PPC440GP

Embedded Processor

User's Manual

Preliminary

IBM®

Eighth Edition (August 2002)

This edition of *IBM PPC440GP Embedded Processor User's Manual* applies to the IBM PPC440GP 32-bit embedded processor, until otherwise indicated in new versions or application notes.

The following paragraph does not apply to the United Kingdom or any country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS MANUAL "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions; therefore, this statement may not apply to you.

IBM does not warrant that the products in this publication, whether individually or as one or more groups, will meet your requirements or that the publication or the accompanying product descriptions are error-free.

This publication could contain technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or program(s) described in this publication at any time.

It is possible that this publication may contain references to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming, or services in your country. Any reference to an IBM licensed program in this publication is not intended to state or imply that you can use only IBM's licensed program. You can use any functionally equivalent program instead.

Requests for copies of this publication and for technical information about IBM products should be made to your IBM Authorized Dealer or your IBM Marketing Representative.

Address comments about this publication to:

IBM Corporation
Department YM5A
P.O. Box 12195
Research Triangle Park, NC 27709

IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

PPC440GP Embedded Processor

User's Manual

SA14-2519-12

July 31, 2002

PowerPC®

Patents and Trademarks

IBM may have patents or pending patent applications covering the subject matter in this publication. The furnishing of this publication does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 208 Harbor Drive, Stamford, CT 06904, United States of America.

The following terms are trademarks of IBM Corporation:

IBM
PowerPC
PowerPC Architecture
PowerPC Embedded Processors
PPC440GP
RISCTrace
RISCWatch
CoreConnect

Other terms which are trademarks are the property of their respective owners.

Printed in the United States of America July 31, 2002

IBM Microelectronics Division
1580 Route 52, Bldg. 504
Hopewell Junction, NY 12533-6351

titleCMP.fm.
July 31, 2002

Contents

Figures

Tables

About This Book

This user's manual provides the architectural overview, programming model, and detailed information about the registers, the instruction set, and operations of the IBM™ PowerPC™ 440GP (PPC440GP™) 32-bit RISC embedded processor.

The PPC440GP RISC embedded processor features:

- Book-E Enhanced PowerPC Architecture™
- PPC440GP processor core operating up to 400 MHz
- Selectable processor bus ratios including but not limited to 3:2, 5:2
- CoreConnect bus architecture, PLB performance monitor and peripheral cores
- 8K Internal SRAM controller (ISC)
- Double data rate (DDR) synchronous DRAM (SDRAM) controller
- Peripheral component interconnect (PCI-X) bridge controller
- External peripheral bus
- Direct Memory Access (DMA) controller
- Memory access layer (MAL) controller
- EMAC to PHY interface (ZMII)
- Two on-chip ethernet ports (EMACs)
- General purpose timer (GPT)
- Two universal interrupt controllers (UICs)
- Two universal asynchronous receiver/transmitters (UARTs)
- Master and slave Inter-integrated circuit (IIC) controllers
- General purpose I/O (GPIO) interface
- External bus master interface (EBMI)
- External bus controller (EBCO)
- JTAG support for board level testing
- Internal processor local bus (PLB) running at SDRAM interface frequency
- Support for PowerPC processor boot from PCI memory
- Extensive development tool support

Who Should Use This Book

This book is for system hardware and software developers, and for application developers who need to understand the PPC440GP. The audience should understand embedded processor design, embedded system design, operating systems, RISC processing, and design for testability.

How to Use This Book

This book describes the PPC440GP device architecture (including instructions and registers), processor core functions, system operations, internal bus functions, and external interfaces. This book contains the following ~~chapters,~~ sections arranged in parts:

- Part IIntroduction

  ~~Chapter 1, "Overview"~~

  ~~Chapter 2, "On-Chip Buses"~~

  ~~Chapter 3, "PLB Performance Monitor"~~

  - *Overview* on page 69
  - *On-Chip Buses* on page 91
  - *PLB Performance Monitor* on page 117

- Part IIPPC440GP RISC Processor

  ~~Chapter 4, "Programming Model"~~

  ~~Chapter 5, "Instruction and Data Caches"~~

  - *Programming Model* on page 145
  - *Instruction and Data Caches* on page 205
  - ~~Chapter 6, "Memory Management"~~*Memory Management* on page 233

- Part IIIPPC440GP System Operations

  ~~Chapter 7, "Reset and Initialization"~~

  ~~Chapter 8, "IIC Bootstrap Controller"~~

  ~~Chapter 9, "Universal Interrupt Controller"~~

  ~~Chapter 10, "Interrupts and Exceptions"~~

  ~~Chapter 11, "Timer Facilities"~~

  ~~Chapter 12, "General Purpose Timers"~~

  - *Reset and Initialization* on page 259
  - *IIC Bootstrap Controller* on page 283
  - *Universal Interrupt Controller* on page 299
  - *Interrupts and Exceptions* on page 337
  - *Timer Facilities* on page 383
  - *General Purpose Timers* on page 393
  - ~~Chapter 13, "Clocking"~~*Clocking* on page 407

  ~~Chapter 14, "Clock and Power Management"~~

  ~~Chapter 15, "Debug Facilities"~~

  - *Clock and Power Management* on page 423
  - *Debug Facilities* on page 429

- Part IVPPC440GP External Interfaces

  ~~Chapter 16, "Internal SRAM Controller"~~

  ~~Chapter 17, "DDR SDRAM Controller"~~

  ~~Chapter 18, "PLB-PCIX Bridge Controller"~~

  ~~Chapter 19, "Direct Memory Access Controller"~~

  ~~Chapter 20, "Memory Access Layer"~~

  ~~Chapter 21, "EMAC to PHY Interface Bridge"~~

Chapter 22, "Ethernet Media Access Controllers"
Chapter 23, "Serial Port Operations"
Chapter 24, "IIC Bus Interface"
Chapter 25, "GPIO Operations"
Chapter 26, "External Bus Master Interface (EBMI)"
Chapter 27, "External Bus Controller"

- *Internal SRAM Controller* on page 463
- *DDR SDRAM Controller* on page 473
- *PLB-PCIX Bridge Controller* on page 535
- *Direct Memory Access Controller* on page 655
- *Memory Access Layer* on page 681
- *EMAC to PHY Interface Bridge* on page 719
- *Ethernet Media Access Controllers* on page 729
- *Serial Port Operations* on page 777
- *IIC Bus Interface* on page 801
- *GPIO Operations* on page 823
- *External Bus Master Interface* on page 831
- *External Bus Controller* on page 855

- Part VReference

Chapter 28, "Instruction Set"
Chapter 29, "Register Summary"
Chapter 30, "Signal Summary"

- *Instruction Set* on page 893
- *Register Summary* on page 1093
- *Signal Summary* on page 1529

This book contains the following appendixes:

Appendix A, "Instruction Summary,"
Appendix B, "PPC440GP Compiler Optimizations,"

- *Instruction Summary* on page 1539
- *PPC440GP Compiler Optimizations* on page 1587

To help readers find material in these chapters, the book contains:

Contents, on page 5.
Figures, on page 31.
Tables, on page 49.
Index, on page X-1.

- *Contents* on page 3
- *Figures* on page 33
- *Tables* on page 55
- *Index* on page 1589

Conventions

The following is a list of notational conventions frequently used in this manual.

$\overline{\text{ActiveLow}}$	An overbar indicates an active-low signal.
n	A decimal number
0x*n*	A hexadecimal number
0b*n*	A binary number
=	Assignment
$\wedge$	AND logical operator
$\neg$	NOT logical operator
$\vee$	OR logical operator
$\oplus$	Exclusive-OR (XOR) logical operator
+	Twos complement addition
–	Twos complement subtraction, unary minus
$\times$	Multiplication
$\div$	Division yielding a quotient
%	Remainder of an integer division; (33 % 32) = 1.
||	Concatenation
=, $\neq$	Equal, not equal relations
<, >	Signed comparison relations
$\overset{u}{<}$, $\overset{u}{>}$	Unsigned comparison relations
if...then...else...	Conditional execution; if *condition* then *a* else *b*, where *a* and *b* represent one or more pseudocode statements. Indenting indicates the ranges of *a* and *b*. If *b* is null, the else does not appear.
do	Do loop. "to" and "by" clauses specify incrementing an iteration variable; "while" and "until" clauses specify terminating conditions. Indenting indicates the scope of a loop.
leave	Leave innermost do loop or do loop specified in a leave statement.
FLD	An instruction or register field
FLD_b	A bit in a named instruction or register field
$FLD_{b:b}$	A range of bits in a named instruction or register field
$FLD_{b,b,\ldots}$	A list of bits, by number or name, in a named instruction or register field
REG_b	A bit in a named register
$REG_{b:b}$	A range of bits in a named register
$REG_{b,b,\ldots}$	A list of bits, by number or name, in a named register
REG[FLD]	A field in a named register

REG[FLD, FLD . . .]	A list of fields in a named register
REG[FLD:FLD]	A range of fields in a named register
GPR(r)	General Purpose Register (GPR) r, where $0 \le r \le 31$.
(GPR(r))	The contents of GPR r, where $0 \le r \le 31$.
DCR(DCRN)	A Device Control Register (DCR) specified by the DCRF field in an **mfdcr** or **mtdcr** instruction
SPR(SPRN)	An SPR specified by the SPRF field in an **mfspr** or **mtspr** instruction
TBR(TBRN)	A Time Base Register (TBR) specified by the TBRF field in an **mftb** instruction
GPRs	RA, RB, . . .
(Rx)	The contents of a GPR, where *x* is A, B, S, or T
(RA|0)	The contents of the register RA or 0, if the RA field is 0.
CR_{FLD}	The field in the condition register pointed to by a field of an instruction.
$c_{0:3}$	A 4-bit object used to store condition results in compare instructions.
^{n}b	The bit or bit value *b* is replicated *n* times.
xx	Bit positions which are don't-cares.
CEIL(x)	Least integer $\ge x$.
EXTS(x)	The result of extending *x* on the left with sign bits.
PC	Program counter.
RESERVE	Reserve bit; indicates whether a process has reserved a block of storage.
CIA	Current instruction address; the 32-bit address of the instruction being described by a sequence of pseudocode. This address is used to set the next instruction address (NIA). Does not correspond to any architected register.
NIA	Next instruction address; the 32-bit address of the next instruction to be executed. In pseudocode, a successful branch is indicated by assigning a value to NIA. For instructions that do not branch, the NIA is CIA +4.
MS(addr, n)	The number of bytes represented by *n* at the location in main storage represented by *addr*.
EA	Effective address; the 32-bit address, derived by applying indexing or indirect addressing rules to the specified operand, that specifies a location in main storage.
EA_b	A bit in an effective address.
$EA_{b:b}$	A range of bits in an effective address.
ROTL((RS),n)	Rotate left; the contents of RS are shifted left the number of bits specified by *n*.
MASK(MB,ME)	Mask having 1s in positions MB through ME (wrapping if MB > ME) and 0s elsewhere.

instruction(EA)	An instruction operating on a data or instruction cache block associated with an EA.

Part I. Introduction

1. Overview

The IBM™ PowerPC™ 440GP 32-bit reduced instruction set computer (RISC) embedded processor, referred to as the PPC440GP, is a system-on-a-chip (SOC) design that integrates a PowerPC 440 embedded processor core with a mix of rich peripheral cores by implementing IBM's high speed CoreConnect™ technology.

This chapter begins with a block diagram of the PPC440GP and goes on to describe the following features with cross references for detailed information included in this book:

- PPC440GP embedded processor features as seen from processor core, peripheral cores and CoreConnect™ technology viewpoint.
- PowerPC Architecture that describes features accessed at user level and supervisor level.
- PPC440GP as a 32-bit implementation of Book-E Enhanced PowerPC Architecture™ optimized for embedded applications.
- A fully integrated PPC440GP processor core implemented in IBM's advanced copper technology with advanced CPU extensions and DSP instructions.
- IBM standard on-chip buses that host both high performance, high bandwidth as well as lower data rate peripherals.
- Several PPC440GP interfaces that provide a peripheral mix for a wide variety of applications.
- Chip level programming model that provides a high degree of user control over configuration and operation of all functional units.
- Powerful debug support for a wide range of hardware and software development tools.

Figure 1-1 illustrates the logical organization of the PPC440GP.

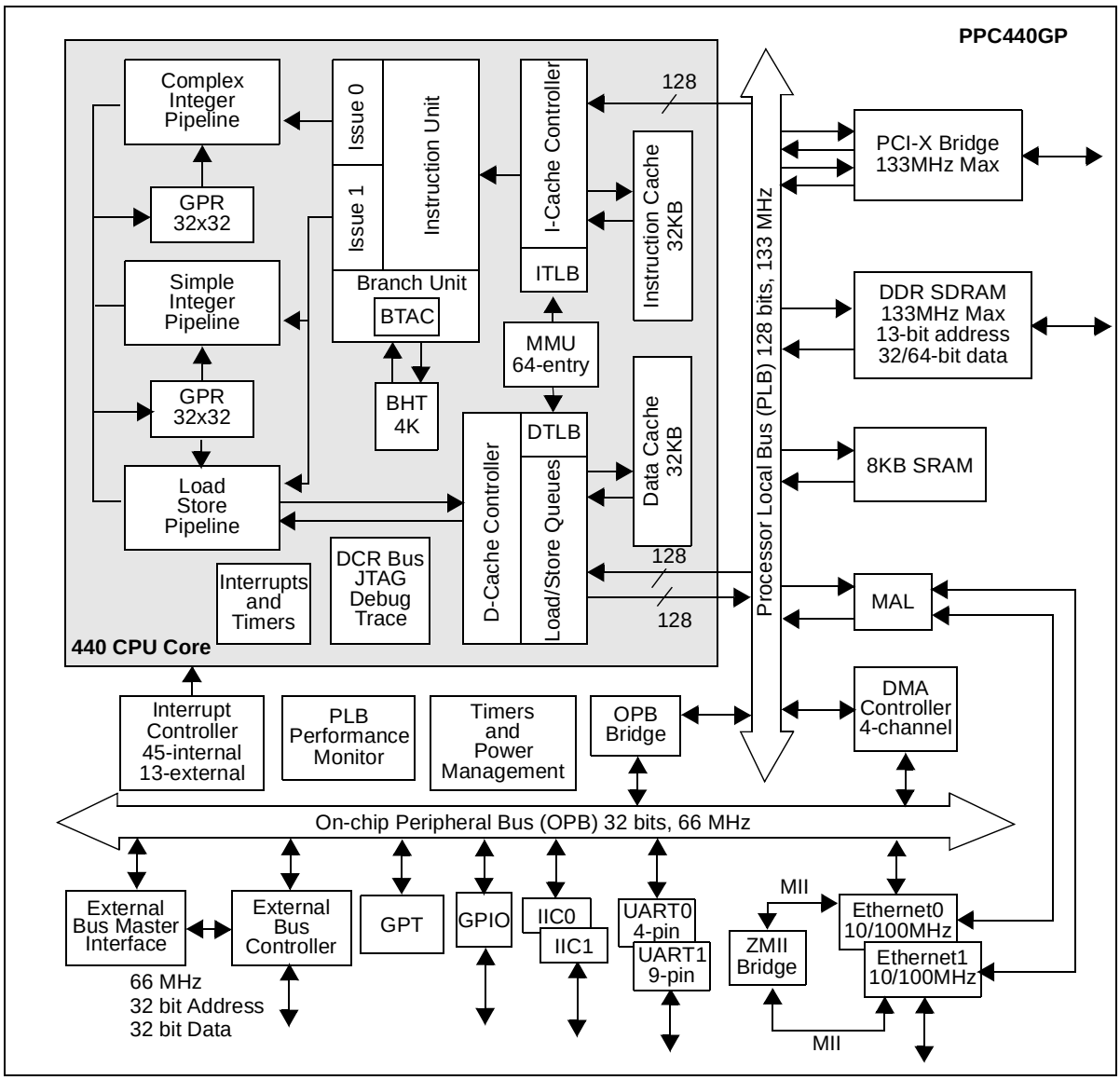


Figure 1-1. PPC440GP Block Diagram

1.1 PPC440GP Embedded Processor Features

The PPC440GP provides high performance and low power consumption. The PPC440GP RISC CPU executes at sustained speeds approaching two instructions per cycle. On-chip peripheral cores reduce chip count and design complexity in systems and improve system throughput. The cpu core combined with wide peripheral mix through the IBM CoreConnect™ technology provides an ideal foundation for systems incorporating system-on-a-chip (SOC) designs as in PPC440GP. This section provides a list of features that are implemented in PPC440GP.

1.1.1 PPC440GP Processor Core Features

- High performance, dual-issue, superscalar 32-bit RISC CPU
  - Superscalar implementation of the Book-E Enhanced PowerPC Architecture
  - Seven stage, highly-pipelined micro-architecture
  - Dual instruction fetch, decode, and out-of-order issue
  - Out-of-order execution
  - High-accuracy dynamic branch prediction utilizing a Branch History Table (BHT)
  - Reduced branch latency using Branch Target Address Cache (BTAC)
  - Three independent pipelines
    - Combined complex integer, system, and branch pipeline
    - Simple integer pipeline
    - Load/store pipeline
  - Single cycle multiply
  - Single cycle multiply-accumulate (new DSP instruction set extensions)
  - Two replicated 6 port (3 read, 3 write) 32x32-bit General Purpose Register (GPR) files
  - Hardware support for all CPU misaligned accesses
  - Full support for both big and little endian byte order
  - Extensive power management designed into core for maximum performance/power efficiency
- Primary caches
  - Independently configurable instruction and data cache arrays
  - Array size offerings: 32KB
  - Single-cycle access
  - 32-byte (eight word) line size
  - Highly-associative (64-way for 32KB)
  - Write-back and write-through operation
  - Control over whether stores will allocate or write-through on cache miss
  - Extensive load/store queues and multiple line fill/flush buffers
  - Non-blocking with up to four outstanding load misses
  - Cache line locking supported
  - Caches can be partitioned to provide separate regions for "transient" instructions and data
  - High associativity permits efficient allocation of cache memory
  - Critical word first data access and forwarding
- Memory Management Unit
  - Separate instruction and data quote sizes for micro-TLB's
  - 64-entry, fully-associative unified TLB array
  - Variable page sizes (1KB-256MB), simultaneously resident in TLB

- 4-bit extended real address for 36-bit (64 GB) addressability
- Flexible TLB management with software page table search
- Storage attribute controls for write-through, caching inhibited, guarded, and byte order (endianness)
- Four user-definable storage attribute controls

- Debug facilities
  - Extensive hardware debug facilities incorporated into the IEEE 1149.1 JTAG port
    - Multiple instruction and data address break points (including range)
    - Data value compare
    - Single-step, branch, trap, and other debug events
  - Non-invasive real-time software trace interface
- Timer facilities
  - 64-bit time base
  - Decrementer with auto-reload capability
  - Fixed interval timer (FIT)
  - Watchdog timer with critical interrupt and/or auto-reset

1.1.2 PPC440GP Peripheral Features

- Internal SRAM controller (ISC)
  - 128-bit PLB slave attachment addressable by any PLB master
  - Transfers data between PLB and internal SRAM
  - One physical bank of 8KB
  - Enables processor to read and write internal registers via DCR bus
  - Complies with PLB guarded memory access
- Double data rate (DDR) synchronous DRAM (SDRAM) controller
  - 2.1GB peak data rate
  - 32 or 64-bit interface with optional ECC
  - x8 or wider devices
  - DDR: 64, 128, 256MB; (4 bank devices)
  - 16MB to 256MG per bank
  - Supports 184 pin dual bank registered DDR DIMMs
  - Page mode accesses w/bank interleaving
  - Configurable paging (supports 8 open pages)
  - Power management
  - SSTL logic interface
- Peripheral component interconnect (PCI-X) bridge controller
  - 64-bit PCI-X address/data bus up to 133 MHz
  - 32 or 64-bit PCI V2.2 at frequencies up to 66 MHz

  - PCI bus power management interface specification designed to PCI bus power
  - Asynchronous clocking between PCI-X and internal buffers
  - Multiple read prefetch and write post buses
  - Simple message passing capability
  - Message signalled interrupts (MSI) support for inbound and outbound interrupts
  - Ability to boot processor from PCI bus memory
- Direct Memory Access (DMA) controller
  - Supports memory to memory, buffered peripheral to memory, and buffered memory to peripheral transfers
  - Four independent DMA channels
  - Scatter/gather capability
  - 128-byte buffer
  - 8-, 16-, 32-bit peripheral support
  - 64-bit addressing
- Memory access layer (MAL) controller
  - No restrictions on buffer alignment
  - Aligned bus accesses to enable burst operation with external memories
  - Configurable receive buffer size (configurable per channel)
  - No minimum transmit buffer size
  - Maximum buffer sizes of 4095 bytes (TX) and 4080 bytes (RX)
  - Up to 256 descriptors in the buffer descriptor table per channel
  - Configures COMMAC according to commands specified in the descriptor status/control field
  - Updates the descriptor status/control field at the end of packet transfer according to the status received from COMMAC
  - Buffer-based interrupt capabilities for each channel
  - Concurrent operation of RX and TX channels
  - Configuration using Device Control Registers (DCRs)
  - Programmable PLB arbitration priority
  - PLB/OPB error detection
- Z media independent interface (ZMII)
  - Support for one MII PHY
  - Support for two RMII PHYs
    - 10Mbps and 100Mbps data rates
    - 50mhz reference clock sourced from EMAC, or an external source, to the PHY
    - Independent 2-bit transmit and receive data paths
  - Support for two PHYs
    - 10Mbps and 100Mbps data rates

    - Half and full duplex operation
    - System clock sharing for multiple EMACs and PHYs
    - Independent 1-bit transmit and receive data paths
- Two on-chip ethernet ports (EMACs)
    - Multi-speed capability for full or half duplex at ~~100~~10/~~10~~ 100 Mbps
    - 2 KB transmit fifo, 4 KB receive fifo
    - One MII or two RMII ports
- General purpose timer (GPT)
    - Provides a separate time base counter and system timers in addition to those defined in the processor core.
    - 32-bit Time Base Counter driven by the OPB bus Clock
    - Five 32-bit compare timers
- Two universal interrupt controllers (UICs)
    - Supports programmable interrupt handling from a variety of sources
    - Support for asynchronous level- or edge-sensitive interrupt types
    - Programmable polarity for all interrupt types
- Two universal asynchronous receiver/transmitters (UARTs)
    - Compatible with the NS 16750
    - 16-byte send FIFO, 16-byte receive FIFO
    - Full duplex operation
    - Programmable baud rate generator
    - Supports 5- to 8-bit word size, 1 or 2 stop bits, even, odd, or no parity
    - One 8-wire interface (UART0) and one 4-wire interface (UART1)
- Master and slave Inter-integrated circuit (IIC) controllers
    - Complies with Phillips I2C specifications
    - Two wire, bi-directional, open-drain, low-speed serial interface
    - 100-kHz and 400-kHz operation
    - 8-bit data transfers
    - 7-bit and 10-bit addressing
- General purpose I/O (GPIO) interface
    - Allows flexible control of up to 23 multiplexed I/Os with user-defined functions
    - Direct control of all functions from registers programmed via memory-mapped addresses
    - Support for 12 external interrupts
    - outputs can be programmed to emulate an open drain driver
- External bus master interface (EBMI)
    - A 32-bit address bus and 32-bit data bus
    - Fully synchronous

- Support for 8-bit, 16-bit, and 32-bit masters
- Support for 20 to 32 bit address bus to reduce system pinout
- Burst protocol support
- Programmable data parity generation/checking
- Special Cycles for access into system DCR facilities
- Arbitration for bus ownership with the system default master

- External bus controller (EBC)
  - Provides direct attachment for most SRAM/Flash type memory and peripheral devices
  - Minimizes the amount of external glue logic needed to communicate with memory and peripheral devices
  - Supports device-paced transfers with optional bus-timeout
  - Support for 8-bit, 16-bit, and 32-bit masters
  - Separate 32-bit address bus

1.1.3 PPC440GP CoreConnect Features

- 128-bit processor local bus (PLB)
  - Fully synchronous high performance 133MHz 64-bit address bus and 128-bit data bus
  - Provides a standard interface between the processor core and the on-chip peripheral cores
  - Supports up to eight masters and any number of slave devices
  - Decoupled address and data buses support split-bus transaction capability for improved bandwidth
  - Bus arbitration-locking mechanism allows for master-driven atomic operations
  - Byte-enable capability allows for unaligned transfers and odd-byte transfers
  - Overlapping of read and write transfers allows two data transfers per clock cycle for maximum bus utilization as well as 4.2 GB/s at 133 MHz.
- 32-bit on-chip peripheral bus (OPB)
  - Fully synchronous 66MHz 36-bit address bus and 32-bit data bus
  - Supports 8-bit, 16-bit and 32-bit masters and slaves
  - Dynamic bus sizing; byte, halfword, and fullword transfers
  - Sequential address (Burst) protocol support
  - Supports multiple OPB bus masters
  - Uses a distributed multiplexer method of attachment instead of tristate drivers to ease manufacturing test
- 32-bit device control register (DCR)bus
  - A simple but flexible interface with distributed multiplexer architecture
  - 10-bit address bus and 32-bit data bus
  - 2-cycle minimum read or write transfers extendable by slave or master
  - Handshake supports clocked asynchronous transfers
  - Slaves may be clocked either faster or slower than the master

- 128-bit processor local bus arbiter core
  - Consists of bus arbitration control unit, a watchdog timer, address path, write data path, and read data path units
  - Arbitration and data steering support for eight masters
  - Programmable read and write address pipelining
  - Programmable high bus utilization feature
  - Fixed and rotating priority schemes via strapping or programming
  - High frequency 3 cycle acknowledge timing protocol
- 32-bit on-chip peripheral bus arbiter core
  - Internal 32-bit address bus and 32-bit data bus core for on-chip peripheral integration
  - Arbitrates for OPB resources for up to four bus masters
  - Dynamic priority reordering mode, implementing a true least recently used (LRU) algorithm
  - Bus parking modes for reduced access latency
- 128-bit PLB to OPB bridge core
  - Enables transfers of data between the PLB and OPB under the direction of PLB master devices
  - External programmable address space via address decode pin
  - Uses 36 bits of PLB ABus which allows for OPB addresses up to 36 bits in size
  - Supports word, double word and quad word burst reads and writes, including fixed-length bursts
  - Supports pipelining for read and write transfers
  - Supports programmable PLB rearbitrate feature
  - Supports PLB bus-speeds at 2x, 3x, 4x, or 5x the frequency of the OPB
- 128-bit OPB to PLB bridge core
  - Enables transfers of data between the OPB and PLB under the direction of OPB master devices
  - 128-bit PLB master interface supports quadword reads, and quadword, doubleword, word, halfword, and byte writes
  - Ability to be mapped to any OPB address space
  - Single-cycle response to OPB reads and writes
  - Supports PLB bus-speeds at 2x, 3x, 4x, or 5x the frequency of the OPB
- PLB performance monitor
  - Hardware for counting certain events associated with PLB transactions
  - Registers to allow selection of master, slave, and generic events to be counted.
  - Power Management Support (Class II).

1.2 PowerPC Architecture

The PowerPC Architecture comprises three levels of standards:

- PowerPC User Instruction Set Architecture (UISA), including the base user-level instruction set, user-level registers, programming model, data types, and addressing modes. This is referred to as Book I of the PowerPC Architecture.
- PowerPC Virtual Environment Architecture, describing the memory model, cache model, cache-control instructions, address aliasing, and related issues. While accessible from the user level, these features are intended to be accessed from within library routines provided by the system software. This is referred to as Book II of the PowerPC Architecture.
- PowerPC Operating Environment Architecture, including the memory management model, supervisor-level registers, and the exception model. These features are not accessible from the user level. This is referred to as Book III of the PowerPC Architecture.

Book I and Book II define the instruction set and facilities available to the application programmer. Book III defines features, such as system-level instructions, that are not directly accessible by user applications. The PowerPC Architecture is described in *The PowerPC Architecture: A Specification for a New Family of RISC Processors*.

The PowerPC Architecture ensures application code compatibility across all PowerPC implementations to help maximize the cross-platform portability of applications developed for PowerPC processors. This is accomplished through compliance with the first level of architectural standard, the PowerPC UISA, which is common for all PowerPC implementations.

1.3 PPC440GP as a PowerPC Implementation

The PPC440GP implements the full, 32-bit fixed-point subset of the Book-E Enhanced PowerPC Architecture. Although it fully complies with these architectural specifications, the 64-bit operations of the architecture are not supported, nor the floating point operations. Within the RISC core, the 64-bit operations and the floating point operations are trapped, and the floating point operations can be emulated using software.The PPC440GP RISC processor core is a high-performance, low-cost, low-power engine that implements the flexible and powerful Book-E Enhanced PowerPC Architecture.

The PPC440GP also provides a number of optimizations and extensions to the lower layers of the Book-E Enhanced PowerPC Architecture. Some of the specific implementation features of the PPC440GP are simply extensions of the Book-E Enhanced PowerPC Architecture. These features are included to enhance performance, integrate functionality, and reduce system complexity in embedded control applications.

1.4 PPC440GP RISC Core Functions

The PPC440GP RISC processor core contains a dual-issue, superscalar, pipelined processing unit, along with other functional elements required by embedded ASIC product specifications. These other functions include memory management, cache control, timers, and debug facilities. The processor local bus (PLB) system interface has been extended to 128 bits and is fully compatible with the IBM CoreConnect on-chip system architecture, providing the framework to efficiently support system-on-a-chip (SOC) designs.

In addition, the PPC440GP RISC processor core is a member of the PowerPC 400 Series of advanced embedded processor cores, which is supported by the PowerPC Embedded Tools Program. In this program, IBM and many third-party vendors offer a full range of robust development tools for embedded applications. Among these are compilers, debuggers, real-time operating systems and logic analyzers.

The PPC440GP includes a seven-stage pipelined PowerPC core, which consists of a three stage, dual-issue instruction fetch and decode unit with attached branch unit, together with three independent, 4-stage pipelines for complex integer, simple integer, and load/store operations, respectively. The PPC440GP also includes a memory management unit (MMU); separate instruction and data cache units; JTAG, debug, and trace logic; and timer facilities.

1.4.1 Superscalar Instruction Unit

The PPC440GP's instruction unit fetches, decodes, and issues two instructions per cycle to any combination of the three execution pipelines and/or the APU interface. The instruction unit includes a branch unit which provides dynamic branch prediction using a branch history table (BHT), as well as a branch target address cache (BTAC). These mechanisms greatly improve the branch prediction accuracy and reduce the latency of taken branches, such that the target of a branch can usually be executed immediately after the branch itself, with no penalty.

1.4.2 Execution Pipelines

The PPC440GP contains three execution pipelines: complex integer, simple integer, and load/store. Each pipeline consists of four stages and can access the six-ported (three read, three write) GPR file. In order to improve performance and avoid contention for the GPR file, there are two identical copies of the file. One is dedicated to the complex integer pipeline, while the other is shared by the simple integer and the load/store pipelines.

The complex integer pipeline handles all arithmetic, logical, branch, and system management instructions (such as interrupt and TLB management, move to/from system registers, and so on). This pipeline also handles multiply and divide operations, and 24 DSP instructions that perform a variety of multiply-accumulate operations. The complex integer pipeline multiply unit can perform 32-bit $\times$ 32-bit multiply operations with single-cycle throughput and three-cycle latency;16-bit $\times$ 32-bit multiply operations have only two-cycle latency. Divide operations take 33 cycles.

The simple integer pipeline can handle most arithmetic and logical operations which do not update the Condition Register (CR).

The load/store pipeline handles all load, store, and cache management instructions. All misaligned operations are handled in hardware, with no penalty on any operation which is contained within an aligned 16-byte region. The load/store pipeline supports all operations to both big-endian and little-endian data regions.

1.4.3 Instruction and Data Cache Controllers

The PPC440GP provides separate instruction and data cache controllers and arrays, which allow concurrent access and minimize pipeline stalls. The storage capacity of the cache arrays, which can range from 8KB–32KB each, depends upon the implementation. Cache sizes are fixed in PPC440GP. Both cache controllers have 32-byte lines, and both are highly-associative, with 64-way set-associativity for 32KB sizes. The PowerPC instruction set provides a rich set of cache management instructions for software-enforced coherency. The PPC440GP implementation also provides special debug instructions that can directly read the tag and data arrays.

The cache controllers connect to the PLB bus for connection to the IBM CoreConnect system-on-a-chip environment.

See "Instruction and Data Caches" on page 5-1.

0.0.0.1 Instruction Cache Controller (ICC)

The ICC delivers two instructions per cycle to the instruction unit of the PPC440GP. The ICC includes a speculative pre-fetch mechanism which can be configured to automatically pre-fetch a burst of up to three additional lines upon any fetch request which misses in the instruction cache. These speculative pre-fetches can be abandoned if the instruction execution branches away from the original instruction stream.

The ICC supports cache line locking, at either an 8-line or 16-line granularity, depending on cache size (16-line for 32KB). Cache sizes are fixed in PPC440GP. In addition, the notion of a "transient" portion of the cache is supported, in which the cache can be configured such that only a limited portion is used for instruction cache lines from memory pages which are designated by a storage attribute from the MMU as being transient in nature. Such memory pages would contain code which is unlikely to be re-used once the processor moves on to the next series of instruction lines, and thus performance may be improved by preventing each series of instruction lines from overwriting all of the "regular" code in the instruction cache.

0.0.0.2 Data Cache Controller (DCC)

The DCC handles all load and store data accesses, as well as the PowerPC data cache management instructions. All misaligned accesses are handled in hardware, with those accesses that are contained within a half-line (16 bytes) being handled as a single request. Load and store accesses which cross a 16-byte boundary are broken into two separate accesses by the hardware.

The data cache can be operated in a store-in (copy-back) or write-through manner, according to the write-through storage attribute specified for the memory page by the MMU. In addition, the DCC supports both "store-with-allocate" and "store-without-allocate" operation, such that store operations which miss in the data cache can either "allocate" the line in the cache by reading it in and storing the new data into the cache, or alternatively bypassing the cache on a miss and simply storing the data to memory. This characteristic can also be specified on a page-by-page basis by a storage attribute in the MMU.

