
0x0140	EIVPR10—External interrupt 10 (IRQ10) vector/priority register	R/W	0x8000_0000	9.3.7.1/9-42
0x0150	EIDR10—External interrupt 10 (IRQ10) destination register	R/W	0x0000_0001	9.3.7.2/9-43
0x0160	EIVPR11—External interrupt 11 (IRQ11) vector/priority register	R/W	0x8000_0000	9.3.7.1/9-42
0x0170	EIDR11—External interrupt 11 (IRQ11) destination register	R/W	0x0000_0001	9.3.7.2/9-43
0x0180–0x01F0	Reserved	—	—	—
0x0200	IIVPR0—Internal interrupt 0 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x0210	IIDR0—Internal interrupt 0 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x0220	IIVPR1—Internal interrupt 1 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x0230	IIDR1—Internal interrupt 1 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x0240	IIVPR2—Internal interrupt 2 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x0250	IIDR2—Internal interrupt 2 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x0260	IIVPR3—Internal interrupt 3 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x0270	IIDR3—Internal interrupt 3 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x0280	IIVPR4—Internal interrupt 4 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x0290	IIDR4—Internal interrupt 4 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x02A0	IIVPR5—Internal interrupt 5 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x02B0	IIDR5—Internal interrupt 5 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x02C0	IIVPR6—Internal interrupt 6 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x02D0	IIDR6—Internal interrupt 6 destination register	R/W	0x0000_0001	9.3.7.4/9-45

Table A-13. PIC Interrupt Source Registers (continued)

PIC Interrupt Source Registers—Block Base Address 0x5_0000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0x02E0	IIVPR7—Internal interrupt 7 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x02F0	IIDR7—Internal interrupt 7 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x0300	IIVPR8—Internal interrupt 8 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x0310	IIDR8—Internal interrupt 8 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x0320	IIVPR9—Internal interrupt 9 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x0330	IIDR9—Internal interrupt 9 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x0340	IIVPR10—Internal interrupt 10 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x0350	IIDR10—Internal interrupt 10 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x0360	IIVPR11—Internal interrupt 11 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x0370	IIDR11—Internal interrupt 11 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x0380	IIVPR12—Internal interrupt 12 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x0390	IIDR12—Internal interrupt 12 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x03A0	IIVPR13—Internal interrupt 13 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x03B0	IIDR13—Internal interrupt 13 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x03C0	IIVPR14—Internal interrupt 14 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x03D0	IIDR14—Internal interrupt 14 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x03E0	IIVPR15—Internal interrupt 15 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x03F0	IIDR15—Internal interrupt 15 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x0400	IIVPR16—Internal interrupt 16 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x0410	IIDR16—Internal interrupt 16 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x0420	IIVPR17—Internal interrupt 17 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x0430	IIDR17—Internal interrupt 17 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x0440	IIVPR18—Internal interrupt 18 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x0450	IIDR18—Internal interrupt 18 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x0460	IIVPR19—Internal interrupt 19 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x0470	IIDR19—Internal interrupt 19 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x0480	IIVPR20—Internal interrupt 20 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x0490	IIDR20—Internal interrupt 20 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x04A0	IIVPR21—Internal interrupt 21 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x04B0	IIDR21—Internal interrupt 21 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x04C0	IIVPR22—Internal interrupt 22 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x04D0	IIDR22—Internal interrupt 22 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x04E0	IIVPR23—Internal interrupt 23 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x04F0	IIDR23—Internal interrupt 23 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x0500	IIVPR24—Internal interrupt 24 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x0510	IIDR24—Internal interrupt 24 destination register	R/W	0x0000_0001	9.3.7.4/9-45

Table A-13. PIC Interrupt Source Registers (continued)

PIC Interrupt Source Registers—Block Base Address 0x5_0000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0x0520	IIVPR25—Internal interrupt 25 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x0530	IIDR25—Internal interrupt 25 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x0540	IIVPR26—Internal interrupt 26 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x0550	IIDR26—Internal interrupt 26 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x0560	IIVPR27—Internal interrupt 27 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x0570	IIDR27—Internal interrupt 27 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x0580	IIVPR28—Internal interrupt 28 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x0590	IIDR28—Internal interrupt 28 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x05A0	IIVPR29—Internal interrupt 29 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x05B0	IIDR29—Internal interrupt 29 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x05C0	IIVPR30—Internal interrupt 30 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x05D0	IIDR30—Internal interrupt 30 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x05E0	IIVPR31—Internal interrupt 31 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x05F0	IIDR31—Internal interrupt 31 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x0600	IIVPR32—Internal interrupt 32 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x0610	IIDR32—Internal interrupt 32 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x0620	IIVPR33—Internal interrupt 33 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x0630	IIDR33—Internal interrupt 33 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x0640	IIVPR34—Internal interrupt 34 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x0650	IIDR34—Internal interrupt 34 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x0660	IIVPR35—Internal interrupt 35 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x0670	IIDR35—Internal interrupt 35 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x0680	IIVPR36—Internal interrupt 36 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x0690	IIDR36—Internal interrupt 36 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x06A0	IIVPR37—Internal interrupt 37 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x06B0	IIDR37—Internal interrupt 37 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x06C0	IIVPR38—Internal interrupt 38 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x06D0	IIDR38—Internal interrupt 38 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x06E0	IIVPR39—Internal interrupt 39 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x06F0	IIDR39—Internal interrupt 39 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x0700	IIVPR40—Internal interrupt 40 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x0710	IIDR40—Internal interrupt 40 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x0720	IIVPR41—Internal interrupt 41 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x0730	IIDR41—Internal interrupt 41 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x0740	IIVPR42—Internal interrupt 42 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x0750	IIDR42—Internal interrupt 42 destination register	R/W	0x0000_0001	9.3.7.4/9-45

Table A-13. PIC Interrupt Source Registers (continued)

PIC Interrupt Source Registers—Block Base Address 0x5_0000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0x0760	IIVPR43—Internal interrupt 43 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x0770	IIDR43—Internal interrupt 43 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x0780	IIVPR44—Internal interrupt 44 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x0790	IIDR44—Internal interrupt 44 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x07A0	IIVPR45—Internal interrupt 45 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x07B0	IIDR45—Internal interrupt 45 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x07C0	IIVPR46—Internal interrupt 46 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x07D0	IIDR46—Internal interrupt 46 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x07E0	IIVPR47—Internal interrupt 47 vector/priority register	R/W	0x8080_0000	9.3.7.3/9-44
0x07F0	IIDR47—Internal interrupt 47 destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x0800–0x15F0	Reserved	—	—	—
0x1600	MIVPR0—Messaging interrupt 0 (MSG 0) vector/priority register	R/W	0x8000_0000	9.3.7.5/9-46
0x1610	MIDR0—Messaging interrupt 0 (MSG 0) destination register	R/W	0x0000_0001	9.3.7.6/9-46
0x1620	MIVPR1—Messaging interrupt 1 (MSG 1) vector/priority register	R/W	0x8000_0000	9.3.7.5/9-46
0x1630	MIDR1—Messaging interrupt 1 (MSG 1) destination register	R/W	0x0000_0001	9.3.7.6/9-46
0x1640	MIVPR2—Messaging interrupt 2 (MSG 2) vector/priority register	R/W	0x8000_0000	9.3.7.5/9-46
0x1650	MIDR2—Messaging interrupt 2 (MSG 2) destination register	R/W	0x0000_0001	9.3.7.6/9-46
0x1660	MIVPR3—Messaging interrupt 3 (MSG 3) vector/priority register	R/W	0x8000_0000	9.3.7.5/9-46
0x1670	MIDR3—Messaging interrupt 3 (MSG 3) destination register	R/W	0x0000_0001	9.3.7.4/9-45
0x1680–0x1BF0	Reserved	—	—	—
0x1C00	MSIVPR0—Shared message signaled interrupt vector/priority register 0	R/W	0x8000_0000	9.3.6.4/9-39
0x1C10	MSIDR0—Shared message signaled interrupt destination register 0	R/W	0x0000_0001	9.3.6.5/9-40
0x1C20	MSIVPR1—Shared message signaled interrupt vector/priority register 1	R/W	0x8000_0000	9.3.6.4/9-39
0x1C30	MSIDR1—Shared message signaled interrupt destination register 1	R/W	0x0000_0001	9.3.6.5/9-40
0x1C40	MSIVPR2—Shared message signaled interrupt vector/priority register 2	R/W	0x8000_0000	9.3.6.4/9-39
0x1C50	MSIDR2—Shared message signaled interrupt destination register 2	R/W	0x0000_0001	9.3.6.5/9-40
0x1C60	MSIVPR3—Shared message signaled interrupt vector/priority register 3	R/W	0x8000_0000	9.3.6.4/9-39
0x1C70	MSDIR3—Shared message signaled interrupt destination register 3	R/W	0x0000_0001	9.3.6.5/9-40
0x1C80	MSIVPR4—Shared message signaled interrupt vector/priority register 4	R/W	0x8000_0000	9.3.6.4/9-39
0x1C90	MSIDR4—Shared message signaled interrupt destination register 4	R/W	0x0000_0001	9.3.6.5/9-40
0x1CA0	MSIVPR5—Shared message signaled interrupt vector/priority register 5	R/W	0x8000_0000	9.3.6.4/9-39
0x1CB0	MSIDR5—Shared message signaled interrupt destination register 5	R/W	0x0000_0001	9.3.6.5/9-40
0x1CC0	MSIVPR6—Shared message signaled interrupt vector/priority register 6	R/W	0x8000_0000	9.3.6.4/9-39
0x1CD0	MSIDR6—Shared message signaled interrupt destination register 6	R/W	0x0000_0001	9.3.6.5/9-40

Table A-13. PIC Interrupt Source Registers (continued)

PIC Interrupt Source Registers—Block Base Address 0x5_0000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0x1CE0	MSIVPR7—Shared message signaled interrupt vector/priority register 7	R/W	0x8000_0000	9.3.6.4/9-39
0x1CF0	MSDIR7—Shared message signaled interrupt destination register 7	R/W	0x0000_0001	9.3.6.5/9-40
0x1D00–0xFFF0	Reserved	—	—	—

A.2.3 PIC—Processor (per-CPU) Registers

Table A-14. PIC Processor (per-CPU) Registers

PIC Processor (per-CPU) Registers—Block Base Address 0x6_0000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0x0000–0x0030	Reserved	—	—	—
0x0040	IPIDR0—Processor core 0 interprocessor 0 dispatch register	W	all zeros	9.3.8.1/9-49
0x0050	IPIDR1—Processor core 0 interprocessor 1 dispatch register			
0x0060	IPIDR2—Processor core 0 interprocessor 2 dispatch register			
0x0070	IPIDR3—Processor core 0 interprocessor 3 dispatch register			
0x0080	CTPR0—Processor core 0 current task priority register	R/W	0x0000_000F	9.3.8.2/9-49
0x0090	WHOAMI0—Processor core 0 who am I register	R	n/a	9.3.8.3/9-50
0x00A0	IACK0—Processor core 0 interrupt acknowledge register	R	all zeros	9.3.8.4/9-51
0x00B0	EOI0—Processor core 0 end of interrupt register	W	all zeros	9.3.8.5/9-51
0x00C0–0x0FF0	Reserved	—	—	—
0x1000–0x1030	Reserved	—	—	—
0x1040	IPIDR0—Processor core 1 interprocessor 0 dispatch register [1]	W	all zeros	9.3.8.1/9-49
0x1050	IPIDR1—Processor core 1 interprocessor 1 dispatch register [1]			
0x1060	IPIDR2—Processor core 1 interprocessor 2 dispatch register [1]			
0x1070	IPIDR3—Processor core 1 interprocessor 3 dispatch register [1]			
0x1080	CTPR1—Processor core 1 current task priority register [1]	R/W	0x0000_000F	9.3.8.2/9-49
0x1090	WHOAMI1—Processor core 1 who am I register [1]	R	n/a	9.3.8.3/9-50
0x10A0	IACK1—Processor core 1 interrupt acknowledge register [1]	R	all zeros	9.3.8.4/9-51
0x10B0	EOI1—Processor core 1 end of interrupt register [1]	W	all zeros	9.3.8.5/9-51
0x10C0–0xFFFC	Reserved	—	—	—

[1] Not supported for single-processor implementations.