The DCC also supports cache line locking and "transient" data, in the same manner as the ICC (see "Instruction Cache Controller (ICC)" above).

The DCC provides extensive load, store, and flush queues, such that up to three outstanding line fills and up to four outstanding load misses can be pending, and the DCC can continue servicing subsequent load and store hits in an out-of-order fashion. Store-gathering can also be performed on caching inhibited, write-through, and "without-allocate" store operations, for up to 16 contiguous bytes. Finally, each cache line has four separate "dirty" bits (one per doubleword), so that the amount of data flushed on cache line replacement can be minimized.

See *Section 5 Instruction and Data Caches* on page 205

1.4.4 Memory Management Unit (MMU)

The PPC440GP supports a flat, 36-bit (64GB) real (physical) address space. This 36-bit real address is generated by the MMU, as part of the translation process from the 41-bit virtual address, which is calculated by the processor core as an instruction fetch or load/store address.

The MMU provides address translation, access protection, and storage attribute control for embedded applications. The MMU supports demand paged virtual memory and other management schemes that require precise control of logical to physical address mapping and flexible memory protection. Working with appropriate system level software, the MMU provides the following functions:

- Translation of the 32-bit effective address space into the 36-bit real address space
- Page level read, write, and execute access control
- Storage attributes for cache policy, byte order (endianness), and speculative memory access
- Software control of page replacement strategy

The translation lookaside buffer (TLB) is the primary hardware resource involved in the control of translation, protection, and storage attributes. It consists of 64 entries, each specifying the various attributes of a given page of the address space. The TLB is fully-associative; the entry for a given page can be placed anywhere in the TLB.

Software manages the establishment and replacement of TLB entries. This gives system software significant flexibility in implementing a custom page replacement strategy. For example, to reduce TLB thrashing or translation delays, software can reserve several TLB entries for globally accessible static mappings. The instruction set provides several instructions for managing TLB entries. These instructions are privileged and the processor must be in supervisor state in order for them to be executed.

The first step in the address translation process is to expand the effective address into a virtual address. This is done by taking the 32-bit effective address and appending to it an 8-bit Process ID (PID), as well as a 1-bit "address space" identifier (AS). The PID value is provided by the PID register (see ~~Chapter 6, "Memory Management"~~ *Memory Management* on page 233). The AS identifier is provided by the Machine State Register (MSR), which contains separate bits for the instruction fetch address space (MSR[IS]) and the data access address space (MSR[DS]). Together, the 32-bit effective address, the 8-bit PID, and the 1-bit AS form a 41-bit virtual address. This 41-bit virtual address is then translated into the 36-bit real address using the TLB.

The MMU divides the address space (whether effective, virtual, or real) into pages. Eight page sizes (1KB, 4KB, 16KB, 64KB, 256KB, 1MB, 16MB, 256MB) are simultaneously supported, such that at any given time the TLB can contain entries for any combination of page sizes. In order for an address translation to occur, a valid entry for the page containing the virtual address must be in the TLB. An attempt to access an address for which no TLB entry exists causes an Instruction (for fetches) or Data (for load/store accesses) TLB Error exception.

To improve performance, both the instruction cache and the data cache maintain separate "shadow" TLBs. The instruction shadow TLB (ITLB) contains four entries, while the data shadow TLB (DTLB) contains eight. These shadow arrays minimize TLB contention between instruction fetch and data load/store operations. The instruction fetch and data access mechanisms only access the main 64-entry unified TLB when a miss occurs in the respective shadow TLB. The penalty for a miss in either of the shadow TLB's is three cycles. Hardware manages the replacement and invalidation of both the ITLB and DTLB; no system software action is required.

Each TLB entry provides separate user state and supervisor state read, write, and execute permission controls for the memory page associated with the entry. If software attempts to access a page for which it does not have the necessary permission, an Instruction (for fetches) or Data (for load/store accesses) Storage exception will occur.

Each TLB entry also provides a collection of storage attributes for the associated page. These attributes control cache policy (such as cachability and write-through vs. copy-back behavior), byte order (big-endian vs. little-endian), and enablement of speculative access for the page. In addition, a set of four storage attributes are provided. These attributes can be used to control various system-level behaviors. They can also be configured to control whether data cache lines are allocated upon a store miss, and whether accesses to a given page should use the "normal" or "transient" portions of the instruction or data cache.

~~See "Memory Management" on page 6-1.~~

See *Section 6 Memory Management* on page 233

1.4.5 Interrupts and Exceptions

An interrupt is the action in which the processor saves its old context (Machine State Register (MSR) and next instruction address) and begins execution at a pre-determined interrupt-handler address, with a modified MSR.Exceptions are the events that may cause the processor to take an interrupt, if the corresponding interrupt type is enabled. Exceptions may be generated by the execution of instructions, or by signals from devices external to the PPC440GP, the internal timer facilities, debug events, or error conditions.

See *Section 10 Interrupts and Exceptions* on page 337

1.4.6 Timers

The PPC440GP contains a Time Base and three timers: a Decrementer (DEC), a Fixed Interval Timer (FIT), and a Watchdog Timer. The Time Base is a 64-bit counter which gets incremented at a frequency either equal to the processor core clock rate or as controlled by a separate asynchronous timer clock input to the core. No interrupt is generated as a result of the Time Base rolling over to zero.

The DEC is a 32-bit register that is decremented at the same rate at which the Time Base is incremented. The user loads the DEC register with a value to create the desired interval. When the register is decremented to zero, a number of actions occur: the DEC stops decrementing, a status bit is set in the Timer Status Register (TSR), and a Decrementer exception is reported to the core's interrupt mechanism. Optionally, the DEC can be programmed to automatically reload the value contained in the Decrementer Auto-Reload register (DECAR), after which the DEC resumes decrementing. The Timer Control Register (TCR) contains the interrupt enable for the Decrementer interrupt.

The FIT generates periodic interrupts based on the transition of a selected bit from the Time Base. Users can select one of four intervals for the FIT period by setting a control field in the TCR to select the appropriate bit from the Time Base. When the selected Time Base bit transitions from 0 to 1, a status bit is set in the TSR and a Fixed Interval Timer exception is reported to the core's interrupt mechanism. The FIT interrupt enable is contained in the TCR.

Similar to the FIT, the Watchdog Timer also generates a periodic interrupt based on the transition of a selected bit from the Time Base. Users can select one of four intervals for the watchdog period, again by setting a control field in the TCR to select the appropriate bit from the Time Base. Upon the first transition from 0 to 1 of the selected Time Base bit, a status bit is set in the TSR and a Watchdog Timer exception is

reported to the core's interrupt mechanism. The Watchdog Timer can also be configured to initiate a hardware reset if a second transition of the selected Time Base bit occurs prior to the first Watchdog exception being serviced. This capability provides an extra measure of recoverability from potential system lock-ups.

~~See "Timer Facilities" on page 11-1.~~

See *Section 11 Timer Facilities* on page 383

1.4.7 Debug Facilities

The PPC440GP debug facilities include debug modes for the various types of debugging used during hardware and software development. Also included are debug events that allow developers to control the debug process. Debug modes and debug events are controlled using debug registers in the chip. The debug registers are accessed either through software running on the processor, or through the JTAG port.

The debug modes, events, controls, and interfaces provide a powerful combination of debug facilities for hardware development tools, such as the RISCWatch™ debugger from IBM. A brief overview of the debug modes and development tool support are provided below.

~~See "Debug Facilities" on page 15-1.~~

See *Section 15 Debug Facilities* on page 429

1.4.7.1 Debug Modes

The PPC440GP supports four debug modes: internal, external, real-time-trace, and debug wait. Each mode supports a different type of debug tool used in embedded systems development. Internal debug mode supports software-based ROM monitors, and external debug mode supports a hardware emulator type of debug. Real-time-trace mode uses the debug facilities to indicate events within a trace of processor execution in real time. Debug wait mode enables the processor to continue to service real-time critical interrupts while instruction execution is otherwise stopped for hardware debug. The debug modes are controlled by Debug Control Register 0 (DBCR0) and the setting of bits in the Machine State Register (MSR).

Internal debug mode supports accessing architected processor resources, setting hardware and software breakpoints, and monitoring processor status. In internal debug mode, debug events can generate debug exceptions, which can interrupt normal program flow so that monitor software can collect processor status and alter processor resources.

Internal debug mode relies on exception-handling software—running on the processor—along with an external communications path to debug software problems. This mode is used while the processor continues executing instructions and enables debugging of problems in application or operating system code. Access to debugger software executing in the processor while in internal debug mode is through a communications port on the processor board, such as a serial port or ethernet connection.

External debug mode supports stopping, starting, and single-stepping the processor, accessing architected processor resources, setting hardware and software breakpoints, and monitoring processor status. In external debug mode, debug events can architecturally "freeze" the processor. While the processor is frozen, normal instruction execution stops, and the architected processor resources can be accessed and altered using a debug tool (such as RISCWatch) attached through the JTAG port. This mode is useful for debugging hardware and low-level control software problems.

1.5 PPC440GP Internal Buses

The on-chip bus architecture, which consists of the processor local bus (PLB), on-chip peripheral bus (OPB), and device control register (DCR) bus, provides a link between the cache units in the processor core and other PLB and OPB master and slave devices used in the PPC440GP.

The PLB is a high performance bus used to access memory through bus interface units. Lower performance peripherals (such as serial ports, general purpose I/O interface and inter integrated circuit controllers) are attached to the OPB. A bridge between the PLB and OPB enables data transfers between PLB masters and OPB slaves.

The DCR bus is used primarily to access status and control registers of the various PLB and OPB masters and slaves. The DCR bus offloads status and control read and write transfers from the PLB.

~~See "On-Chip Buses" on page 2-1.~~

See *Section 2 On-Chip Buses* on page 91

1.6 PPC440GP Interfaces

A brief description of various PPC440GP interfaces (including high performance and low performance peripheral cores) used in this system-on-a-chip follows with a cross reference to more detailed information available in this book.

1.6.1 Reset Interface

Resetting the PPC440GP is accomplished by asserting of reset inputs into the PPC440GP processor core, which is not reset using a scan-flush mechanism. Attempts to reset the PPC440GP core using scan-flush result in indeterminate logic states in the PPC440GP processor core. However, a scan-flush mechanism can *initialize* the PPC440GP. Some customers may wish to *initialize* the PPC440GP out of its undefined state to remove Xs in simulation (a practice *not* endorsed by the PPC440GP design group, because this can mask code and hardware initialization problems).A customer implementing a scan-flush *initialization* must still properly reset the PPC440GP by the described methods after the scan-flush initialization.

~~See "Reset and Initialization" on page 7-1.~~

See *Section 7 Reset and Initialization* on page 259

1.6.2 JTAG Interface

The PPC440GP JTAG port is enhanced to support the attachment of a debug tool such as the RISCWatch product from IBM. Through the JTAG test access port, and using the debug facilities designed into the PPC440GP, a debug workstation can single-step the processor and interrogate internal processor state to facilitate hardware and software debugging. The enhancements comply with the IEEE 1149.1 specification for vendor-specific extensions, and are therefore compatible with standard JTAG hardware for boundary-scan system testing.

If the ASIC designer wishes to incorporate JTAG chip testing capability, or other optional features of IEEE Standard 1149.1, such as the use of a Device ID Register, a parallel TAP port must be built to work with the PPC440GP, and PPC440GP-specific instruction decodes must be identified so that the ASIC design can combine PPC440GP functions with additional desired functions. Contact IBM Embedded PowerPC Technical Support; ppcsupp@us.ibm.com) for details.

1.6.3 Trace Interface

The trace interface enables the connection of an external trace tool, such as RISCWatch, and allows for user-extended trace functions. It is important to note that users can have full trace capability without adding ASIC logic, although including some trace control logic in the ASIC can provide some benefits.

To use the RISCTrace feature of RISCWatch, the C440_trcExecutionStatus(0:4), C440_trcBranchStatus(0:2), and C440_trcTraceStatus(0:6) outputs must be sent to chip I/O, along with the C440_trcCycle signal for a total of 16 chip I/Os. It is the responsibility of the ASIC designer to ensure that the C440_trcCycle is properly adjusted to be used as a clock to the RISCTrace capture device. Or, alternatively, the data can be properly delayed to meet setup/hold times when the C440_trcCycle signal is used directly as the data capture clock. The C440_trcCycle signal, as well as the other trace outputs, switch at 1/4 the core clock frequency, based on a 4:1 core to trace clock ratio.

To save chip I/O, trace outputs can be multiplexed with other outputs, as long as the other outputs are not required during debug/trace. The mux select for these outputs should be controlled by a DCR. Even if the trace outputs are not multiplexed, a DCR should enable and disable the trace outputs to minimize chip and system I/O switching when trace is not in use.

1.6.4 PLB Performance Monitor

The PLB performance monitor (PPM) provides hardware for counting certain events associated with PLB transactions. The contents of the counter(s) data may be read by software and used to analyze and enhance PLB performance, or as a software debug mechanism.

The PPM can perform two types of event counting: occurrence and duration. Occurrence counting is accomplished via a set of counters that increment their value once for each occurrence of a selected event, until a predefined timer has expired. The timer value is programmable via a set of cycle count registers and is capable of generating an external interrupt upon expiration. Duration counting is accomplished via separate registers that increment on every clock cycle that a pre-selected event is active. When enabled, each counter is capable of generating an external interrupt once that counter stops incrementing. Event selections and counter controls are performed via the control, status, and individual counter selection registers.

~~See "PLB Performance Monitor" on page 3-1.~~

See *Section 3 PLB Performance Monitor* on page 117

1.6.5 Bootstrap Controller

The IIC bootstrap controller provides a flexible alternative to traditional strapping pins as a solution for reset configuration. Immediately following System Reset, the IIC bootstrap controller, if enabled, sequentially reads 16 bytes from an IIC slave device accessible from the IIC0 interface. The first 8 bytes of data read is configuration data necessary for initializing the PLL settings, clock ratios, boot location and boot width. The second 8 bytes is user definable configuration data. If the bootstrap controller is not enabled, a set of default strap settings are used.

See *Section 8 IIC Bootstrap Controller* on page 283

1.6.6 Clock and Power Management

The PPC440GP provides a clock and power management (CPM) controller that reduces power dissipation by stopping clocks in unused or dormant functional units. Use of the CPM controller requires careful programming and special consideration to avoid compromising system and functional unit integrity.

See *Section 14 Clock and Power Management* on page 423

1.6.7 Internal SRAM Controller ~~Interface~~

The Internal SRAM Controller (ISC) transfers data between the Processor Local Bus (PLB) and internal SRAM of multiple ranges. The SRAM controller provides support for up to 4 banks of physical memory, each bank configurable to 4, 8, 16, 32, 64, or 128 KB. Width of the SRAM bus is 128 bits.

The SRAM controller has a Device Control Register (DCR) interface to allow a processor to read and write internal registers specific to the SRAM controller to program the attributes of each of the four banks.

~~See "Internal SRAM Controller" on page 16-1.~~

See *Section 16 Internal SRAM Controller* on page 463

1.6.8 DDR_SDRAM Memory Controller ~~Interface~~

The HSPLB_DDR memory controller supports Double Data Rate (DDR) SDRAMs, consists of a PLB slave interface and a DCR interface, and can provide 32- or 64-bit interface to SDRAM memory with optional Error Checking and Correction (ECC).

The controller supports page mode operation with bank interleaving and maintains up to four open pages. The controller supports up to four 512 MB logical banks in limited configurations, providing memory up to 2 GB. Global memory timings, address and bank sizes, and memory addressing modes are programmable. System power can be reduced by placing the DDR_SDRAM controller in sleep and/or self-refresh mode.

~~See "DDR SDRAM Controller" on page 17-1.~~

See *Section 17 DDR SDRAM Controller* on page 473

1.6.9 PCI-X Bridge Controller ~~Interface~~

The PLB-PCI Bridge provides an interface between the Peripheral Component Interconnect (PCI) bus and the Processor Local Bus (PLB). It enables PCI initiators to access PLB slaves and PLB masters to access PCI targets. PCI initiators and PCI targets may be in PCI conventional or PCI-X mode.

The PLB-PCI Bridge also contains an internal register set that can be accessed by both PLB masters and PCI initiators. The PLB-PCI Bridge has many optional and programmable features that enable it to be configured for different applications. These options and features are set using strapping pins or using the internal registers. The PLB-PCI Bridge may be used in "host-bridge" mode or "adapter-bridge" mode. The PLB-PCI Bridge is a "bridge" device, in that it only generates transfers as a master on the PLB or the PCI bus in response to decoding a transfer as a slave on the other bus.

~~See "PLB-PCIX Bridge Controller" on page 18-1.~~

0.0.1 DMA Controller Interface

See *Section 18 PLB-PCIX Bridge Controller* on page 535

1.6.10 Direct Memory Access Controller

The Direct Memory Access (DMA) controller is a Processor Local Bus (PLB) and On-chip Peripheral Bus (OPB) master which supports the autonomous transfer of data between memory and peripherals and from memory-to-memory. The controller provides four DMA channels, each of which has an independent set of configuration registers. Each channel has its own control, count and control, source address, destination address, and scatter/gather address registers. Once these registers are programmed by the PPC440 processor, the DMA controller performs the requested data transfer without the need for processor intervention.

The four DMA channels also support scatter/gather transfers. During a scatter/gather transfer the configuration registers for a particular DMA channel are automatically loaded from a data structure in memory instead of being individually programmed. Since the scatter/gather address register is updated in this process, the channel can optionally reconfigure itself for another transfer when the current one completes.

See "Direct Memory Access Controller" on page 19-1.

See *Section 19 Direct Memory Access Controller* on page 655

1.6.11 Memory Access Layer Interface

The Memory Access Layer (MAL) is a hardware core that manages data transfers between packet-oriented communications cores, also known as COMMACs (communications media access controllers), and memory. To communicate with software device drivers, MAL utilizes a buffer descriptor ring structure in memory. A software device driver uses the buffer descriptor structure to inform MAL about buffer locations and packet or buffer status. MAL uses the buffer descriptors to convey packet transfer status from the COMMAC core back to the software device driver. Each MAL channel requires its own buffer descriptor table ring structure in memory.

The PPC440GP MAL manages the transfer of packets between the two Ethernet Media Access Controllers (EMAC0 and EMAC1) and the memory attached to the PPC440GP (SDRAM or SRAM). The primary function of MAL is to move packets directly between memory and a COMMAC core with minimal involvement of the processor core.

See "Memory Access Layer" on page 20-1.

0.0.2 ZMII Bridge Controller Interface

See *Section 20 Memory Access Layer* on page 681

1.6.12 EMAC to PHY Bridge

The PPC440GP provides an EMAC to PHY Interface Bridge (ZMII) that connects the ethernet media access controllers (EMACs) to standard physical devices (PHYs). Media independent interface (MII) and management data interface (MDI) signals to and from an EMAC are passed to the ZMII bridge. The ZMII bridge passes MII and MDI signals through to the PHY, or formats the MII signals to support the reduced media independent interface (RMII) or serial media independent interface (SMII).

Depending on the ZMII configuration, ZMII provides one of the following off-chip interfaces: one MII, two RMII, or two SMII. The interfaces are mutually exclusive. Interface selection is controlled by the Function Enable Register (ZMII0_FER). RMII and SMII significantly reduce the external signal count required to support multiple EMAC ports. While the MII requires 16 additional pins for a single interface, RMII requires only seven additional pins (six data and control and a common clock) and SMII requires only four additional pins (two data, sync, and a clock). Configuring the ZMII bridge for SMII or RMII allows the use of both EMACs using a reduced number of external pins.

~~RMII and SMII significantly reduce the external signal count required to support multiple EMAC ports. While the MII requires 16 additional pins for a single interface, RMII requires only seven additional pins (six data and control and a common clock) and SMII requires only four additional pins (two data, sync, and a clock). Configuring the ZMII bridge for SMII or RMII allows the use of both EMACs using a reduced number of external pins.~~

~~See "EMAC to PHY Interface Bridge" on page 21-1.~~

See *Section 21 EMAC to PHY Interface Bridge* on page 719

1.6.13 Ethernet Media Access Controller ~~Interface~~

The PPC440GP provides two ethernet media access controllers (EMACs) that are generic implementations of the Ethernet Media Access Control (MAC) protocol complying with ANSI/IEEE Std 802.3 and IEEE 802.3u supplement. EMAC supports half-duplex (CSMA/CD) and full-duplex operation for 10-Mbps and 100-Mbps operations.

Both EMACs are implemented identically, with the exception of register addresses. Each EMAC provides two on-chip peripheral bus (OPB) slave interfaces. The first OPB interface provides access to the EMAC configuration and status registers. The PLB/OPB bridge enables the processor core to access these registers.

~~See "Ethernet Media Access Controllers" on page 22-1.~~

See *Section 22 Ethernet Media Access Controllers* on page 729

1.6.14 General Purpose ~~Timer Interface~~Timers

The General Purpose Timer (GPT) is an on-chip peripheral bus (OPB) soft core that provides a separate time base counter and additional system timers beyond those defined in the PowerPC® architecture. Five compare timers are implemented in PPC440GP. The GPT core is an OPB compliant soft core that provides additional system timers. A time-base is provided by the 32-bit Time Base Counter (TBC) register which continually increments by one unless written or reset. ~~From it's maximum value, it rolls to 0. GPT_chipReset resets the entire GPT core, including the TBC. TBCreset resets just the TBC.~~

~~See "General Purpose Timers" on page 12-1.~~

See *Section 12 General Purpose Timers* on page 393

1.6.15 Universal Interrupt ~~Controller Interface~~Controllers

The PPC440GP contains two universal interrupt controllers (UIC0 and UIC1) that provide all necessary control, status, and communication between the various internal and external interrupt sources and the processor core.

The UICs support 48 internal interrupts and 14 external interrupts. Status reporting (using the UIC Status Register [UIC*x*_SR]) is provided to ensure that systems software can determine the current and interrupting state of the system and respond appropriately. Software can generate interrupts to simplify software development and for diagnostics.

~~See "Universal Interrupt Controller" on page 9-1.~~

See *Section 9 Universal Interrupt Controller* on page 299

1.6.16 Serial Port Operations ~~Interface~~

The PPC440GP contains two universal asynchronous receiver/transmitters (UARTs) which provide two full-duplex serial interfaces to support communications with serial peripheral devices. Each UART is compatible with the National Semiconductor (NS) 16750 chip, and includes a 16-byte send and a 16-byte receive FIFO.

The UART performs serial-to-parallel conversion on data characters received from a peripheral device, and parallel-to-serial conversion on data characters received from the processor. The processor can read the complete status of the UART at any time during the functional operation. Status information reported includes the type and condition of the transfer operations being performed by the UART, as well as any error conditions, such as parity, overrun, framing, and break interrupt.

~~See "Serial Port Operations" on page 23-1.~~

See *Section 23 Serial Port Operations* on page 777

~~0.0.3 IIC Interface~~

1.6.17 Inter-Integrated Circuit Bus

The PPC440GP provides two inter-integrated circuit (IIC) bus interfaces, IIC0 and IIC1, complying with specifications contained in the Phillips ® Semiconductors document The I2C bus and how to use it (including specifications) (1995 update).

Each IIC bus has a two wire, bi-directional, open-drain, low-speed serial interface. The serial clock (IIC0SClk and IIC1SClk) and serial data (IIC0SDA and IIC1SDA) lines are bidirectional, to support multiple bus masters and to mix high- and low-speed devices on the same bus.

~~See "IIC Bus Interface" on page 24-1.~~

See *Section 24 IIC Bus Interface* on page 801

1.6.18 GPIO Operations ~~Interface~~

The GPIO Controller is an OPB macro that controls up to 32 bidirectional module I/O pins with user-programmable functions. To reduce the quantity of module I/O on the PPC440GP package there are no dedicated general purpose I/Os. Each of the GPIOs has been assigned a function pin such as an interrupt.

~~See "GPIO Operations" on page 25-1.~~

See *Section 25 GPIO Operations* on page 823

1.6.19 External Bus Master ~~Interface~~

The External Peripheral Bus Interface (EXPB) is used as a mechanism for external master devices to gain access to internal system facilities. It is also used to attach external slave peripherals such as ROM and SRAM, DMA device paced memory devices, and DMA peripheral devices. The External slave peripherals are accessed by the External Bus Controller (EBCO) that acts as the default EXPB master. The external master device must arbitrate for ownership of the EXPB through the External Bus Master Interface (EBMI).

~~See "External Bus Master Interface (EBMI)" on page 26-1.~~

See *Section 26 External Bus Master Interface* on page 831

1.6.20 External Bus Controller ~~Interface~~

The PPC440GP External Bus Controller attached to the OPB (EBCO) provides direct attachment for most SRAM/Flash type memory and peripheral devices. The interface minimizes the amount of external glue logic needed to communicate with memory and peripheral devices. This reduces the embedded system device count, circuit board area, and cost.

To eliminate off-chip address decoding, the EBCO provides eight programmable chip selects that enable system designers to locate memory and peripherals within the PPC440GP memory map. Chip select, data bus, and associated control signal timings are programmable for both single and burst transfers. For peripherals with variable timing requirements, the EBCO supports device-paced transfers with optional bus-timeout. System design is further simplified through dynamic bus sizing, which supports seamlessly attaching 8-, 16-, and 32-bit wide memories and peripherals. Whenever a size mismatch exists between a read or write operation and the externally attached device, the EBCO automatically packs or unpacks data as appropriate.

Using the External Bus Master Interface (EBMI) core, external masters may arbitrate and gain access to the peripheral interface. Once an external master owns the peripheral interface, it can read and write all OPB-addressable memory using the External Peripheral Bus (EXPB) protocol, with the exception of devices controlled by the EBCO. Typical destinations for external master transactions are PCI address space and SDRAM memory. For EBCO-attached peripherals and memory, the external master is required to directly control the target.

~~See "External Bus Controller" on page 27-1.~~

See *Section 27 External Bus Controller* on page 855

1.7 PPC440GP Supported Configurability

The PowerPC instruction set and register set implemented in PPC440GP provide a high degree of user control over configuration and operation of the PPC440GP functional units.

1.7.1 PPC440GP Addressing Modes

As a 32-bit implementation of the Book-E Enhanced PowerPC Architecture, the PPC440GP implements a uniform 32-bit effective address (EA) space. Effective addresses are expanded into virtual addresses and then translated to 36-bit (64GB) real addresses by the memory management unit (see ~~Chapter 6, "Memory Management,"~~*Memory Management* on page 233 for more information on the translation process). The organization of the real address space into a physical address space is described in ~~Chapter 4, "Programming Model."~~*Programming Model* on page 145

The PPC440GP generates an effective address whenever it executes a storage access, branch, cache management, or translation lookaside buffer (TLB) management instruction, or when it fetches the next sequential instruction.

Bytes in storage are numbered consecutively starting with 0. Each number is the address of the corresponding byte. Data storage operands accessed by the integer load/store instructions may be bytes, halfwords, words, or—for load/store multiple and string instructions—a sequence of words or bytes, respectively. Data storage operands accessed by auxiliary processor unit (APU) load/store instructions can be bytes, halfwords, words, doublewords, or quadwords. The address of a storage operand is the address of its first byte (that is, of its lowest-numbered byte). Byte ordering can be either big-endian or little-endian, as controlled by the endian storage attribute (see ~~"Byte Ordering" on page 4-5~~*Byte Ordering* on page 149; also see ~~"Endian (E)" on page 6-15~~*Endian (E)* on page 246 for more information on the endian storage attribute).

The PPC440GP supports the following addressing modes, which enable efficient retrieval and storage of data in memory:

- Base plus displacement addressing
- Indexed addressing
- Base plus displacement addressing and indexed addressing, with update

In the base plus displacement addressing mode, an effective address (EA) is formed by adding a displacement to a base address contained in a GPR (or to an implied base of 0). The displacement is an immediate field in an instruction.

In the indexed addressing mode, the EA is formed by adding an index contained in a GPR to a base address contained in a GPR (or to an implied base of 0).

The base plus displacement and the indexed addressing modes also have a "with update" mode. In "with update" mode, the effective address calculated for the current operation is saved in the base GPR, and can be used as the base in the next operation. The "with update" mode relieves the processor from repeatedly loading a GPR with an address for each piece of data, regardless of the proximity of the data in memory.

1.7.2 PPC440GP Register Set

The PPC440GP registers can be grouped into basic categories based on function and access mode: general purpose registers (GPRs), special purpose registers (SPRs), time base registers (TBRs), machine state register (MSR), condition register (CR), device control registers (DCRs), and memory-mapped I/O (MMIO) registers.

~~Chapter 29, "Register Summary,"~~*Register Summary* on page 1093 provides a register diagram and a register field description table for each register.

1.7.2.1 General Purpose Registers

The PPC440GP provides 32 general purpose registers (GPRs) each containing 32 bits. Data from the data cache or memory can be loaded into GPRs using integer load instructions; the contents of GPRs can be stored to the data cache or memory using integer store instructions. Most of the integer instructions reference GPRs. The GPRs are also used as targets and sources for most of the instructions which read and write the other register types.

~~"Integer Processing" on page 4-53~~*Section 4.6 Integer Processing* on page 192 provides more information on integer operations and the use of GPRs.

1.7.2.2 Special Purpose Registers

Special Purpose Registers (SPRs) are directly accessed using the **mtspr** and **mfspr** instructions. In addition, certain SPRs may be updated as a side-effect of the execution of various instructions. For example, the Integer Exception Register (XER) (see ~~"Integer Exception Register (XER)" on page 4-53~~*Integer Exception Register (XER)* on page 192) is an SPR which is updated with arithmetic status (such as carry and overflow) upon execution of certain forms of integer arithmetic instructions.

SPRs control the use of the debug facilities, timers, interrupts, memory management, caches, and other architected processor resources. ~~Table 29-2, "Special Purpose Registers Sorted by SPR Number," on page 29-4~~*Table 29-2 Special Purpose Registers Sorted by SPR Number* on page 1095 shows the mnemonic, name, and number for each SPR, in order by SPR number. Each of the SPRs is described in more detail within the section or chapter covering the function with which it is associated. See ~~Table 4-4, "Register Categories," on page 4-13~~*Table 4-4 Register Categories* on page 157 for a cross-reference to the associated document section for each register.

1.7.2.3 Time Base Registers

The PPC440GP provides a 64-bit time base as described in ~~"Timer Facilities" on page 11-1~~*Timer Facilities* on page 383. The time base is implemented as two 32-bit time base registers (TBRs). The low-order32 bits of the time base are read from the TBL and high-order 32 bits are read from the TBU. User-mode access to the TBRs is read-only and there is no explicitly privileged read access to the time base.

The **mftb** instruction reads from TBL and TBU. (Writing the time base is accomplished by moving the contents of a GPR to a pair of SPRs, which are also called TBL and TBU, using the **mtspr** instruction.

1.7.2.4 Machine State Register

The Machine State Register (MSR) is a register of its own unique type that controls important chip functions, such as the enabling or disabling of various interrupt types.

The MSR can be written from a GPR using the **mtmsr** instruction. The contents of the MSR can be read into a GPR using the **mfmsr** instruction. The MSR[EE] bit (external interrupt enable) can be set or cleared atomically using the **wrtee** or **wrteei** instructions. The MSR contents are also automatically saved, altered, and restored by the interrupt-handling mechanism. See ~~"Machine State Register (MSR)" on page 10-7~~*Machine State Register (MSR)* on page 343 for more detailed information on the MSR and the function of each of its bits.

1.7.2.5 Condition Register

The Condition Register (CR) is a 32-bit register of its own unique type and is divided up into eight, independent 4-bit fields (CR0–CR7). The CR may be used to record certain conditional results of various arithmetic and logical operations. Subsequently, conditional branch instructions may designate a bit of the CR as one of the branch conditions (see ~~"Branch Processing" on page 4-46~~*Branch Processing* on page 185). Instructions are also provided for performing logical bit operations and for moving fields within the CR.

See ~~"Condition Register (CR)" on page 4-50~~*Condition Register (CR)* on page 189 for more information on the various instructions which can update the CR.

1.7.2.6 Device Control Registers

Device Control Registers (DCRs) are on-chip registers that control various PPC440GP system functions, such as the operation of PPC440GP buses, peripherals, and certain PPC440GP processor core behaviors. The DCR access instructions are **mtdcr** (move to device control register) and **mfdcr** (move from device control register), which move data between GPRs and the DCRs. ~~See "Device Control Registers" on page 4-16.~~See *Device Control Registers* on page 159

1.7.2.7 Memory-Mapped I/O Registers

The memory-mapped I/O (MMIO) registers are associated with PPC440GP peripherals and are accessed using load and store instructions. MMIO registers are mapped into the system memory and are used to control, configure, and hold status for various PPC440GP functional units. ~~See "Memory-Mapped Input/Output Registers" on page 4-25.~~See *Memory-Mapped Input/Output Registers* on page 168

1.8 PPC440GP Signals

All PPC440GP signals are listed towards the end of this book under two tables, one in alphabetical order and the other by interface category. ~~See "Signal Summary" on page 30-1.~~See *Signal Summary* on page 1529

1.9 Development Tool Support

The PPC440GP provides powerful debug support for a wide range of hardware and software development tools.

The OS Open real-time operating system debugger is an example of an operating system-aware debugger, implemented using software traps.

RISCWatch is an example of a development tool that uses the external debug mode, debug events, and the JTAG port to support hardware and software development and debugging.

The RISCTrace™ feature of RISCWatch is an example of a development tool that uses the real-time trace capability of the PPC440GP.

2. On-Chip Buses

The on-chip bus architecture, which consists of the processor local bus (PLB), on-chip peripheral bus (OPB), and device control register (DCR) bus, provides a link between the cache units in the PPC440GP embedded processor core and other PLB and OPB master and slave devices used in the PPC440GP.

The PLB is a high performance bus used to access memory through bus interface units. The PLB master and slave assignments for the PPC440GP are listed in ~~"PLB Master and Slave Assignments" on page 2-2~~*PLB Master and Slave Assignments* on page 92. High performance peripherals such as PCI-X bridge, DDR SDRAM controller, SRAM controller, media access layer and DMA controller) are attached to the PLB.