A.3 Serial RapidIO

The serial RapidIO module uses 128 Kbytes of CCSR space (0xC_0000–0xD_FFFF).

A.3.1 RapidIO—Architectural Registers

Table A-15. RapidIO Architectural Registers

RapidIO Architectural Registers—Block Base Address 0xC_0000				
Offset	**Register**	**Access**	**Reset[1]**	**Section/Page**
0x0000	Device identity capability register (DIDCAR)	R	0x701*n*_0002 = MPC8641	15.6.1.1/15-10
0x0004	Device information capability register (DICAR)	R	0x*nnnn_nnnn*	15.6.1.2/15-11
0x0008	Assembly identity capability register (AIDCAR)	R/W	0x*nnnn_nnnn*	15.6.1.3/15-12
0x000C	Assembly information capability register (AICAR)	R/W	0x0000_0000	15.6.1.4/15-12
0x0010	Processing element features capability register (PEFCAR)	R	0xE0F8_00*n*9	15.6.1.5/15-13
0x0018	Source operations capability register (SOCAR)	R	0x0600_FCF0	15.6.1.6/15-14
0x001C	Destination operations capability register (DOCAR)	R	0x0000_FCF4	15.6.1.7/15-15
0x0040	Mailbox command and status register (MCSR)	R	0x2020_0000	15.6.1.8/15-16
0x0044	Port -Write and doorbell command and status register (PWDCSR)	R	0x2000_0020	15.6.1.9/15-17
0x004C	Processing element logical layer control command and status register (PELLCCSR)	R	0x0000_0001	15.6.1.10/15-19
0x005C	Local configuration space base address 1 command and status register (LCSBA1CSR)	R/W	0x0000_0000	15.6.1.11/15-19
0x0060	Base device ID command and status register (BDIDCSR)	R/W	0x00*nn_nnnn*	15.6.1.12/15-20
0x0068	Host base device ID lock command and status register (HBDIDLCSR)	Special	0x0000_FFFF	15.6.1.13/15-21
0x006C	Component tag command and status register (CTCSR)	R/W	0x0000_0000	15.6.1.14/15-21
Extended Features Space				
1x/4x LP-Serial				
0x0100	Port maintenance block header 0 (PMBH0)	R	0x0600_0001	15.6.2.1/15-22
0x0120	Port link time-out control command and status register (PLTOCCSR)	R/W	0xFFFF_FF00	15.6.2.2/15-22
0x0124	Port response time-out control command and status register (PRTOCCSR)	R/W	0xFFFF_FF00	15.6.2.3/15-23
0x013C	General control command and status register (GCCSR)	R/W	0x*n*000_0000	15.6.2.4/15-23
0x0140	Link maintenance request command and status register (LMREQCSR)	R/W	0x0000_0000	15.6.2.5/15-24
0x0144	Link maintenance response command and status register (LMRESPCSR)	R	0x0000_0000	15.6.2.6/15-25
0x0148	Local ackID status command and status register (LASCSR)	Mixed	0x0000_0000	15.6.2.7/15-25
0x0158	Error and status command and status register (ESCSR)	Mixed	0x0000_0001	15.6.2.8/15-26

Table A-15. RapidIO Architectural Registers (continued)

RapidIO Architectural Registers—Block Base Address 0xC_0000				
Offset	**Register**	**Access**	**Reset[1]**	**Section/Page**
0x015C	Control command and status register (CCSR)	R/W	0x5060_0001	15.6.2.9/15-28
Error Reporting, Logical				
0x0600	Error reporting block header (ERBH)	R	0x0000_0007	15.6.3.1/15-30
0x0608	Logical/Transport layer error detect command and status register (LTLEDCSR)	R/W	0x0000_0000	15.6.3.2/15-30
0x060C	Logical/Transport layer error enable command and status register (LTLEECSR)	R/W	0x0000_0000	15.6.3.3/15-32
0x0614	Logical/Transport layer address capture command and status register (LTLACCSR)	R/W	0x0000_0000	15.6.3.4/15-33
0x0618	Logical/Transport layer device ID capture command and status register (LTLDIDCCSR)	R/W	0x0000_0000	15.6.3.5/15-34
0x061C	Logical/Transport layer control capture command and status register (LTLCCCSR)	R/W	0x0000_0000	15.6.3.6/15-35
Error Reporting, Physical				
0x0640	Error detect command and status register (EDCSR)	R/W	0x0000_0000	15.6.4.1/15-36
0x0644	Error rate enable command and status register (ERECSR)	R/W	0x0000_0000	15.6.4.2/15-36
0x0648	Error capture attributes command and status register (ECACSR)	R/W	0x0000_0000	15.6.4.3/15-37
0x064C	Packet/control symbol error capture command and status register 0 (PCSECCSR0)	R/W	0x0000_0000	15.6.4.4/15-38
0x0650	Packet error capture command and status register 1 (PECCSR1)	R/W	0x0000_0000	15.6.4.5/15-39
0x0654	Packet error capture command and status register 2 (PECCSR2)	R/W	0x0000_0000	15.6.4.6/15-39
0x0658	Packet error capture command and status register 3 (PECCSR3)	R/W	0x0000_0000	15.6.4.7/15-40
0x0668	Error rate command and status register (ERCSR)	R/W	0x8000_0000	15.6.4.8/15-40
0x066C	Error rate threshold command and status register (ERTCSR)	R/W	0xFFFF_0000	15.6.4.9/15-41
0x0670 –0x067C	Reserved	—	—	—

1 Values indicated with *n* are read from configuration signals at reset; see register description for details.

A.3.2 RapidIO—Implementation Registers

Table A-16. RapidIO Implementation Registers

RapidIO Implementation Registers—Block Base Address 0xD_0000				
Offset	**Register**	**Access**	**Reset[1]**	**Section/Page**
General				
0x0004	Logical layer configuration register (LLCR)	R/W	0x0000_0000	15.6.5.1/15-42
0x0010	Error / port-write interrupt status register (EPWISR)	R	0x0000_0000	15.6.5.2/15-43
0x0020	Logical retry error threshold configuration register (LRETCR)	R/W	0x0000_00FF	15.6.5.3/15-43
0x0080	Physical retry error threshold configuration register (PRETCR)	R/W	0x0000_00FF	15.6.5.4/15-44
0x0100	Alternate device ID command and status register (ADIDCSR)	R/W	0x0000_0000	15.6.5.5/15-44
0x0120	Accept-all configuration register (AACR)	R/W	0x*n*000_000*n*	15.6.5.6/15-45
0x0124	Logical Outbound Packet time-to-live configuration register (LOPTTLCR)	R/W	0x0000_0000	15.6.5.7/15-45
0x0130	Implementation error command and status register (IECSR)	w1c	0x0000_0000	15.6.5.8/15-46
0x0140	Physical configuration register (PCR)	R/W	0x0000_8010	15.6.5.9/15-47
0x0158	Serial link command and status register (SLCSR)	w1c	0x0000_0000	15.6.5.10/15-47
0x0160	Serial link error injection configuration register (SLEICR)	R/W	0x0000_0000	15.6.5.11/15-48
Revision Control Register				
0x0BF8	IP Block Revision Register 1 (IPBRR1)	R	0x01C0_0000	15.6.6.1/15-49
0x0BFC	IP Block Revision Register 2 (IPBRR2)	R	0x0000_0000	15.6.6.2/15-49
ATMU				
0x0C00	RapidIO outbound window translation address register 0 (ROWTAR0)	R/W	0xFF80_0000	15.6.7.2/15-52
0x0C04	RapidIO outbound window translation extended address register 0 (ROWTEAR0)	R/W	0x0000_003F	15.6.7.3/15-53
0x0C10	RapidIO outbound window attributes register 0 (ROWAR0)	R/W	0x8004_4023	15.6.7.5/15-54
0x0C20	RapidIO outbound window translation address register 1 (ROWTAR1)	R/W	0x0000_0000	15.6.7.2/15-52
0x0C24	RapidIO outbound window translation extended address register 1 (ROWTEAR1)	R/W	0x0000_0000	15.6.7.3/15-53
0x0C28	RapidIO outbound window base address register 1 (ROWBAR1)	R/W	0x0000_0000	15.6.7.4/15-54
0x0C30	RapidIO outbound window attributes register 1 (ROWAR1)	R/W	0x0004_4023	15.6.7.5/15-54
0x0C34	RapidIO outbound window segment 1 register 1 (ROWS1R1)	R/W	0x0044_0000	15.6.7.6/15-56
0x0C38	RapidIO outbound window segment 2 register 1 (ROWS2R1)	R/W	0x0044_0000	15.6.7.6/15-56
0x0C3C	RapidIO outbound window segment 3 register 1 (ROWS3R1)	R/W	0x0044_0000	15.6.7.6/15-56
0x0C40 – 0x0CFC	RapidIO outbound window 2 through outbound window 7	—	—	—

Table A-16. RapidIO Implementation Registers (continued)

RapidIO Implementation Registers—Block Base Address 0xD_0000				
Offset	**Register**	**Access**	**Reset[1]**	**Section/Page**
0x0D00	RapidIO outbound window translation address register 8 (ROWTAR8)	R/W	0x0000_0000	15.6.7.2/15-52
0x0D04	RapidIO outbound window translation extended address register 8 (ROWTEAR8)	R/W	0x0000_0000	15.6.7.3/15-53
0x0D08	RapidIO outbound window base address register 8 (ROWBAR8)	R/W	0x0000_0000	15.6.7.4/15-54
0x0D10	RapidIO outbound window attributes register 8 (ROWAR8)	R/W	0x0004_4023	15.6.7.5/15-54
0x0D14	RapidIO outbound window segment 1 register 8 (ROWS1R8)	R/W	0x0044_0000	15.6.7.6/15-56
0x0D18	RapidIO outbound window segment 2 register 8 (ROWS2R8)	R/W	0x0044_0000	15.6.7.6/15-56
0x0D1C	RapidIO outbound window segment 3 register 8 (ROWS3R8)	R/W	0x0044_0000	15.6.7.6/15-56
0x0D60	RapidIO Inbound window translation address register 4 (RIWTAR4)	R/W	0x0000_0000	15.6.7.7/15-57
0x0D68	RapidIO Inbound window base address register 4 (RIWBAR4)	R/W	0x0000_0000	15.6.7.8/15-58
0x0D70	RapidIO inbound window attributes register 4 (RIWAR4)	R/W	0x0004_4021	15.6.7.9/15-59
0x0D80–0x0DBC	RapidIO inbound window 3 through inbound window 2			
0x0DC0	RapidIO inbound window translation address register 1 (RIWTAR1)	R/W	0x0000_0000	15.6.7.7/15-57
0x0DC8	RapidIO inbound window base address register 1 (RIWBAR1)	R/W	0x0000_0000	15.6.7.8/15-58
0x0DD0	RapidIO inbound window attributes register 1 (RIWAR1)	R/W	0x0004_4021	15.6.7.9/15-59
0x0DE0	RapidIO inbound window translation address register 0 (RIWTAR0)	R/W	0x0000_0000	15.6.7.7/15-57
0x0DF0	RapidIO inbound window attributes register 0 (RIWAR0)	Mixed	0x8004_4021	15.6.7.9/15-59
RapidIO Message Unit				
RapidIO Outbound Message Unit 0 Registers				
0x3000	Outbound message 0 mode register (OM0MR)	R/W	0x0000_0000	15.7.1.1/15-61
0x3004	Outbound message 0 status register (OM0SR)	Mixed	0x0000_0000	15.7.1.2/15-63
0x3008	Extended outbound message 0 descriptor queue dequeue pointer address register (EOM0DQDPAR)	R/W	0x0000_0000	15.7.1.3/15-64
0x300C	Outbound message 0 descriptor queue dequeue pointer address register (OM0DQDPAR)	R/W	0x0000_0000	15.7.1.3/15-64
0x3010	Extended outbound message 0 source address register (EOM0SAR)	R/W	0x0000_0000	15.7.1.5/15-67
0x3014	Outbound message 0 source address register (OM0SAR)	R/W	0x0000_0000	15.7.1.5/15-67
0x3018	Outbound message 0 destination port register (OM0DPR)	R/W	0x0000_0000	15.7.1.6/15-68
0x301C	Outbound message 0 destination attributes Register (OM0DATR)	R/W	0x0000_0000	15.7.1.7/15-69

Table A-16. RapidIO Implementation Registers (continued)

RapidIO Implementation Registers—Block Base Address 0xD_0000				
Offset	**Register**	**Access**	**Reset[1]**	**Section/Page**
0x3020	Outbound message 0 double-word count register (OM0DCR)	R/W	0x0000_0000	15.7.1.8/15-69
0x3024	Extended outbound message 0 descriptor queue enqueue pointer address register (EOM0DQEPAR)	R/W	0x0000_0000	15.7.1.4/15-66
0x3028	Outbound message 0 descriptor queue enqueue pointer address register (OM0DQEPAR)	R/W	0x0000_0000	15.7.1.4/15-66
0x302C	Outbound message 0 retry error threshold configuration register (OM0RETCR)	R/W	0x0000_0000	15.7.1.9/15-70
0x3030	Outbound message 0 multicast group register (OM0MGR)	R/W	0x0000_0000	15.7.1.10/15-71
0x3034	Outbound message 0 multicast list register (OM0MLR)	R/W	0x0000_0000	15.7.1.11/15-71
RapidIO Inbound Message Unit 0 Registers				
0x3060	Inbound message 0 mode register (IM0MR)	R/W	0x0000_0000	15.7.2.1/15-72
0x3064	Inbound message 0 status register (IM0SR)	Mixed	0x0000_0002	15.7.2.2/15-74
0x3068	Extended inbound message 0 frame queue dequeue pointer address register (EIM0FQDPAR)	R/W	0x0000_0000	15.7.2.3/15-75
0x306C	Inbound message 0 frame queue dequeue pointer address register (IM0FQDPAR)	R/W	0x0000_0000	15.7.2.3/15-75
0x3070	Extended inbound message 0 frame queue enqueue pointer address register (EIM0FQEPAR)	R/W	0x0000_0000	15.7.2.4/15-76
0x3074	Inbound message 0 frame queue enqueue pointer address register (IM0FQEPAR)	R/W	0x0000_0000	15.7.2.4/15-76
0x3078	Inbound message 0 maximum interrupt report interval register (IM0MIRIR)	R/W	0xFFFF_FF00	15.7.2.5/15-77
RapidIO Outbound Message Unit 1 Registers				
0x3100	Outbound message 1 mode register (OM1MR)	R/W	0x0000_0000	15.7.1.1/15-61
0x3104	Outbound message 1 status register (OM1SR)	Mixed	0x0000_0000	15.7.1.2/15-63
0x3108	Extended outbound message 1 descriptor queue dequeue pointer address register (EOM1DQDPAR)	R/W	0x0000_0000	15.7.1.3/15-64
0x310C	Outbound message 1 descriptor queue dequeue pointer address register (OM1DQDPAR)	R/W	0x0000_0000	15.7.1.3/15-64
0x3110	Extended outbound message 1 source address register (EOM1SAR)	R/W	0x0000_0000	15.7.1.5/15-67
0x3114	Outbound message 1 source address register (OM1SAR)	R/W	0x0000_0000	15.7.1.5/15-67
0x3118	Outbound message 1 destination port register (OM1DPR)	R/W	0x0000_0000	15.7.1.6/15-68
0x311C	Outbound message 1 destination attributes register (OM1DATR)	R/W	0x0006_0000	15.7.1.7/15-69
0x3120	Outbound message 1 double-word count register (OM1DCR)	R/W	0x0000_0000	15.7.1.8/15-69
0x3124	Extended outbound message 1 descriptor queue enqueue pointer address register (EOM1DQEPAR)	R/W	0x0000_0000	15.7.1.4/15-66
0x3128	Outbound message 1 descriptor queue enqueue pointer address register (OM1DQEPAR)	R/W	0x0000_0000	15.7.1.4/15-66

Table A-16. RapidIO Implementation Registers (continued)

RapidIO Implementation Registers—Block Base Address 0xD_0000				
Offset	**Register**	**Access**	**Reset[1]**	**Section/Page**
0x312C	Outbound message 1 retry error threshold configuration register (OM1RETCR)	R/W	0x0000_0000	15.7.1.9/15-70
0x3130	Outbound message 1 multicast group register (OM1MGR)	R/W	0x0000_0000	15.7.1.10/15-71
0x3134	Outbound message 1 multicast list register (OM1MLR)	R/W	0x0000_0000	15.7.1.11/15-71
RapidIO Inbound Message Unit 1 Registers				
0x3160	Inbound message 1 mode register (IM1MR)	R/W	0x0000_0000	15.7.2.1/15-72
0x3164	Inbound message 1 status register (IM1SR)	Mixed	0x0000_0002	15.7.2.2/15-74
0x3168	Extended inbound message 1 frame queue dequeue pointer address register (EIM1FQDPAR)	R/W	0x0000_0000	15.7.2.3/15-75
0x316C	Inbound message 1 frame queue dequeue pointer address register (IM1FQDPAR)	R/W	0x0000_0000	15.7.2.3/15-75
0x3170	Extended inbound message 1 frame queue enqueue pointer address register (EIM1FQEPAR)	R/W	0x0000_0000	15.7.2.4/15-76
0x3174	Inbound message 1 frame queue enqueue pointer address register (IM1FQEPAR)	R/W	0x0000_0000	15.7.2.4/15-76
0x3178	Inbound message 1 maximum interrupt report interval register (IM1MIRIR)	R/W	0xFFFF_FF00	15.7.2.5/15-77
RapidIO Doorbell Registers				
0x3400	Outbound doorbell mode register (ODMR)	R/W	0x0000_0000	15.7.3.1/15-78
0x3404	Outbound doorbell status register (ODSR)	Mixed	0x0000_0000	15.7.3.2/15-79
0x3408–0x3414	Reserved	—	—	—
0x3418	Outbound doorbell destination port register (ODDPR)	R/W	0x0000_0000	15.7.3.3/15-79
0x341C	Outbound doorbell destination attributes register (ODDATR)	R/W	0x0000_0000	15.7.3.4/15-80
0x3420–0x3428	Reserved	—	—	—
0x342C	Outbound doorbell retry error threshold configuration register (ODRETCR)	R/W	0x0000_0000	15.7.3.5/15-81
0x3430–0x345C	Reserved	—	—	—
0x3460	Inbound doorbell mode register (IDMR)	R/W	0x0000_0000	15.7.4.1/15-82
0x3464	Inbound doorbell status register (IDSR)	Mixed	0x0000_0002	15.7.4.2/15-83
0x3468	Extended inbound doorbell queue dequeue pointer address register (EIDQDPAR)	R/W	0x0000_0000	15.7.4.3/15-84
0x346C	Inbound doorbell queue dequeue Pointer address register (IDQDPAR)	R/W	0x0000_0000	15.7.4.3/15-84
0x3470	Extended inbound doorbell queue enqueue pointer address register (EIDQEPAR)	R/W	0x0000_0000	15.7.4.4/15-85
0x3474	Inbound doorbell Queue enqueue pointer address register (IDQEPAR)	R/W	0x0000_0000	15.7.4.4/15-85

Table A-16. RapidIO Implementation Registers (continued)

RapidIO Implementation Registers—Block Base Address 0xD_0000				
Offset	**Register**	**Access**	**Reset[1]**	**Section/Page**
0x3478	Inbound doorbell maximum interrupt report interval register (IDMIRIR)	R/W	0xFFFF_FF00	15.7.4.5/15-86
0x347C	Reserved	—	—	—
RapidIO Port-Write Registers				
0x34E0	Inbound port-write mode register (IPWMR)	R/W	0x0000_0000	15.7.5.1/15-87
0x34E4	Inbound port-write status register (IPWSR)	Mixed	0x0000_0000	15.7.5.2/15-88
0x34E8	Extended inbound port-write queue base address register (EIPWQBAR)	R/W	0x0000_0000	15.7.5.3/15-88
0x34EC	Inbound port-write queue base address register (IPWQBAR)	R/W	0x0000_0000	15.7.5.3/15-88

1 Values indicated with *n* are read from configuration signals at reset; see register description for details.

A.4 Device-Specific Utilities

The device-specific utilities registers control functions that are not particular to a functional unit but to the device as a whole; they occupy the highest 256 Kbytes of CCSR space (0xE_0000–0xF_FFFF).

A.4.1 Global Utilities

Table A-17. Global Utilities Registers

Global Utilities—Block Base Address 0xE_0000				
Offset	**Register**	**Access**	**Reset [1]**	**Section/Page**
Power-On Reset Configuration Values				
0x000	PORPLLSR—POR PLL ratio status register	R	0x00*nn*_*n*0*nn*	17.4.1.1/17-6
0x004	PORBMSR—POR boot mode status register	R	0x*nnnn*_0000	17.4.1.2/17-7
0x008	PORIMPCR—POR I/O impedance control register	Mixed	0x000*n*_007F	17.4.1.3/17-8
0x00C	PORDEVSR—POR I/O device status register	R	see ref***.	17.4.1.4/17-9
0x010	PORDBGMSR—POR debug mode status register	R	see ref.	17.4.1.5/17-12
0x020	PORCIR—POR configuration information register	R	see ref.	17.4.1.6/17-13
Signal Multiplexing and GPIO Controls				
0x030	GPIOCR—GPIO control register	R/W	0x0000_0000	17.4.1.7.1/17-14
0x040	GPOUTDR—General-purpose output data register	R/W	0x0000_0000	17.4.1.7.2/17-15
0x050	GPINDR—General-purpose input data register	R	0x*nnnn*_*nnnn*	17.4.1.7.3/17-15
0x060	PMUXCR—Alternate function signal multiplex control	R/W	0x0000_0000	17.4.1.8/17-16
Device Disables				

Table A-17. Global Utilities Registers (continued)

Global Utilities—Block Base Address 0xE_0000				
Offset	**Register**	**Access**	**Reset** [1]	**Section/Page**
0x070	DEVDISR—Device disable control	R/W	0x0000_0000	17.4.1.9/17-17
Power Management Registers				
0x080	POWMGTCSR—Power management status and control register	Mixed	0x0000_0000	17.4.1.10/17-20
Interrupt and Reset Status and Control Registers				
0x090	MCPSUMR—Machine check summary register	w1c	0x0000_0000	17.4.1.11/17-22
0x094	RSTRSCR—Reset request status and control register	Mixed	0x0000_0000	17.4.1.12/17-23
Version Registers				
0x0A0	PVR—Processor version register	R	0x8004_*nnnn*	17.4.1.13/17-24
0x0A4	SVR—System version register	R	0x8090_00*nn* (MPC8641) 0x8090_01*nn* (MPC8641D)	17.4.1.14/17-24
Reset Registers				
0x0B0	RSTCR—Reset control register	R/W	0x0000_0000	17.4.1.15/17-25
DDR Clock Control				
0xB20	DDR1CLKDR—DDRC1 Clock Disable Register	R/W	0x0000_0000	17.4.1.16/17-26
0xB28	DDR2CLKDR—DDRC2 Clock Disable Register	R/W	0x0000_0000	17.4.1.17/17-26
Clock and SerDes Control				
0xE00	CLKOCR—Clock out control register	R/W	0x0000_0000	17.4.1.18/17-27
0xF04	SRDS1CR0—SerDes1 control register 0	R/W	0x*nn*00_*nn*00	17.4.1.19/17-28
0xF08	SRDS1CR1—SerDes1 control register 1	R/W	0x0000_0000	17.4.1.20/17-30
0xF40	SRDS2CR0—SerDes2 control register 0	R/W	0x*nn*00_*nn*00	17.4.1.21/17-31
0xF44	SRDS2CR1—SerDes2 control register 1	R/W	0x0000_0000	17.4.1.22/17-32

[1] Bits indicated with *n* are set from configuration signals.