Lower performance peripherals (such as serial ports, ethernet controller, external bus controller, general purpose timer, general purpose I/O interface and inter integrated circuit controllers) are attached to the OPB. A bridge between the PLB and OPB enables data transfers between PLB masters and OPB slaves. DMA peripherals can also be OPB peripherals.

The DCR bus is used primarily to access status and control registers of the various PLB and OPB masters and slaves. The DCR bus off-loads status and control read and write transfers from the PLB. However, PPC440GP contains peripherals such as, DDR SDRAM, SRAM, serial port, ethernet controllers and others that use memory mapped registers which are accessed using load/store instructions. ~~See "Device Control Register (DCR) Interface" on page 2-25.~~See *Device Control Register (DCR) Interface* on page 114

The following publications, which are available from your IBM representative and on the IBM Microelectronics technical library (www.chips.ibm.com), describe the on-chip bus architecture:

- *The CoreC*onnect Bus Architecture
- Processor Local Bus Architecture Specifications
- On-Chip Peripheral Bus Architecture Specifications
- Device Cont*rol Register Bus Architecture Specifications*

The PPC440GP block diagram (~~Figure 1-1 on page 1-2~~*Figure 1-1* on page 70) illustrates the on-chip bus structure of the PPC440GP.

2.1 Processor Local Bus

The PLB is a high-performance on-chip bus. The PLB supports read and write data transfers between master and slave devices equipped with a PLB interface and connected through PLB signals.

Each PLB master is attached to the PLB through separate address, read data and write data buses, and transfer qualifier signals. PLB slaves are attached to the PLB through shared, but decoupled, address, read data and write data buses, and transfer control and status signals for each data bus.

Access to the PLB is granted through a central arbitration mechanism that enables masters to compete for bus ownership. This arbitration mechanism provides for fixed and fair priority schemes.

Timing for all PLB signals is provided by a clock source that is shared by all PLB masters and slaves.

2.1.1 PLB Features

- Overlapping of read and write transfers allows two data transfers per clock cycle for maximum bus utilization
- Decoupled address and data buses support split-bus transaction capability for improved bandwidth
- Extendable address pipelining reduces overall bus latency by allowing the latency associated with a new request to be overlapped with an ongoing data transfer in the same direction
- Late master request abort capability reduces latency associated with aborted requests
- Four levels of request priority for each master allow PLB implementations with various arbitration schemes
- Byte-enable capability allows for unaligned transfers and odd-byte transfers.
- Support for 16-, 32-, and 64-byte line data transfers
- Guarded and unguarded memory transfers allow a slave device to enable or disable the prefetching of instructions or data
- DMA buffered, peripheral to memory, memory to peripheral, and DMA memory to memory operations are supported

2.1.2 PLB Master and Slave Assignments

Table 2-1 lists the PLB masters and slaves provided in the PPC440GP.

Table 0-1. PPC440GP PLB Master Assignments

PLB Agent	PLB Masters and Slaves
Processor core instruction cache unit (ICU)	Master
Processor core data cache read unit (DCU)	Master
Processor core data cache write unit (DCU)	Master
Peripheral component interconnect (PCI-X) bridge controller	Master/Slave
Double date rate (DDR) SDRAM controller	Slave
Internal sram controller (ISC)	Slave
Media access layer (MAL)	Master
Direct memory access (DMA) controller	Master
PLB to OPB bridge controller	Slave
OPB to PLB bridge controller	Master

Table 2-1. PPC440GP PLB Master Assignments

PLB Agent	PLB Masters and Slaves
Processor core instruction cache unit (ICU)	Master
Processor core data cache read unit (DCU)	Master
Processor core data cache write unit (DCU)	Master
Peripheral component interconnect (PCI-X) bridge controller	Master/Slave
Double date rate (DDR) SDRAM controller	Slave
Internal sram controller (ISC)	Slave

Table 2-1. PPC440GP PLB Master Assignments

PLB Agent	PLB Masters and Slaves
Media access layer (MAL)	Master
Direct memory access (DMA) controller	Master
PLB to OPB bridge controller	Slave
OPB to PLB bridge controller	Master

2.1.3 PLB Master Priority Assignment

Each PLB master can be programmed to use one of four priority levels during PLB transfers, enabling the system designer to tune PLB transfer priorities to the requirements of a particular application. For example, if an application always requires DMA PCI to SDRAM transfers to have the lowest latency, the DMA PCI master can be programmed to the highest PLB master priority. This causes the PLB arbiter to grant DMA PCI access requests before granting the access requests of any other master.

> **Programming Note:** PLB master priority assignments, which are application-dependent, must be considered carefully to prevent potential lockouts of lower priority masters. For most applications, assigning a priority of 0b10 to each master is a useful starting point.

A register associated with each master controls the priority of that master. Table 2-2 lists the PLB masters and the register fields controlling the priority of the masters. Priorities range from 0b00 (lowest) to 0b11 (highest).

Table 0-2. Registers Controlling PLB Master Priority Assignments

Master ID	Description	Register Field	Comments
0	Processor core instruction cache unit (ICU)	CPC0_CR1[IRF] CPC0_CR1[IRT] CPC0_CR1[IRS]	IcuRdFetch priority setting IcuRdTouch priority setting IcuRdSpec priority setting
1	Processor core data cache read unit (DCU)	CPC0_CR1[DRU] CPC0_CR1[DRT] CPC0_CR1[DRNC] CPC0_CR1[DRLC]	DcuRdUrgent priority setting DcuRdTouch priority setting DcuRdNonCache priority DcuRdLdCache priority setting
2	Processor core data cache write unit (DCU)	CPC0_CR1[DWF] CPC0_CR1[DWS] CPC0_CR1[DWU]	DcuWrFlush priority setting DcuWrStore priority setting DcuWrUrgent priority setting
3	Peripheral component interconnect (PCIX) bridge controller	PCI_BRGOPT1[PLREQ]	
4	Reserved		
5	Media access layer (MAL)	MAL0_CFG[PLBP]	
6	Direct memory access (DMA) controller	DMA0_CR0[CP] DMA0_CR1[CP] DMA0_CR2[CP] DMA0_CR3[CP]	Unique priorities can be assigned to each DMA channel
7	OPB to PLB bridge controller	OPB0_CTRL[PRI]	

Table 2-2. Registers Controlling PLB Master Priority Assignments

Master ID	Description	Register Field	Comments
0	Processor core instruction cache unit (ICU)	CPC0_CR1[IRF] CPC0_CR1[IRT] CPC0_CR1[IRS]	IcuRdFetch priority setting IcuRdTouch priority setting IcuRdSpec priority setting
1	Processor core data cache read unit (DCU)	CPC0_CR1[DRU] CPC0_CR1[DRT] CPC0_CR1[DRNC] CPC0_CR1[DRLC]	DcuRdUrgent priority setting DcuRdTouch priority setting DcuRdNonCache priority DcuRdLdCache priority setting
2	Processor core data cache write unit (DCU)	CPC0_CR1[DWF] CPC0_CR1[DWS] CPC0_CR1[DWU]	DcuWrFlush priority setting DcuWrStore priority setting DcuWrUrgent priority setting
3	Peripheral component interconnect (PCIX) bridge controller	PCI_BRGOPT1[PLREQ]	
4	Reserved		
5	Media access layer (MAL)	MAL0_CFG[PLBP]	
6	Direct memory access (DMA) controller	DMA0_CR0[CP] DMA0_CR1[CP] DMA0_CR2[CP] DMA0_CR3[CP]	Unique priorities can be assigned to each DMA channel
7	OPB to PLB bridge controller	OPB0_CTRL[PRI]	

See ~~"PLB Arbiter Control Register (PLB0_ACR)" on page 2-7~~*PLB Arbiter Control Register (PLB0_ACR)* on page 97 for information about programming the PLB0_ACR to control PLB priority mode and priority order, which determine how the PLB arbitrates simultaneous PLB bus access requests having equal priorities.

2.1.3.1 PLB Master Priority Register (CPC0_PLB)

The PLB master priority register (CPC0_PLB) is a a 32-bit read/write register that describes PLB master and alternate priority setting. ~~Figure 2-1~~Figure 2-1 describes CPC0_PLB register bits.

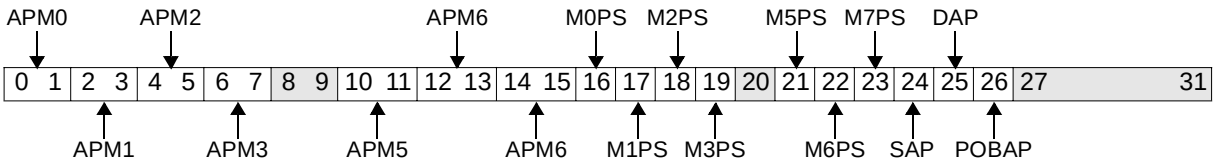


Figure 0-1. PLB Master Priority Register (CPC0_PLB)

0:1	APM0	Alternate PLB master priority setting 0	Alternate ICU read priority setting
2:3	APM1	Alternate PLB master priority setting 1	Alternate DCU read priority setting
4:5	APM2	Alternate PLB master priority setting 2	Alternate DCU write priority setting
6:7	APM3	Alternate PLB master priority setting 3	ALternate PCIX priority setting
8:9		Reserved	
10:11	APM5	Alternate PLB master priority setting 5	Alternate MAL priority setting
12:13	APM6	Alternate PLB master priority setting 6	Alternate DMA priority setting

14:15	APM7	Alternate PLB master priority setting 7	Alternate OPB to PLB bridge priority setting
16	M0PS	Master 0 Priority Setting 0 Master 0 controls own priority setting 1 Alternate PLB master priority setting	ICU read priority setting
17	M1PS	Master 1 Priority Setting 0 Master 1 controls own priority setting 1 Alternate PLB master priority setting	DCU read priority setting
18	M2PS	Master 2 Priority Setting 0 Master 2 controls own priority setting 1 Alternate PLB master priority setting	DCU write priority setting
19	M3PS	Master 3 Priority Setting 0 Master 3 controls own priority setting 1 Alternate PLB master priority setting	PCIX priority setting
20		Reserved	
21	M5PS	Master 5 Priority Setting 0 Master 5 controls own priority setting 1 Alternate PLB master priority setting	MAL priority setting
22	M6PS	Master 6 Priority Setting 0 Master 6 controls own priority setting 1 Alternate PLB master priority setting	DMA priority setting
23	M7PS	Master 7 Priority Setting 0 Master 7 controls own priority setting 1 Alternate PLB master priority setting	OPB to PLB bridge priority setting
24	SAP	SRAM Address Pipelining 0 Address pipelining disabled to SRAM slave 1 Address pipelining enabled to SRAM slave	
25	DAP	DDRSDRAM Address Pipelining 0 Address pipelining disabled to DDR SDRAM slave 1 Address pipelining enabled to DDR SDRAM slave	
26	POBAP	PLB to OPB Bridge Address Pipelining 0 Address pipelining disabled to PLB to OPB bridge slave 1 Address pipelining enabled to PLB to OPB bridge slave	
27:31		Reserved	

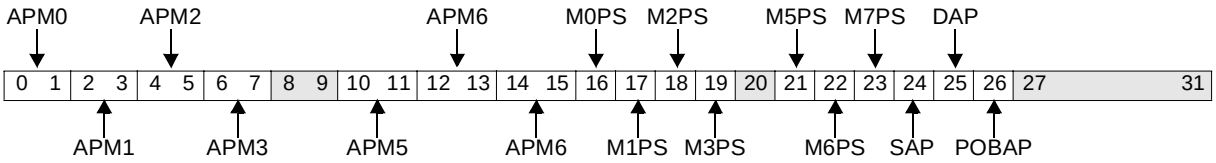


Figure 2-1. PLB Master Priority Register (CPC0_PLB)

0:1	APM0	Alternate PLB master priority setting 0	Alternate ICU read priority setting
2:3	APM1	Alternate PLB master priority setting 1	Alternate DCU read priority setting
4:5	APM2	Alternate PLB master priority setting 2	Alternate DCU write priority setting
6:7	APM3	Alternate PLB master priority setting 3	ALternate PCIX priority setting
8:9		Reserved	
10:11	APM5	Alternate PLB master priority setting 5	Alternate MAL priority setting
12:13	APM6	Alternate PLB master priority setting 6	Alternate DMA priority setting
14:15	APM7	Alternate PLB master priority setting 7	Alternate OPB to PLB bridge priority setting
16	M0PS	Master 0 Priority Setting 0 Master 0 controls own priority setting 1 Alternate PLB master priority setting	ICU read priority setting
17	M1PS	Master 1 Priority Setting 0 Master 1 controls own priority setting 1 Alternate PLB master priority setting	DCU read priority setting
18	M2PS	Master 2 Priority Setting 0 Master 2 controls own priority setting 1 Alternate PLB master priority setting	DCU write priority setting
19	M3PS	Master 3 Priority Setting 0 Master 3 controls own priority setting 1 Alternate PLB master priority setting	PCIX priority setting
20		Reserved	
21	M5PS	Master 5 Priority Setting 0 Master 5 controls own priority setting 1 Alternate PLB master priority setting	MAL priority setting
22	M6PS	Master 6 Priority Setting 0 Master 6 controls own priority setting 1 Alternate PLB master priority setting	DMA priority setting
23	M7PS	Master 7 Priority Setting 0 Master 7 controls own priority setting 1 Alternate PLB master priority setting	OPB to PLB bridge priority setting
24	SAP	SRAM Address Pipelining 0 Address pipelining disabled to SRAM slave 1 Address pipelining enabled to SRAM slave	

25	DAP	DDRSDRAM Address Pipelining 0 Address pipelining disabled to DDR SDRAM slave 1 Address pipelining enabled to DDR SDRAM slave
26	POBAP	PLB to OPB Bridge Address Pipelining 0 Address pipelining disabled to PLB to OPB bridge slave 1 Address pipelining enabled to PLB to OPB bridge slave
27:31		Reserved

*2.1.3.2 Control Register 1 (CPC0_CR1)_*

CPC0_CR1 is a 32-bit read/write register that controls processor core priorities and DDR delay tuning. ~~Figure 2-2~~Figure 2-2 describes CPC0_CR1 register bit definitions.

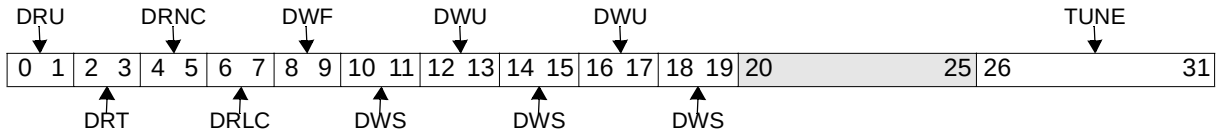


Figure 0-2. Control Register 1 (CPC0_CR1)

0:1	DRU	DcuRdUrgent priority setting used by CPU in some scenarios
2:3	DRT	DcuRdTouch priority setting used by CPU in some scenarios
4:5	DRNC	DcuRdNonCache priority setting used by CPU in some scenarios
6:7	DRLC	DcuRdLdCache priority setting used by CPU in some scenarios
8:9	DWF	DcuWrFlush priority setting used by CPU in some scenarios
10:11	DWS	DcuWrStore priority setting used by CPU in some scenarios
12:13	DWU	DcuWrUrgent priority setting used by CPU in some scenarios
14:15	IRF	IcuRdFetch priority setting used by CPU in some scenarios
16:17	IRT	IcuRdTouch priority setting used by CPU in some scenarios
18:19	IRS	IcuRdSpec priority setting used by CPU in some scenarios
20:25		Reserved

26:31	TUNE	TUNE (5:0) bits for DDR delay line element

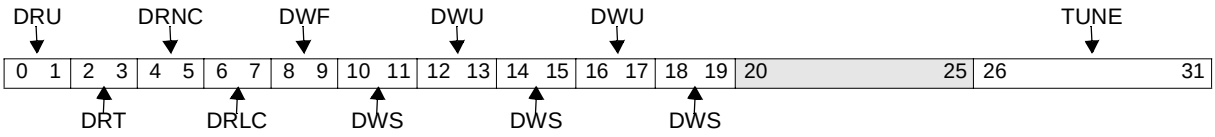


Figure 2-2. Control Register 1 (CPC0_CR1)

0:1	DRU	DcuRdUrgent priority setting used by CPU in some scenarios
2:3	DRT	DcuRdTouch priority setting used by CPU in some scenarios
4:5	DRNC	DcuRdNonCache priority setting used by CPU in some scenarios
6:7	DRLC	DcuRdLdCache priority setting used by CPU in some scenarios
8:9	DWF	DcuWrFlush priority setting used by CPU in some scenarios
10:11	DWS	DcuWrStore priority setting used by CPU in some scenarios
12:13	DWU	DcuWrUrgent priority setting used by CPU in some scenarios
14:15	IRF	IcuRdFetch priority setting used by CPU in some scenarios
16:17	IRT	IcuRdTouch priority setting used by CPU in some scenarios
18:19	IRS	IcuRdSpec priority setting used by CPU in some scenarios
20:25		Reserved
26:31	TUNE	TUNE (5:0) bits for DDR delay line element

2.1.3.3 Master Interrupt Request Register 0 (CPC0_MIRQ0)

CPC0_MIRQ0 is a 32-bit read/write register that controls interrupt request from master 0, 1, 2, 3, and 5. ~~Figure 2-3~~Figure 2-3 describes CPC0_MIRQ0 register bits.

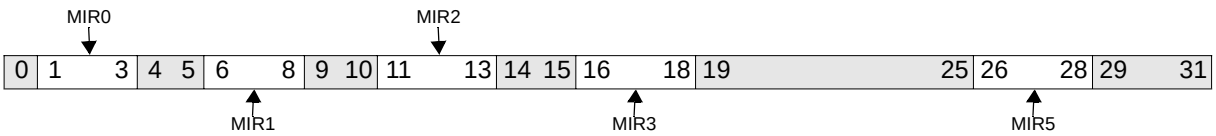


Figure 0-3. Master Interrupt Request Register 0 (CPC0_MIRQ0)

0		Reserved

1:3	MIR0	Master interrupt request 0	ICU read interrupt request from DDR, PCI, and PLB to OPB bridge controller.
4:5		Reserved	
6:8	MIR1	Master interrupt request 1	DCU read interrupt request from DDR, PCI, and PLB to OPB bridge controller.
9:10		Reserved	
11:13	MIR2	Master interrupt request 2	DCU write interrupt request from DDR, PCI, and PLB to OPB bridge controller.
14:15		Reserved	
16:18	MIR3	Master interrupt request 3	PCIX controller interrupt request from DDR, PCI, and PLB to OPB bridge controller
19:25		Reserved	
26:28	MIR5	Master interrupt request 5	MAL controller interrupt request from DDR, PCI, and PLB to OPB bridge controller
29:31		Reserved	

Figure 2-3. Master Interrupt Request Register 0 (CPC0_MIRQ0)

0		Reserved	
1:3	MIR0	Master interrupt request 0	ICU read interrupt request from DDR, PCI, and PLB to OPB bridge controller.
4:5		Reserved	
6:8	MIR1	Master interrupt request 1	DCU read interrupt request from DDR, PCI, and PLB to OPB bridge controller.
9:10		Reserved	
11:13	MIR2	Master interrupt request 2	DCU write interrupt request from DDR, PCI, and PLB to OPB bridge controller.
14:15		Reserved	
16:18	MIR3	Master interrupt request 3	PCIX controller interrupt request from DDR, PCI, and PLB to OPB bridge controller
19:25		Reserved	
26:28	MIR5	Master interrupt request 5	MAL controller interrupt request from DDR, PCI, and PLB to OPB bridge controller
29:31		Reserved	

2.1.3.4 Master Interrupt Request Register 1 (CPC0_MIRQ1)

CPC0_MIRQ1 is a 32-bit read/write register that controls interrupt request from master 6 and 7. Figure 2-4 describes CPC0_MIRQ1 register bits.

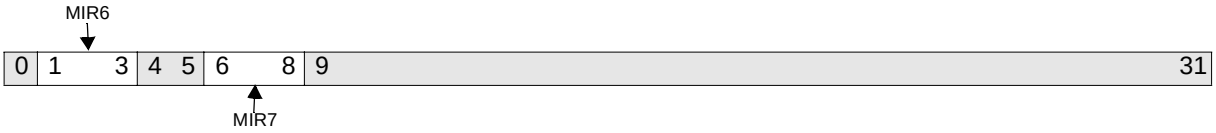


Figure 0-4. Master Interrupt Request Register 1 (CPC0_MIRQ1)

0		Reserved	
1:3	MIR6	Master interrupt request 6	DMA controller interrupt request from DDR, PCI, and PLB to OPB bridge controller.
4:5		Reserved	
6:8	MIR7	Master interrupt request 7	OPB to PLB bridge controller interrupt request from DDR, PCI, and PLB to OPB bridge controller.
9:31		Reserved	

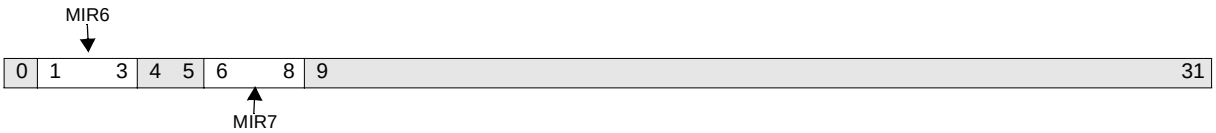


Figure 2-4. Master Interrupt Request Register 1 (CPC0_MIRQ1)

0		Reserved	
1:3	MIR6	Master interrupt request 6	DMA controller interrupt request from DDR, PCI, and PLB to OPB bridge controller.
4:5		Reserved	
6:8	MIR7	Master interrupt request 7	OPB to PLB bridge controller interrupt request from DDR, PCI, and PLB to OPB bridge controller.
9:31		Reserved	

2.1.4 PLB Arbiter Registers

PLB arbiter registers are DCRs accessed using the **mfdcr** and **mtdcr** instructions.

Table 2-3 summarizes the PLB arbiter DCRs.

Table 2-3. PLB Arbiter Registers

Mnemonic	Register Name	Address	Access	Page
PLB0_REVID	PLB Arbiter Revision ID	0x082	R/O	2-797
PLB0_ACR	PLB Arbiter Control Register	0x083	R/W	2-797
PLB0_BESR	PLB Error Status Register	0x084	R/Clear	2-898
PLB0_BEARL	PLB Error Address Register Low	0x086	R/O	2-11100
PLB0_BEARH	PLB Error Address Register High	0x087	R/O	2-11101

2.1.4.1 PLB Revision ID Register (PLB0_REVID)

The PLB Revision ID Register (PLB0_REVID) is a 32-bit read-only register which contains the revision ID of the PLB arbiter core. The contents of the PREV can be accessed by using the move from device control register (**mfdcr)** instruction.

The PLB0_REVID can be accessed with CPU_dcrAddr(6:9) = 0x2. The PLB0_REV is not affected by Reset.

Figure 2-5 shows bit definitions for PLB0_REVID.Figure 2-5—

Figure 0-5. PLB Revision ID Register (PLB0_REVID)

0:31	RevId	This value indicates what revision level this core is. PLB Arbiter core Revision ID = 0x(C27E5031)

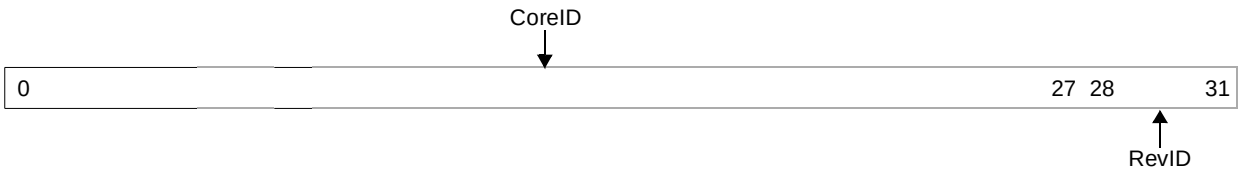


Figure 2-5. PLB Arbiter Revision ID Register (PLB0_REVID)

0:27	CoreID	Core ID	Corresponds to the technology.
28:31	RevId	Revision ID	Corresponds to the revision level.

2.1.4.2 PLB Arbiter Control Register (PLB0_ACR)

The PLB0_ACR controls the modes of operation for the arbiter. The priority mode, priority order, high bus utilization, read pipeline enable, and write pipeline enable are contained in this register. Figure 2-6 describes PLB0_ACR register bits.

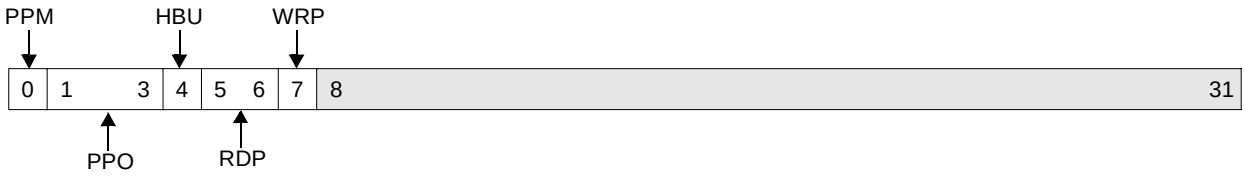


Figure 0-6. PLB Arbiter Control Register (PLB0_ACR)

0	PPM	PLB Priority Mode 0 Fixed 1 Fair	
1:3	PPO	PLB Priority Order 000 Masters 0, 1, 2, 3, 5, 6, 7 001 Masters 1, 2, 3, 5, 6, 7, 0 010 Masters 2, 3, 5, 6, 7, 0, 1 011 Masters 3, 5, 6, 7, 0, 1, 2 100 Masters 5, 6, 7, 0, 1, 2, 3 101 Masters 5, 6, 7, 0, 1, 2, 3, 110 Masters 6, 7, 0, 1, 2, 3, 5 111 Masters 7, 0, 1, 2, 3, 5, 6	The PLB priority order (PPO) will remain a constant value whenPLB priority mode (PPM) bit 0 is set to 0 (fixed). However, the PLB priority order (PPO) bits 1:3 are constantly changing when PLB priority mode (PPM) bit 0 is set to 1 (fair)
4	HBU	High Bus Utilization 0 Disabled 1 Enabled	If read and write pipelining are disabled this feature has no effect on arbiter operation.
5:6	RDP	Read Pipeline Control 00 Read pipelining disabled 01 2 Deep read pipe 10 3 Deep read pipe 11 4 Deep read pipe	
7	WRP	Write Pipeline Control 0 Write Pipeline Disabled 1 2 Deep Write Pipe	
8:31		Reserved	

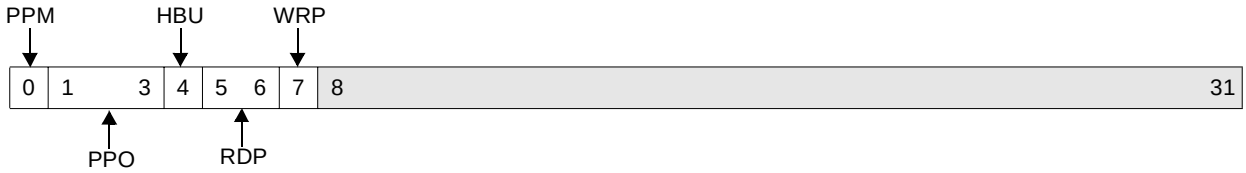


Figure 2-6. PLB Arbiter Control Register (PLB0_ACR)

0	PPM	PLB Priority Mode 0 Fixed 1 Fair	
1:3	PPO	PLB Priority Order 000 Masters 0, 1, 2, 3, 5, 6, 7 001 Masters 1, 2, 3, 5, 6, 7, 0 010 Masters 2, 3, 5, 6, 7, 0, 1 011 Masters 3, 5, 6, 7, 0, 1, 2 100 Masters 5, 6, 7, 0, 1, 2, 3 101 Masters 5, 6, 7, 0, 1, 2, 3, 110 Masters 6, 7, 0, 1, 2, 3, 5 111 Masters 7, 0, 1, 2, 3, 5, 6	The PLB priority order (PPO) will remain a constant value whenPLB priority mode (PPM) bit 0 is set to 0 (fixed). However, the PLB priority order (PPO) bits 1:3 are constantly changing when PLB priority mode (PPM) bit 0 is set to 1 (fair)
4	HBU	High Bus Utilization 0 Disabled 1 Enabled	If read and write pipelining are disabled this feature has no effect on arbiter operation.
5:6	RDP	Read Pipeline Control 00 Read pipelining disabled 01 2 Deep read pipe 10 3 Deep read pipe 11 4 Deep read pipe	
7	WRP	Write Pipeline Control 0 Write Pipeline Disabled 1 2 Deep Write Pipe	
8:31		Reserved	

2.1.4.3 PLB Error Status Register (PLB0_BESR)

The read/clear PLB0_BESR identifies time-out errors on PLB bus transfers, the master initiating the transfer, and the type of transfer.

Each PLB0_BESR[PTE*n*] field (*n* is the master ID) can be locked by the master. Once locked, PLB0_BESR [PTE*n*] fields cannot be updated if a subsequent error occurs until the corresponding PLB0_BESR [FLCK*n*] field is cleared. To clear a PLB0_BESR field, write 1 to the field. Writing 0 to a PLB0_BESR field does not affect the field. Figure 2-7 describes PLB0_BESR register bits.

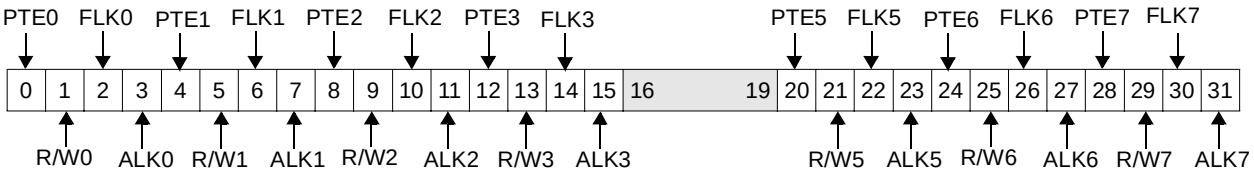


Figure 0-7. PLB Error Status Register (PLB0_BESR)

0	PTE0	Master 0 PLB Timeout Error Status 0 No master 0 timeout error 1 Master 0 timeout error	Master 0 is the read-only instruction cache unit (ICU)
1	R/W0	Master 0 Read/Write Status 0 Master 0 error operation was a write 1 Master 0 ICU error operation was a read	
2	FLK0	Master 0 PLB0_BESR Field Lock 0 Master 0 PLB0_BESR field is unlocked 1 Master 0 field is locked	
3	ALK0	Master 0 PLB0_BEAR Address Lock 0 Master 0 PLB0_BEAR is unlocked 1 Master 0 PLB0_BEAR is locked	
4	PTE1	Master 1 PLB Timeout Error Status 0 No master 1 timeout error 1 Master 1 timeout error	Master 1 is the read-only data cache unit (DCU)
5	R/W1	Master 1 Read/Write Status 0 Master 1 error operation was a write 1 Master 1 error operation was a read	
6	FLK1	Master 1PLB0_BESR Field Lock 0 Master 1 PLB0_BESR field is unlocked 1 Master 1 PLB0_BESR field is locked	
7	ALK1	Master 1 PLB0_BEAR Address Lock 0 Master 1 PLB0_BEAR is unlocked 1 Master 1 PLB0_BEAR is locked	
8	PTE2	Master 2 PLB Timeout Error Status 0 No master 2 timeout error 1 Master 2 timeout error	Master 2 is the write-only data cache unit (DCU)
9	R/W2	Master 2 Read/Write Status 0 Master 2 error operation was a write 1 Master 2 error operation was a read	
10	FLK2	Master 2 PLB0_BESR Field Lock 0 Master 2 PLB0_BESR field is unlocked 1 Master 2 PLB0_BESR field is locked	
11	ALK2	Master 2 PLB0_BEAR Address Lock 0 Master 2 PLB0_BEAR is unlocked 1 Master 2 PLB0_BEAR is locked	

12	PTE3	Master 3 PLB Timeout Error Status 0 No Master 3 timeout error 1 Master 3 timeout error	Master 3 is the PCIX bridge controller
13	R/W3	Master 3 Read/Write Status 0 Master 3 error operation was a write 1 Master 3 error operation was a read	
14	FLK3	Master 3 PLB0_BESR Field Lock 0 Master 3 PLB0_BESR field is unlocked 1 Master 3 PLB0_BESR field is locked	
15	ALK3	Master 3 PLB0_BEAR Address Lock 0 Master 3 PLB0_BEAR is unlocked 1 Master 3 PLB0_BEAR is locked	
16:19		Reserved	
20	PTE5	Master 5 PLB Timeout Error Status 0 No master 5 timeout error 1 Master 5 timeout error	Master 5 is the MAL controller
21	R/W5	Master 5 Read/Write Status 0 Master 5 error operation was a write 1 Master 5 error operation was a read	
22	FLK5	Master 5 PLB0_BESR Field Lock 0 Master 5 PLB0_BESR field is unlocked 1 Master 5 PLB0_BESR field is locked	
23	ALK5	Master 5 PLB0_BEAR Address Lock 0 Master 5 PLB0_BEAR is unlocked 1 Master 5 PLB0_BEAR is locked	
24	PTE6	Master 6 PLB Timeout Error Status 0 No master 6 timeout error 1 Master 6 timeout error	Master 6 is the DMA controller
25	R/W6	Master 6 Read/Write Status 0 Master 6 error operation was a write 1 Master 6 error operation was a read	
26	FLK6	Master 6 PLB0_BESR Field Lock 0 Master 6 PLB0_BESR field is unlocked 1 Master 6 PLB0_BESR field is locked	
27	ALK6	Master 6 PLB0_BEAR Address Lock 0 Master 6 PLB0_BEAR is unlocked 1 Master 6 PLB0_BEAR is locked	
28	PTE7	Master 7 PLB Timeout Error Status 0 No Master 7 timeout error 1 Master 7 timeout error	Master 7 is the OPB to PLB bridge controller
29	R/W7	Master 7 Read/Write Status 0 Master 7 error operation was a write 1 Master 7 error operation was a read	

30	FLK7	Master 7 PLB0_BESR Field Lock 0 Master 7 PLB0_BESR field is unlocked 1 Master 7 PLB0_BESR field is locked
31	ALK7	Master 7 PLB0_BEAR Address Lock 0 Master 7 PLB0_BEAR is unlocked 1 Master 7 PLB0_BEAR is locked

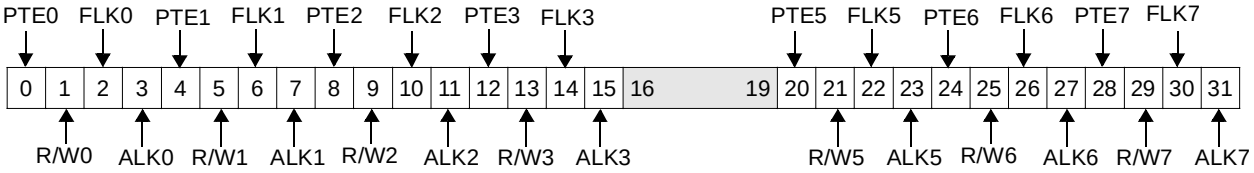


Figure 2-7. PLB Error Status Register (PLB0_BESR)

0	PTE0	Master 0 PLB Timeout Error Status 0 No master 0 timeout error 1 Master 0 timeout error	Master 0 is the read-only instruction cache unit (ICU)
1	R/W0	Master 0 Read/Write Status 0 Master 0 error operation was a write 1 Master 0 ICU error operation was a read	
2	FLK0	Master 0 PLB0_BESR Field Lock 0 Master 0 PLB0_BESR field is unlocked 1 Master 0 field is locked	
3	ALK0	Master 0 PLB0_BEAR Address Lock 0 Master 0 PLB0_BEAR is unlocked 1 Master 0 PLB0_BEAR is locked	
4	PTE1	Master 1 PLB Timeout Error Status 0 No master 1 timeout error 1 Master 1 timeout error	Master 1 is the read-only data cache unit (DCU)
5	R/W1	Master 1 Read/Write Status 0 Master 1 error operation was a write 1 Master 1 error operation was a read	
6	FLK1	Master 1PLB0_BESR Field Lock 0 Master 1 PLB0_BESR field is unlocked 1 Master 1 PLB0_BESR field is locked	
7	ALK1	Master 1 PLB0_BEAR Address Lock 0 Master 1 PLB0_BEAR is unlocked 1 Master 1 PLB0_BEAR is locked	
8	PTE2	Master 2 PLB Timeout Error Status 0 No master 2 timeout error 1 Master 2 timeout error	Master 2 is the write-only data cache unit (DCU)
9	R/W2	Master 2 Read/Write Status 0 Master 2 error operation was a write 1 Master 2 error operation was a read	
10	FLK2	Master 2 PLB0_BESR Field Lock 0 Master 2 PLB0_BESR field is unlocked 1 Master 2 PLB0_BESR field is locked	

11	ALK2	Master 2 PLB0_BEAR Address Lock 0 Master 2 PLB0_BEAR is unlocked 1 Master 2 PLB0_BEAR is locked	
12	PTE3	Master 3 PLB Timeout Error Status 0 No Master 3 timeout error 1 Master 3 timeout error	Master 3 is the PCIX bridge controller
13	R/W3	Master 3 Read/Write Status 0 Master 3 error operation was a write 1 Master 3 error operation was a read	
14	FLK3	Master 3 PLB0_BESR Field Lock 0 Master 3 PLB0_BESR field is unlocked 1 Master 3 PLB0_BESR field is locked	
15	ALK3	Master 3 PLB0_BEAR Address Lock 0 Master 3 PLB0_BEAR is unlocked 1 Master 3 PLB0_BEAR is locked	
16:19		Reserved	
20	PTE5	Master 5 PLB Timeout Error Status 0 No master 5 timeout error 1 Master 5 timeout error	Master 5 is the MAL controller
21	R/W5	Master 5 Read/Write Status 0 Master 5 error operation was a write 1 Master 5 error operation was a read	
22	FLK5	Master 5 PLB0_BESR Field Lock 0 Master 5 PLB0_BESR field is unlocked 1 Master 5 PLB0_BESR field is locked	
23	ALK5	Master 5 PLB0_BEAR Address Lock 0 Master 5 PLB0_BEAR is unlocked 1 Master 5 PLB0_BEAR is locked	
24	PTE6	Master 6 PLB Timeout Error Status 0 No master 6 timeout error 1 Master 6 timeout error	Master 6 is the DMA controller
25	R/W6	Master 6 Read/Write Status 0 Master 6 error operation was a write 1 Master 6 error operation was a read	
26	FLK6	Master 6 PLB0_BESR Field Lock 0 Master 6 PLB0_BESR field is unlocked 1 Master 6 PLB0_BESR field is locked	
27	ALK6	Master 6 PLB0_BEAR Address Lock 0 Master 6 PLB0_BEAR is unlocked 1 Master 6 PLB0_BEAR is locked	
28	PTE7	Master 7 PLB Timeout Error Status 0 No Master 7 timeout error 1 Master 7 timeout error	Master 7 is the OPB to PLB bridge controller
29	R/W7	Master 7 Read/Write Status 0 Master 7 error operation was a write 1 Master 7 error operation was a read	

30	FLK7	Master 7 PLB0_BESR Field Lock 0 Master 7 PLB0_BESR field is unlocked 1 Master 7 PLB0_BESR field is locked
31	ALK7	Master 7 PLB0_BEAR Address Lock 0 Master 7 PLB0_BEAR is unlocked 1 Master 7 PLB0_BEAR is locked

2.1.4.4 PLB Error Address Register (PLB0_BEARL)

The read-only PLB0_BEARL contains the lower 32 bits of the address of the access on which a bus time-out error occurred.

The PLB0_BEARL can be locked by the master. Once locked, the PLB0_BEARL cannot be updated, if a subsequent error occurs, until all PLB0_BESR[FLCK*n*] fields are cleared (*n* is the master ID). ~~Figure 2-8~~Figure 2-8 describes PLB0_BEARL register bits.

Figure 0-8. PLB Error Address Register (PLB0_BEARL)

0:31		Lower address of bus timeout error

Figure 2-8. PLB Error Address Register (PLB0_BEARL)

0:31		Lower address of bus timeout error

2.1.4.5 PLB Error Address Register High (PLB0_BEARH)

The read-only PLB0_BEARH contains the upper 32 bits of the address of the access on which a bus time-out error occurred.

The PLB0_BEARH can be locked by the master. Once locked, the PLB0_BEARH cannot be updated, if a subsequent error occurs, until all PLB0_BESR[FLCK*n*] fields are cleared (*n* is the master ID). Figure 2-9 describes PLB0_BEARH register bits.

Figure 0-9. PLB Error Address Register (PLB0_BEARH)

0:31		Upper address of bus timeout error

Figure 2-9. PLB Error Address Register (PLB0_BEARH)

0:31		Upper address of bus timeout error

2.1.5 PLB to OPB Bridge Registers

The PLB to OPB bridge registers are DCRs accessed using the **mfdcr** and **mtdcr** instructions.

Table 2-4 lists the PLB to OPB bridge registers.

Table 2-4. PLB ~~Arbiter~~ to OPB Bridge Registers

Mnemonic	Register Name	Address	Access	Page
POB0_BESR0	PLB to OPB Bridge Error Status Register 0 (Master IDs 0,1, 2, 3)	0x090	R/Clear	~~2-12~~102
POB0_BEARL	PLB to OPB Bridge Error Address Register Low	0x092	R/O	~~2-14~~104
POB0_BEARH	PLB to OPB Bridge Error Upper Address Register High	0x093	R/O	~~2-15~~105
POB0_BESR1	PLB to OPB Bridge Error Status Register (Master IDs 4, 5)	0x094	R/Clear	~~2-15~~105
POB0_CONFG	PLB to OPB Bridge Error Configuration Register	0x096	R/Clear	~~2-17~~107
POB0_LATENCY	PLB to OPB Bridge Burst Latency Timer	0x098	R/Clear	~~2-17~~107
POB0_REVID	PLB to OPB Bridge Revision ID Register	0x09A	R/O	~~2-18~~108

2.1.5.1 PLB to OPB Bridge Error Status Register 0 (POB0_BESR0)

The PLB to OPB bridge writes error information into the appropriate POB0_BESR*m* register. (For master IDs 0, 1, 2, and 3, *m* = 0; for master IDs 4 and 5, *m* = 1.)

POB0_BESR*m* fields can be locked using the POB0_BESR*m*[FLK*n*] and POB0_BESR*m*[ALK*n*] fields (*n* is the master ID). Once locked, the POB0_BESR*m* fields associated with a master cannot be overwritten if a subsequent error occurs until the locking fields are cleared. To clear a lock, write 1 to the POB0_BESR*m*[FLK*n*] and POB0_BESR*m*[ALK*n*] fields that are set. Writing 0 to a lock field does not affect the field

~~Figure 2-10~~Figure 2-10 shows bit definitions for POB0_BESR0

–

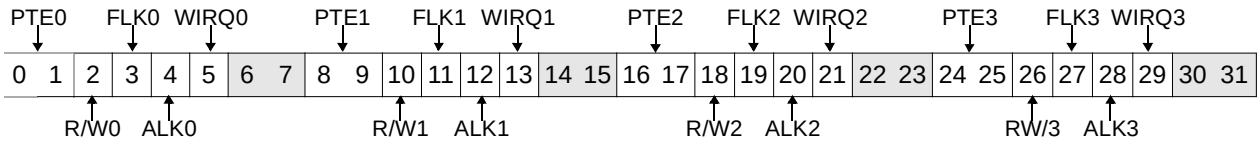


Figure 0-10. PLB to OPB Bridge Error Status Register 0 (POB0_BESR0)

0:1	PTE0	PLB Timeout Error Status Master 0 00 No master 0 error occurred 01 Master 0 timeout error occurred 10 Master 0 slave error occurred 11 Reserved	Master 0 is the read-only instruction cache unit (ICU)

2	R/W0	Read Write Status Master 0 0 Master 0 error operation is a read 1 Master 0 error operation is a write	
3	FLK0	POB0_BESR0 Field Lock Master 0 0 Master 0 POB0_BESR0 field is unlocked 1 Master 0 POB0_BESR0 field is locked	
4	ALK0	POB0_BEAR Address Lock Master 0 0 Master 0 POB0_BEAR address is unlocked 1 Master 0 POB0_BEAR address is locked	
5	WIRQ0	Write Error Interrupt Master 0 0 No write error detected - master 0 interrupt request is inactive 1 Write error detected - master 0 interrupt request is active	
6:7		Reserved	
8:9	PTE1	PLB Timeout Error Status Master 1 00 No master 1 error occurred 01 Master 1 timeout error occurred 10 Master 1 slave error occurred 11 Reserved	Master 1 is the read-only data cache unit (DCU)
10	R/W1	Read/Write Status Master 1 0 Master 1 error operation is a read 1 Master 1 error operation is a write	
11	FLK1	POB0_BESR0 Field Lock Master 1 0 Master 1 POB0_BESR0 field is unlocked 1 Master 1 POB0_BESR0 field is locked	
12	ALK1	POB0_BEAR Address Lock Master 1 0 Master 1 POB0_BEAR address is unlocked 1 Master 1 POB0_BEAR address is locked	
13	WIRQ1	Write Error Interrupt Master 1 0 No write error detected - master 1 interrupt request is inactive 1 Write error detected - master 1 interrupt request is active	
14:15		Reserved	
16:17	PTE2	PLB Timeout Error Status Master 2 00 No master 2 error occurred 01 Master 2 timeout error occurred 10 Master 2 slave error occurred 11 Reserved	Master 2 is the write-only data cache unit (DCU)
18	R/W2	Read/Write Status Master 2 0 Master 2 error operation is a read 1 Master 2 error operation is a write	
19	FLK2	POB0_BESR0 Field Lock Master 2 0 Master 2 POB0_BESR0 field is unlocked 1 Master 2 POB0_BESR0 field is locked	

20	ALK2	POB0_BEAR Address Lock Master 2 0 Master 2 POB0_BEAR address is unlocked 1 Master 2 POB0_BEAR address is locked	
21	WIRQ2	Write Error Interrupt Master 2 0 No write error detected - master 2 interrupt request is inactive 1 Write error detected - master 2 interrupt request is active	
22:23		Reserved	
24:25	PTE3	PLB Timeout Error Status Master 3 00 No master 3 error occurred 01 Master 3 timeout error occurred 10 Master 3 slave error occurred 11 Reserved	Master 3 is the PCIX bridge controller
25	R/W3	Read/Write Status Master 3 0 Master 3 error operation is a read 1 Master 3 error operation is a write	
26	FLK3	POB0_BESR0 Field Lock Master 3 0 Master 3 POB0_BESR0 field is unlocked 1 Master 3 POB0_BESR0 field is locked	
27	ALK3	POB0_BEAR Address Lock Master 3 0 Master 3 POB0_BEAR address is unlocked 1 Master 3 POB0_BEAR address is locked	
29	WIRQ3	Write Error Interrupt Master 3 0 No write error detected - master 3 interrupt request is inactive 1 Write error detected - master 3 interrupt request is active	
30:31		Reserved	

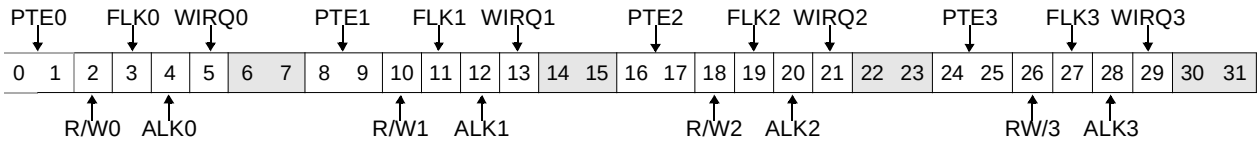


Figure 2-10. PLB to OPB Bridge Error Status Register 0 (POB0_BESR0)

0:1	PTE0	PLB Timeout Error Status Master 0 00 No master 0 error occurred 01 Master 0 timeout error occurred 10 Master 0 slave error occurred 11 Reserved	Master 0 is the read-only instruction cache unit (ICU)
2	R/W0	Read Write Status Master 0 0 Master 0 error operation is a write 1 Master 0 error operation is a read	

3	FLK0	POB0_BESR0 Field Lock Master 0 0 Master 0 POB0_BESR0 field is unlocked 1 Master 0 POB0_BESR0 field is locked	
4	ALK0	POB0_BEAR Address Lock Master 0 0 Master 0 POB0_BEAR address is unlocked 1 Master 0 POB0_BEAR address is locked	
5	WIRQ0	Write Error Interrupt Master 0 0 No write error detected - master 0 interrupt request is inactive 1 Write error detected - master 0 interrupt request is active	
6:7		Reserved	
8:9	PTE1	PLB Timeout Error Status Master 1 00 No master 1 error occurred 01 Master 1 timeout error occurred 10 Master 1 slave error occurred 11 Reserved	Master 1 is the read-only data cache unit (DCU)
10	R/W1	Read/Write Status Master 1 0 Master 1 error operation is a write 1 Master 1 error operation is a read	
11	FLK1	POB0_BESR0 Field Lock Master 1 0 Master 1 POB0_BESR0 field is unlocked 1 Master 1 POB0_BESR0 field is locked	
12	ALK1	POB0_BEAR Address Lock Master 1 0 Master 1 POB0_BEAR address is unlocked 1 Master 1 POB0_BEAR address is locked	
13	WIRQ1	Write Error Interrupt Master 1 0 No write error detected - master 1 interrupt request is inactive 1 Write error detected - master 1 interrupt request is active	
14:15		Reserved	
16:17	PTE2	PLB Timeout Error Status Master 2 00 No master 2 error occurred 01 Master 2 timeout error occurred 10 Master 2 slave error occurred 11 Reserved	Master 2 is the write-only data cache unit (DCU)
18	R/W2	Read/Write Status Master 2 0 Master 2 error operation is a write 1 Master 2 error operation is a read	
19	FLK2	POB0_BESR0 Field Lock Master 2 0 Master 2 POB0_BESR0 field is unlocked 1 Master 2 POB0_BESR0 field is locked	
20	ALK2	POB0_BEAR Address Lock Master 2 0 Master 2 POB0_BEAR address is unlocked 1 Master 2 POB0_BEAR address is locked	
21	WIRQ2	Write Error Interrupt Master 2 0 No write error detected - master 2 interrupt request is inactive 1 Write error detected - master 2 interrupt request is active	
22:23		Reserved	

24:25	PTE3	PLB Timeout Error Status Master 3 Master 3 is the PCIX bridge controller 00 No master 3 error occurred 01 Master 3 timeout error occurred 10 Master 3 slave error occurred 11 Reserved
26	R/W3	Read/Write Status Master 3 0 Master 3 error operation is a write 1 Master 3 error operation is a read
27	FLK3	POB0_BESR0 Field Lock Master 3 0 Master 3 POB0_BESR0 field is unlocked 1 Master 3 POB0_BESR0 field is locked
28	ALK3	POB0_BEAR Address Lock Master 3 0 Master 3 POB0_BEAR address is unlocked 1 Master 3 POB0_BEAR address is locked
29	WIRQ3	Write Error Interrupt Master 3 0 No write error detected - master 3 interrupt request is inactive 1 Write error detected - master 3 interrupt request is active
30:31		Reserved

2.1.5.2 PLB to OPB Bridge Error Address Register Low (POB0_BEARL)

The read-only POB0_BEARL reports the lower 32 bits of the address of a PLB to OPB transfer that results in an error. The PLB to OPB bridge writes the error address in the POB0_BEARL, unless the associated POB0_BESR*m*[ALCK*n*] field is set (*m* is either 0 or 1, depending on the master ID specified by *n*). Once locked, the PLB to OPB bridge cannot write POB0_BEARL until all POB0_BESR*m*[ALCK*n*] fields that are set are cleared. Figure 2-11 describes POB0_BEARL register bits. Figure 2-11 shows bit definitions for POB0_BEARL

–

0	31

Figure 0-11. PLB to OPB Bridge Error Address Register Low (POB0_BEARL)

0:31	0x0B2	Lower address of bus error

Figure 2-11. PLB to OPB Bridge Error Address Register Low (POB0_BEARL)

0:31	0x0B2	Lower address of bus error

2.1.5.3 PLB to OPB Bridge Error Address Register High (POB0_BEARH)

The read-only POB0_BEARH reports the upper four address bits of a PLB to OPB transfer that results in an error. The PLB to OPB bridge writes the error address in the POB0_BEARH, unless the associated POB0_BESR*m*[ALCK*n*] field is set (*m* is either 0 or 1, depending on the master ID specified by *n*). Once locked, the PLB to OPB bridge cannot write POB0_BEARH until all POB0_BESR*m*[ALCK*n*] fields that are set are cleared. ~~Figure 2-12~~Figure 2-12 describes POB0_BEARH register bits.

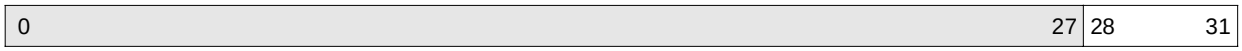


Figure 0-12. PLB to OPB Bridge Error Address Register High (POB0_BEARH)

0:27		Reserved
28:31	10'h0B3	Upper address of bus error

Figure 2-12. PLB to OPB Bridge Error Address Register High (POB0_BEARH)

0:27		Reserved
28:31	10'h0B3	Upper address of bus error

2.1.5.4 PLB to OPB Bridge Error Status Register 1(POB0_BESR1)

~~Figure 2-13~~Figure 2-13 shows bit definitions for POB0_BESR1.

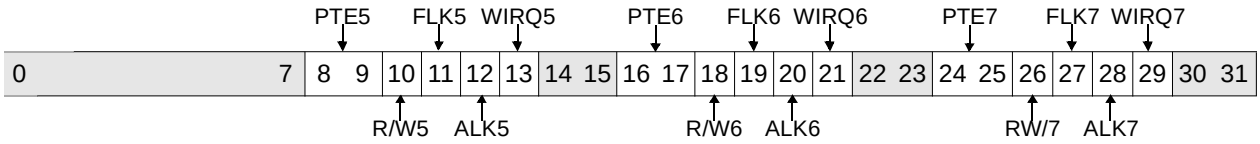


Figure 0-13. PLB to OPB Bridge Error Status Register 1 (POB0_BESR1)

0:7		Reserved	
8:9	PTE5	PLB Timeout Error Status Master 5 00 No master 5 error occurred 01 Master 5 timeout error occurred 10 Master 5 slave error occurred 11 Reserved	Master 5 is the MAL controller
10	R/W5	Read/Write Status Master 5 0 Master 5 error operation is a read 1 Master 5 error operation is a write	
11	FLK5	POB0_BESR0 Field Lock Master 5 0 Master 5 POB0_BESR0 field is unlocked 1 Master 5 POB0_BESR0 field is locked	
12	ALK5	POB0_BEAR Address Lock Master 5 0 Master 5 POB0_BEAR address is unlocked 1 Master 5 POB0_BEAR address is locked	
13	WIRQ5	Write Error Interrupt Master 5 0 No write error detected - master 5 interrupt request is inactive 1 Write error detected - master 5 interrupt request is active	
14:15		Reserved	
16:17	PTE6	PLB Timeout Error Status Master 6 00 No master 6 error occurred 01 Master 6 timeout error occurred 10 Master 6 slave error occurred 11 Reserved	Master 6 is the DMA controller
18	R/W6	Read/Write Status Master 6 0 Master 6 error operation is a read 1 Master 6 error operation is a write	
19	FLK6	POB0_BESR0 Field Lock Master 6 0 Master 6 POB0_BESR0 field is unlocked 1 Master 6 POB0_BESR0 field is locked	

20	ALK6	POB0_BEAR Address Lock Master 6 0 Master 6 POB0_BEAR address is unlocked 1 Master 6 POB0_BEAR address is locked	
21	WIRQ6	Write Error Interrupt Master 6 0 No write error detected - master 6 interrupt request is inactive 1 Write error detected - master 6 interrupt request is active	
22:23		Reserved	
24:25	PTE7	PLB Timeout Error Status Master 7 00 No master 7 error occurred 01 Master 7 timeout error occurred 10 Master 7 slave error occurred 11 Reserved	Master 7 is the OPB to PLB bridge controller
25	R/W7	Read/Write Status Master 7 0 Master 7 error operation is a read 1 Master 7 error operation is a write	
26	FLK7	POB0_BESR0 Field Lock Master 7 0 Master 7 POB0_BESR0 field is unlocked 1 Master 7 POB0_BESR0 field is locked	
27	ALK7	POB0_BEAR Address Lock Master 7 0 Master 7 POB0_BEAR address is unlocked 1 Master 7 POB0_BEAR address is locked	
29	WIRQ7	Write Error Interrupt Master 7 0 No write error detected - master 7 interrupt request is inactive 1 Write error detected - master 7 interrupt request is active	
30:31		Reserved	

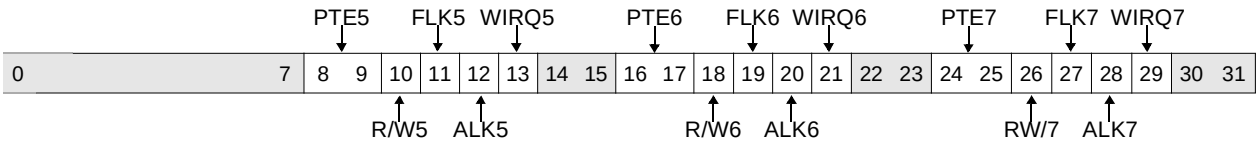


Figure 2-13. PLB to OPB Bridge Error Status Register 1 (POB0_BESR1)

0:7		Reserved	
8:9	PTE5	PLB Timeout Error Status Master 5 00 No master 5 error occurred 01 Master 5 timeout error occurred 10 Master 5 slave error occurred 11 Reserved	Master 5 is the MAL controller

10	R/W5	Read/Write Status Master 5 0 Master 5 error operation is a write 1 Master 5 error operation is a read
11	FLK5	POB0_BESR1 Field Lock Master 5 0 Master 5 POB0_BESR1 field is unlocked 1 Master 5 POB0_BESR1 field is locked
12	ALK5	POB0_BEAR Address Lock Master 5 0 Master 5 POB0_BEAR address is unlocked 1 Master 5 POB0_BEAR address is locked
13	WIRQ5	Write Error Interrupt Master 5 0 No write error detected - master 5 interrupt request is inactive 1 Write error detected - master 5 interrupt request is active
14:15		Reserved
16:17	PTE6	PLB Timeout Error Status Master 6 Master 6 is the DMA controller 00 No master 6 error occurred 01 Master 6 timeout error occurred 10 Master 6 slave error occurred 11 Reserved
18	R/W6	Read/Write Status Master 6 0 Master 6 error operation is a write 1 Master 6 error operation is a read
19	FLK6	POB0_BESR1 Field Lock Master 6 0 Master 6 POB0_BESR1 field is unlocked 1 Master 6 POB0_BESR1 field is locked
20	ALK6	POB0_BEAR Address Lock Master 6 0 Master 6 POB0_BEAR address is unlocked 1 Master 6 POB0_BEAR address is locked
21	WIRQ6	Write Error Interrupt Master 6 0 No write error detected - master 6 interrupt request is inactive 1 Write error detected - master 6 interrupt request is active
22:23		Reserved
24:25	PTE7	PLB Timeout Error Status Master 7 Master 7 is the OPB to PLB bridge controller 00 No master 7 error occurred 01 Master 7 timeout error occurred 10 Master 7 slave error occurred 11 Reserved
26	R/W7	Read/Write Status Master 7 0 Master 7 error operation is a write 1 Master 7 error operation is a read
27	FLK7	POB0_BESR1 Field Lock Master 7 0 Master 7 POB0_BESR1 field is unlocked 1 Master 7 POB0_BESR1 field is locked
28	ALK7	POB0_BEAR Address Lock Master 7 0 Master 7 POB0_BEAR address is unlocked 1 Master 7 POB0_BEAR address is locked

29	WIRQ7	Write Error Interrupt Master 7 0 No write error detected - master 7 interrupt request is inactive 1 Write error detected - master 7 interrupt request is active
30:31		Reserved

2.1.5.5 PLB to OPB Bridge Configuration Register (POB0_CONFG)

Figure 2-14 shows bit definitions for POB0_CONFG.Figure 2-14—

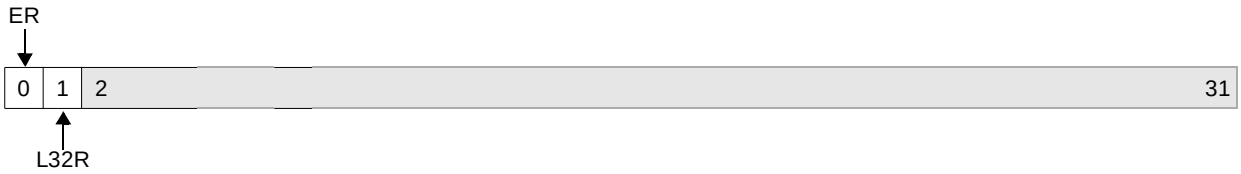


Figure 0-14. PLB to OPB Bridge Configuration Register (POB0_CONFG)

0	ER	Enable rearbitration 0 PLB to OPB bridge rearbitration is enabled 1 PLB to OPB bridge rearbitration is disabled
1	L32R	Line 32 bit read 0 PLB to OPB bridge reads line operations at requested size of master 1 PLB to OPB bridge reads line operations at 32 bit slave size regardless of requesting master size
2:31		Reserved

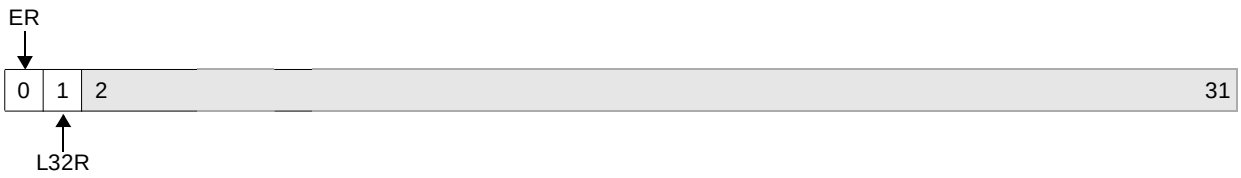


Figure 2-14. PLB to OPB Bridge Configuration Register (POB0_CONFG)

0	ER	Enable rearbitration 0 PLB to OPB bridge rearbitration is enabled 1 PLB to OPB bridge rearbitration is disabled
1	L32R	Line 32 bit read 0 PLB to OPB bridge reads line operations at requested size of master 1 PLB to OPB bridge reads line operations at 32 bit slave size regardless of requesting master size
2:31		Reserved

2.1.5.6 PLB to OPB Bridge Burst Latency Timer (POB0_LATENCY)

~~Figure 2-15~~Figure 2-15 shows bit definitions for POB0_LATENCY

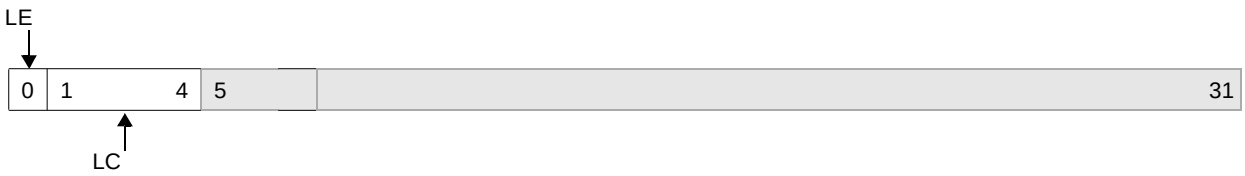


Figure 0-15. PLB to OPB Bridge Burst Latency Timer Register (POB0_LATENCY)

0	LE	Latency Enable 0 Latency timer disabled 1 Latency timer enabled
1:4	LC	Latency Count 0000 - Minimum count value. PLB to OPB bridge will count 16 OPB_xferAcks during a read or write burst sequence and if OPB_pendReq is sampled high at the end of the count - then the burst sequence is terminated. 1111 - Maximum count value. PLB to OPB bridge will count 16 blocks of 16 OPB_xferAcks, (or 256 xferAcks) during a read or write burst sequence, and if OPB_pendReq is sampled high at the end of count - then the burst sequence is terminated.
5:31		Reserved

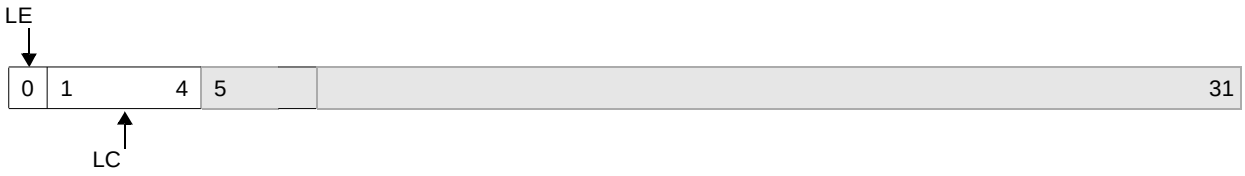


Figure 2-15. PLB to OPB Bridge Burst Latency Timer Register (POB0_LATENCY)

0	LE	Latency Enable 0 Latency timer disabled 1 Latency timer enabled
1:4	LC	Latency Count 0000 - Minimum count value. PLB to OPB bridge will count 16 OPB_xferAcks during a read or write burst sequence and if OPB_pendReq is sampled high at the end of the count - then the burst sequence is terminated. 1111 - Maximum count value. PLB to OPB bridge will count 16 blocks of 16 OPB_xferAcks, (or 256 xferAcks) during a read or write burst sequence, and if OPB_pendReq is sampled high at the end of count - then the burst sequence is terminated.
5:31		Reserved

2.1.5.7 PLB to OPB Bridge Revision ID (POB0_REVID)

Figure 2-16 shows bit definitions for POB0_REVID.Figure 2-16—

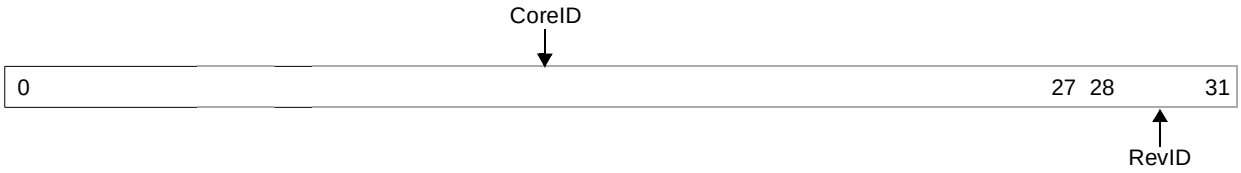


Figure 0-16. PLB to OPB Bridge Revision ID Register (POB0_REVID)

0:27	CoreID	Core ID value indicates what technology and what type of core this is. The core ID value for this core is C27E502.
28:31	RevId	This value indicates what revision level this core is. The revision ID for this core is 1.

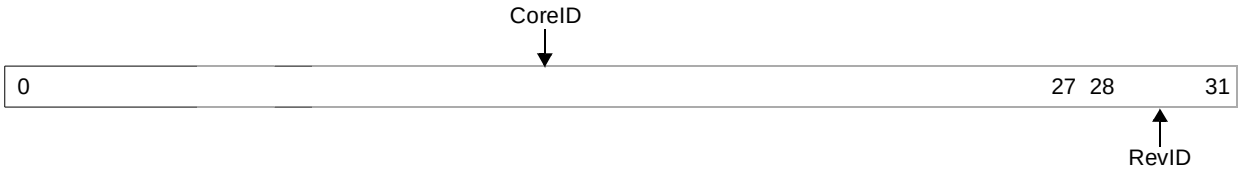


Figure 2-16. PLB to OPB Bridge Revision ID Register (POB0_REVID)

0:27	CoreID	Core ID	Corresponds to the technology.
28:31	RevId	Revision ID	Corresponds to the revision level.

2.2 On-Chip Peripheral Bus

The OPB is used to attach peripherals that do not require the bandwidth of the PLB. The OPB does not connect directly to the PPC440GP processor core, which accesses peripherals attached to the OPB through a PLB-to-OPB bridge.