A.4.2 Performance Monitor

Table A-18. Performance Monitor Registers

Performance Monitor—Block Base Address 0xE_1000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0x000	PMGC0—Performance monitor global control register	R/W	0x0000_0000	18.3.2.1/18-5
0x010	PMLCA0—Performance monitor local control register A0	R/W	0x0000_0000	18.3.2.2/18-5
0x014	PMLCB0—Performance monitor local control register B0	R/W	0x0000_0000	18.3.2.2/18-5
0x018	PMC0 (lower)—Performance monitor counter 0 lower	R/W	0x0000_0000	18.3.3.1/18-9
0x01C	PMC0 (upper)—Performance monitor counter 0 upper	R/W	0x0000_0000	18.3.3.1/18-9
0x020	PMLCA1—Performance monitor local control register A1	R/W	0x0000_0000	18.3.2.2/18-5
0x024	PMLCB1—Performance monitor local control register B1	R/W	0x0000_0000	18.3.2.2/18-5
0x028	PMC1—Performance monitor counter 1	R/W	0x0000_0000	18.3.3.1/18-9
0x030	PMLCA2—Performance monitor local control register A2	R/W	0x0000_0000	18.3.2.2/18-5
0x034	PMLCB2—Performance monitor local control register B 2	R/W	0x0000_0000	18.3.2.2/18-5
0x038	PMC2—Performance monitor counter 2	R/W	0x0000_0000	18.3.3.1/18-9
0x040	PMLCA3—Performance monitor local control register A3	R/W	0x0000_0000	18.3.2.2/18-5
0x044	PMLCB3—Performance monitor local control register B3	R/W	0x0000_0000	18.3.2.2/18-5
0x048	PMC3—Performance monitor counter 3	R/W	0x0000_0000	18.3.3.1/18-9
0x050	PMLCA4—Performance monitor local control register A4	R/W	0x0000_0000	18.3.2.2/18-5
0x054	PMLCB4—Performance monitor local control register B4	R/W	0x0000_0000	18.3.2.2/18-5
0x058	PMC4—Performance monitor counter 4	R/W	0x0000_0000	18.3.3.1/18-9
0x060	PMLCA5—Performance monitor local control register A5	R/W	0x0000_0000	18.3.2.2/18-5
0x064	PMLCB5—Performance monitor local control register B 5	R/W	0x0000_0000	18.3.2.2/18-5
0x068	PMC5—Performance monitor counter 5	R/W	0x0000_0000	18.3.3.1/18-9
0x070	PMLCA6—Performance monitor local control register A6	R/W	0x0000_0000	18.3.3.1/18-9
0x074	PMLCB6—Performance monitor local control register B6	R/W	0x0000_0000	18.3.2.2/18-5
0x078	PMC6—Performance monitor counter 6	R/W	0x0000_0000	18.3.3.1/18-9
0x080	PMLCA7—Performance monitor local control register A7	R/W	0x0000_0000	18.3.2.2/18-5
0x084	PMLCB7—Performance monitor local control register B7	R/W	0x0000_0000	18.3.2.2/18-5
0x088	PMC7—Performance monitor counter 7	R/W	0x0000_0000	18.3.3.1/18-9
0x090	PMLCA8—Performance monitor local control register A8	R/W	0x0000_0000	18.3.2.2/18-5
0x094	PMLCB8—Performance monitor local control register B8	R/W	0x0000_0000	18.3.2.2/18-5
0x098	PMC8—Performance monitor counter 8	R/W	0x0000_0000	18.3.3.1/18-9
0x0A0	PMLCA9—Performance monitor local control register A9	R/W	0x0000_0000	18.3.2.2/18-5

Table A-18. Performance Monitor Registers (continued)

Performance Monitor—Block Base Address 0xE_1000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0x0A4	PMLCB9—Performance monitor local control register B9	R/W	0x0000_0000	18.3.2.2/18-5
0x0A8	PMC9—Performance monitor counter 9	R/W	0x0000_0000	18.3.3.1/18-9

A.4.3 Watchpoint Monitor and Trace Buffer

Table A-19. Watchpoint Monitor and Trace Buffer Registers

Watchpoint Monitor and Trace Buffer—Block Base Address 0xE_2000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
Watchpoint Monitor Registers				
0x000	WMCR0—Watchpoint monitor control register 0	R/W	0x0000_0000	19.3.1.1/19-11
0x004	WMCR1—Watchpoint monitor control register 1	R/W	0x0000_0000	19.3.1.1/19-11
0x008	Watchpoint Monitor Address High Register (WMAHR)	R/W	0x0000_0000	19.3.1.2/19-13
0x00C	WMAR—Watchpoint monitor address register	R/W	0x0000_0000	19.3.1.3/19-13
0x010	Watchpoint Monitor Address Mask High Register (WMAMHR)	R/W	0x0000_0000	19.3.1.4/19-14
0x014	WMAMR—Watchpoint monitor address mask register	R/W	0x0000_0000	19.3.1.5/19-14
0x018	WMTMR—Watchpoint monitor transaction mask register	R/W	0x0000_0000	19.3.1.6/19-15
0x01C	WMSR—Watchpoint monitor status register	R/W	0x0000_0000	19.3.1.7/19-16
Trace Buffer Registers				
0x040	TBCR0—Trace buffer control register 0	R/W	0x0000_0000	19.3.2.1/19-17
0x044	TBCR1—Trace buffer control register 1	R/W	0x0000_0000	19.3.2.1/19-17
0x048	Trace Buffer Address High Register (TBAHR)	R/W	0x0000_0000	19.3.2.2/19-19
0x04C	TBAR—Trace buffer address register	R/W	0x0000_0000	19.3.2.3/19-20
0x050	Trace Buffer Address Mask High Register (TBAMHR)	R/W	0x0000_0000	19.3.2.4/19-20
0x054	TBAMR—Trace buffer address mask register	R/W	0x0000_0000	19.3.2.5/19-21
0x058	TBTMR—Trace buffer transaction mask register	R/W	0x0000_0000	19.3.2.6/19-21
0x05C	TBSR—Trace buffer status register	R/W	0x0000_0000	19.3.2.7/19-22
0x060	TBACR—Trace buffer access control register	R/W	0x0000_0000	19.3.2.8/19-23
0x064	TBADHR—Trace buffer access data high register	R/W	0x0000_0000	19.3.2.9/19-23
0x068	TBADR—Trace buffer access data register	R/W	0x0000_0000	19.3.2.10/19-24
Context ID Registers				
0x0A0	PCIDR—Programmed context ID register	R/W	0x0000_0000	19.3.3.1/19-25

Table A-19. Watchpoint Monitor and Trace Buffer Registers (continued)

Watchpoint Monitor and Trace Buffer—Block Base Address 0xE_2000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0x0A4	CCIDR—Current context ID register	R/W	0x0000_0000	19.3.3.2/19-25
Other Registers				
0x0B0	TOSR—Trigger output source register	R/W	0x0000_0000	19.3.4.1/19-26

Appendix B
Revision History

This appendix provides a list of the major differences between revisions of the *MPC8641D Integrated Host Processor Reference Manual*.

B.1 Changes From Revision 1 to Revision 2

Major changes to this document from Revision 1 to Revision 2 are as follows:

NOTE

All references are to the current (Revision 2) document.

Section/Page	Changes
1.3.1/1-9	Changed “Total latency...” bullet under “On-chip level 2 (L2) cache ...” to 11.5 and 12.5 cycles when ECC is enabled.
1.3.2/1-14	In list of features, changed DDR data rate to 600 MHz and 300 MHz.
1.5.1.2/11-29	Removed Figure 1-8, “Key Areas for Bandwidth Performance.”
2.2.2/2-6	Added Section 2.2.2, “Precedence of Local Access Windows.”
2.4/2-10	Added a note, as follows: “The CCSR window always takes precedence over all local access windows. However, the CCSR window must not overlap an LAW that maps to the DDR controller. Otherwise, undefined behavior occurs.”
2.4.1/2-11	Added Section 2.4.1, “Accessing CCSR Memory from the Local Processor.”
3.2/3-24	In Table 3-3, “MPC8641D Reset Configuration Signals,” changed default settings for cfg_sys_pll[0:3] to 1111, cfg_core_pll[0:3] to 1111, and cfg_core_pll[4] to 1.
4.3.1.3.1/4-8	In Figure 4-4, “Boot Page Translation Register (BPTR),” changed offset of BPTR from 0x010 to 0x020.
4.4.2/4-9	Revised note to reference recommendations in the hardware specifications.
4.4.2/4-9	Revised step 4.
4.4.3.3/4-13	In Table 4-11, “e600 Core Clock PLL Ratios,” changed default setting for cfg_core_pll[0:4] to 11111.
4.4.3.4/4-14	In Table 4-12, “Core 1 Enable,” changed default setting for cfg_sys_pll[0:3] to to 1111.
4.4.3.7/4-15	Clarified that the POR configuration option only affects the outbound ATMU window 3 of PCI Express controller 1.
5.3.4.2.1/5-31	Added Section 5.3.4.2.1, “Sources of tea_ assertion.”

6.3.6.3.2/6-87	Changed the last sentence of the fourth paragraph to say the following: “Note that the broadcast of **tlbie** and **tlbsync** instructions is enabled by the setting of HID1[ABE].”
7.4.1.3/7-6	In Table 7-4, “PCR Field Descriptions,” revised bit description for PORT1_EN to state it is not intended to dynamically enabled and disabled. In addition,in PORT1_PRI description, “Highest priority level” should have the encoding 10.
7.4.1.4/7-7	In Table 7-5, “EDR Field Descriptions,” revised LAE field description to identify that **tlbie** broadcasts can generate LAEs.
7.4.1.6/7-9	In Table 7-7, “EATR Field Descriptions,” revised bit encodings for PCI Express 2 to now say, “PCI Express 2/RapidIO,” and changed the former RapidIO encoding to reserved.
7.4.1.6/7-9	In Table 7-7, “EATR Field Descriptions,” revised the SRC_ID field description to now use the following encodings: 10000 Core 0 (instruction/data) 10001 Reserved 10010 Core 1 (instruction/data) 10011–10100 Reserved
8.3.1/8-3	In Table 8-1, “DDR Memory Interface Signal Summary,” added a note clarifying that some devices that implement two DDR controllers may share MDVAL and MSRCID[0:4] signals between both controllers; other devices may offer a set for each controller.
8.3.2.2/8-10	In Table 8-4, “Clock Signals—Detailed Signal Descriptions,” added the following to the MCKE description: “The MCKE signals should be connected to the same rank of memory as the corresponding MCS and MODT signals. For example, MCKE[0] should be connected to the same rank of memory as MCS[0] and MODT[0].”
8.4.1.7/8-22	In Table 8-12, “DDR_SDRAM_CFG Field Descriptions,” in 8_BE field description, modified note to say the following: “DDR1 (SDRAM_TYPE = 010) must use 8-beat bursts when using 32-bit bus mode (32_BE = 1) and 4-beat bursts when using 64-bit bus mode; DDR2 (SDRAM_TYPE = 011) must use 4-beat bursts, even when using 32-bit bus mode.”
8.4.1.7/8-22	In Table 8-12, “DDR_SDRAM_CFG Field Descriptions,” added new programming requirement for DDR_SDRAM_CFG[HSE] such that this bit should not be set if using automatic calibration.
8.4.1.14/8-31	Changed introductory sentence for Figure 8-15 to say the following: “The DDR SDRAM clock control configuration register, shown in Figure 8-15, provides a 1/8-cycle clock adjustment.”

8.4.1.28/8-40	In Figure 8-29, “Memory Data Path Read Capture ECC Register (CAPTURE_ECC),” extended bit field for ECE from 24:31 to 16:31. Added detailed bit field description after generic ECE statement.
8.4.1.32/8-43	In Table 8-38, “CAPTURE_ATTRIBUTES Field Descriptions,” modified TSIZ field description.
8.4.1.32/8-43	In Table 8-38, “CAPTURE_ATTRIBUTES Field Descriptions,” revised bit encodings for TSRC field.
8.5.6/8-64	Added note after first paragraph clarifying system board requirements when using registered DIMMs.
8.5.11/8-70	Added the following text to the end of the section: “In 32-bit mode, Table 8-55 is split into 2 halves. The first half, consisting of rows 0–31, is used to calculate the ECC bits for the first 32 data bits of any 64-bit granule of data. This always applies to the odd data beats on the DDR data bus. The second half of the table, consisting of rows 32–63, is used to calculate the ECC bits for the second 32 bits of any 64-bit granule of data. This always applies to the even data beats on the DDR data bus.”
8.6.1/8-74	In Table 8-59, “Programming Differences between Memory Types,” for ODT_PD_EXIT, changed it to be set to 0001 for DDR1; for FOUR_ACT, changed it to be set for 00001 for DDR1.
Chapter 9/9-1	Changed all instances of “Message Shared” to “Message Signaled” throughout. In addition, revised description for Activity bit to reference the respective xxVPRs throughout.
9.1.1/9-1	In Figure 9-1, “Interrupt Sources Block Diagram Features,” changed reference in Internal Interrupts block to Table 9-2, “Internal Interrupt Assignments.”
9.2.2/9-9	In Table 9-3, “Interrupt Signals—Detailed Signal Descriptions,” removed the following note from IRQ*n*: “If IRQ[0:3] or IRQ[4:7] are used to receive INTx signals from PCI Express port 1 and port 2, respectively, as a root complex, the polarity and sense of each of these signals must be set to be active-high and level-sensitive.”
9.3.1.6/9-21	Added clarification to the proper used of bits in the PIR register. Namely, core reset request should not be cleared until the requested core reset has occurred.
9.3.1.6/9-21	Added the following sentence in paragraph before Figure 9-8: “However, if one core is used to reset another one, the core being reset can effectively be held off indefinitely from issuing its initial boot vector fetch to the platform by leaving its appropriate PIR[Px] bit asserted.”
9.3.6.3/9-38	In Figure 9-34, “Shared Message Signaled Interrupt Index Register (MSIIR),” changed location of SRS and IBS fields in MSIIR register. Note that for the MPC8641D Rev 2.0 and earlier, the SRS and IBS fields occupy bits 24-31.
9.3.7.1/9-42	Removed last sentence regarding polarity and sense for IRQ[0:7] and PEX INTx sharing. This is fully covered in Section 9.4.5, “PCI Express INTx.”

9.3.7.1/9-42	In Table 9-39, “EIVPRn Field Descriptions,” removed note in bit description for P field requiring it to be set to active high for PCI Express INTx.
9.4.3/9-57	Removed Section 9.4.3.1, “Shared Message Signaled Interrupts,” and Section 9.4.3.2, “PCI Express INTx.”
10.3.1.2/10-6	In Figure 10-3, “I^2C Frequency Divider Register (I2CFDR),” changed reset value of I2CFDR to “All zeros.”
10.3.1.2/10-6	In Table 10-5, “I2CFDR Field Descriptions,” revised FDR field description.
10.3.1.5/10-10	In Table 10-8, “I2CDR Field Descriptions,” in DATA description, modified last sentence to say, “Note that in both master receive and slave receive modes, the very first read is always a dummy read.”
10.4.5/10-17	Revised description of serial bit clock in description.
10.5.4/10-22	Removed sentence, “For 1-byte transfers, a dummy read should be performed by the interrupt service routine.”
11.3.1.3/11-6	In Table 11-7, “Baud Rate Examples,” replaced 667 MHz entries with 600 MHz entries.
12.3.1.13/12-27	Added Figure 12-16, “Transfer Error Attributes Register (LTEATR).”
12.3.1.13/12-27	In Table 12-19, “LTEATR Field Descriptions,” added fields XA (bits 28–29) and Reserved (bit 30).
12.3.1.13/12-27	In Table 12-19, “LTEATR Field Descriptions,” revised bit encodings in SRC_ID field description.
12.4.4.4.1/12-64	In Table 12-28, “RAM Word Field Descriptions,” added the following note to fields LOOP and AMX: “AMX must not change values in any RAM word which begins a loop.”
12.4.4.4.7/12-69	Added the following note: “AMX must not change values in any RAM word which begins a loop.”
13.3/13-4	Modified first bullet, removing specific maximum data clock frequency ratios, referring reader to the device hardware specificiations document for specific maximum frequencies. In addition, changed “eTSECs 0 and 1, 2, and 3,” to “eTSECs 1, 2, 3, and 4.”
13.4/13-6	In Table 13-1, “eTSECn Network Interface Signal Properties,” added MII to designated modes during which TXD[7:4] and RXD[7:4] are unused.
13.4/13-6	In Table 13-1, “eTSECn Network Interface Signal Properties,” for RGMII and RTBI protocols, changed description of TSEC_RX_ER from “Unused, output driven low” to “Unused.”
13.4/13-6	In Table 13-1, “eTSECn Network Interface Signal Properties,” modified TSEC*n*_TX_CLK signal description, removing specific maximum receive clock frequency ratios; referring reader to the device hardware specificiations document for specific maximum frequencies.
13.4.1/13-8	In Table 13-2, “eTSEC Signals—Detailed Signal Descriptions,” for signal TSEC*n*_GTX_CLK, removed RMII text from all devices: “In RMII mode,

	TSEC*n*_GTX_CLK feeds back the effective transmit clock according to the interface, 100Base-T is 25 MHz and 10Base-T is 2.5 MHz.”
13.4.1/13-8	In Table 13-2, “eTSEC Signals—Detailed Signal Descriptions,” changed TSEC*n*_TX_EN signal description from “rising and falling edges of the TSEC*n*_TX_CLK, respectively” to “rising and falling edges of the TSEC*n*_GTX_CLK, respectively.”
13.4.1/13-8	In Table 13-2, “eTSEC Signals—Detailed Signal Descriptions,” changed the “State Meaning” description for the TSEC*n*_CRS.
13.4.1/13-8	In Table 13-2, “eTSEC Signals—Detailed Signal Descriptions,” modified TSEC*n*_RX_CLK signal description, removing specific maximum receive clock frequency ratios; referring reader to the device hardware specificiations document for specific maximum frequencies.
13.5.2/13-13	Removed eTSEC FIFO Control and Status Registers.
13.5.3.1.2/13-24	In Table 13-6, “TSEC_ID2 Field Descriptions,” changed description for TSEC_INT field.
13.5.3.1.6/13-32	In Table 13-11, “ECNTRL Field Descriptions,” and Table 13-12, “eTSEC Interface Configurations,” updated RMM field description.
13.5.3.2.1/13-37	In Table 13-16, “TCTRL Field Descriptions,” clarified TFC_PAUSE field description.
13.5.3.2.2/13-39	In Table 13-17, TSTAT Field Descriptions,” replaced the following sentence in the introduction: “Only enabled rings can indicate halt state” with the following: “The halt bit only has meaning for enabled rings.” In addition, changed THLTn descriptions from, “This bit is set only on a general error condition (as in IEVENT[TXE]) or...” to “This bit is set only on a general error condition (as in IEVENT[TXE]), regardless of TQUEUE[ENn], or...”
13.5.3.3.1/13-49	Table 13-27, “RCTRL Field Descriptions,” clarified RSF field description.
13.5.3.3.4/13-55	In Figure 13-25, “RQUEUE Register Definition,” and Table 13-30, “RQUEUE Field Descriptions,” removed EX0–EX7 fields.
13.5.3.3.5/13-55	In Table 13-31, “RBIFX Field Descriptions,” modified B*n*CTL field descriptions to clarify that arbitrary extraction of preamble is not supported in FIFO modes.
13.5.3.3.8/13-59	In Table 13-34, “RQFPR Field Descriptions,” updated ETY field description with the following note: “Packets with a value in the length/type field greater than 1500 and less than 1536 are treated as payload length. If the eTSEC is used in a network where there are packets carring a type designation between 1500 and 1536 (note there are none currently publically defined by IANA), then the S/W must confirm the parser and filer results by checking the type/length field after the packet has been written to memory to see if it falls in this range.”
13.5.3.3.8/13-59	In Table 13-34, “RQFPR Field Descriptions,” added the following text to IPF field description:

	"See the descriptions of receive FCB fields IP and PRO in Section 13.6.4.3, "Receive Path Off-Load," for more information on determining the status of received packets for which IPF is set."
13.5.3.3.8/13-59	In Table 13-34, "RQFPR Field Descriptions," added clarification to ETY field description describing parser/filer behavior with ethertype field (parsed into ETY), as follows: "Note that the eTSEC filer gets multiple packet attributes as a result of parsing the packet. The behavior of the eTSEC is that it will pull the innermost ethertype found in the packet; this means that in many supported protocols, it is impossible to create a filer rule that will match on the outer ethertype. There are four cases that need to be highlighted: 1. The jumbo ethertype (0x8870)—In this case, the eTSEC assumes that the following header is LLC/SNAP. LLC/SNAP has an associated etherype, and the ETY field will be populated with that ethertype. This makes it impossible to file on jumbo frames. In this case, one can use arbitrary extracted bytes to pull the outermost Ethertype. 2. The PPPoE ethertype described above. 3. The VLAN tag ethertype (0x8100)—In this case, one can use the PID1 VLN bit to indicate that the packet had a VLAN tag. 4. MPLS tagged packets; this is a defect as described in IPGear 435. In this case, one can use arbitrary extraction bytes to compare to the actual ethertype if a filer rule is intending to file based on an MPLS label existence."
13.5.3.5.2/13-69	In Table 13-41, "MACCFG2 Field Descriptions," clarified that TX preamble length may be from 0x3 to 0xF.
13.5.3.5.2/13-69	In Table 13-41, "MACCFG2 Field Descriptions," added note to PAD/CRC field description to include, "This bit must be set when in half-duplex mode (MACCFG2[Full Duplex] is cleared)."
13.5.3.6.26/13-93	Changed Table 13-81, "TPKT Field Descriptions," to show field TPKT as 22 bits (bits 10–31).
13.5.3.8.1/13-109	In Table 13-106, "FIFOCFG Field Descriptions," corrected IPG field description.
13.5.3.9.1/13-111	In Figure 13-103, "ATTR Register Definition," and Table 13-107, "ATTR Field Descriptions," removed ELCWT and BDLWT fields.
13.5.3.9.1/13-111	Removed Section 13.5.3.10.2, "Attribute Extract Length and Extract Index Register (ATTRELI)."
13.5.3.10/13-112	Added Section 13.5.3.10, "Lossless Flow Control Configuration Registers."
13.5.4.3.10/13-125	In Table 13-121, "TBICON Field Descriptions," updated/clarified the asserted state description for the Clock Select field.
13.6.2/13-136	Corrected the last bullet of this section regarding minimum inter-packet gap requirements.

13.6.3.8/13-154	Clarified last sentence of Magic Packet Mode description to include multicast packets.
13.6.3.11/13-158	Revised description of inter-frame gap timing.
13.6.3.13/13-158	In Table 13-140, “Reception Errors,” removed the following note for Parser error: **“Note:** Any values in the length/type field between 1500 and 1536 will be treated as a length, however, only illegal packets exist with this length/type since these are not valid lengths and not valid types. These are treated by the MAC logic as out of range. Software must confirm the parser and filer results by checking the type/length field after the packet has been written to memory to see if it falls in this range.”
13.6.4.3/13-162	In Table 13-141, “Rx Frame Control Block Descriptions,” modified description of PRO, as follows: “If IP = 1, PRO is set as follows: PRO=0xFF for a fragment header or a back to back route header PRO=0x*nn* for an unrecognized header, where *nn* is the next protocol field PRO=<TCP/UDP header>, as defined in the IANA specification, if TCP or UDP header is found If IP = 0, PRO is undefined.”
13.6.4.3/13-162	In Table 13-141, “Rx Frame Control Block Descriptions,” added the following text to IP field description, as follows: “If S/W is relying on the RxFCB for the parse results, any RxFCB[IP] bits set with the corresponding RxFCB[PRO] = 0xFF indicates a fragmented packet (or that this packet had a back-to-back IPv6 routing extension header. Additionally, RQFPR[IPF] (see Section 13.5.3.3.8, “Receive Queue Filer Table Property Register (RQFPR)”) indicates that the packet was fragmented.”
13.6.4.3/13-162	In Table 13-141, “Rx Frame Control Block Descriptions,” added the following text PRO field description, as follows: “Note that the eTSEC parser logic stops further parsing when encountering an IP datagram that has indicated that it has fragmented the upper layer protocol. This in general means that there is likely no layer 4 header following the IP header and extension headers. eTSEC leaves the RxFCB[PRO] and RQFPR[L4P] fields 0xFF in this case, which usually means that there was no IP header seen. In this case RxFCB[IP] and optionally RxFCB[IP6] will be set. IP header checksumming will operate and perform as intended. Most of the time, the eTSEC will update the RxFCB[PRO] field and RQFPR[L4P] fileds with whatever value was found in the protocol field of the IP header. See Section 13.5.3.3.8, “Receive Queue Filer Table Property Register (RQFPR),” for a description of RQFPR.”
13.6.6.2.1/13-176	Completed last sentence of second paragraph, as follows:

	"As soon as the hardware consumes a BD (by writing it back to memory), RBPTR*n* will advance and the free BD count will reflect the correct number of available free BDs."
13.6.7.1/13-177	Changed sentence in first paragraph to say, "Because of pre-fetching, a minimum of four buffer descriptors per ring are required."
13.6.7.3/13-181	In Table 13-149, "Receive Buffer Descriptor Field Descriptions," added recommendation to use 64-byte aligned receive buffer pointer addresses to description of Rx Data Buffer Pointer (offset 4–7, bits 0–31).
13.7.1.8/13-210	In Table 13-170, "16-Bit FIFO Interface Mode Signal Configuration (eTSECs 1 and 2)," and Table 13-171, "16-Bit FIFO Interface Mode Signal Configuration (eTSECs 3 and 4)," removed support for 3.3-V FIFO interface.
14.3.1.1/14-10	In Table 14-5, "MRn Field Descriptions," clarified CS field description.
14.4.1.3/14-33	Added clarification to third paragraph, describing external control functionality.
14.4.1.3/14-33	Added clarification to first bullet of fifth paragraph, describing the use of the signal, $\overline{\text{DMA\_DREQ}}$.
14.4.1.4/14-34	Modified second paragraph .
14.4.2/14-36	Added note that a single DMA transfer in any of the direct or chaining modes must not cross a 16GB (34-bit) address boundary.
14.5/14-42	Removed bullets also found in Section 14.5.1, "Unusual DMA Scenarios."
Chapter 15/15-1	Change all instances of OMMR[MM] to OMDATR[MM] throughout.
15.6.2.4/15-23	In Table 15-19, "GCCSR Field Descriptions," added notes clarifying that although the register is R/W it is, in fact, a status register (not a control register). As such, manually changing the value of bitfields does not affect logical behavior.
15.6.4.3/15-37	In Table 15-33, "ECACSR Field Descriptions," clarified description of ECI field.
15.10.2.1.7/15-172	In Table 15-130, "Outbound Doorbell Hardware Errors," replaced three instances of ODMR[EIE] with OM*n*MR[EIE].
16.1.1.1/16-2	Added note regarding checking the link status before issuing outbound transactions after reset or when recovering from a linkdown event.
16.3.1/16-5	In Table 16-3, "PCI Express Memory-Mapped Register Map," added footnote for reset value when alternate boot vector is selected.
16.3.1/16-5	In Table 16-3, "PCI Express Memory-Mapped Register Map," changed reset value for PEXOWAR3 to 0x0000_0000.
16.3.2.3/16-11	In Table 16-6, "PEX_OTB_CPL_TOR Field Descriptions," and Table 16-7, "PEX_CONF_RTY_TOR Field Descriptions," clarified description of TC (timeout counter) units for different clock frequencies.
16.3.5.1/16-19	In Table 16-14, "PCI Express Outbound Translation Address Registers (PEXOTARn)," Table 16-15, "PCI Express Outbound Translation Extended Address Registers (PEXOTEARn)," and Table 16-16, "PCI Express Outbound Window Base Address Registers (PEXOWBARn)," added footnote for reset value when alternate boot vector is selected.

16.3.5.1.4/16-22	In Figure 16-18, “PCI Express Outbound Window Attributes Registers 1–4 (PEXOWARn),” revised footnote for reset value of PEXOWAR3 when alternate boot vector is not selected.
16.3.6.1/16-29	In Table 16-23, “PCI Express Error Detect Register Field Descriptions,” added note recommending hot reset after a completion time-out is detected to bit description for PCT.
16.3.6.4/16-35	In Table 16-26, “PCI Express Error Capture Status Register Field Descriptions,” revised bit encodings for GSID field.
16.3.6.7/16-39	In Table 16-23, “PCI Express Error Detect Register Field Descriptions,” Table 16-24, “PCI Express Error Interrupt Enable Register Field Descriptions,” and Table 16-25, “PCI Express Error Disable Register Field Descriptions,” changed descriptions for OD0, OD1, and OD2 to “Internal platform transaction information. Reserved for factory debug.”
16.3.7.2/16-44	In the first paragraph, added statement that the PCI Express Controller Internal CSR registers are not accessible by inbound PCI Express configuration transactions.
16.3.9.11/16-74	In Table 16-86, “PCI Express Link Control Register Field Description,” augmented field description for RL (bit 5).
16.3.10/16-82	Added note to Figure 16-101, “PCI Express Extended Configuration Space,” stating that the PCI Express Controller Internal CSRs are not accessable by inbound PCI Express configuration transactions and any attempts to access these registers will return all 0s.
16.3.10/16-82	In Figure 16-101, “PCI Express Extended Configuration Space,” revised range for Internal CSR space to 0x400–0x6FF.
16.3.10.2/16-83	In Table 16-99, “PCI Express Uncorrectable Error Status Register Field Description,” added note recommending hot reset after a completion time-out is detected to bit description for CTO.
16.3.10.18/16-95	Clarified that the reset value of the Configuration Ready Register is defined during POR configuration (host/agent mode and CPU boot configuration).
16.4.1.9.1/16-103	In Table 16-122, changed entries for Assert INTA, INTB, INTC, INTD and Deassert INTA, INTB, INTC, INTD messages to supported in EP mode column and added, ”Sent upstream by endpoint,” to description.
16.4.1.10/16-107	Added Section 16.4.1.10, “Error Handling.”
16.4.1.10.3/16-111	In Table 16-126, “Error Conditions,” revised entry for Outbound response/Internal platform response with error to describe behavior when poison data occurs in the middle of the packet.
16.4.2.1.3/16-113	Revised paragraph, as follows: “Software can generate outbound assert or deassert INTx message transactions by using the outbound ATMU mechanism described in Section 16.4.1.9.1, “Outbound ATMU Message Generation.”
16.4.6/16-116	Added Section 16.4.6, “Link Down.”

17.4.1.2/17-7	In Figure 17-2, “POR Boot Mode Status Register (PORBMSR),” changed default value for ROM_LOC to *nnnn*.
17.4.1.4/17-9	In Table 17-7, “PORDEVSR Field Descriptions,” in entry for ECW3 = 1 changed “8-bit FIFO” to “16-bit FIFO.”
17.4.1.7/17-14	Revised section.
17.4.1.7.2/17-15	Revised section.
17.4.1.9/17-17	In Table 17-14, “DEVDISR Field Descriptions,” added note instructing software to place a core in sleep mode prior to disabling it with the DEVDISR.
17.4.1.10/17-20	Added Figure 17-12, “Power Management Control & Status Register (POWMGTCSR).”
17.4.1.20/17-30	In Figure 17-22, “SerDes 1 Control Register 1 (SRDS1CR1),” Table 17-26, “SRDS1CR1 Field Descriptions,” Figure 17-24, “SerDes 2 Control Register 1 (SRDS2CR1),” and Table 17-28, “SRDS2CR1 Field Descriptions,” added LBSELTYPE field.
18.3.1/18-3	Exchanged address offsets for PMC0 upper and PMC0 lower in Table 18-1, “Control Register Memory Map,” and Figure 18-7, “Performance Monitor Counter Register 0 (PMC0).”
18.4.7/18-14	In Table 18-10, “Performance Monitor Events Assignment,” revised entries for DDR1 event Ref:13 and DDR2 event Ref:24. In addition removed entries for MCM events C6:17, C8:15, C4:22, C4:17, C6:18, and C7:15.
19.4.1/19-27	Revised source and target ID encodings in Table 19-30, “Source and Target ID Values.”

B.2 Changes From Revision 0 to Revision 1

Major changes to this document from Revision 0 to Revision 1 are as follows:

NOTE

All references are to the Revision 1 document.

Section, Page	Changes
Throughout	Changed maximum local bus controller frequency to 133 MHz.
Chapter 3/3-1	Changed D1_MDM[0:7] and D2_MDM[0:7] to output-only (O) in all signal tables.
4.4.3.1/4-11	In Table 4-9, “MPX Clock PLL Ratio,” changed MPX Clock : SYSCLK Ratio for cfg_sys_pll[0:3] = 0000 to “Reserved.” The 16:1 ratio is not supported.
4.4.3.7/4-14	Changed outbound window affected by alternate boot vector from window 1 to window 3.
4.4.3.8/4-15	Revised several entries in Table 4-16. Specifically, added a new (non-reserved) entry for 0000 and changed the reference clock requirements on SerDes2 to 100 MHz for settings of 0011, 0110, 0111, 1010, 1011, 1100, and 1110.

4.4.4.2/4-23	For Serial RapidIO interface at 2.5 Gbps and 1.25 Gbps bit rate, changed the reference clock frequency entries to 100 MHz
7.4/7-4	In Table 7-1, changed DBCR reset value to 0x0000_0003.
7.4.1.1/7-4	In Figure 7-2 and Table 7-2, deleted A_PAR_MODE and A_STRM_CNT fields.
7.4.1.2/7-6	In Figure 7-3 and Table 7-3, deleted D_PAR_MODE and D_STRM_CNT fields, added new field CFG_DDR32.
7.4.1.4/7-7	In Figure 7-5 and Table 7-5, deleted DPE and APE bits from EDR.
7.4.1.5/7-8	In Figure 7-6 and Table 7-6, deleted DPEE and APEE bits from EER.
7.5.2/7-12	Deleted reference to A_STRM_CNT in second paragraph.
7.6/7-13	Deleted last paragraph concerning A_STRM_CNT
8.1/8-1	Updated the first sentence of chapter as follows: The two fully programmable DDR SDRAM controllers support most JEDEC standard x8, x16, or x32 DDR and DDR2 memories available.
8.3.1/8-3	Revised Table 8-2 to include alternate auto-precharge on MA8 when DDR_SDRAM_CFG[PCHB8] = 1.
8.4.1.2/8-13	The descriptions of fields ODT_RD_CFG and ODT_WR_CFG were missing from CS*n*_CONFIG. The affected rows have been updated.
8.4.1.6/8-20	In Table 8-11, in the field description for CPO (bits 4–8), changed the description for CPO = 11111 to Reserved. The setting of all ones (automatic calibration) for TIMING_CFG_2[CPO] is now reserved.
8.4.1.6/8-20	In Table 8-11, in the field description for FOUR_ACT (bits 26–31), deleted the sentence after, “Must be set to 0001 for DDR1.” That is, the description for TIMING_CFG_2[FOUR_ACT] should read, “Window for four activates (t_{FAW}). This is applied to DDR2 with eight logical banks only. Must be set to 0001 for DDR1.”
8.4.1.7/8-22	In Table 8-11, replaced the entry for 8_BE.
8.4.1.8/8-25	In Table 8-12, added notes to RD_EN and 2T_EN field descriptions stating that these fields must not both be set at the same time. Also revised description of 32_BE field to include necessary change to MCM DBCR[CFG_DDR32] bit when operating in 32-bit bus mode.
8.4.1.19/8-35	The text describing the use of DDRCDR_1 was revised. In addition, Figure 8-20, and Table 8-25 were revised.
8.5.3/8-58	In the sixth bulleted paragraph (Mode register set), after the sentence that says, “A burst length of 8 is supported for DDR1 memory only.” added the following sentence: For DDR2 in 32-bit bus mode, all 32-byte burst accesses from the platform are split into two 16-byte (that is, 4 beat) accesses to the SDRAMs in the memory controller.
8.6.1/8-75	In Table 8-58, under 8_BE (8-beat burst enable), changed the differences entry for DDR2 to state, “Should be set to 0 regardless of bus width. In 32-bit bus mode,

32-byte burst accesses from the platform will be split into two 16-byte (4-beat) bursts to the SDRAMs."

9.3.6.3/9-37 Replaced the second sentence of the introduction with the following:

When MSIIR is written, MSIIR[SRS] selects the register in which an interrupt bit is to be set; MSIIR[IBS] selects the shared interrupt field in the selected MSIR register to be set.