2.2.1 OPB Features

The on-chip peripheral bus features:

- A 36-bit address bus and a 32-bit data bus
- Dynamic bus sizing; byte, halfword, and fullword transfers
- Byte and halfword duplication for byte and halfword transfers
- Single-cycle transfer of data between OPB bus master and OPB slaves
- Sequential address (burst) protocol support
- Devices on the OPB may be memory mapped, act as DMA peripherals, or support both transfer methods

- A 16-cycle fixed bus timeout provided by the OPB arbiter
- Bus parking for reduced latency
- Bus arbitration overlapped with last cycle of bus transfers

2.2.2 OPB Master Assignments

Table 2-5 lists the OPB masters. (The numbering reflects that the PPC440GP implements three of four masters that are supported by the OPB.

Table 2-5. PPC440GP OPB Master Assignments

OPB Agents	Description
PLB to OPB Bridge Controller	BGO (master 0)
External Bus Master Interface	EBMI (master 1)
Direct Memory Access Controller	DMA (master 2)

2.2.3 OPB Arbiter Registers

The OPB arbiter contains the MMIO registers summarized in Table 2-6.

Table 2-6. PLB Arbiter Registers

Mnemonic	Register Name	Address	Access	Page
OPBA0_PR	OPB Arbiter Priority Register	0x140000600	R/W	~~2-19~~109
OPBA0_CR	OPB Arbiter Control Register	0x1400000601	R/W	~~2-21~~111

2.2.3.1 OPB Arbiter Priority Register (OPBA0_PR)

The OPBA0_PR assigns priorities to the OPB master IDs. Figure 2-17 describes OPBA0_PR register bits.

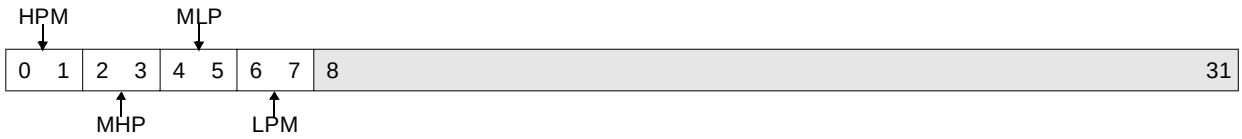


Figure 0-17. OPB Arbiter Priority Register (OPBA0_PR)

0:1	HPM	High Priority Master ID 00 Master ID 0of OPB master device connected to M0_request and OPB_M0Grant 01 Master ID of OPB master device connected to M1_request and OPB_M1GrantReserved 10 Master ID of OPB master device connected to M2_request and OPB_M2GrantReserved 11 Master ID of OPB master device connected to M3_request and OPB_M3GrantReserved	At reset, this priority is assigned to PLB to OPB bridge.
2:3	MHP	Medium High Priority master ID 00 Master ID of OPB master device connected to M0_request and OPB_M0Grant 01 Master ID of OPB master device connected to M1_request and OPB_M1Grant 10 Master ID of OPB master device connected to M2_request and OPB_M2Grant 11 Master ID of OPB master device connected to M3_request and OPB_M3Grant	At reset, this priority is assigned to the external bus master interface.
4:5	MLP	LowMedium Low Priority master ID 00 Master ID of OPB master device connected to M0_request and OPB_M0GrantReserved 01 Master ID of OPB master device connected to M1_request and OPB_M1GrantReserved 10 Master ID 2of OPB master device connected to M2_request and OPB_M2Grant 11 Master ID of OPB master device connected to M3_request and OPB_M3GrantReserved	At reset, this priority is assigned to DMA

6:7	LPM	Low Priority Master ID 00 Master ID of OPB master device connected to M0_request and OPB_M0Grant 01 Master ID of OPB master device connected to M1_request and OPB_M1Grant 10 Master ID of OPB master device connected to M2_request and OPB_M2Grant 11 Master ID of OPB master device connected to M3_request and OPB_M3Grant
8:31		Reserved

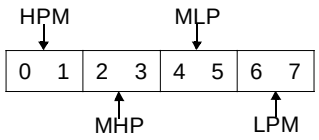


Figure 2-17. OPB Arbiter Priority Register (OPBA0_PR)

0:1	HPM	High Priority Master ID 00 Master ID 0of OPB master device connected to M0_request and OPB_M0Grant 01 Master ID of OPB master device connected to M1_request and OPB_M1GrantReserved 10 Master ID of OPB master device connected to M2_request and OPB_M2GrantReserved 11 Master ID of OPB master device connected to M3_request and OPB_M3GrantReserved	At reset, this priority is assigned to PLB to OPB bridge.
2:3	MHP	Medium High Priority master ID 00 Master ID of OPB master device connected to M0_request and OPB_M0Grant 01 Master ID of OPB master device connected to M1_request and OPB_M1Grant 10 Master ID of OPB master device connected to M2_request and OPB_M2Grant 11 Master ID of OPB master device connected to M3_request and OPB_M3Grant	At reset, this priority is assigned to the external bus master interface.

4:5	MLP	Medium Low Priority master ID 00 Master ID of OPB master device connected to M0_request and OPB_M0GrantReserved 01 Master ID of OPB master device connected to M1_request and OPB_M1GrantReserved 10 Master ID 2of OPB master device connected to M2_request and OPB_M2Grant 11 Master ID of OPB master device connected to M3_request and OPB_M3GrantReserved	At reset, this priority is assigned to DMA
6:7	LPM	Low Priority Master ID 00 Master ID of OPB master device connected to M0_request and OPB_M0Grant 01 Master ID of OPB master device connected to M1_request and OPB_M1Grant 10 Master ID of OPB master device connected to M2_request and OPB_M2Grant 11 Master ID of OPB master device connected to M3_request and OPB_M3Grant	

2.2.3.2 OPB Arbiter Control Register (OPBA0_CR)

The OPBA0_CR fields controls updating of the OPBA0_PR (described in "OPB Arbiter Priority Register (OPBA0_PR)" on page -109). Figure 2-18 describes OPBA0_CR register bits.

—

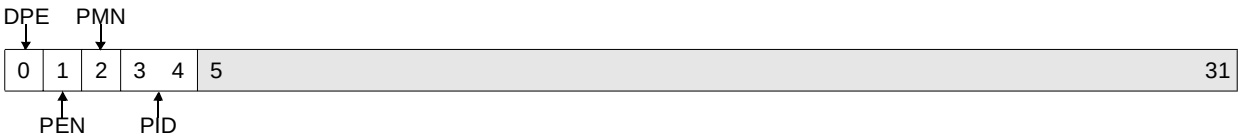


Figure 0-18. OPB Arbiter Control Register (OPBA0_CR)

0	DPE	Dynamic Priority Enable 0 Dynamic priority disabled 1 Dynamic priority enabled
1	PEN	Park Enable 0 Park disabled 1 Park enabled

2	PMN	Park on Master Not Last 0 Park on master last 1 Park on master not last	
3:4	PID	Parked Master ID 00 Master ID 0of OPB master device connected to M0_request and OPB_M0Grant 01 Master ID of OPB master device connected to M1_request and OPB_M1GrantReserved 10 Master ID 2of OPB master device connected to M2_request and OPB_M2Grant 11 Master ID of OPB master device connected to M3_request and OPB_M3GrantReserved	Master 0 is the PLB to OPB bridge controller; master 1 is the external bus master interface, and master 2 is the DMA controller. Master 3 is unused.
5:31		Reserved	

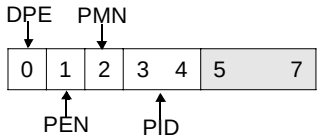


Figure 2-18. OPB Arbiter Control Register (OPBA0_CR)

0	DPE	Dynamic Priority Enable 0 Dynamic priority disabled 1 Dynamic priority enabled	
1	PEN	Park Enable 0 Park disabled 1 Park enabled	
2	PMN	Park on Master Not Last 0 Park on master last 1 Park on master not last	
3:4	PID	Parked Master ID 00 Master ID 0of OPB master device connected to M0_request and OPB_M0Grant 01 Master ID of OPB master device connected to M1_request and OPB_M1GrantReserved 10 Master ID 2of OPB master device connected to M2_request and OPB_M2Grant 11 Master ID of OPB master device connected to M3_request and OPB_M3GrantReserved	Master 0 is the PLB to OPB bridge controller; master 1 is the external bus master interface, and master 2 is the DMA controller. Master 3 is unused.
5:7		Reserved	

2.2.4 OPB to PLB Bridge Registers

OPB to PLB bridge provides registers which are summarized in Table 2-7

Table 2-7. OPB to PLB Bridge Registers

Mnemonic	Register Name	Address	Access	Page
OPB0_BCTRL	OPB to PLB Bridge Control Register	0x0A8	R/W	~~2-22~~112
OPB0_BSTAT	OPB to PLB Bridge Status Register	0x0A9	R/O	~~2-23~~112
OPB0_BEARL	OPB to PLB Bridge Error Address Register Low	0x0AA	R/O	~~2-24~~113
OPB0_BEARH	OPB to PLB Bridge Error Address Register High	0x0AB	R/O	~~2-25~~114
OPB0_REVID	OPB to PLB Bridge Revision ID Register	0x0AC	R/O	~~2-25~~114

2.2.4.1 OPB to PLB Bridge Control Register (OPB0_BCTRL)

OPB to PLB bridge control register (OPB0_CTRL) contains the control bits for the OPB to PLB bridge core. It is read from and written to via the DCR bus. ~~Figure 2-19~~Figure 2-19 shows OPB0_CTRL bit definitions.

Figure 0-19. OPB to PLB Bridge Control Register (OPB0_CTRL)

0:1	PRI	PLB Priority Bits. Determines the priority of PLB requests. Reset to value on PGM_priority inputs	These bits determine the priority of PLB requests. They are directly connected to the PLB_priority(0:1) bits during a PLB request. During reset, these bits are set to the value present on the PGM_priority input (set by the system designer).
2:31		Reserved	

Figure 2-19. OPB to PLB Bridge Control Register (OPB0_BCTRL)

0:1	PRI	PLB Priority Bits. Determines the priority of PLB requests. Reset to value on PGM_priority inputs	These bits determine the priority of PLB requests. They are directly connected to the PLB_priority(0:1) bits during a PLB request. During reset, these bits are set to the value present on the PGM_priority input (set by the system designer).
2:31		Reserved	

2.2.4.2 OPB to PLB Bridge Status Register (OPB0_BSTAT)

The OPB to PLB bridge status register (OPB0_BSTAT) contains the error bit for the OPB to PLB bridge core. To clear the status register, a "1" must be loaded into those register bits that are to be cleared. Writing a "0" to any bit in the status register will not affect the state of the bit. ~~Figure 2-20~~Figure 2-20 shows OPB0_BSTAT bit definitions

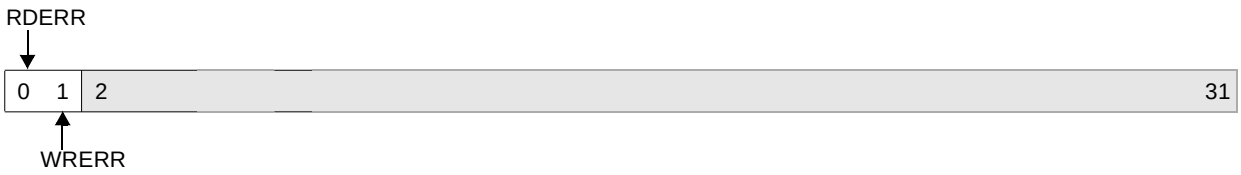


Figure 0-20. OPB to PLB Bridge Status Register (OPB0_STAT)

0	RDERR	Read Error Bit Reset to 0	This error bit is set whenever a PLB slave device reports a read error to the bridge. The error is additionally reported to the OPB master through BGI_errAck. The OPB to PLB bridge status register read error bit should be polled to detect errors, and the SEAR and SEGR of the PLB slave consulted for details of the error condition.
1	WRERR	Write Error Bit Reset to 0	This error bit is set whenever a PLB slave device reports a write error to the bridge. For posted single writes and errors occurring after the deassertion of OPB_seqAddr on burst writes, no error is reported to the OPB master through BGI_errAck. An error is reported to the OPB master through BGI_errAck for a previous write error if OPB_seqAddr is still asserted (but note that the error does not correspond to the transfer during which it is reported).
2:31		Reserved	

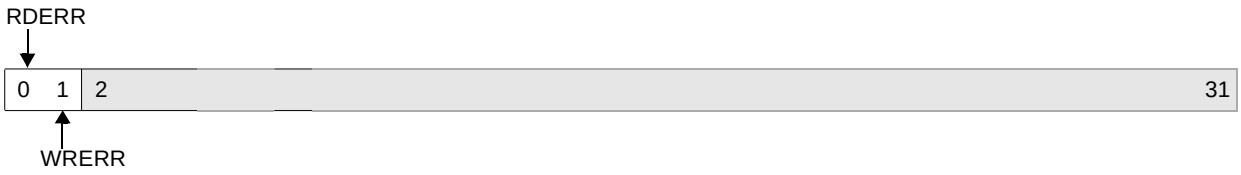


Figure 2-20. OPB to PLB Bridge Status Register (OPB0_BSTAT)

0	RDERR	Read Error Bit Reset to 0	This error bit is set whenever a PLB slave device reports a read error to the bridge. The error is additionally reported to the OPB master through BGI_errAck. The OPB to PLB bridge status register read error bit should be polled to detect errors, and the SEAR and SEGR of the PLB slave consulted for details of the error condition.
1	WRERR	Write Error Bit Reset to 0	This error bit is set whenever a PLB slave device reports a write error to the bridge. For posted single writes and errors occurring after the deassertion of OPB_seqAddr on burst writes, no error is reported to the OPB master through BGI_errAck. An error is reported to the OPB master through BGI_errAck for a previous write error if OPB_seqAddr is still asserted (but note that the error does not correspond to the transfer during which it is reported).
2:31		Reserved	

2.2.4.3 OPB to PLB Bridge Error Address Register Low (OPB0_BEARL)

The OPB to PLB bridge error address register low OPB0_BEARL is a read-only register. As part of its error reporting, the PLB to OPB bridge will load the OPB0_BEARL with the error address if the ALCK, (address lock bit), is not already set. ~~Figure 2-21~~Figure 2-21 describes OPB0_BEARL register bits.

Figure 0-21. OPB to PLB Bridge Error Address Register Low (OPB0_BEARL)

0:31	0x0B2	Lower address of bus error

Figure 2-21. OPB to PLB Bridge Error Address Register Low (OPB0_BEARL)

0:31	0x0B2	Lower address of bus error

2.2.4.4 OPB to PLB Bridge Error Address Register High (OPB0_BEARH)

The OPB to PLB bridge error address register high OPB0_BEARH is a read-only register. As part of its error reporting, the PLB to OPB bridge will load the GEARU with the error address if the ALCK, (address lock bit), is not already set. ~~Figure 2-22~~Figure 2-22 describes OPB0_BEARH register bits.

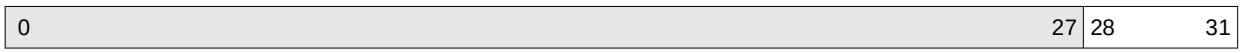


Figure 0-22. OPB to PLB Bridge Error Address Register High (OPB0_BEARH)

0:27		Reserved
28:31	10'h0B3	Upper address of bus error

Figure 2-22. OPB to PLB Bridge Error Address Register High (OPB0_BEARH)

0:27		Reserved
28:31	10'h0B3	Upper address of bus error

2.2.4.5 OPB to PLB Bridge Revision ID Register (OPB0_REVID)

The OPB to PLB bridge revision ID register (OPB0_REVID) is a read-only register. ~~Figure 2-23~~Figure 2-23 describes OPB0_REVID register bits.

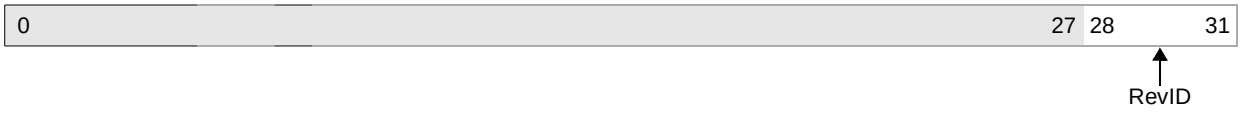


Figure 0-23. OPB to PLB Bridge Revision ID Register (OPB0_REVID)

0:27		Reserved
28:31	0x0B2	This value indicates what revision level this core is. The revision ID for this core is 1.

Figure 2-23. OPB to PLB Bridge Revision ID Register (OPB0_REVID)

0:27		Reserved
28:31	0x0B2	This value indicates what revision level this core is. The revision ID for this core is 1.

2.3 Device Control Register (DCR) Interface

The DCR interface provides a mechanism for the PPC440GP to setup other on-chip facilities. For example, programmable resources in an external bus interface unit may be configured for usage with various memory devices according to their transfer characteristics and address assignments. DCR's are accessed through the use of the PowerPC **mfdcr** and **mtdcr** instructions.

The interface is interlocked with control signals such that it may be connected to peripheral units that may be clocked at different frequencies from the processor core. The design allows for future expansion of the non-core facilities without changing the I/O on either the Core or the ASIC peripherals.

The DCR interface also allows the RISC processor core to communicate with peripheral devices without using the PLB interface, thereby avoiding the impact to the primary system bus bandwidth, and without additional segmentation of the useable address map.

3. PLB Performance Monitor

The PLB performance monitor (PPM) provides hardware for counting certain events associated with PLB bus transactions. The PPM consists of a set of counters whose contents may be read by software and used to analyze and enhance PLB performance, or used as a software debug mechanism.

The PPM can perform both event-occurrence counting and event-duration counting. Occurrence counting is accomplished via a set of counters that increment their value once for each occurrence of a selected event, until a predefined timer has expired. The timer value is programmable via a cycle register and is capable of generating an external interrupt upon expiration.

Duration counting is accomplished via separate registers that increment on every clock cycle that a pre-selected event is active.

Each counter can be individually enabled, and is capable of generating an external interrupt once that counter has reached its maximum value. Event selections and counter controls are performed via the control, status, and individual counter selection registers.

3.1 PPM Features

The PLB performance monitor (PPM) features:

- Selection Registers to simultaneously track events from up to four selectable PLB masters, up to four PLB slaves and up to four external generic events or pipeline-stage events.
- Four 24-bit master-event counters for counting occurrences of selectable PLB master events. Each master-event counter increments independently once for each PLB clock cycle that the selected event is asserted.
- Four 24-bit slave-event counters for counting occurrences of selectable PLB slave events. Each slave-event counter increments independently once for each PLB clock cycle that the selected event is asserted.
- Four 24-bit generic-event counters for counting either pipeline-depth-level occurrences or generic events external to the PPM core. A selection register allow software to choose which event(s) to monitor. These counters allow events from any of four external events and/or pipeline-depth-levels to be counted concurrently.
- A 24-bit cycle counter that decrements on the PLB clock allows for a maximum sample period of approximately ~~120 msecs~~ 20 secs @133Mhz.
- An interrupt status register and an external interrupt signal that is triggered from any/all counter overruns.
- Master input ports to connect up to eight PLB masters. Each master input port receives signals associated with a PLB Master n,(n = 0, 1,. 7).
- Slave Input ports to connect up to eight PLB slaves. Each slave input port receives PLB slave signals associated with a PLB Slave n, (n=0, 1,...7) prior to the PLB slave OR logic.
- Configuration bits for selecting priority matching, address matching, transfer-size matching, and/or read/write matching allow finger granularity when tracking certain PLB master events. Configuration bits for selecting MasterID matching also allow finer granularity when tracking certain PLB slave events.
- Programmable Address Matching Registers that allow finer granularity when tracking PLB events. If enabled, PLB events containing a PLB address matching the 64-bit contents of the Address Matching Register(s) are tracked.

- Control and Status registers allowing software full control of PPM functionality, as well as a means of monitoring the status of each counter.
- Two sets of duration counters allow concurrent tracking of up to two event-duration measurements from any of the eight slaves. Event-duration measurements include counting of Read, Write, and/or Wait tenures associated with transactions from any of eight possible PLB slaves. Each event duration counter set contains an individual counter for tracking the Minimum Duration value, Maximum Duration value, Total Duration value, and the Total Event Occurrences.
- Pipeline-depth tracking feature allows for independent monitoring of hits-per-pipeline depth level for both read and write data buses. PLB implementations having a read-pipeline depth no greater than four, and a write-pipeline depth no greater than two are supported.
- Minimal bus loading. PLB signals feeding into the PPM's input interface are gated, therefore minimizing the additional PLB signal loading to just a single additional load per signal.
- Power Management (Class I) support for Low Power consumption mode (LPM). When sleep-mode is desired, the PPM can be placed into pseudo-active mode where all internal logic is halted with the exception of the pipeline-depth-tracking logic, thus eliminating unnecessary power consumption.
- Additional power management feature for Ultra-Low-Power consumption mode (ULPM). When not used, the PPM can be placed into inactive mode, where all internal clocking is halted with the exception of the DCR register access which is used for awakening.

Figure 3-1 describes a logical organization of the PLB performance monitor.

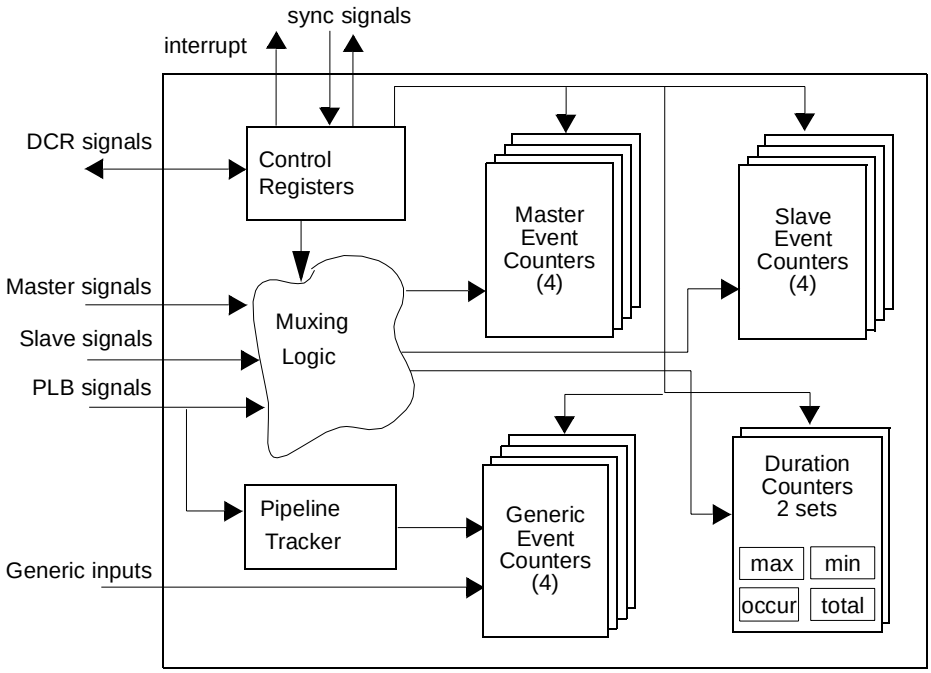


Figure 3-1. PPM Block Diagram

3.2 PLB Performance Counter Registers

All PPM registers are accessed using the **mtdcr** and **mfdcr** instructions. The PPM registers are accessed through the PPM0_ADDR and PPM0_DATA registers using an indirect addressing method. PPM0_CFGADDR and PPM0_CFGDATA are described in Table 3-1.

Table 3-1. PPM DCR Addresses

Mnemonic	DCR Address	Access	Register	Page
PPM0_CFGADDR	0x016	R/W	PPM Address Register	3-4120
PPM0_CFGDATA	0x017	R/W	PPM Data Register	3-5121

To read or write one of the indirectly accessed PPM registers, software must first write the target register address offset into the PPM0_ADDR register. The target register can then be read using **mfdcr** or written using **mtdcr** at the DCR address for PPM0_DATA. Indirectly accessed PPM registers and their address offsets are listed in Table 3-2.

Table 3-2. PPM Indirectly Accessed Registers

Mnemonic	Offset	Access	Register	Page
PPM0_ISR	0x0	R	Interrupt Status Register	3-5121
PPM0_CR	0x2	R/W	Control Register	3-7123
PPM0_CCR	0x3	R/W	Cycle Control Register	3-9125
PPM0_UAR	0x4	R/W	Upper Address Register	3-9125
PPM0_LAR	0x5	R/W	Lower Address Register	3-10 126
PPM0_UAMR	0x6	R/W	Upper Address Mask Register	3-11127
PPM0_LAMR	0x7	R/W	Lower Address Mask Register	3-11127
PPM0_RIDR	0x8	R	Revision ID Register	3-12 128
PPM0_MCSR0	0x9	R/W	Master Event Counter Selection Register 0	3-12 128
PPM0_MCSR1	0xA	R/W	Master Event Counter Selection Register 1	3-12 128
PPM0_MCSR2	0xB	R/W	Master Event Counter Selection Register 2	3-12 128
PPM0_MCSR3	0xC	R/W	Master Event Counter Selection Register 3	3-12 128

Table 3-2. PPM Indirectly Accessed Registers

Mnemonic	Offset	Access	Register	Page
PPM0_SCSR0	0x11	R/W	Slave Event Counter Selection Register 0	~~3-14~~ 130
PPM0_SCSR1	0x12	R/W	Slave Event Counter Selection Register 1	~~3-14~~ 130
PPM0_SCSR2	0x13	R/W	Slave Event Counter Selection Register 2	~~3-14~~ 130
PPM0_SCSR3	0x14	R/W	Slave Event Counter Selection Register 3	~~3-14~~ 130
PPM0_GCSR0	0x19	R/W	Generic Event Counter Selection Register 0	~~3-17~~ 133
PPM0_GCSR1	0x1A	R/W	Generic Event Counter Selection Register 1	~~3-17~~ 133
PPM0_GCSR2	0x1B	R/W	Generic Event Counter Selection Register 2	~~3-17~~ 133
PPM0_GCSR3	0x1C	R/W	Generic Event Counter Selection Register 3	~~3-17~~ 133
PPM0_MCR0	0x1D	R/W	Master Event Counter Register 0	~~3-17~~ 133
PPM0_MCR1	0x1E	R/W	Master Event Counter Register 1	~~3-17~~ 133
PPM0_MCR2	0x1F	R/W	Master Event Counter Register 2	~~3-17~~ 133
PPM0_MCR3	0x20	R/W	Master Event Counter Register 3	~~3-17~~ 133
PPM0_SCR0	0x25	R/W	Slave Event Counter Register 0	~~3-18~~ 134
PPM0_SCR1	0x26	R/W	Slave Event Counter Register 1	~~3-18~~ 134
PPM0_SCR2	0x27	R/W	Slave Event Counter Register 2	~~3-18~~ 134
PPM0_SCR3	0x28	R/W	Slave Event Counter Register 3	~~3-18~~ 134
PPM0_GCR0	0x2D	R/W	Generic Pipeline Event Counter Register 0	~~3-19~~ 135
PPM0_GCR1	0x2E	R/W	Generic Pipeline Event Counter Register 1	~~3-19~~ 135
PPM0_GCR2	0x2F	R/W	Generic Pipeline Event Counter Register 2	~~3-19~~ 135
PPM0_GCR3	0x30	R/W	Generic Pipeline Event Counter Register 3	~~3-19~~ 135

Table 3-2. PPM Indirectly Accessed Registers

Mnemonic	Offset	Access	Register	Page
PPM0_DCSR0	0x31	R/W	Duration Event Counter Selection Register 0	~~3-19~~ 135
PPM0_DCSR1	0x32	R/W	Duration Event Counter Selection Register 1	~~3-19~~ 135
PPM0_DCMXR0	0x33	R/W	Duration Event Counter Max Register 0	~~3-20~~ 136
PPM0_DCMXR1	0x34	R/W	Duration Event Counter Max Register 1	~~3-21~~ 137
PPM0_DCMNR0	0x35	R/W	Duration Event Counter Min Register 0	~~3-21~~ 137
PPM0_DCMNR1	0x36	R/W	Duration Event Counter Minimum Register 1	~~3-21~~ 137
PPM0_DCTVR0	0x37	R/W	Duration Event Counter Total Value Register 0	~~3-22~~ 138
PPM0_DCTVR1	0x38	R/W	Duration Event Counter Total Value Register 1	~~3-22~~ 138
PPM0_DCOTR0	0x39	R/W	Duration Event Counter Occurrence Total Register 0	~~3-23~~ 139
PPM0_DCOTR1	0x3A	R/W	Duration Event Counter Occurrence Total Register 1	~~3-23~~ 139

.

3.2.1 PPM Configuration Address Register (PPM0_CFGADDR)

The PPM0_CFGADDR Register is a 32-bit read/write register which holds an 8-bit offset associated with one of the PPM's configuration registers.

~~Figure 3-2~~Figure 3-2 describes PPM0_CFGADDR register bits.

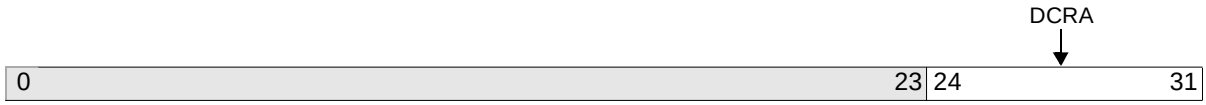


Figure 0-1. PPM Configuration Address Register (PPM0_CFGADDR)

0:23			
24:31	DCRA	8-bit PPM Register Offset Value	This value can range from 0x000000 to 0x7F. Its contents are used as the indirect DCR address for accessing a PPM register.

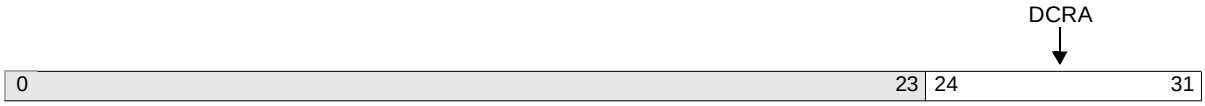


Figure 3-2. PPM Configuration Address Register (PPM0_CFGADDR)

0:23			
24:31	DCRA	8-bit PPM Register Offset Value	This value can range from 0x000000 to 0x7F. Its contents are used as the indirect DCR address for accessing a PPM register.

3.2.2 PPM Configuration Data Register (PPM0_CFGDATA)

The PPM0_CFGDATA Register is a 32-bit read/write register which holds a 32-bit read/write data to/from one of the PPM's configuration registers.

~~Figure 3-3~~Figure 3-3 describes PPM0_CFGDATA register bits.

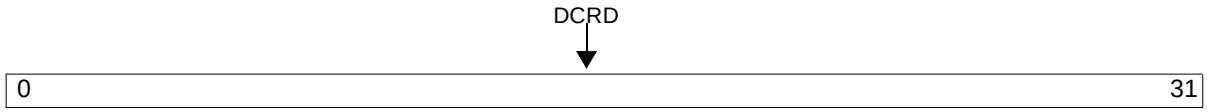


Figure 0-2. PPM Configuration Data Register (PPM0_CFGDATA)

0:31	DCRD	32-bit Data Value	This value can range from 0x00000000 to 0xFFFFFFFF. Its contents contain the value of the PPM register as indicated by the PPM_CFGADDR register contents.

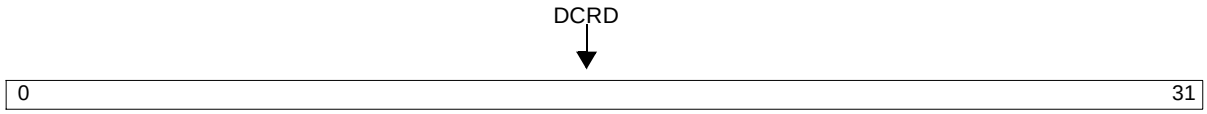


Figure 3-3. PPM Configuration Data Register (PPM0_CFGDATA)

0:31	DCRD	32-bit Data Value	This value can range from 0x00000000 to 0xFFFFFFFF. Its contents contain the value of the PPM register as indicated by the PPM_CFGADDR register contents.

3.2.3 Interrupt Status Register (PPM0_ISR)

The Interrupt Status Register (PPM_ISR) is a read-only 32-bit register where PPM interrupts are captured and held until bits are intentionally reset. The only way to reset a bit in this register is to clear the corresponding counter register. The contents of the ISR can be accessed by using the move from device control register (**mfdcr)** and move to device control register (**mtdcr)** instructions.

The default value of this register is 0h

~~Figure 3-4~~*Figure 3-4* describes PPM0_ISR register bits.

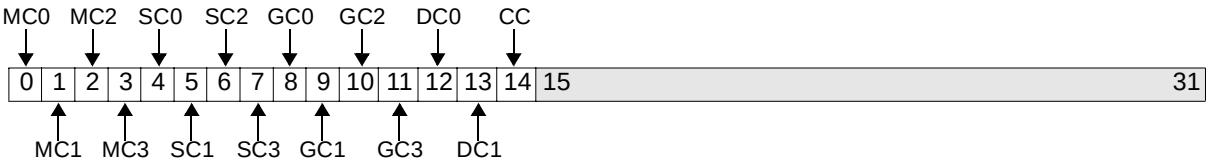


Figure 0-3. Interrupt Status Register (PPM0_ISR)

0	MC0	Master Counter 0 Interrupt Status 0 Master counter 0 interrupt did not occur 1 Master counter 0 interrupt occurred
1	MC1	Master Counter 1 Interrupt Status 0 Master counter 1 interrupt did not occur 1 Master counter 1 interrupt occurred
2	MC2	Master Counter 2 Interrupt Status 0 Master counter 2 interrupt did not occur 1 Master counter 2 interrupt occurred
3	MC3	Master Counter 3 Interrupt Status 0 Master counter 3 interrupt did not occur 1 Master counter 3 interrupt occurred
4	SC0	Slave Counter 0 Interrupt Status 0 Slave counter 0 interrupt did not occur 1 Slave counter 0 interrupt occurred
5	SC1	Slave Counter 1Interrupt Status 0 Slave counter 1 interrupt did not occur 1 Slave counter 1 interrupt occurred
6	SC2	Slave Counter 2 Interrupt Status 0 Slave counter 2 interrupt did not occur 1 Slave counter 2 interrupt occurred

7	SC3	Slave Counter 3 Interrupt Status 0 Slave counter 3 interrupt did not occur 1 Slave counter 3 interrupt occurred
8	GC0	Generic Counter 0 Interrupt Status 0 Generic counter 0 interrupt did not occur 1 Generic counter 0 interrupt occurred
9	GC1	Generic Counter 1 Interrupt Status 0 Generic counter 1 interrupt did not occur 1 Generic counter 1 interrupt occurred
10	GC2	Generic Counter 2 Interrupt Status 0 Generic counter 2 interrupt did not occur 1 Generic counter 2 interrupt occurred
11	GC3	Generic Counter 3 Interrupt Status 0 Generic counter 3 interrupt did not occur 1 Generic counter 3 interrupt occurred
12	DC0	Duration Counter 0 Interrupt Status 0 Duration counter 0 interrupt did not occur 1 Duration counter 0 interrupt occurred
13	DC1	Duration Counter 1 Interrupt Status 0 Duration counter 1 interrupt did not occur 1 Duration counter 1 interrupt occurred
14	CC	Cycle Counter Interrupt Status 0 Cycle counter interrupt did not occur 1 Cycle counter interrupt occurred
15:31		Reserved

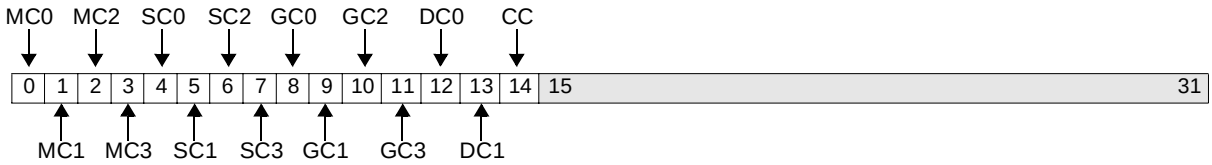


Figure 3-4. Interrupt Status Register (PPM0_ISR)

0	MC0	Master Counter 0 Interrupt Status 0 Master counter 0 interrupt did not occur 1 Master counter 0 interrupt occurred
1	MC1	Master Counter 1 Interrupt Status 0 Master counter 1 interrupt did not occur 1 Master counter 1 interrupt occurred
2	MC2	Master Counter 2 Interrupt Status 0 Master counter 2 interrupt did not occur 1 Master counter 2 interrupt occurred

3	MC3	Master Counter 3 Interrupt Status 0 Master counter 3 interrupt did not occur 1 Master counter 3 interrupt occurred
4	SC0	Slave Counter 0 Interrupt Status 0 Slave counter 0 interrupt did not occur 1 Slave counter 0 interrupt occurred
5	SC1	Slave Counter 1Interrupt Status 0 Slave counter 1 interrupt did not occur 1 Slave counter 1 interrupt occurred
6	SC2	Slave Counter 2 Interrupt Status 0 Slave counter 2 interrupt did not occur 1 Slave counter 2 interrupt occurred
7	SC3	Slave Counter 3 Interrupt Status 0 Slave counter 3 interrupt did not occur 1 Slave counter 3 interrupt occurred
8	GC0	Generic Counter 0 Interrupt Status 0 Generic counter 0 interrupt did not occur 1 Generic counter 0 interrupt occurred
9	GC1	Generic Counter 1 Interrupt Status 0 Generic counter 1 interrupt did not occur 1 Generic counter 1 interrupt occurred
10	GC2	Generic Counter 2 Interrupt Status 0 Generic counter 2 interrupt did not occur 1 Generic counter 2 interrupt occurred
11	GC3	Generic Counter 3 Interrupt Status 0 Generic counter 3 interrupt did not occur 1 Generic counter 3 interrupt occurred
12	DC0	Duration Counter 0 Interrupt Status 0 Duration counter 0 interrupt did not occur 1 Duration counter 0 interrupt occurred
13	DC1	Duration Counter 1 Interrupt Status 0 Duration counter 1 interrupt did not occur 1 Duration counter 1 interrupt occurred
14	CC	Cycle Counter Interrupt Status 0 Cycle counter interrupt did not occur 1 Cycle counter interrupt occurred
15:31		Reserved

3.2.4 Control Register (PPM0_CR)

The Control Register (PPM0_CR) is a 32-bit read/write register which allows full control of the PLB performance monitor. The contents of the CR can be accessed by using the move from device control register (**mfdcr)** and move to device control register (**mtdcr)** instructions.

This register must be the last register programmed. The default value of this register is 0h.

~~Figure 3-5~~*Figure 3-5* describes PPM0_CR register bits.

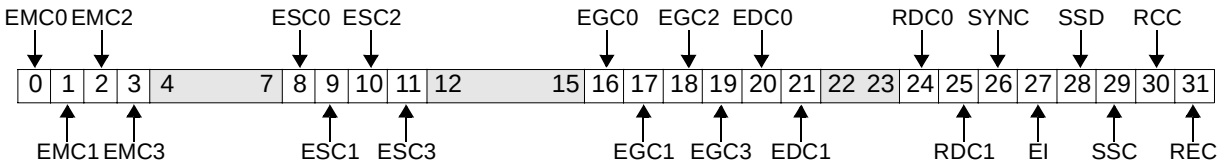


Figure 0-4. Control Register (PPM0_CR)

0	EMC0	Enable Master Counter 0 0 Disable master counter 0. 1 Enable master counter 0.
1	EMC1	Enable Master Counter 1 0 Disable master counter 1. 1 Enable master counter 1.
2	EMC2	Enable Master Counter 2 0 Disable master counter 2. 1 Enable master counter 2.
3	EMC3	Enable Master Counter 3 0 Disable master counter 3. 1 Enable master counter 3.
4:7		Reserved
8	ESC0	Enable Slave Counter 0 0 Disable slave counter 0. 1 Enable slave counter 0.
9	ESC1	Enable Slave Counter 1 0 Disable slave counter 1. 1 Enable slave counter 1.
10	ESC2	Enable Slave Counter 2 0 Disable slave counter 2. 1 Enable slave counter 2.
11	ESC3	Enable Slave Counter 3 0 Disable slave counter 3. 1 Enable slave counter 3.
12:15		Reserved
16	EGC0	Enable Generic Counter 0 0 Disable generic counter 0. 1 Enable generic counter 0.
17	EGC1	Enable Generic Counter 1 0 Disable generic counter 1. 1 Enable generic counter 1.
18	EGC2	Enable Generic Counter 2 0 Disable generic counter 2. 1 Enable generic counter 2.

19	EGC3	Enable Generic Counter 3 0 Disable generic counter 3. 1 Enable generic counter 3.
20	EDC0	Enable Duration Counter Set 0 0 Disable duration counter 0. 1 Enable duration counter 0.
21	EDC1	Enable Duration Counter Set 1 0 Disable duration counter 1. 1 Enable duration counter 1.
22:23		Reserved
24	RDC0	Reset Event Counters 0 No action. 1 Reset all DC0 counters.
25	RDC1	Reset Event Counters 0 No action. 1 Reset all DC1 counters.
26	SYNC	Sync enable 0 Disable sync-out functionality. 1 Enable sync-out functionality.
27	EI	Enable Interrupts 0 Disable interrupts. 1 Enable interrupts
28	SSD	Start/Stop Duration Events 0 Stop all counters programmed to operate in duration mode. 1 Start all counters programmed to operate in duration mode.
29	SSC	Start/Stop Cycle Counter 0 Stop decrementing cycle counter. 1 Start decrementing cycle counter.
30	RCC	Reset Cycle Counters 0 No action. 1 Reset cycle counter.
31	REC	Reset Event Counters 0 No action. 1 Reset all MCn, SCn, and GCn counters.

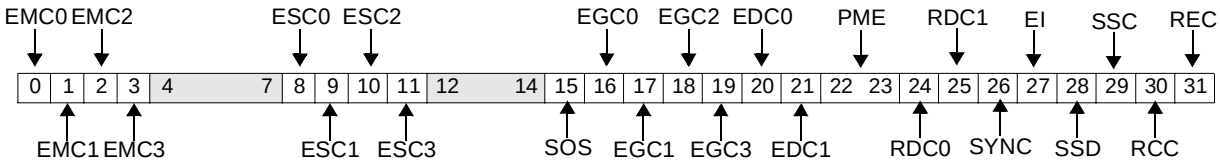


Figure 3-5. Control Register (PPM0_CR)

0	EMC0	Enable Master Counter 0 0 Disable master counter 0. 1 Enable master counter 0.
1	EMC1	Enable Master Counter 1 0 Disable master counter 1. 1 Enable master counter 1.
2	EMC2	Enable Master Counter 2 0 Disable master counter 2 1 Enable master counter 2
3	EMC3	Enable Master Counter 3 0 Disable master counter 3 1 Enable master counter 3
4:7		Reserved
8	ESC0	Enable Slave Counter 0 0 Disable slave counter 0 1 Enable slave counter 0
9	ESC1	Enable Slave Counter 1 0 Disable slave counter 1 1 Enable slave counter 1
10	ESC2	Enable Slave Counter 2 0 Disable slave counter 2 1 Enable slave counter 2
11	ESC3	Enable Slave Counter 3 0 Disable slave counter 3 1 Enable slave counter 3
12:14		Reserved
15	SOS	Synchout Selector 0 Drives PPM_synchOutSSC signal with value of SSC bit and PPM_synchOutSSD signal with value of SSD bit. 1 Asserts PPM_synchOutSSC signal when master counter 0, slave counter 0, generic counter 0, or cycle counter has overflow/underrun its value. Asserts PPM_synchOurSSD signal when duration counter 0 occurrence total counter, duration counter 1 occurrence total counter, duration counter o total value counter, and duration counter 1 total value counter has overrun their contents. **Note:** The associated interrupts must be enabled for this feature to work.
16	EGC0	Enable Generic Counter 0 0 Disable generic counter 0 1 Enable generic counter 0
17	EGC1	Enable Generic Counter 1 0 Disable generic counter 1 1 Enable generic counter 1
18	EGC2	Enable Generic Counter 2 0 Disable generic counter 2. 1 Enable generic counter 2.

19	EGC3	Enable Generic Counter 3 0 Disable generic counter 3. 1 Enable generic counter 3.
20	EDC0	Enable Duration Counter Set 0 0 Disable duration counter 0. 1 Enable duration counter 0.
21	EDC1	Enable Duration Counter Set 1 0 Disable duration counter 1. 1 Enable duration counter 1.
22:23	PME	Power Mode Enable 00 Normal mode 01 Low power mode 10 Ultra low power mode 11 Ultra low power mode
24	RDC0	Reset Event Counters 0 No action. 1 Reset all DC0 counters.
25	RDC1	Reset Event Counters 0 No action. 1 Reset all DC1 counters.
26	SYNC	SyncIn enable 0 Disable syncIn functionality. 1 Enable syncIn functionality.
27	EI	Enable Interrupts 0 Disable interrupts. 1 Enable interrupts
28	SSD	Start/Stop Duration Events 0 Stop all counters 1 Start all counters
29	SSC	Start/Stop Cycle Counter 0 Stop decrementing cycle counter. 1 Start decrementing cycle counter.
30	RCC	Reset Cycle Counters 0 No action. 1 Reset cycle counter.
31	REC	Reset Event Counters 0 No action. 1 Reset all MCn, SCn, and GCn counters.

3.2.5 Cycle Counter Register (PPM0_CCR)

The Cycle Counter Register (PPM0_CCR) is a 32-bit read/write register which holds a time value for PPM sampling. When enabled, the content of the CCR register value is decremented once on each PLB clock cycle. The contents of the CCR register can be accessed by using the move from device control register (**mfdcr)** and the move to device control register (**mtdcr**) instruction.

The default value for this register is FFFFFFh (max value).

~~Figure 3-6~~Figure 3-6 describes PPM0_CCR register bits.

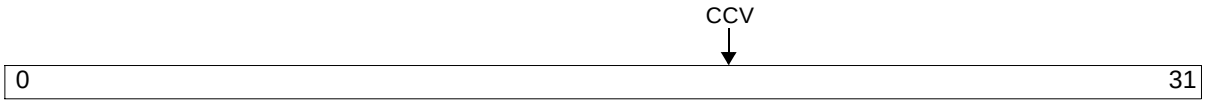


Figure 0-5. Cycle Control Register (PPM0_CCR)

0:31	CCV	32-bit Clock Count Value	This value can range from 0x0h to 0xFFFFFFFFh. Its contents are decremented by 1 (for every PLB clock) if the enable bit in the CR register is active. This counter will stop decrementing when a value of 0x0h reached.

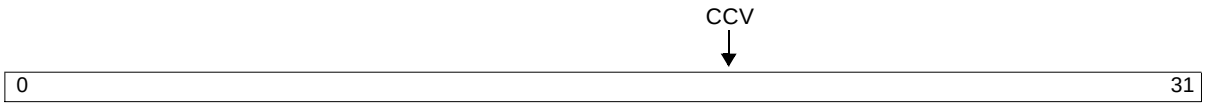


Figure 3-6. Cycle Control Register (PPM0_CCR)

0:31	CCV	32-bit Clock Count Value	This value can range from 0x0h to 0xFFFFFFFFh. Its contents are decremented by 1 (for every PLB clock) if the enable bit in the CR register is active. This counter will stop decrementing when a value of 0x0h reached.

3.2.6 Upper Address Register (PPM0_UAR)

The Upper Address Register (PPM0_UAR) is a 32-bit register that contains a value that is compared against the upper 32 bits of the 64-bit PLB address bus. When an MCounterN is programmed to use the address-matching feature, that counter will only track events where the PLB address matches the contents of this register and the lower address register(LAR), collectively. This register is used along with the upper address mask register (UAMR), lower address register (LAR), and the lower address mask register (LAMR). The contents of the UAR can be accessed by using the move from device control register (**mfdcr)** and the move to device control register (**mtdcr**) instruction.

The default value of this register is 0h.

~~Figure 3-7~~Figure 3-7 describes PPM0_UAR register bits.

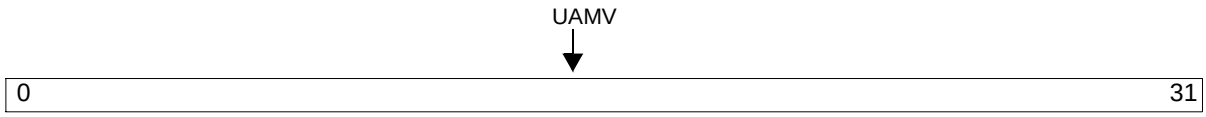


Figure 0-6. Upper Address Register (PPM0_UAR)

0:31	UAMV	Upper Address Match Value	If enabled, this value is matched against the upper PLB address bits UAbus(0-31) of the PLB transaction event.

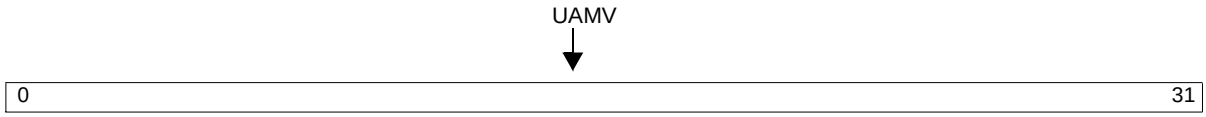


Figure 3-7. Upper Address Register (PPM0_UAR)

0:31	UAMV	Upper Address Match Value	If enabled, this value is matched against the upper PLB address bits UAbus(0-31) of the PLB transaction event.

3.2.7 Lower Address Register (PPM0_LAR)

The Lower Address Register (LAR) is a 32-bit register that contains a value that is compared against the lower 32 bits (0-31) of the 64-bit PLB address bus. When an MCounterN is programmed to use this feature (enabled by EAM bit in MCounterN Selection register(s)), that counter will only track events where the PLB address matches the contents of this register and the upper address register, collectively. This register is used along with the lower address mask register (LAMR), upper address register (UAR), and the upper address mask register (UAMR). The contents of the LAR can be accessed by using the move from device control register (**mfdcr)** and the move to device control register (**mtdcr**) instruction.

The default value of this register is 0h.

~~Figure 3-8~~Figure 3-8 describes PPM0_LAR register bits.

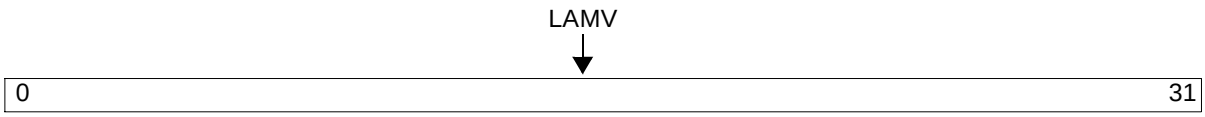


Figure 0-7. Lower Address Register (PPM0_LAR)

0:31	LAMV	Lower Address Match Value	If enabled, this value is match against the upper address bits of the PLB transaction event. Its contents can range from 0x00000000 to 0xFFFFFFF.

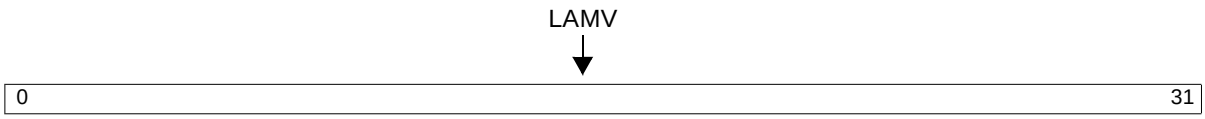


Figure 3-8. Lower Address Register (PPM0_LAR)

0:31	LAMV	Lower Address Match Value	If enabled, this value is match against the upper address bits of the PLB transaction event. Its contents can range from 0x00000000 to 0xFFFFFFF.

3.2.8 Upper Address Mask Register (PPM0_UAMR)

The Upper Address Mask Register (UAMR) is a 32-bit register used to individually isolate bit-level address comparison when using the address matching feature. Each bit in this register corresponds directly with an associated bit in the UAR register. This register is used along with the UAR match register. The contents of the UAMR can be accessed by using the move from device control register (**mfdcr)** and the move to device control register (**mtdcr**) instruction.

The default value of this register is FFFFFFFFh.

~~Figure 3-9~~Figure 3-9 describes PPM0_UAMR register bits.

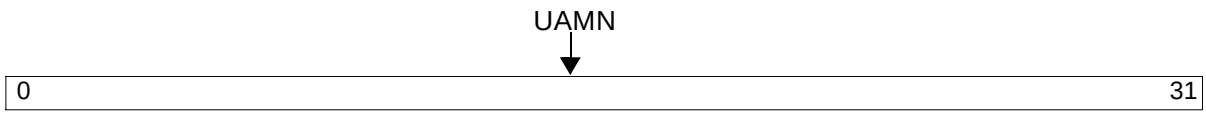


Figure 0-8. Upper Address Mask Register (PPM0_UAMR)

0:31	UAMn	Upper Address Bit-n Mask 0 Disable bit-n matching. 1 Enable bit-n matching.	If enabled, the corresponding bit in theUAR register is match against the same address bit-n of the PLB upper address bus.

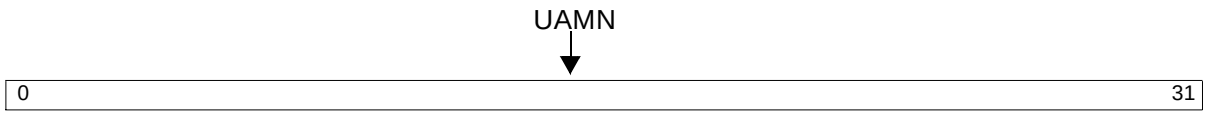


Figure 3-9. Upper Address Mask Register (PPM0_UAMR)

0:31	UAMn	Upper Address Bit-n Mask 0 Disable bit-n matching. 1 Enable bit-n matching.	If enabled, the corresponding bit in theUAR register is match against the same address bit-n of the PLB upper address bus.

3.2.9 Lower Address Mask Register (PPM0_LAMR)

The Lower Address Mask Register (LAMR) is a 32-bit register used to individually isolate bit-level address comparison when using the address matching feature. Each bit in this register corresponds directly with an associated bit in the LAR register. This register is used along with the LAR match register described in a previous section. The contents of the LAMR can be accessed by using the move from device control register (**mfdcr)** and the move to device control register (**mtdcr**) instruction.

The default value of this register is FFFFFFFFh

~~Figure 3-10~~Figure 3-10 describes PPM0_LAMR register bits.

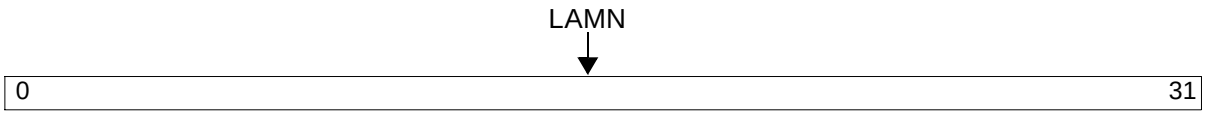


Figure 0-9. Lower Address Mask Register (PPM0_LAMR)

0:31	LAMn	Lower AddressBit-n Mask 0 Disable bit-n matching. 1 Enable bit-n matching.	If enabled, the corresponding bit in the LAR register is match against the same address bit-n of the PLB lower address bus.

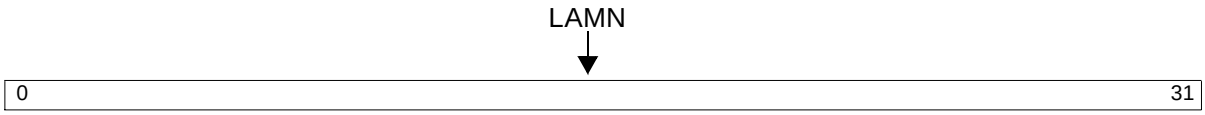


Figure 3-10. Lower Address Mask Register (PPM0_LAMR)

0:31	LAMn	Lower AddressBit-n Mask 0 Disable bit-n matching. 1 Enable bit-n matching.	If enabled, the corresponding bit in the LAR register is match against the same address bit-n of the PLB lower address bus.

3.2.10 Revision ID Register (PPM0_RIDR)

The Revision ID Register is a 32-bit read-only register used to identify the current design level implemented for this PPM core. The contents of the RID can be accessed by using the move from device control register (**mfdcr)** and the move to device control register (**mtdcr**) instruction.

The default value of this register is C27E341x h, where x is the revision #. The default value for x in PPC440GP is 1.

~~Figure 3-11~~Figure 3-11 describes PPM0_RIDR register bits.

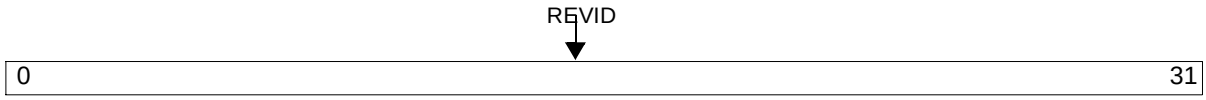


Figure 0-10. Revision ID Register (PPM0_RIDR)

0:31	REVID	RevID value: 0xC27E341x	Read Only field, where x is the Revision #. The default vaule for x in is 1.

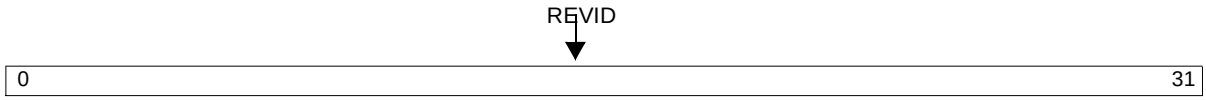


Figure 3-11. Revision ID Register (PPM0_RIDR)

0:31	REVID	RevID value: 0xC27E341x	Read Only field, where x is the Revision #. The default vaule for x in PPC440GP is 1.

3.2.11 Master Event Counter Selection Register 0:3 (PPM0_MCSR0-PPM0_MCSR3)

The Master Event Counter Selection Registers 0:3 (MCSR0:MCSR3) are 32-bit read/write registers which allow event selections for corresponding master counters. The contents of the MCSR0:MCSR3 can be accessed by using the move from device control register (**mfdcr)** and move to device control register (**mtdcr)** instructions.

The default value of this register is 0h.

~~Figure 3-12~~*Figure 3-12* describes PPM0_MCSR0-PPM0_MCSR3 register bits.

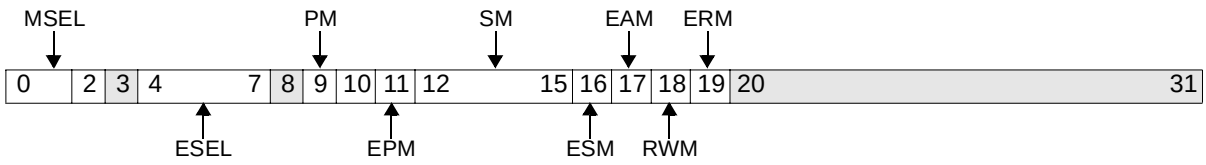


Figure 0-11. Master Event Counter Selection Register 0:7 (PPM0_MCSR0-PPM0_MCSR7)

0:2	MSEL	Master Selection 000 Selects Master 0 001 Selects Master 1 010 Selects Master 2 011 Selects Master 3 100 Selects Master 4 101 Selects Master 5 110 Selects Master 6 111 Selects Master 7	Selects which master's signal transitions to count with this counter set. Master 0 is ICU Read Master 1 is DCU Read Master 2 is DCU Write Master 3 is PCIX Master 4 is reserved Master 5 is MAL Master 6 is DMA Master 7 is OPB to PLB bridge
3		Reserved	
4:7	ESEL	Master Event Selection 0000 Selects PLB_MnAddrAck 0001 Selects PLB_MnRdDAck 0010 Selects PLB_MnWrDAck 0011 Selects PLB_MnRdErr 0100 Selects PLB_MnWrErr 0101 Selects PLB_MnRequest 0110 Selects PLB_MnAbort 0111 Selects PLB_MnBuslock 1000 Selects Mn_WrBurst 1001 Selects Mn_RdBurst 1010-1111 Reserved	Selects the master's PLB event to be tracked. (refer to Table 3-4, for signal event qualifications)
8		Reserved	
9:10	PM	Priority Matching 00 Lowest priority 01 Low priority 10 High priority 11 Highest priority	Sets the priorty decode value, which will be used if EPM bit is set.
11	EPM	Enable Priority Matching 0 Disable priority matching 1 Enable priority matching	Selects the priority bits to be used when tracking an event **Note:** This feature is functional when MES bits=0101 only. This bit is ignored for all other MES values.

12:15	SM	PLB transaction Size Matching 0000 One to 16 bytes 0001 4-word line 0010 8-word line 0011 16-word line 0100 Reserved 0101 Reserved 0110 Reserved 0111 Reserved 1000 Burst transfer 1001 Burst transfer 1010 Burst transfer 1011 Burst transfer 1100 Burst transfer 1101 Burst transfer 1110 Reserved 1111 Reserved	Sets the PLB transaction size decode value, which will be used during event tracking if the ESM bit is set.
16	ESM	Enable Size Matching 0 Disable transaction Size matching 1 Enable transaction Size matching	Selects the SM bits to be used when tracking an event **Note:** This feature is functional when MES bits=0000 only. This bit is ignored for all other values.
17	EAM	Enable Address Matching 0 Disable address matching feature 1 Enable address matching feature	Selects the contents of the UAR/LAR registers to be used when tracking an event **Note:** This feature is functional when MES bits=0000 only. This bit is ignored for all other values.
18	RWM	RnW Matching 0 Write tranastion matching 1 Read transaction matching	Selects the transaction matching criteria for Mn_Request transactions only.
19	ERM	Enable RnW Matching 0 Disable matching 1 Enable matching	Selects the priority bits to be used when tracking an event **Note:** This feature is functional when MES bits=0000 or 0110 only. This bit is ignored for all other values.
20:31		Reserved	

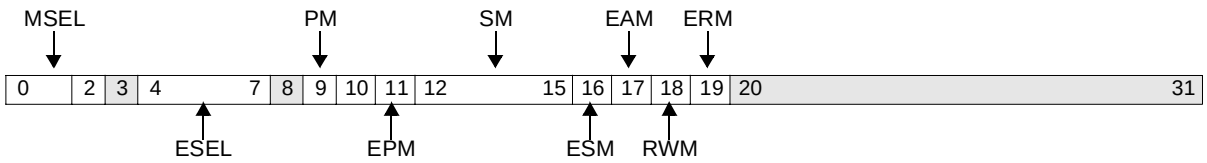


Figure 3-12. Master Event Counter Selection Register 0:7 (PPM0_MCSR0-PPM0_MCSR7)

0:2	MSEL	Master Selection 000 Selects Master 0 001 Selects Master 1 010 Selects Master 2 011 Selects Master 3 100 Selects Master 4 101 Selects Master 5 110 Selects Master 6 111 Selects Master 7	Selects which master's signal transitions to count with this counter set. Master 0 is ICU Read Master 1 is DCU Read Master 2 is DCU Write Master 3 is PCIX Master 4 is reserved Master 5 is MAL Master 6 is DMA Master 7 is OPB to PLB bridge
3		Reserved	
4:7	ESEL	Master Event Selection 0000 Selects PLB_MnAddrAck 0001 Selects PLB_MnRdDAck 0010 Selects PLB_MnWrDAck 0011 Selects PLB_MnRdErr 0100 Selects PLB_MnWrErr 0101 Selects PLB_MnRequest 0110 Selects PLB_MnAbort 0111 Selects PLB_MnBuslock 1000 Selects Mn_WrBurst 1001 Selects Mn_RdBurst 1010-1111 Reserved	Selects the master's PLB event to be tracked. (refer to Table 3-4, for signal event qualifications)
8		Reserved	
9:10	PM	Priority Matching 00 Lowest priority 01 Low priority 10 High priority 11 Highest priority	Sets the priorty decode value, which will be used if EPM bit is set.
11	EPM	Enable Priority Matching 0 Disable priority matching 1 Enable priority matching	Selects the priority bits to be used when tracking an event **Note:** This feature is functional when MES bits=0101 only. This bit is ignored for all other MES values.
12:15	SM	PLB transaction Size Matching 0000 One to 16 bytes 0001 4-word line 0010 8-word line 0011 16-word line 0100 Reserved 0101 Reserved 0110 Reserved 0111 Reserved 1000 Burst transfer 1001 Burst transfer 1010 Burst transfer 1011 Burst transfer 1100 Burst transfer 1101 Burst transfer 1110 Reserved 1111 Reserved	Sets the PLB transaction size decode value, which will be used during event tracking if the ESM bit is set.

16	ESM	Enable Size Matching 0 Disable transaction Size matching 1 Enable transaction Size matching	Selects the SM bits to be used when tracking an event **Note:** This feature is functional when MES bits=0000 only. This bit is ignored for all other values.
17	EAM	Enable Address Matching 0 Disable address matching feature 1 Enable address matching feature	Selects the contents of the UAR/LAR registers to be used when tracking an event **Note:** This feature is functional when MES bits=0000 only. This bit is ignored for all other values.
18	RWM	RnW Matching 0 Write tranastion matching 1 Read transaction matching	Selects the transaction matching criteria for Mn_Request transactions only.
19	ERM	Enable RnW Matching 0 Disable matching 1 Enable matching	Selects the priority bits to be used when tracking an event **Note:** This feature is functional when MES bits=0000 or 0110 only. This bit is ignored for all other values.
20:31		Reserved	

3.2.12 Slave Event Counter Selection Register 0:7 (PPM0_SCSR0-PPM0_SCSR7)

The Slave Event Counter Selection Registers 0:7 (SCSR0:SCSR7) are 32-bit read/write registers which allow event selections for corresponding slave counters. The contents of the SCSR0:SCSR7 can be accessed by using the move from device control register (**mfdcr)** and move to device control register (**mtdcr)** instructions.

The default value of this register is 0h.

~~Figure 3-13~~*Figure 3-13* describes PPM0_SCSR0-PPM0_SCSR7 register bits.

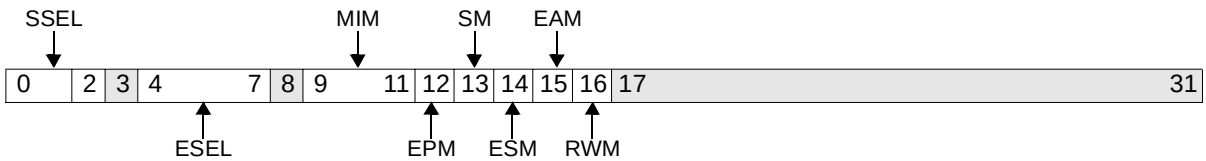


Figure 0-12. Slave Event Counter Selection Register 0:3 (PPM0_SCSR0-PPM0_SCSR3)

0:2	SSEL	Slave Selection 000 Selects Slave 0 001 Selects Slave 1 010 Selects Slave 2 011 Selects Slave 3 100 Selects Slave 4 101 Selects Slave 5 110 Selects Slave 6 111 Selects Slave 7	Selects which Slave's signal transitions to count with this counter set. Slave 0 is DDR_SDRAM Slave 1 is PCIX Slave 2 is SRAM Slave 3 is reserved Slave 4 is reserved SLave 5 is PLB to OPB bridge Slave 6 is reserved Slave 7 is reserved
3		Reserved	
4:7	ESEL	Slave Event Selection 0000 Selects PLB_SlnAddrAck 0001 Selects PLB_SlnRdDAck 0010 Selects PLB_SlnWrDAck 0011 Selects PLB_SlnRearbitrate 0100 Selects Sln_Wait 0101 Selects Sln_WrComp 0110 Selects Sln_RdComp 0111-1111 Reserved	Selects the PLB slave event to be tracked.