9.3.6.3/9-37 The settings for IBS were wrong. The affected row was revised as follows:

Table 9-36. MSIIR Field Descriptions

Bits	Name	Description
27–31	IBS	Interrupt register select—Selects the bit to set in the MSIR 0000 Set field SH0 (bit 31) 0001 Set field SH1 (bit 30) 0010 Set field SH2 (bit 29) ... 1111 Set field SH31 (bit 0)

9.4.4/9-57 Added new Section 9.4.4, "Shared Message Signaled Interrupts."

9.4.5/9-57 Added new Section 9.4.5, "PCI Express INTx."

10.4.5/10-17 Revised the entire section (Boot Sequencer Mode).

12.3/12-8 In Table 12-3, corrected the local bus block base address (0x0_5000).

12.3.1.1/12-10 The field XBA (bits 17–18) was missing from BR*n*. The field description for XBA was updated in Table 12-4 as follows:

Table 12-4. BR*n* Field Descriptions

Bits	Name	Description
17–18	XBA	Extended base address. Extends BA by a further 2 bits such that 19 bits of each base register are compared to the 34-bit transaction address. XBA provides the extra 2 msb's (i.e. {XBA,BA} represents the full base address). Used with the extended address mask bits OR*n*[XAM].

12.3.1.2.2/12-13 The field XAM (bits 17–18) was missing from OR*n*. Also, the descriptions of fields CSNT and ACS were missing. The field descriptions in Table 12-6 should appear as follows:

Table 12-6. OR*n*—GPCM Field Descriptions

Bits	Name	Description
17–18	XAM	Extended address mask. Masks the corresponding XBA bits in the BR*n* register, effectively extending the address mask (AM) by 2 bits.

Table 12-6. OR*n*—GPCM Field Descriptions (continued)

Bits	Name	Description
20	CSNT	Chip select negation time. Determines when $\overline{\text{LCS}n}$ and $\overline{\text{LWE}}$ are negated during an external memory write access handled by the GPCM, provided that ACS ≠ 00 (when ACS = 00, only $\overline{\text{LWE}}$ is affected by the setting of CSNT). This helps meet address/data hold times for slow memories and peripherals. 0 $\overline{\text{LCS}n}$ and $\overline{\text{LWE}}$ are negated normally. 1 $\overline{\text{LCS}n}$ and $\overline{\text{LWE}}$ are negated one quarter of a bus clock cycle earlier.
21–22	ACS	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. 00 $\overline{\text{LCS}n}$ is output at the same time as the address lines. Note that this overrides the value of CSNT such that CSNT=0. 01 Reserved. 10 $\overline{\text{LCS}n}$ is output a quarter bus clock cycle after the address lines. 11 $\overline{\text{LCS}n}$ is output a half bus clock cycle after the address lines.

12.3.1.2.3/12-15 The field XAM (bits 17–18) was missing from OR*n*. The field description in Table 12-7 should appear as follows:

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

Bits	Name	Description
17–18	XAM	Extended address mask. Masks the corresponding XBA bits in the BR*n* register, effectively extending the address mask (AM) by 2 bits.

12.3.1.2.4/12-16 The field XAM (bits 17–18) was missing from OR*n*. The field description in Table 12-8 should appear as follows:

Table 12-8. OR*n*—SDRAM Field Descriptions

Bits	Name	Description
17–18	XAM	Extended address mask. Masks the corresponding XBA bits in the BR*n* register, effectively extending the address mask (AM) by 2 bits.

12.3.1.11/12-25 In Table 12-17, removed references to HID1[RFXE]; it does not exist on the e600 core.

12.3.1.12/12-26 In Table 12-18, removed references to HID1[RFXE]; it does not exist on the e600 core.

13.2/13-1 In the feature list, revised the first sub-bullet under the seventh bullet (“Full- and half-duplex Ethernet support....” to read as follows:

— IEEE 802.3 full-duplex flow control (software programmed PAUSE frame generation and recognition)

13.3/13-4 Replaced the first bullet in the list of operational modes with the following:

- Ethernet and FIFO operation

  The ECNTRL register’s FIFO mode enable bit (ECNTRL[FIFM]) allows bypass of the Ethernet MAC and enables I/O through the FIFO interface sharing the normal GMII signals. All four eTSECs support an 8-bit FIFO interface independently. Pairs of GMII ports can be combined to create a 16-bit full-duplex interface. If configured in FIFO mode, the FIFOCFG register determines

operation. In FIFO mode data is transferred synchronously with respect to the external data clock, whose maximum is defined by a ratio of 4.2:1 (platform:TxClk) in GMII mode and a ratio of 3.2:1 (platform:TxClk) in encoded mode.

In the same section, replaced the last bullet (loop back mode) with the following:

- Internal loop back supported for all interfaces except when configured for half-duplex operation

13.4/13-6 In Table 13-1, the description of TSECn_RX_ER was modified as follows:

Table 13-1. eTSEC*n* Network Interface Signal Properties

Signal Name	Function	Reset State
TSEC*n*_RX_ER	GMII, MII, RMII—Receive error, input TBI—RGC bit 9, input FIFO—Receive error or receive frame control bit, input RGMII, RTBI—Unused, output driven zero	—

In Table 11-2, the descriptions of TSEC*n*_RX_CLK and TSEC*n*_TX_CLK were modified as follows:

Table 13-2. eTSEC Signals—Detailed Signal Descriptions

Signal	I/O	Description
TSEC*n*_RX_CLK	I	Receive clock. In MII or RGMII mode, the receive clock TSEC*n*_RX_CLK is a continuous clock (2.5, 25, or 125 MHz) that provides a timing reference for TSEC*n*_RX_DV, TSEC*n*_RXD, and TSEC*n*_RX_ER. In TBI mode, TSEC*n*_RX_CLK is the input for a 62.5 MHz PMA receive clock, 0 split phase with PMA_RX_CLK1 and is supplied by the SerDes. In RTBI mode it is a 125 MHz receive clock. In RMII mode this clock is not used for the receive clock, as RMII uses a shared reference clock. In FIFO mode the receive clock is a continuous clock whose maximum is defined by a ratio of 4.2:1 (platform:RxClk) in GMII mode and a ratio of 3.2:1 (platform:RxClk) in encoded mode.
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 eTSEC and provided to the PHY and the MAC. The TSEC*n*_TX_CLK is generated in the PHY and provided to the MAC. In TBI mode, this signal is PMA receive clock 1 at 62.5 MHz, split phase with PMA_RX_CLK0, and is supplied by the SerDes. In RMII mode this signal is the reference clock shared between transmit and receive, and is supplied by the PHY. In FIFO mode the transmit clock is a continuous clock whose maximum is defined by a ratio of 4.2:1 (platform:TxClk) in GMII mode and a ratio of 3.2:1 (platform:TxClk) in encoded mode. This signal is not used in the eTSEC RTBI or RGMII modes.

13.5.3.1.3/13-23 All fields of register IEVENT were corrected to be w1c (write-one-to-clear).

13.5.3.1.8/13-34 Replaced the description of DMACTRL[GTS] with the following:

Graceful transmit stop. If this bit is set, the Ethernet controller stops transmission after all frames that are currently in the Tx FIFO or scheduled have been transmitted, and the GTSC interrupt in the IEVENT register is asserted. A frame that has started reading buffer descriptors or data from memory will be read

to completion and transmitted before the GTSC interrupt occurs. However, if no frame has been scheduled for transmission and the Tx FIFO is empty, the GTSC interrupt is asserted immediately. Once transmission has completed, clearing GTS will “restart” transmit.

13.5.3.2.1/13-37 Removed the sections, “FIFO Receive Pause Start Threshold Register (FIFO_RX_PAUSE),” and, “FIFO Receive Pause Shut-off Threshold Register (FIFO_RX_PAUSE_SHUTOFF).”

13.5.3.3.2/13-41 All fields of register TSTAT were corrected to be w1c (write-one-to-clear).

13.5.3.3.3/13-43 In the first paragraph, removed reference to unsupported CFA field.

13.5.3.3.3/13-43 Replaced the description of DFVLAN[TAG] with the following:

Table 13-24. DFVLAN Field Descriptions

Bits	Name	Description
0–15	TAG	This is the default Ethertype used to tag VLAN frames. On transmit, this tag is inserted ahead of the VLAN control word; TAG should be set to 0x8100 for IEEE 802.1Q VLAN. On receive, an Ethertype matching TAG or an Ethertype of 0x8100 marks a VLAN-tagged frame. Note that if using DFVLAN to set a custom ethertype (that is, using a value other than 0x8100), packets received with a custom tag are not counted by any of the RMON counters. Affected counters include TRMGV, RMCA, RBCA, RXCF, RXPF, RXUO, RALN, RFLR, ROVR, RJBR, TMCA, TBCA, TXPF, TXCF.

13.5.3.7.19/13-89 Replaced the description of RCSE[RCSE] with the following:

Receive false carrier counter. Counts the number of times that the carrier sense condition was lost or never asserted when attempting to transmit a frame on a particular interface.

The count represented by an instance of this object is incremented at most once per transmission attempt, even if the carrier sense condition fluctuates during a transmission attempt. The event is reported along with the statistics generated on the next received frame, as defined by a 1 on TSEC*n*_RX_ER and an 0xE on TSEC*n*_RXD. Only one false carrier condition can be detected and logged between frames.

13.5.3.4.1/13-49 The descriptions of VLEX, FILREN and PRSDEP were updated.

13.5.3.4.2/13-51 All fields of register RSTAT were corrected to be w1c (write-one-to-clear).

13.5.3.4.8/13-59 Bits 28 (formerly TUC) and 29 (formerly TUV) in Figure 13-35 were relabeled as reserved. (They were correctly indicated as reserved in Table 13-38.)

In the same section, in Table 13-38, added the following note to the description of RQFPR[ETY] (PID 0111, bits 16–31):

Using the filer to match ETY does not work in the case of PPPoe packets, because the PPPoe ethertype in the original packet, 0x8864, is always overwritten with the PPP protocol field. Thus, matches on ETY == 0x8864 always fail.

Instead, software should use PID=1 fields IP4 (ETY = 0x0021) and IP6 (ETY = 0x0057) to distinguish PPPoE session packets carrying IPv4 and IPv6 datagrams. Other PPP protocols are encoded in the ETY field, but many of them overlap with real ethertype definitions. Consult IANA and IEEE for possible ambiguities.

In the same section added the following note to the description of RQFPR[TOS] (PID 1010, bits 16–31) in Table 13-38:

Note that for IPv6 the Traffic Class field is extracted using the IP header definition in RFC 2460. IPv6 headers formed using the earlier RFC 1883 have a different format and must be handled with software

13.5.3.4.9/13-62 Added the following text to the description of MRBLR[MRBL] (Table 13-39):

MRBL must be set, together with the number of buffer descriptors, to ensure adequate space for received frames. See Section 11.5.3.6.5, “Maximum Frame Length Register (MAXFRM),” for further discussion.

13.5.3.6.2/13-69 In Table 13-45, revised the description of MACCFG2[Huge Frame].

13.5.3.6.5/13-73 In Table 13-48, revised the description of MAXFRM[Maximum Frame].

13.5.3.7.19/13-89 In Table 13-78, revised the description of RCSE[RCSE].