8		Reserved	
9:11	MIM	Master ID Matching 000 Selects Master 0 001 Selects Master 1 010 Selects Master 2 011 Selects Master 3 100 Selects Master 4 101 Selects Master 5 110 Selects Master 6 111 Selects Master 7	Sets the MasterID decode value, which will be used if EMIM bit is set. Master 0 is ICU Read Master 1 is DCU Read Master 2 is DCU Write Master 3 is PCIX Master 4 is Reserved Master 5 is MAL Master 6 is DMA Master 7 is OPB to PLB Bridge
12	EMIM	Enable Master ID Matching 0 Disable master ID matching 1 Enable master ID matching	Selects the MIM bits to be used when tracking an event **Note:** This feature is functional when SES bits=0000 only. This bit is ignored for all other SES values.
13	RWM	RnW Matching 0 Write tranastion matching 1 Read transaction matching	Selects the Read or Write value that will be used when ERM bit is set.
14	ERM	Enable RnW Matching 0 Disable matching 1 Enable matching	Selects the RWM bit to be used when tracking an event **Note:** This feature is functional when SES bits=0000 or 0011 only. This bit is ignored for all other SES values.
15	AVM	Address Valid Matching 0 PAValid matching 1 SAValid matching	Selects either the PAValid or SAValid signals to used when the EVM bit is set.
16	EVM	Enable Address Valid Matching 0 Disable matching 1 Enable matching	Enables the Primary of Secondary Address Valid matching feature to be used when tracking an event **Note:** This feature is functional when SES bits= 0011 only. This bit is ignored for all other SES values.
17:31		Reserved	

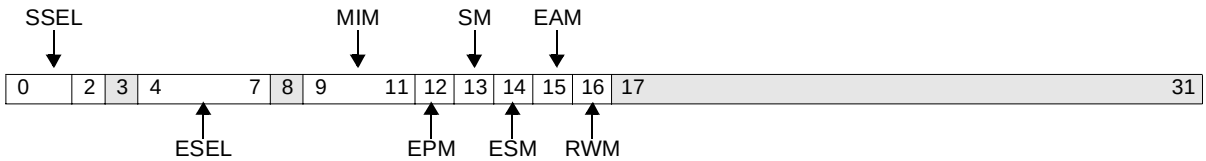


Figure 3-13. Slave Event Counter Selection Register 0:3 (PPM0_SCSR0-PPM0_SCSR3)

0:2	SSEL	Slave Selection 000 Selects Slave 0 001 Selects Slave 1 010 Selects Slave 2 011 Selects Slave 3 100 Selects Slave 4 101 Selects Slave 5 110 Selects Slave 6 111 Selects Slave 7	Selects which Slave's signal transitions to count with this counter set. Slave 0 is DDR_SDRAM Slave 1 is PCIX Slave 2 is SRAM Slave 3 is reserved Slave 4 is reserved SLave 5 is PLB to OPB bridge Slave 6 is reserved Slave 7 is reserved
3		Reserved	
4:7	ESEL	Slave Event Selection 0000 Selects PLB_SlnAddrAck 0001 Selects PLB_SlnRdDAck 0010 Selects PLB_SlnWrDAck 0011 Selects PLB_SlnRearbitrate 0100 Selects Sln_Wait 0101 Selects Sln_WrComp 0110 Selects Sln_RdComp 0111-1111 Reserved	Selects the PLB slave event to be tracked.

8		Reserved	
9:11	MIM	Master ID Matching 000 Selects Master 0 001 Selects Master 1 010 Selects Master 2 011 Selects Master 3 100 Selects Master 4 101 Selects Master 5 110 Selects Master 6 111 Selects Master 7	Sets the MasterID decode value, which will be used if EMIM bit is set. Master 0 is ICU Read Master 1 is DCU Read Master 2 is DCU Write Master 3 is PCIX Master 4 is Reserved Master 5 is MAL Master 6 is DMA Master 7 is OPB to PLB Bridge
12	EMIM	Enable Master ID Matching 0 Disable master ID matching 1 Enable master ID matching	Selects the MIM bits to be used when tracking an event **Note:** This feature is functional when SES bits=0000 only. This bit is ignored for all other SES values.
13	RWM	RnW Matching 0 Write tranastion matching 1 Read transaction matching	Selects the Read or Write value that will be used when ERM bit is set.
14	ERM	Enable RnW Matching 0 Disable matching 1 Enable matching	Selects the RWM bit to be used when tracking an event **Note:** This feature is functional when SES bits=0000 or 0011 only. This bit is ignored for all other SES values.
15	AVM	Address Valid Matching 0 PAValid matching 1 SAValid matching	Selects either the PAValid or SAValid signals to used when the EVM bit is set.
16	EVM	Enable Address Valid Matching 0 Disable matching 1 Enable matching	Enables the Primary of Secondary Address Valid matching feature to be used when tracking an event **Note:** This feature is functional when SES bits= 0011 only. This bit is ignored for all other SES val-ues.
17:31		Reserved	

3.2.13 Generic Event Counter Selection Register 0:3 (PPM0_GCSR0-PPM0_GCSR3)

The Generic/Pipeline Event Counter Selection Registers 0:3 (GCSR0:GCSR3) are 32-bit read/write registers which allow selections for generic events corresponding to an external device or a pipeline-depth signal corresponding to one of four levels. The contents of the GCSR0:GCSR3 can be accessed by using the move from device control register (**mfdcr)** and move to device control register (**mtdcr)** instructions.

The default value of this register is 0h.

~~Figure 3-14~~*Figure 3-14* describes PPM0_GCSR0-PPM0_GCSR3 register bits.

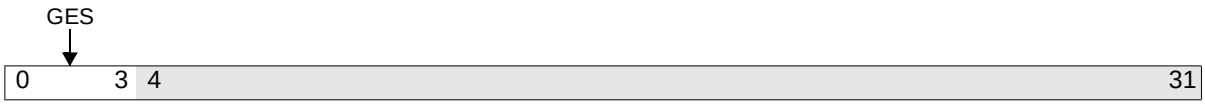


Figure 0-13. Generic Event Counter Selection Registers 0:3 (PPM0_GCSR0-PPM0_GCSR3)

0-3	GES	Generic Event Selection 0000 Selects G0_input 0001 Selects G1_input 0010 Selects G2_input 0011 Selects G3_input 0100 Selects write-pipeline-depth 1 0101 Selects write-pipeline-depth 2 0110 Selects read-pipeline-depth 1 0111 Selects read-pipeline-depth 2 1000 Selects read-pipeline-depth 3 1001 Selects read-pipeline-depth 4	Selects the external signal n, or the pipeline-depth n signal as input to this counter. **Note:** Using the pipeline depth selection(s) one can determine the depth of PLB pipelining that has been reached under specific applications.
4:31		Reserved	

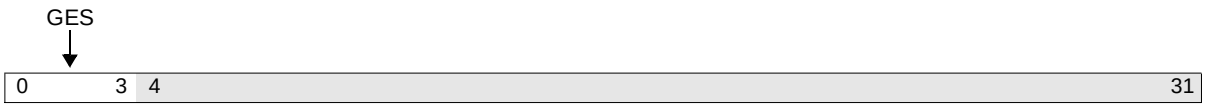


Figure 3-14. Generic Event Counter Selection Registers 0:3 (PPM0_GCSR0-PPM0_GCSR3)

0-3	GES	Generic Event Selection 0000 Selects G0_input 0001 Selects G1_input 0010 Selects G2_input 0011 Selects G3_input 0100 Selects write-pipeline-depth 1 0101 Selects write-pipeline-depth 2 0110 Selects read-pipeline-depth 1 0111 Selects read-pipeline-depth 2 1000 Selects read-pipeline-depth 3 1001 Selects read-pipeline-depth 4	Selects the external signal n, or the pipeline-depth n signal as input to this counter. **Note:** Using the pipeline depth selection(s) one can determine the depth of PLB pipelining that has been reached under specific applications.
4:31		Reserved	

3.2.14 Master Event Counter Register 0:3 (PPM0_MCR0-PPM0_MCR3)

The Master Event Counter Registers 0:3 (MCR0:MCR3) are 32-bit read/write register which that will increment according to the operation mode as indicated in the MCounterN Selection Register. In occurrence mode, these counters will increment by 1 for each occurrence of its preselected event. The value will continue to increment until the contents of the Cycle Counter register = 0x0. In duration mode, these registers increment by 1 for each PLB clock cycle until the first pre-selected event occurs as defined in ~~"PLB Event-Duration Measurements" on page 3-25~~*PLB Event-Duration Measurements* on page 141. Through program control, an interrupt can be generated once the contents of this register reaches a value of FFFFFFh.The contents of the MCR0:MCR3 can be accessed by using the move from device control register (**mfdcr**) and the move to device control register (**mtdcr**) instruction.

The default value of this register is 0h.

~~Figure 3-15~~Figure 3-15 describes PPM0_MCR0-PPM0_MCR3 register bits.

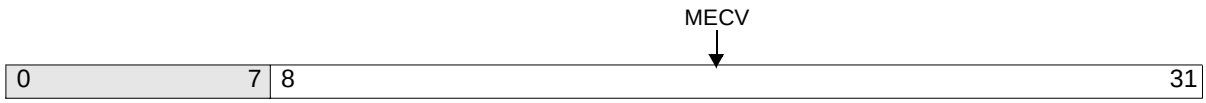


Figure 0-14. Master Event Counter Registers 0:3 (PPM0_MCR0-PPM0_MCR3)

0:7		Reserved	
8:31	MECV	Master Event Counter Value	This value can range from 0x000000 to 0xFFFFFF. Its contents are incremented once for each event occurrence as selected in its corresponding Mcounter Selection register. This register will only be updated if its corresponding MCn enable bit in the CR register is active.

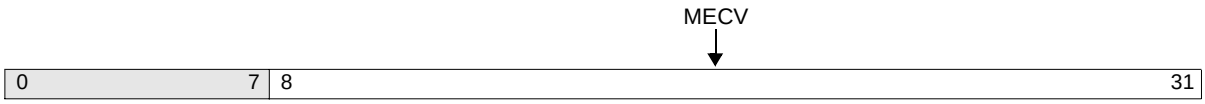


Figure 3-15. Master Event Counter Registers 0:3 (PPM0_MCR0-PPM0_MCR3)

0:7		Reserved	
8:31	MECV	Master Event Counter Value	This value can range from 0x000000 to 0xFFFFFF. Its contents are incremented once for each event occurrence as selected in its corresponding Mcounter Selection register. This register will only be updated if its corresponding MCn enable bit in the CR register is active.

3.2.15 SLave Event Counter Register 0:3 (PPM0_SCR0-PPM0_SCR3)_

The Slave Event Counter Registers 0:7 (SCR0:SCR3) are 32-bit read/write register that will increment by 1 for each occurrence of its event as dictated by the corresponding "SCounterN selection register". When enabled, this register will continue to increment until the contents of the Cycle Counter register = 0x0. Through program control, an interrupt can be generated once the contents of this register reaches a value of FFFFFFh. The contents of the SCR0:SCR3 can be accessed by using the move from device control register (**mfdcr**) and the move to device control register (**mtdcr**) instruction.

The default value of this register is 0h.

~~Figure 3-16~~Figure 3-16 describes PPM0_SCR0-PPM0_SCR3 register bits.

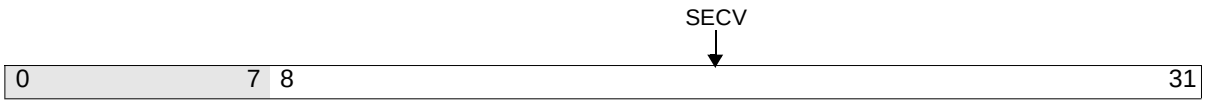


Figure 0-15. Slave Event Counter Registers 0:3 (PPM0_SCR0-PPM0_SCR3)

0:7		Reserved	
8:31	SECV	Slave Event Counter Value	This value can range from 0x000000 to 0xFFFFFF. Its contents are incremented once for each event occurrence as selected in its corresponding SCounter Selection register. This register can only be updated if its corresponding SCn enable bit in the CR register is active

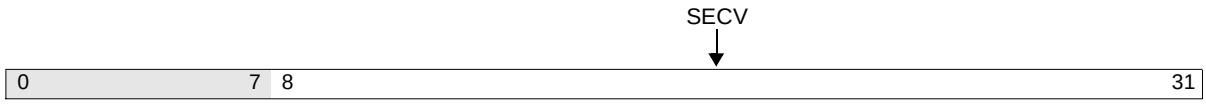


Figure 3-16. Slave Event Counter Registers 0:3 (PPM0_SCR0-PPM0_SCR3)

0:7		Reserved	
8:31	SECV	Slave Event Counter Value	This value can range from 0x000000 to 0xFFFFFF. Its contents are incremented once for each event occurrence as selected in its corresponding SCounter Selection register. This register can only be updated if its corresponding SCn enable bit in the CR register is active

3.2.16 Generic Pipeline Event Counter Register 0:3 (PPM0_GCR0-PPM0_GCR3)

The Generic Pipeline Event Counter Registers 0:3 (GCR0:GCR3) are 32-bit read/write register that will increment by 1 for each occurrence of its event as dictated by the corresponding "GCounterN selection register". When enabled, this register will continue to increment until the contents of the Cycle Counter register = 0x0. The contents of the GCR0:GCR3 can be accessed by using the move from device control register (**mfdcr**) and the move to device control register (**mtdcr**) instruction.

The default value of this register is 0h.

~~Figure 3-17~~Figure 3-17 describes PPM0_GCR0-PPM0_GCR3 register bits.

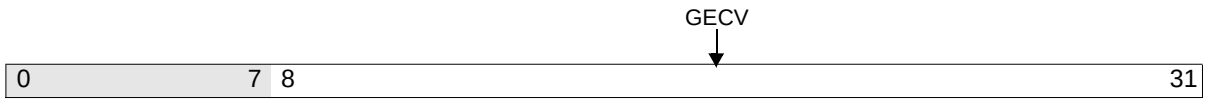


Figure 0-16. Generic Pipeline Event Counter Registers 0:3 (PPM0_GCR0-PPM0_GCR3)

0:7		Reserved	
8:31	GECV	Generic Event Counter Value	This value can range from 0x000000 to 0xFFFFFF. Its contents are incremented once for each event occurrence as selected in its corresponding GCounter Selection register. This register can only be updated if its corresponding GCn enable bit in the CR register is active

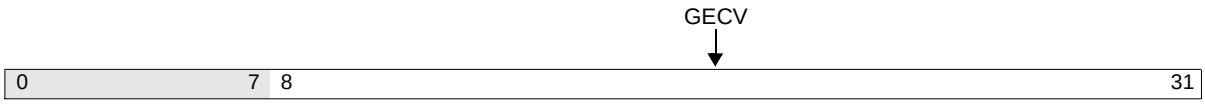


Figure 3-17. Generic Pipeline Event Counter Registers 0:3 (PPM0_GCR0-PPM0_GCR3)

0:7		Reserved	
8:31	GECV	Generic Event Counter Value	This value can range from 0x000000 to 0xFFFFFF. Its contents are incremented once for each event occurrence as selected in its corresponding GCounter Selection register. This register can only be updated if its corresponding GCn enable bit in the CR register is active

3.2.17 Duration Counter Selection Register 0:1 (PPM0_DCSR0-PPM0_DCSR1)

The Duration Counter Selection Registers 0:1 (DCSR0:DCSR1) are 32-bit read/write registers which allow event selections for corresponding duration counters. The contents of the DCSR0:DCSR1 can be accessed by using the move from device control register (**mfdcr)** and move to device control register (**mtdcr)** instructions.

The default value of this register is 0h.

~~Figure 3-18~~Figure 3-18 describes PPM0_DCSR0-PPM0_DCSR1 register bits.

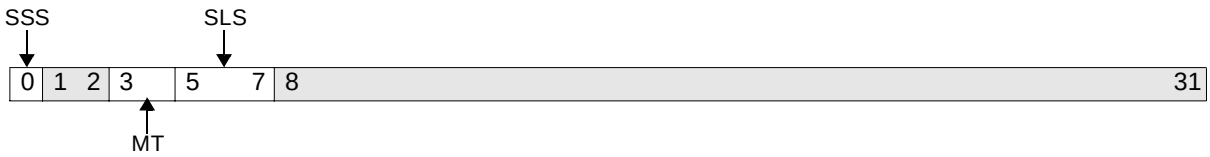


Figure 0-17. Duration Counter Selection Registers 0:1 (PPM0_DCSR0-PPM0_DCSR1)

0	SSS	Single Shot Selection 0 Disable single shot measurement 1 Enable single shot measurement	Selects which Slave's signal transitions to count with this counter set.
1:2		Reserved	
3:4	MT	Measurement Type 00 Write tenure 01 Read tenure 10 Wait tenure 11 Reserved	Selects the type of slave duration measurement to be performed (see Table 3-4).

5:7	SLS	Slave Selection 000 Selects SI0 001 Selects SI1 010 Selects SI2 011 Selects SI3 100 Selects SI4 101 Selects SI5 110 Selects SI6 111 Selects SI7	Selects the desired PLB Slave device in which the measurement is to be performed Slave 0 is DDR_SDRAM Slave 1 is PCIX Slave 2 is SRAM Slave 3 is reserved Slave 4 is reserved SLave 5 is PLB to OPB bridge Slave 6 is reserved Slave 7 is reserved
8:31		Reserved	

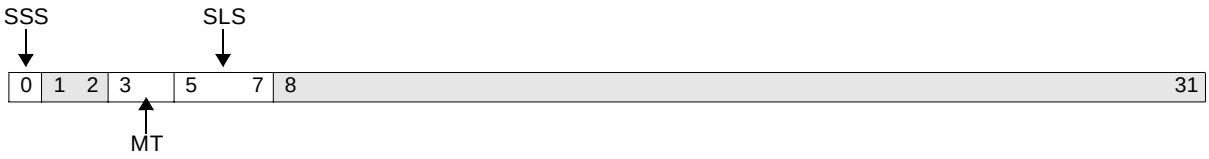


Figure 3-18. Duration Counter Selection Registers 0:1 (PPM0_DCSR0-PPM0_DCSR1)

0	SSS	Single Shot Selection 0 Disable single shot measurement 1 Enable single shot measurement	Selects which Slave's signal transitions to count with this counter set.
1:2		Reserved	
3:4	MT	Measurement Type 00 Write tenure 01 Read tenure 10 Wait tenure 11 Reserved	Selects the type of slave duration measurement to be performed (see Table 3-4).
5:7	SLS	Slave Selection 000 Selects SI0 001 Selects SI1 010 Selects SI2 011 Selects SI3 100 Selects SI4 101 Selects SI5 110 Selects SI6 111 Selects SI7	Selects the desired PLB Slave device in which the measurement is to be performed Slave 0 is DDR_SDRAM Slave 1 is PCIX Slave 2 is SRAM Slave 3 is reserved Slave 4 is reserved SLave 5 is PLB to OPB bridge Slave 6 is reserved Slave 7 is reserved
8:31		Reserved	

3.2.18 Duration Counter Maximum Register 0:1 (PPM0_DCMXR0-PPM0_DCMXR1)

The Duration Counter Max Registers 0:1 (DCMXR0:DCMXR1) are 32-bit read/write registers that will contain the last calculated maximum duration for the currently selected duration-event being monitored. Through program control, an interrupt can be generated once the contents of this register reaches a value of FFFFFFh. The contents of this register can be accessed by using the move from device control register (**mfdcr**) and the move to device control register (**mtdcr**) instruction.

The default value of this register is 0h.

~~Figure 3-19~~Figure 3-19 describes PPM0_DCMXR0-PPM0_DCMXR1 register bits.

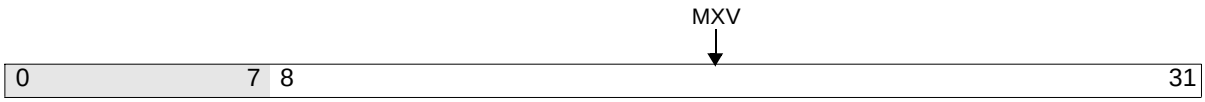


Figure 0-18. Duration Counter Maximum Registers 0:1 (PPM0_DCMXR0-PPM0_DCMXR1)

0:7		Reserved	
8:31	MXV	Highest Max value	This value can range from 0x0 to 0xFFFFFF. Its contents are the highest maximum duration value calculated via the selection in its corresponding Duration Selection register. This register can only be updated if its corresponding enable bit in the CR register is active

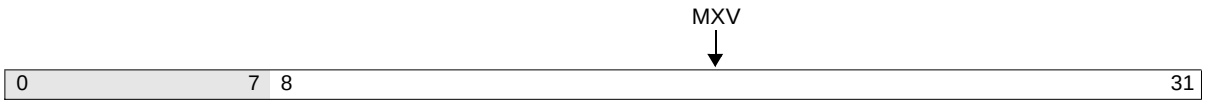


Figure 3-19. Duration Counter Maximum Registers 0:1 (PPM0_DCMXR0-PPM0_DCMXR1)

0:7		Reserved	
8:31	MXV	Highest Max value	This value can range from 0x0 to 0xFFFFFF. Its contents are the highest maximum duration value calculated via the selection in its corresponding Duration Selection register. This register can only be updated if its corresponding enable bit in the CR register is active

3.2.19 Duration Counter Minimum Register 0:1 (PPM0_DCMNR0-PPM0_DCMNR1)_

The Duration Counter Min Registers 0:1 (DCMNR0:DCMNR1) are 32-bit read/write registers that will contain the last calculated minimum duration for the currently selected duration-event being monitored. Through program control, an interrupt can be generated once the contents of this register reaches a value of FFFFFFFh. The contents of this register can be accessed by using the move from device control register (**mfdcr**) and the move to device control register (**mtdcr**) instruction.

The default value of this register is 0h.

~~Figure 3-20~~Figure 3-20 describes PPM0_DCMNR0-PPM0_DCMNR1 register bits.

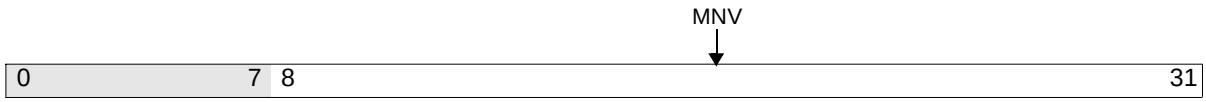


Figure 0-19. Duration Counter Minimum Registers 0:1 (PPM0_DCMNR0-PPM0_DCMNR1)

0:7		Reserved	
8:31	MNV	Least Min value	This value can range from 0x000000 to 0xFFFFFF. Its contents are the least minimum duration value calculated via the selection in its corresponding Duration Selection register. This register can only be updated if its corresponding enable bit in the CR register is active

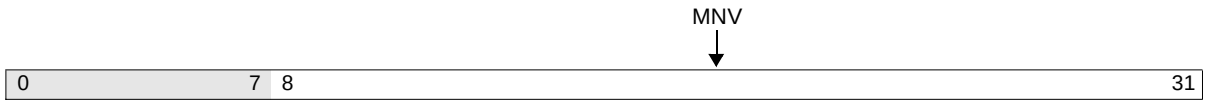


Figure 3-20. Duration Counter Minimum Registers 0:1 (PPM0_DCMNR0-PPM0_DCMNR1)

0:7		Reserved	
8:31	MNV	Least Min value	This value can range from 0x000000 to 0xFFFFFF. Its contents are the least minimum duration value calculated via the selection in its corresponding Duration Selection register. This register can only be updated if its corresponding enable bit in the CR register is active

3.2.20 Duration Counter Total Value Register 0:1 (PPM0_DCTVR0-PPM0_DCTVR1)

The Duration Counter Total Value Registers 0:1 (DCTVR0:DCTVR1) are 32-bit read/write registers that will contain the running total value of all durations for the currently selected duration-event being monitored. Through program control, an interrupt can be generated once the contents of this register reaches a value of FFFFFFFh. The contents of this register can be accessed by using the move from device control register (**mfdcr**) and the move to device control register (**mtdcr**) instruction.

The default value of this register is 0h.

~~Figure 3-21~~Figure 3-21 describes PPM0_DCTVR0-PPM0_DCTVR1 register bits.

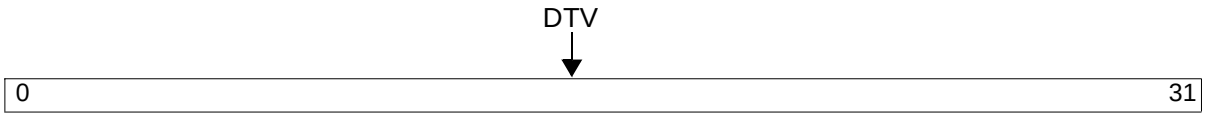


Figure 0-20. Duration Total Value Register 0:1 (PPM0_DCTVR0-PPM0_DCTVR1)

0:31	DTV	Duration Total Value	This value can range from 0x0 to 0xFFFFFFFF. The value is incremented in accordance with the duration values calculated via the selection in its corresponding Duration Selection register. This register can only be updated if its corresponding enable bit in the CR register is active.

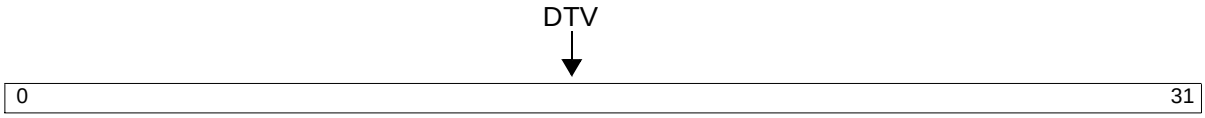


Figure 3-21. Duration Total Value Register 0:1 (PPM0_DCTVR0-PPM0_DCTVR1)

0:31	DTV	Duration Total Value	This value can range from 0x0 to 0xFFFFFFFF. The value is incremented in accordance with the duration values calculated via the selection in its corresponding Duration Selection register. This register can only be updated if its corresponding enable bit in the CR register is active.

3.2.21 Duration Counter Occurrence Total Register 0:1 (PPM0_DCOTR0-PPM0_DCOTR1)

The Duration Counter Occurrence Total Registers 0:1 (DCOTR0:DCOTR1) are 32-bit read/write registers that will contain the total number of event occurrences for the currently selected duration-event being monitored. Through program control, an interrupt can be generated once the contents of this register reaches a value of FFFFFFh. The contents of this register can be accessed by using the move from device control register (**mfdcr**) and the move to device control register (**mtdcr**) instruction.

The default value of this register is 0h.

~~Figure 3-22~~Figure 3-22 describes PPM0_DCOTR0-PPM0_DCOTR1 register bits.

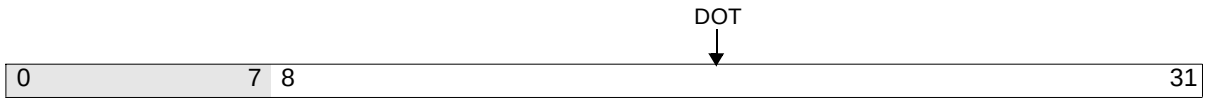


Figure 0-21. Duration Counter Occurrence Total Registers 0:1 (PPM0_DCOTR0-1)

0:7		Reserved	
8:31	DOT	Total number of event occurrences	This value can range from 0x0 to 0xFFFFFF. Its contents are incremented once for each occurrence of an event as selected in the corresponding Duration Selection register. This register can only be updated if its corresponding enable bit in the CR register is active

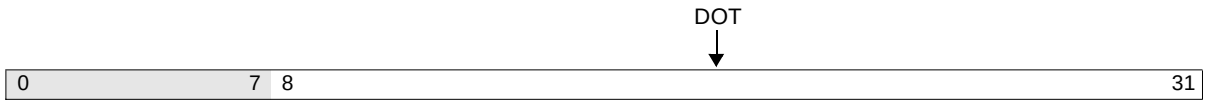


Figure 3-22. Duration Counter Occurrence Total Registers 0:1 (PPM0_DCOTR0-1)

0:7		Reserved	
8:31	DOT	Total number of event occurrences	This value can range from 0x0 to 0xFFFFFF. Its contents are incremented once for each occurrence of an event as selected in the corresponding Duration Selection register. This register can only be updated if its corresponding enable bit in the CR register is active

3.3 Monitoring of PLB Events

The PLB performance monitor can monitor selected events simultaneously via a set of event counters. The event definitions are described in Table 3-3 below. Bits residing in the event selection registers allow software to select which event to count.

3.3.1 PLB Event-Occurrence Qualification

When validating PLB transactions as acceptable PPM events for monitoring, certain conditions apply. These events along with their qualifying signals and conditions are shown inTable 3-3.

Table 3-3. PLB Event Occurrence

PPM Event	Qualifications and Conditions
PLB_MnAddrAck (non-specific)	Signal active & SYS_plbClk (rising edge)
PLB_MnAddrAck (w/ address matching)	Signal active & SYS_plbClk (rising edge) & Masked (PLB_ABus(0:31) == UA register) & Masked(PLB_ABus(0:31) == LAR register)
PLB_MnAddrAck (w/ rd./war matching)	Signal active & SYS_plbClk (rising edge) & (RnW bit of MCounterN Selection register == Mn_RnW value)
PLB_MnAddrAck (w/ address and rd/wr matching)	Signal active & SYS_plbClk (rising edge) & Masked (PLB_ABus(0:31) == UA register) & Masked(PLB_ABus(0:31) == LAR register) & (RnW bit of MCounterN Selection register == Mn_RnW value)
PLB_MnAddrAck (w/ msize matching)	Signal active & SYS_plbClk (rising edge) & (Mn_Size(0:3) = value of SM bits in specified MCounterN Selection register)
PLB_MnAddrAck (w/ address and size matching)	Signal active & SYS_plbClk (rising edge) & Masked (PLB_ABus(0:31) == UA register) & Masked(PLB_ABus(0:31) == LAR register) & (Mn_Size(0:3) = value of SM bits in specified MCounterN Selection register)
PLB_MnAddrAck (w/ rd/wr and size matching)	Signal active & SYS_plbClk (rising edge) &(RnW bit of MCounterN Selection register == Mn_RnW value) & (Mn_Size(0:3) = value of SM bits in specified MCounterN Selection register)
PLB_MnAddrAck (w/ address, rd/wr, and size matching)	Signal active & SYS_plbClk (rising edge) & Masked(PLB_ABus(0:31) == UA register) & Masked(PLB_ABus(0:31) == LAR register) & (RnW bit of MCounterN Selection register == Mn_RnW value) & (Mn_Size(0:3) = value of SM bits in specified MCounterN Selection register)
PLB_MnRdAck	Signal active & SYS_plbClk (rising edge)
PLB_MnWrDAck	Signal active & $\overline{\text{Mn\_Abort}}$ & SYS_plbClk(rising edge)
PLB_MnRdErr	Signal active & SYS_plbClk (rising edge)
PLB_MnWrErr	Signal active & SYS_plbClk (rising edge)
Mn_Request (non-specific)	Signal (rising_edge) & $\overline{\text{Mn\_Abort}}$ & $\overline{\text{PLB\_MnRearbitrate}}$ & $\overline{\text{PLB\_MnTimeout}}$ & SYS_plbClk (rising edge)
Mn_Request (w/ priority matching)	Signal (rising_edge) & $\overline{\text{Mn\_Abort}}$ & $\overline{\text{PLB\_MnRearbitrate}}$ & $\overline{\text{PLB\_MnTimeout}}$ & SYS_plbClk (rising edge) &PLB_MnPriority(0:1) = value of PM bits in Specified MCounterN Selection register)
Mn_Abort (non-specific)	Signal active & PLB_MnRequest & SYS_plbClk (rising edge)
Mn_Abort (w rd/wr matching)	Signal active & PLB_MnRequest & SYS_plbClk (rising edge) & (RnW bit of MCounterN Selection register == Mn_RnW value)

Table 3-3. PLB Event Occurrence (continued)

PPM Event	Qualifications and Conditions
Mn_WrBurst	Signal active & SYS_plbClk (rising edge)
Mn_RdBurst	Signal active & SYS_plbClk (rising edge)
Sln_AddrAck (non-specific)	Signal active & SYS_plbClk (rising edge)
Sln_AddrAck (w/ MasterID matching)	Signal active & SYS_plbClk (rising edge) & (PLB_MasterID(0:2) = value of MIM bits in specified SCounterN Selection register)
Sln_RdDAck	Signal active & SYS_plbClk (rising edge)
Sln_WrDAck	Signal active & SYS_plbClk (rising edge)
Sln_rearbitrate (non-specific)	Signal active & SYS_plbClk (rising edge)
Sln_rearbitrate (w/ rd/wr matching)	Signal active & SYS_plbClk (rising edge) & (RnW bit of MCounterN Selection register == Mn_RnW value)
Sln_rearbitrate (w/ PAValid)	Signal active & SYS_plbClk (rising edge) & PAValid
Sln_rearbitrate (w/ SAValid)	Signal active & SYS_plbClk (rising edge) & SAValid
Sln_rearbitrate (w/ PAValid and rd/wr matching)	Signal active & SYS_plbClk (rising edge) & PAValid & (RnW bit of MCounterN Selection register == Sln_RnW value)
Sln_rearbitrate (w/ SAValid and rd/wr matching)	Signal active & SYS_plbClk (rising edge) & SAValid & (RnW bit of MCounterN Selection register == Sln_RnW value)
Sln_wrComp	Signal active & SYS_plbClk (rising edge) & $\overline{\text{Mn\_Abort}}$
Sln_rdComp	Signal active & SYS_plbClk (rising edge) & $\overline{\text{Mn\_Abort}}$
Sln_Wait	Signal active (rising edge only) & $\overline{\text{SLn\_Addrack}}$
G0_input	Signal active & SYS_plbClk (rising edge)
G1_input	Signal active & SYS_plbClk (rising edge)
G2_input	Signal active & SYS_plbClk (rising edge)
G3_input	Signal active & SYS_plbClk (rising edge)
Pipeline_depth 1 event (write bus)	PAValid & Sln_Addrack (w/o Abort) & $\overline{\text{PLB\_RnW}}$ & SYS_plbClk (rising edge)
Pipeline_depth 2 event (write bus)	SAValid & Sln_Addrack (w/o Abort) & $\overline{\text{PLB\_ RnW}}$ & SYS_plbClk (rising edge)
Pipeline_depth_1 event (read bus)	PAValid & Sln_Addrack (w/o Abort) & PLB_RnW & SYS_plbClk (rising edge)
Pipeline_depth_2 event (read bus)	First (SAValid & Sln_Addrack (w/o Abort) & PLB_RnW & SYS_plbClk (rising edge)) without Sln_rdComp occurrence AND current pipeline-depth is level 1.
Pipeline_depth 3 event (read bus)	First (SAValid & Sln_Addrack (w/o Abort) & PLB_RnW & SYS_plbClk (rising edge)) without Sln_rdComp occurrence AND current pipeline-depth is level 2.
Pipeline_depth 4 event (read bus)	(SAValid & Sln_Addrack (w/o Abort) & PLB_RnW & SYS_plbClk (rising edge)) AND current pipeline-depth is level 3.