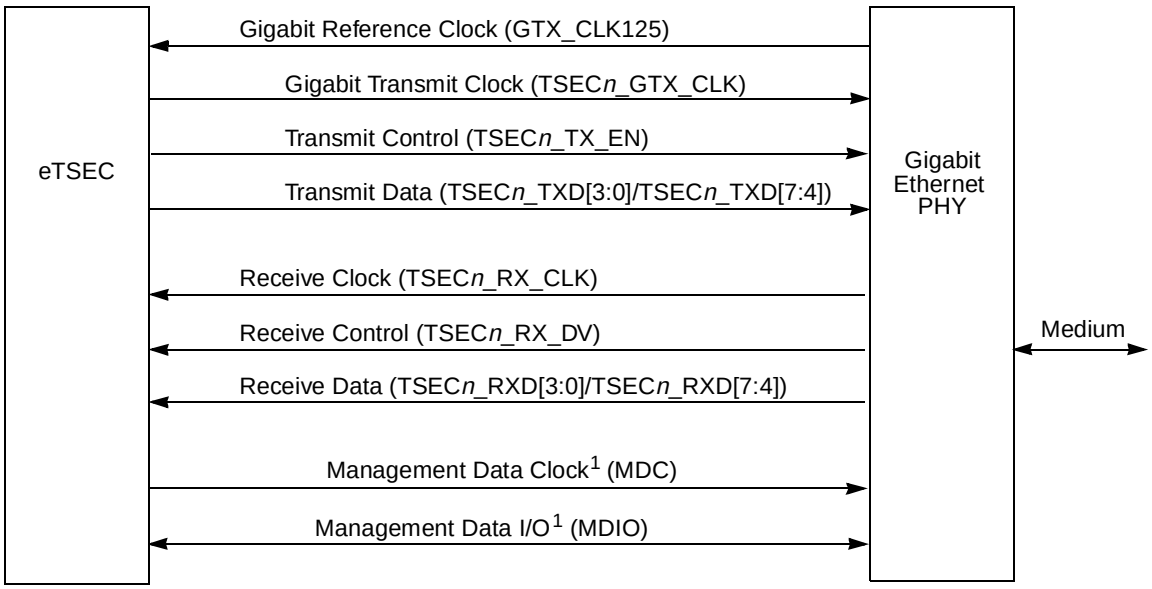
13.5.3.7.25/13-92 Added the following note to the description of TBYT[TBYT]:

Note that the value of TBYT may be greater than the actual number of bytes transmitted if the frame is truncated because it exceeds MAXFRM.

13.6.1.4/13-128 Added the following sentence after the fifth sentence:

Note that the GTX_CLK125 input must be provided at 125 MHz for an RGMII interface, regardless of operation speed (1 Gbps, 100 Mbps, or 10 Mbps).

Also, replaced Figure 13-123 with the following:



1 The management signals (MDC and MDIO) are common to all of the gigabit Ethernet controllers module connections in the system, assuming that each PHY has a different management address.

Figure 11-125. eTSEC-RGMII Connection

13.6.3.2/13-143 Clarified step 3 to note that MACCFG1[SOFT_RESET] must remain set for at least 3 TX clocks prior to clearing in step 4.

13.6.6.2/13-171 In Table 13-149, revised the description of TxBD field TR.

13.6.6.3/13-175 Added the following paragraph after the first paragraph of this section, “Receive Buffer Descriptors (RxBD)”:

The number of buffer descriptors in a ring is set by using the W bit to indicate that the next buffer wraps back to the beginning of the ring. See Section 13.5.3.6.5, “Maximum Frame Length Register (MAXFRM),” for information on setting the size of the buffer descriptor ring.

In the same section, revised the description of RxBD fields LG and TR in Table 13-150.

14.3.1.1/14-10 In Table 14-5, delete the sentence, “Defined only if MR*n*[EMP_EN] is set.” from the field description for MR*n*[BWC].

14.3.1.4/14-15 Add the following sentence to the end of the second paragraph:

Note that ATMU bypass mode is not supported in RapidIO large transport system size.

14.3.1.6/14-18 Add the following sentence to the end of the second paragraph:

Note that ATMU bypass mode is not supported in RapidIO large transport system size.

15.6.1.3/15-12 Changed all fields in the register AIDCAR to be read/write

15.6.1.4/15-12 Changed all fields in the register AICAR to be read/write.

15.6.1.5/15-13 The values of PEFCAR[CTLS] were reversed. The affected row should appear as follows:

Table 15-6. PEFCAR Field Descriptions

Bit	Name	Description
27	CTLS	This bit reflects the selected RapidIO common transport system size. The RapidIO common transport system size is determined at power-on reset. See Section 4.4.3.16, “RapidIO System Size,” for POR configuration details. 0 PE only supports common transport small systems (up to 256 devices). 1 PE supports common transport large systems (up to 65,536 devices).

15.6.2.2/15-22 Clarified relationship between timeout field value and platform clock (field value decrements at 1/2 platform clock) for PLTOCCSR.

15.6.2.3/15-23 Clarified relationship between timeout field value and platform clock (field value decrements at 1/2 platform clock) for PRTOCCSR.

15.6.5.3/15-43 Clarified relationship between timeout field value and platform clock (field value decrements at 1/2 platform clock) for LOPTTLCR.

15.6.5.9/15-47 In Figure 15-48, changed access for CCP (bit 27) to read/write.

15.6.7/15-50 Added note regarding when booting from serial RapidIO, that outbound ATMU window 0 must be used.

15.6.7.1/15-50 Added note that subsegments are only supported when multiple segments are chosen.

15.6.7.5/15-54 Added sentence concerning NSSEG being valid only when ROWAR*n*[NSEG] is non-zero to the description of ROWAR*n*[NSSEG].

15.7.1.4/15-66 Added the following text to the end of the section (after the bulleted list):

When software initializes these registers, the queue to which they point must be aligned on a boundary equal to number of queue entries × 32 bytes (size of each queue descriptor). For example, if there are eight entries in the queue, the register must be 256-byte aligned.

The number of queue entries is set in OM*n*MR[CIRQ_SIZ]; see Section 15.7.1.1, “Outbound Message n Mode Registers (OMnMR)”).

15.7.2.3/15-75 Added the following text after the first paragraph:

When software initializes these registers, the queue to which they point must be aligned on a boundary equal to number of queue entries × frame size. For example, if there are eight entries in the queue and the frame size is 128 bytes, the queue must be 1024-byte aligned.

The number of queue entries is set in IMnMR[CIRQ_SIZ], and the frame size is set in IMnMR[FRM_SIZ]. See Section 15.7.2.1, “Inbound Message n Mode Registers (IMnMR)”).

15.7.2.4/15-76 Added the following text to the end of the section (after the bulleted list):

When software initializes these registers, the queue to which they point must be aligned on a boundary equal to number of queue entries × frame size. For example, if there are eight entries in the queue and the frame size is 128 bytes, the queue must be 1024-byte aligned.

The number of queue entries is set in IM*n*MR[CIRQ_SIZ], and the frame size is set in IM*n*MR[FRM_SIZ]. See Section 15.7.2.1, “Inbound Message n Mode Registers (IMnMR)”).

15.7.2.5/15-77 Replaced the last sentence of the first paragraph with the following:

The reset value is the maximum time-out interval. The timer decrements at one-half the MMC clock rate; thus at an MMC clock rate of 400 MHz, the maximum time-out value is approximately 83.9 ms.

15.9.4.2.1/15-149 Added the following text at the end of the third bullet:

These pairs of registers must also be queue size aligned (that is, the queue must be aligned on a boundary equal to number of queue entries × 32 bytes (size of each queue descriptor)). For example, if there are 16 entries in the queue, the queue must be 512-byte aligned.

The number of queue entries is set in OM*n*MR[CIRQ_SIZ]; see Section 15.7.1.1, “Outbound Message n Mode Registers (OMnMR)”).

15.9.4.2.6/15-151 Added the following text at the end of the second sentence:

(in other words, on a boundary equal to number of queue entries × 32 bytes (size of each queue descriptor))

15.9.4.2.14/15-155 Added an error for “Queue misaligned” to Table 15-127.

15.9.5.1/15-157 Added the following text to the first bullet:

These also must be queue size aligned (in other words, the queue to which they point must be aligned on a boundary equal to number of queue entries × frame size). For example, if there are eight entries in the queue and the frame size is 128 bytes, the queue must be 1024-byte aligned.

15.9.5.5.5/15-165 Added an error for “Queue misaligned” to Table 15-129.

16.1.1.2/16-3 Revised the description of splitting transactions when sending data back to the PCI Express link.

16.3.2.4/16-11 In Table 16-7, revised the the description for PEX_CONF_RTY_TOR[RD].

16.3.2.5/16-12	In Figure 16-6, removed the IPOL field (bit 16), by marking it reserved. Also, added a new field, SB_EN, at bit 17.
16.3.2.5/16-12	In Table 16-8, removed the name and description for IPOL (bit 16) by marking it reserved. Also, added the name and description for new bit 17, SB_EN.
16.3.3.1/16-13	In Table 16-9, added the following note to the field description for reserved bit 20 Note that during normal operation, this bit may be set (falsely). The bit may be ignored and cleared (w1c) without consequence.
16.3.5.1.4/16-22	In Table 16-18, added a note restricting the translation target to the description of Configuration Write and I/O Write encodings for the WTT field.
16.3.5.2.6/16-27	In Figure 16-23, renamed PEXIWAR*n* bits 8-11 to TRGT. This was done for documentation consistency; there is no functional change. Also, in Table 16-22, rename the field TRGT.
16.3.5.2.6/16-27	In Table 16-22, added a note to the end of the field description for bits 8-11 (TRGT) restricting the translation target.
16.3.7/16-42	Added the following after the first paragraph: Note that external configuration transactions should not be attempted until the link has successfully trained. Software can poll the LTSSM state status register (PEX_LTSSM_STAT) to check the status of link training before issuing external configuration requests.
16.3.7.1/16-42	Revised the sentence in the first paragraph regarding obtaining the PCI Express controller's bus number.
16.3.10.14/16-90	Revised text in the beginning of the section to describe how the actual PCI Express controller clock frequency is determined.
16.4.1.2/16-98	Completely rewrote the sections formerly titled, "Outbound Byte Swapping," "Inbound Byte Swapping," and replaced them with the new section titled, "Byte Ordering."
16.4.2.1.2/16-107	Completely rewrote Section 16.4.2.1.2, "Hardware MSI Generation."
16.4.2.1.4/16-108	Replaced the last sentence of the section with the following: A write to the ATMU window containing the MSI address with the appropriate data value will generate the desired MSI transaction to the remote RC.
16.4.2.2.1/16-108	Completely rewrote Section 16.4.2.2.1, "INTx Message Handling."
16.5/16-111	Added new Section 16.5.1, "Boot Mode and Inbound Configuration Transactions."
17.4.1.1/17-6	In Figure 17-1, PORPLLSR[Plat_Ratio] was corrected to show it spanning bits 16–30; bit 31 is reserved. Note that the field description in Table 17-4 was correct.
17.4.1.3/17-8	In Table 17-6, revised the description for PORIMPSCR[LBC_Z].
17.4.1.9/17-17	Revised the first paragraph with the following, and removed the first two sentences of the second paragraph:

DEVDISR, shown in Figure 17-12, contains disable bits for various functional blocks. All functional blocks are enabled after reset; unneeded blocks can be disabled to reduce power consumption or allow their signals to be used as general-purpose I/O signals. See Section 17.5.1.2, “Shutting Down Unused Blocks,” and Section 17.4.1.7.1, “General-Purpose I/O Control Register (GPIOCR),” for more information.

When a block is disabled using the DEVDISR register, its clocks are shut down to save power. Blocks disabled by DEVDISR must not be re-enabled without a hard reset. The timebase disable (TB) control in DEVDISR may be set and cleared without requiring a hard-reset. However, because changing other bits in DEVDISR could cause errant behavior, it is recommended that DEVDISR be accessed using a read-modify-write operation when changing TB.

Note that a block can be enabled in DEVDISR and disabled functionally. The SerDes blocks are enabled or disabled based on the SerDes port selection (cfg_io_ports[0:2]) configuration inputs. See Section 4.4.3, “Power-On Reset Configuration,” for additional details.

17.4.1.9.1/17-19	Added new Section 17.4.1.9.1, “Using DEVDISR[TB] to Synchronize the Timebases of Multiple Cores.”
17.4.1.14/17-24	Revised description and format of SVR.
17.5.1.2/17-35	Revised the second paragraph to include description of setting and clearing DEVDISR[TB].
17.5.1.3.1/17-36	Revised description of returning from interrupt to state that MSR[POW] remains cleared upon **rfi**.
17.5.1.3.2/17-36	Revised description of returning from interrupt to state that MSR[POW] remains cleared upon **rfi**.
17.5.1.6.1/17-37	Revised description of returning from interrupt to state that MSR[POW] remains cleared upon **rfi**.

Glossary

The glossary contains an alphabetical list of terms, phrases, and abbreviations used in this reference manual.

A

Architecture. A detailed specification of requirements for a processor or computer system. It does not specify details of how the processor or computer system must be implemented; instead it provides a template for a family of compatible *implementations*.

Atomic access. A bus access that attempts to be part of a read-write operation to the same address uninterrupted by any other access to that address (the term refers to the fact that the transactions are indivisible). The 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*.

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, 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.

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.

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*.

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.

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

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

Z