Note: In the above context, pipeline-depth level differs from pipeline-depth event, in that the depth level refers to the highest depth containing an outstanding plb transaction, while the depth event refers to the depth in which a newly accepted plb transaction is placed.

3.3.2 PLB Event-Duration Measurements

If event-duration monitoring is desired, the Duration counters can be individually enabled to perform this function. The following table shows the different types of event-duration measurements that the PPM can be programmed to monitor. Note: The duration counters are programmable to operate in "single-shot" mode, where the counter will stop incrementing its value once the desired event has been captured, or "running total" mode, where the counters will continually start and stop counting on each event occurrence. All duration counters that have been programmed to operate in the "running total" mode will continue to function, until the global start/stop duration (SSD) bit in the Control Register has been cleared, or the Cycle Counter has timed out.

Table 3-4. PLB Event Duration Measurement

Duration Measurement Type	Measurement Qualifiers	Measurements Taken
Sln Write Tenure	# of clock cycles between assertion of Sln_AddrAck(qualified) and assertion of Sln_WrComp (qualified) signal	(1) Total clock cycle count(s), (2) Total # of occurrences (3) # of clock cycles in shortest measurement (Min), (4) # of clocks cycles in longest measurement (Max), (5) # of Sln_WrDAcks**
Sln Read Tenure	# of clock cycles between assertion of Sln_AddrAck and assertion of Sln_RdComp signal	(1) Total clock cycle count, (2) Total # of occurrences, (3) # of clock cycles in shortest measurement (Min), (4) # of clocks cycles in longest measurement (Max), (5) # of Sln_RdDAcks**
Sln Wait Tenure	(# of clock cycles between assertion and deassertion of Sln_Wait signal) & Sln_AddrAck	(1) Total clock cycle count(s), (2) Total # of occurrences, (3) # of clock cycles in shortest measurement (Min), (4) # of clocks cycles in longest measurement (Max),

**This measurement is to be performed in separate event counter registers as described in ~~"PLB Event-Occurrence Qualification" on page 3-23~~*PLB Event-Occurrence Qualification* on page 139.

Part II. PPC440GP RISC Processor

5. Instruction and Data Caches

The PPC440GP provides separate instruction and data cache controllers and arrays, which allow concurrent access and minimize pipeline stalls. The storage capacity of both cache arrays is 32KB. Both cache controllers have 32-byte lines, and both are highly associative, having 64-way set-associativity. The PowerPC instruction set provides a rich set of cache management instructions for software-enforced coherency. The PPC440GP implementation also provides special debug instructions that can directly read the tag and data arrays. The cache controllers interface to the processor local bus (PLB) for connection to the IBM CoreConnect system-on-a-chip environment.

The rest of this chapter provides more detailed information about the operation of the instruction and data cache controllers and arrays.

5.1 Cache Array Organization and Operation

The instruction and data cache arrays are organized identically, although the fields of the tag and data portions of the arrays are slightly different because the functions of the arrays differ, and because the instruction cache is virtually tagged while the data cache has real tags.

The organization of the cache into "ways" and "sets" is as follows. There are 64 ways in each set, with a set consisting of all 64 lines (one line from each way) at which a given memory location can reside. Conversely, there are 16 sets in each way, with a way consisting of 16 lines (one from each set).

~~Table 5-1 on page 5-1~~*Table 5-1* illustrates the ways and sets of the cache arrays. The tag field for each line in each way holds the high-order address bits associated with the line that currently resides in that way. The middle-order address bits form an index to select a specific set of the cache, while the five lowest-order address bits form a byte-offset to choose a specific byte (or bytes, depending on the size of the operation) from the 32-byte cache line.

Table 5-1. Instruction and Data Cache Array Organization

	Way 0	Way 1	• • •	Way 62	Way 63
Set 0	Line 0	Line 16	• • •	Line 992	Line 1008
Set 1	Line 1	Line 17	• • •	Line 993	Line 1009
• • •	• • •	• • •	• • •	• • •	• • •
Set 14	Line 14	Line 30	• • •	Line 1006	Line 1022
Set 15	Line 15	Line 31	• • •	Line 1007	Line 1023

In the cache array, an effective address (EA) is divided into three parts: tag, set, and byte offset. See ~~Figure 5-7~~*Figure 5-7* and ~~Figure 5-8 on page 5-17~~*Figure 5-8* on page 219 for instruction cache tag address bits, and ~~Figure 5-9~~*Figure 5-9* and ~~Figure 5-10 on page 5-28~~*Figure 5-10* on page 232 for data cache tag address bits. Also, see ~~"Instruction Cache Synonyms" on page 5-12~~*Instruction Cache Synonyms* on page 215 for details on instruction cache synonyms associated with the use of virtual tags for the instruction cache. $A_{23:26}$ are the set address bits, and $A_{27:31}$ are the byte offset address bits.

5.1.1 Cache Line Replacement Policy

Memory addresses are specified as being cacheable or caching inhibited on a page basis, using the caching inhibited (I) storage attribute (see ~~"Caching Inhibited (I)" on page 6-14~~*Caching Inhibited (I)* on page 245). When a program references a cacheable memory location and that location is not already in the cache (a *cache miss*), the line may be brought into the cache (a *cache line fill* operation) and placed into any one of the ways within the set selected by the middle portion of the address (address bits $EA_{23:26}$ select the set). If the particular way within the set already contains a valid line from some other address, the existing line is removed and replaced by the newly referenced line from memory. The line being replaced is referred to as the *victim*.

The way selected to be the victim for replacement is controlled by a field within a Special Purpose Register (SPR). There is a separate "victim index field" for each set within the cache. The registers controlling the victim selection are shown in Figure 5-1.

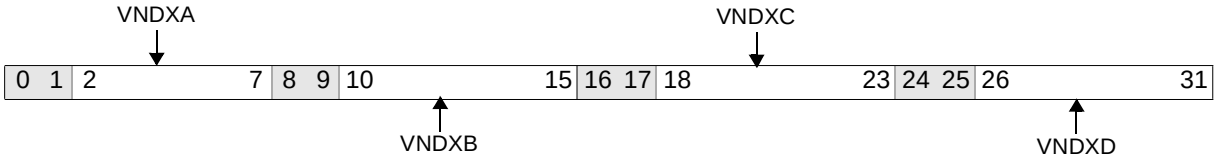


Figure 0-1. Instruction Cache Normal Victim Registers (INV0–INV3)
Instruction Cache Transient Victim Registers (ITV0–ITV3)
Data Cache Normal Victim Registers (DNV0–DNV3)
Data Cache Transient Victim Registers (DTV0–DTV3)

0:1		Reserved	
2:7	VNDXA	Victim Index A (for cache lines with EA[25:26] = 0b00)	
8:9		Reserved	
10:15	VNDXB	Victim Index A (for cache lines with EA[25:26] = 0b01)	
16:17		Reserved	
18:23	VNDXC	Victim Index A (for cache lines with EA[25:26] = 0b10)	
24:25		Reserved	
26:31	VNDXD	Victim Index A (for cache lines with EA[25:26] = 0b11)	

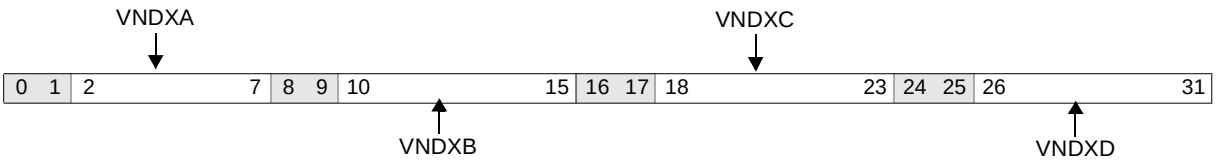


Figure 5-1. Instruction Cache Normal Victim Registers (INV0–INV3) Instruction Cache Transient Victim Registers (ITV0–ITV3) Data Cache Normal Victim Registers (DNV0–DNV3) Data Cache Transient Victim Registers (DTV0–DTV3)

0:1		Reserved	
2:7	VNDXA	Victim Index A (for cache lines with EA[25:26] = 0b00)	
8:9		Reserved	
10:15	VNDXB	Victim Index B (for cache lines with EA[25:26] = 0b01)	
16:17		Reserved	
18:23	VNDXC	Victim Index C (for cache lines with EA[25:26] = 0b10)	
24:25		Reserved	
26:31	VNDXD	Victim Index D (for cache lines with EA[25:26] = 0b11)	

Note: Each of the victim index fields consist of six bits, as there are 64 ways in 32KB cache. Unusued bits of the victim selection registers are reserved.

Each of the 16 SPRs illustrated in Figure 5-1 can be written from a GPR using **mtspr**, and can be read into a GPR using **mfspr**. In general, however, these registers are initialized by software once at startup, and then are managed automatically by hardware after that. Specifically, every time a new cache line is placed into the cache, the appropriate victim index field (as controlled by the type of access and the particular cache set being updated) is first referenced to determine which way within that set should be replaced. Then, that same field is incremented such that the ways within that set are replaced in a *round-robin* fashion as each new line is brought into that set. When the victim index field value reaches the index of the last way (according to the size of the cache and the type of access being performed), the value is *wrapped back* to the index of the first way for that type of access. The first and last ways for the different types of accesses are controlled by fields in a pair of *victim limit* SPRs, one for each cache (see ~~"Cache Locking and Transient Mechanism" on page 5-3~~*Cache Locking and Transient Mechanism* on page 207 for more information).

The victim index field that is used varies according to the type of access and the address of the cache line. ~~Table 5-2~~*Table 5-2* describes the correlation between the victim index fields and different access types, and addresses.

Table 5-2. Victim Index Field Selection

Address$_{23:26}$	Victim Index Field[1,~~2~~]
0	xxV0[VNDXA]$_{~~2:7~~}$
1	xxV0[VNDXB]$_{~~2:7~~}$
2	xxV0[VNDXC]$_{~~2:7~~}$
3	xxV0[VNDXD]$_{~~2:7~~}$
4	xxV1[VNDXA]$_{~~2:7~~}$
5	xxV1[VNDXB]$_{~~2:7~~}$
6	xxV1[VNDXC]$_{~~2:7~~}$
7	xxV1[VNDXD]$_{~~2:7~~}$
8	xxV2[VNDXA]$_{~~2:7~~}$
9	xxV2[VNDXB]$_{~~2:7~~}$
10	xxV2[VNDXC]$_{~~2:7~~}$
11	xxV2[VNDXD]$_{~~2:7~~}$
12	xxV3[VNDXA]$_{~~2:7~~}$
13	xxV3[VNDXB]$_{~~2:7~~}$
14	xxV3[VNDXC]$_{~~2:7~~}$
15	xxV3[VNDXD]$_{~~2:7~~}$

Note 1: "xx" refers to "IN", "IT", "DN", or "DT", depending on whether the access is to the instruction or data cache, and whether the access is "normal" or "transient." (~~See "Cache Locking and Transient Mechanism" on page 5-3.~~See *Cache Locking and Transient Mechanism* on page 207)

~~**Note 2:** Bits 2:7 of the victim index fields are used because the cache arrays have 64 ways. Unused bits of the victim index fields are reserved.~~

Note 3:

5.1.2 Cache Locking and Transient Mechanism

Both caches support locking, at a "way" granularity. Any number of ways can be locked, from 0 ways to 63. At least one way must always be left unlocked, for use by cacheable line fills.

In addition, a portion of each cache can be designated as a "transient" region, by specifying that only a limited number of ways are used for cache lines from memory pages that are identified as being transient in nature by a storage attribute from the MMU (see ~~Chapter 6, "Memory Management"~~*Memory Management* on page 233). For the instruction cache, such memory pages can be used for code sequences that are unlikely to be reused once the processor moves on to the next series of instruction lines. Thus, performance may be improved by preventing each series of instruction lines from overwriting the rest of the "regular" code in the instruction cache. Similarly, for the data cache, transient pages can be used for large "streaming" data struc-

tures, such as multimedia data. As each piece of the data stream is processed and written back to memory, the next piece can be brought in, overwriting the previous (now obsolete) cache lines instead of displacing other areas of the cache, which may contain other data that should remain in the cache.

A set of fields in a pair of victim limit registers specifies which ways of the cache are used for normal accesses and/or transient accesses, as well as which ways are locked. These registers, Instruction Cache Victim Limit (IVLIM) and Data Cache Victim Limit (DVLIM), are illustrated in Figure 5-2. They can be written from a GPR using **mtspr**, and can be read into a GPR using **mfspr**.

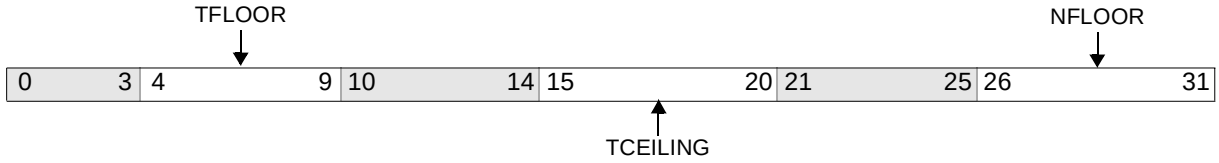


**Figure 0-2. Instruction Cache Victim Limit (IVLIM)
Data Cache Victim Limit (DVLIM)**

0:3		Reserved	
4:9	TFLOOR	Transient Floor	
10:14		Reserved	
15:20	TCEILING	Transient Ceiling	
21:25		Reserved	
26:31	NFLOOR	Normal Floor	

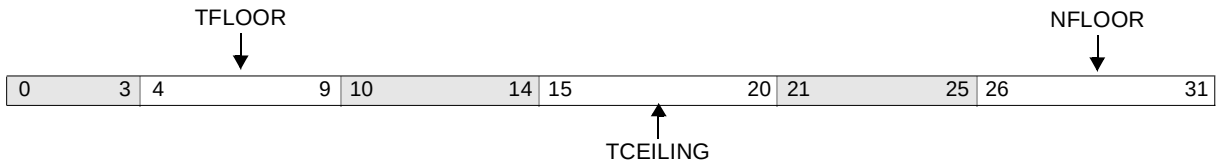


Figure 5-2. Instruction Cache Victim Limit (IVLIM) Data Cache Victim Limit (DVLIM)

0:3		Reserved	
4:9	TFLOOR	Transient Floor	
10:14		Reserved	
15:20	TCEILING	Transient Ceiling	
21:25		Reserved	
26:31	NFLOOR	Normal Floor	

When a cache line fill occurs as the result of a normal memory access (that is, one *not* marked as transient using the U1 storage attribute from the MMU; see ~~Chapter 6, "Memory Management"~~*Memory Management on page 233*), the cache line to be replaced is selected by the corresponding victim index field from one of the normal victim index registers (INV0–INV3 for instruction cache lines, DNV0–DNV3 for data cache lines). As the processor increments any of these normal victim index fields according to the round-robin mechanism

described in ~~"Cache Line Replacement Policy" on page 5-2~~*Cache Line Replacement Policy* on page 206, the values of the fields are constrained to lie within the range specified by the NFLOOR field of the corresponding victim limit register, and the last way of the cache. That is, when one of the normal victim index fields is incremented past the last way of the cache, it wraps back to the value of the NFLOOR field of the associated victim limit register.

Similarly, when a cache line fill occurs as the result of a transient memory access, the cache line to be replaced is selected by the corresponding victim index field from one of the transient victim index registers (ITV0–ITV3 for instruction cache lines, DTV0–DTV3 for data cache lines). As the processor core increments any of these transient victim index fields according to the round-robin replacement mechanism, the values of the fields are constrained to lie within the range specified by the TFLOOR and the TCEILING fields of the corresponding victim limit register. That is, when one of the transient victim index fields is incremented past the TCEILING value of the associated victim limit register, it wraps back to the value of the TFLOOR field of that victim limit register.

Given the operation of this mechanism, if both the NFLOOR and TFLOOR fields are set to 0, and the TCEILING is set to the index of the last way of the cache, then all cache line fills—both normal and transient—are permitted to use the entire cache, and nothing is locked. Alternatively, if both the NFLOOR and TFLOOR fields are set to values greater than 0, the lines in those ways of the cache whose indexes are between 0 and the lower of the two floor values are effectively *locked*, as no cache line fills (neither normal nor transient) will be allowed to replace the lines in those ways. Yet another example is when the TFLOOR is lower than the NFLOOR, and the TCEILING is lower than the last way of the cache. In this scenario, the ways between the TFLOOR and the NFLOOR contain only transient lines, while the ways between the NFLOOR and the TCEILING may contain either normal or transient lines, and the ways from the TCEILING to the last way of the cache contain only normal lines.

Programming Note: It is a programming error for software to program the TCEILING field to a value lower than that of the TFLOOR field. Furthermore, software must initialize each of the normal and transient victim index fields to values that are between the ranges designated by the respective victim limit fields, prior to performing any cacheable accesses intended to utilize these ranges.

In order to setup a locked area within the data cache, software must perform the following steps (the procedure for the instruction cache is similar, with **icbt** instructions substituting for **dcbt** instructions):

1. Execute **msync** and then **isync** to guarantee all previous cache operation have completed.
2. Mark all TLB entries associated with memory pages which are being used to perform the locking function as caching-inhibited. Leave the TLB entries associated with the memory pages containing the data which is to be locked into the data cache marked as cacheable, however.
3. Execute **msync** and then **isync** again, to cause the new TLB entry values to take effect.
4. Set both the NFLOOR and the TFLOOR values to the index of the first way which should be locked, and set the TCEILING value to the last way of the cache.
5. Set each of the normal and transient victim index fields to the same value as the NFLOOR and TFLOOR.
6. Execute **dcbt** instructions to the cache lines within the cacheable memory pages which contain the data which is to be locked in the data cache. The number of **dcbt** instructions executed to any given set should not exceed the number of ways which will exist in the locked region (otherwise not all of the lines will be able to be simultaneously locked in the data cache). Remember that when a series of **dcbt** instructions are executed to sequentially increasing addresses (with the address increment being the size of a cache

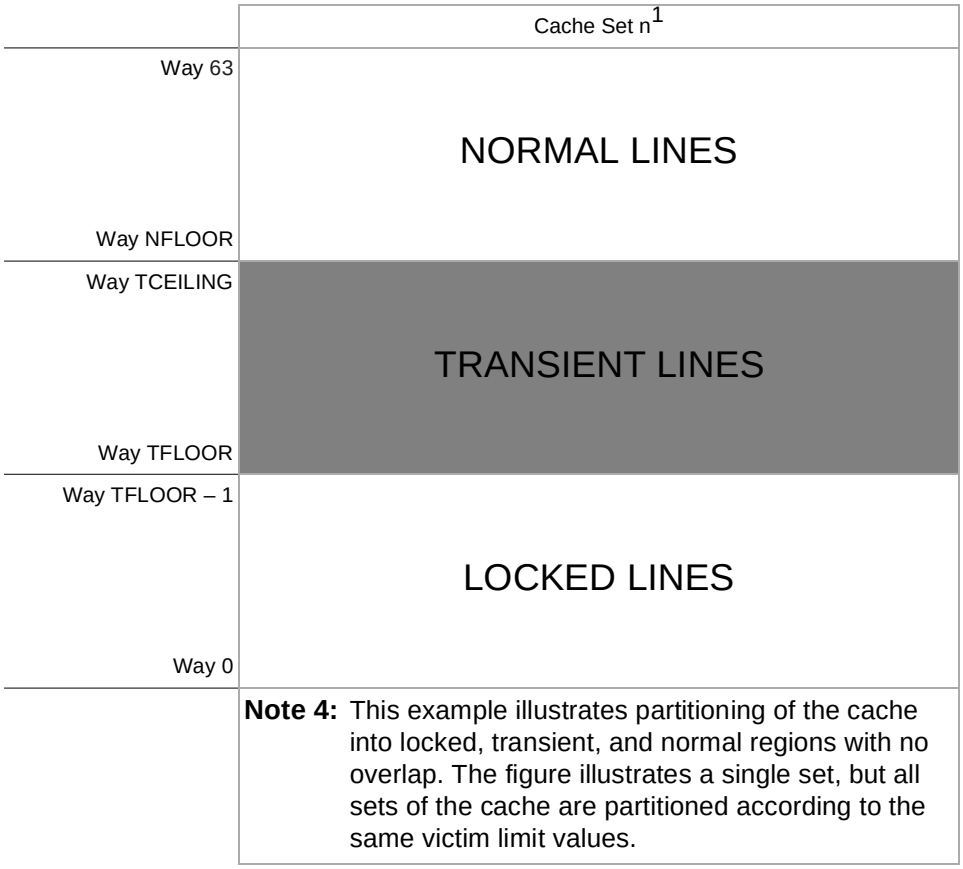
block -- 32 bytes), it takes sixteen such **dcbt** operations (one for each set) before the next way of the initial set will be targeted again.

7. Execute **msync** and then **isync** again, to guarantee that all of the **dcbt** operations have completed and updated the corresponding victim index fields.
8. Set the NFLOOR, TFLOOR, and TCEILING values to the desired indices for the operating normal and transient regions of the cache. Both the NFLOOR and the TFLOOR values should be set higher than the highest locked way of the data cache; otherwise, subsequent normal and/or transient accesses could overwrite a way containing a line which was to be locked.
9. Set each of the normal and transient victim index fields to the value of the NFLOOR and TFLOOR, respectively.
10. Restore the cacheability of the memory pages which were used to perform the locking function to the desired operating values, by clearing the caching-inhibited attribute of the TLB entries which were updated in step 2.
11. Execute **msync** and then **isync** again, to cause the new TLB entry values to take effect.

The ways of the data cache whose indices are below the lower of the NFLOOR and TFLOOR values will now be locked.

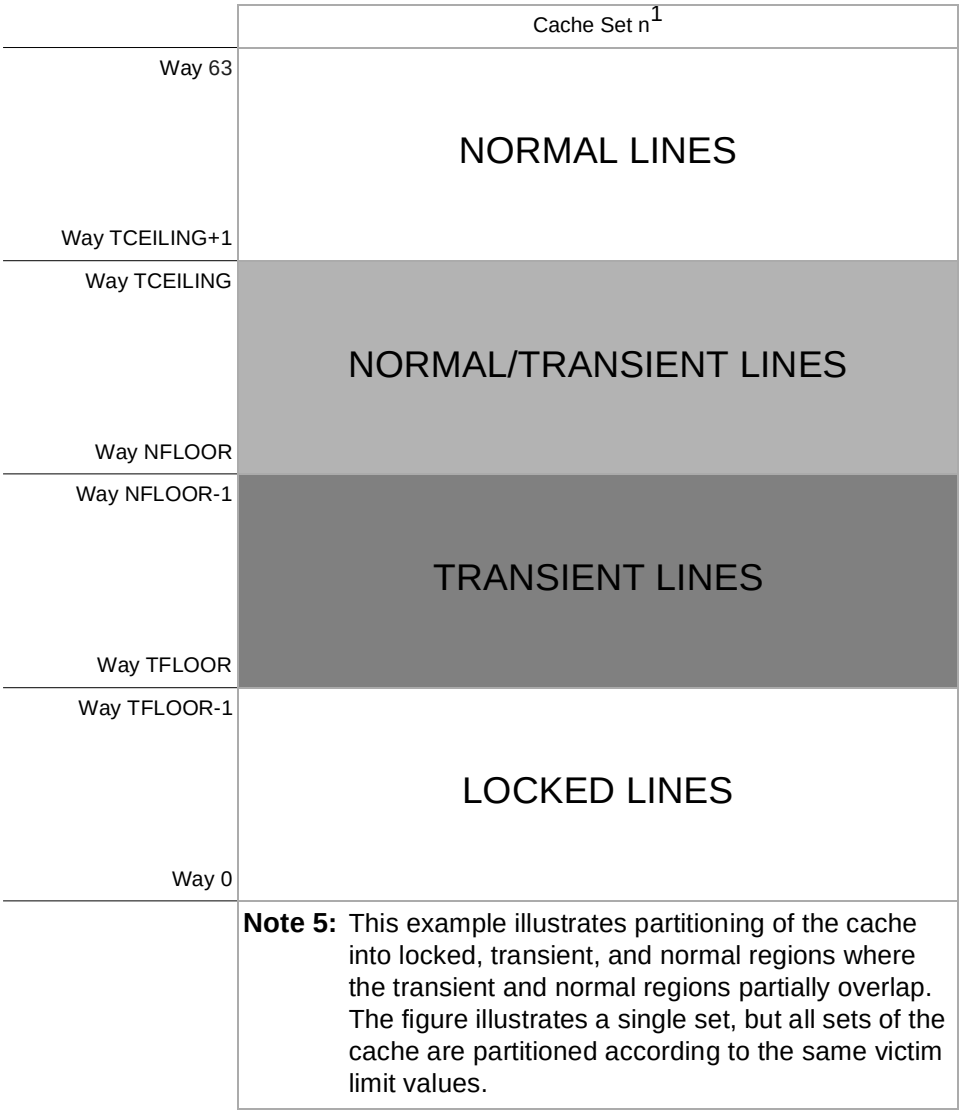
Figure 5-3 and Figure 5-4 illustrate two of these examples of the use of the locking and transient mechanisms. Other configurations are possible, given the ability to program each of the victim limit fields to different relative values, although some configurations are not necessarily useful or practical.

Figure 5-3. Cache Locking and Transient Mechanism (Example 1)[1]



Note 4: This example illustrates partitioning of the cache into locked, transient, and normal regions with no overlap. The figure illustrates a single set, but all sets of the cache are partitioned according to the same victim limit values.

Figure 5-4. Cache Locking and Transient Mechanism (Example 2)



Note 5: This example illustrates partitioning of the cache into locked, transient, and normal regions where the transient and normal regions partially overlap. The figure illustrates a single set, but all sets of the cache are partitioned according to the same victim limit values.

5.2 Instruction Cache Controller

The instruction cache controller (ICC) delivers two instructions per cycle to the instruction unit of the PPC440GP. The ICC interfaces to the PLB using a 128-bit read interface. The ICC handles frequency synchronization between the PPC440GP and the PLB, and can operate at any ratio of *n*:1, *n*:2, and *n*:3, where *n* is an integer greater than the corresponding denominator.

The ICC provides a speculative prefetch mechanism which can be configured to automatically prefetch a burst of up to three additional lines upon any fetch request which misses in the instruction cache.

The ICC also handles the execution of the PowerPC instruction cache management instructions, for touching (prefetching) or invalidating cache lines, or for flash invalidation of the entire cache. Resources for controlling and debugging the instruction cache operation are also provided.

The rest of this section describes each of these functions in more detail.

5.2.1 ICC Operations

When the ICC receives an instruction fetch request from the instruction unit of the PPC440GP, the ICC simultaneously searches the instruction cache array for the cache line associated with the virtual address of the fetch request, and translates the virtual address into a real address (see ~~Chapter 6, "Memory Management"~~*Memory Management* on page 233 for information about address translation). If the requested cache line is found in the array (a cache *hit*), the pair of instructions at the requested address are returned to the instruction unit. If the requested cache line is *not* found in the array (a cache *miss*), the ICC sends a request for the entire cache line (32 bytes) to the instruction PLB interface, using the real address. Note that the entire ~~cache~~ 32-bytecache line is requested, even if the caching inhibited (I) storage attribute is set for the memory page containing that cache line (see ~~"Caching Inhibited (I)" on page 6-14~~*Caching Inhibited (I)* on page 245). Also note that the request to the instruction PLB interface is sent using the specific instruction address requested by the instruction unit, so that the memory subsystem may read the cache line *target word first* and supply the requested instructions before retrieving the rest of the cache line.

As the ICC receives each portion of the cache line from the instruction PLB interface, it is placed into the instruction cache line fill data (ICLFD) buffer. Instructions from this buffer may be *bypassed* to the instruction unit as requested, without waiting for the entire cache line to be filled. Once the entire cache line has been filled into the buffer, and assuming that the memory page containing that line is cacheable, it is written into the instruction cache. If the memory page containing the line is caching inhibited, the line will remain in the ICLFD until it is displaced by a subsequent request for another cache line (either cachable or caching inhibited).

If a memory subsystem error (such as an address time-out, invalid address, or some other type of hardware error external to the PPC440GP) occurs during the filling of the cache line, the line will not be written into the instruction cache, although instructions from the line may still be forwarded to the instruction unit from the ICLFD. Later, if execution of an instruction from that line is attempted, an Instruction Machine Check exception will be reported, and a Machine Check interrupt (if enabled) will result. See ~~"Machine Check Interrupt" on page 10-19~~*Machine Check Interrupt* on page 354 for more information on Machine Check interrupts.

Once a request for a cache line read has been accepted by the instruction PLB interface and acknowledged on the PLB, the entire line read will be performed and the line will be written into the instruction cache (assuming no error occurs on the read), regardless of whether or not the instruction stream branches (or is interrupted) away from the line being read. This behavior is due to the nature of the PLB architecture, and the fact that once started, a cache line read request type cannot be abandoned. This does not mean, however, that the ICC will wait for this cache line read to complete before responding to a new request from the instruction unit (due, perhaps, to a branch redirection, or an interrupt). Instead, the ICC will immediately access the cache to determine if the cache line at the new address requested by the instruction unit is already in the cache. If so, the requested pair of instructions from this line will immediately be forwarded to the instruction unit, while the ICC in parallel continues to fill the previously requested cache line. In other words, the instruction cache is completely *non-blocking*.

If the newly requested cache line is instead a miss in the instruction cache, the ICC will immediately attempt to abort the previous cache line read request on the instruction PLB interface. If the previous cache line read request has not yet been acknowledged on the PLB bus, the old request will be aborted and the new request will be made. if the previous cache line read request has already been acknowledged, then as previously stated it cannot be abandoned, but the ICC will immediately present the request for the new cache line, such that it may be serviced immediately after the previous cache line read is completed.

Programming Note:

It is a programming error for an instruction fetch request to reference a valid cache line in the instruction cache if the caching inhibited storage attribute is set for the memory page containing the cache line. The result of attempting to execute an instruction from such an access is undefined. After processor reset, hardware automatically sets the caching inhibited storage attribute for the memory page containing the reset address, and also automatically flash invalidates the instruction cache. Subsequently, lines will not be placed into the instruction cache unless they are accessed by reference to a memory page for which the caching inhibited attribute has been turned off. If software subsequently turns on the caching inhibited storage attribute for such a page, software must make sure that no lines from that page remain valid in the instruction cache, before attempting to fetch and execute instructions from the (now caching inhibited) page.

5.2.2 Speculative Prefetch Mechanism

The ICC can be configured to automatically prefetch up to three more cache lines upon (in addition to the line being requested by the instruction unit) in response to a cache miss. This speculative prefetch only occurs on requests for lines from cacheable memory pages, and then only if enabled by the setting of certain fields in the Core Configuration Register 0 (CCR0) (see ~~Figure 5-5 on page 5-14~~*Figure 5-5* on page 216).

CCR0[ICSLC] specifies the number of additional cache lines (from 0 to 3) to speculatively prefetch upon an instruction cache miss. If this field is non-zero, upon an instruction cache miss, the ICC will first check the cache to see whether the additional lines are themselves already in the cache. If not, then the ICC will present a fixed-length burst request to the instruction PLB interface, requesting the additional cache line(s). The burst request is presented after the cache line request for the initial cache line requested by the instruction unit is presented and acknowledged on the PLB.

The speculative line fill mechanism will not request lines past the end of the minimum memory page size, which is 1KB. That is, if the line requested by the instruction unit is at or near the end of an aligned 1KB boundary, the speculative prefetch mechanism will only request those additional lines specified by the CCR0[ICSLC] field that are also within the same 1KB page of memory. This allows the speculative prefetch mechanism to operate without having to access the Memory Management Unit (MMU) for a translation for the next page address.

Another field in the CCR0 register, CCR0[ICSLT], specifies a *threshold* value that is used to determine whether the speculative burst request should be abandoned prior to completion, as a result of a change in direction in the instruction stream (such as a branch or interrupt). If the instruction unit requests a new cache line and the new request is a hit in the instruction cache, both the original line fill request and any speculative burst request associated with it will be unaffected. However, if the new cache line requested by the instruction unit is a miss in the instruction cache, any prior request which has not yet been acknowledged by the PLB will be abandoned. If a prior speculative burst request has already been acknowledged by the PLB, the value of CCR0[ICSLT] determines if and when the speculative burst request will be terminated. CCR0[ICSLT] specifies the number of *doublewords* (8-byte units) of the *current* cache line which must already have been received by the ICC, in order that the ~~burst~~ filling of the current cache line will *not* be ~~terminated~~terminated (note that in this context, the term "current" refers to the cache line with which the next PLB data transfer is associated, at the time that the ICC determines that it needs to request a new line). That is, if the ICC has already received the number of doublewords indicated by CCR0[ICSLT], the ICC will not terminate the burst until it has received that entire cache line. All additional lines beyond the one in progress at the time that the ICC determines that it needs to request a new line *will* be abandoned. For example, if CCR0[ICSLC] is set to 3, and the ICC is in the middle of receiving the data for the first of the three speculative lines at the time that the new instruction cache miss request is received from the instruction unit, the second and third lines of the speculative burst will be abandoned, and whether the first of the speculative lines is abandoned is controlled by CCR0[ICSLT].

Since cache lines contain 32 bytes, there are four doublewords in each cache line. Thus, CCR0[ICSLT] can be set to a value from 0 to 3. If CCR0[ICSLT] = 0, the current ~~speculative~~ line fill will be completed regardless of how many doublewords have already been received. Similarly, if CCR0[ICSLT] = 3, the current line fill will be abandoned if only two or fewer doublewords have been received by the ICC.

If at the time that the ICC determines that it needs to request a new line and abandon a speculative burst request, the ICC has still not received all of the data associated with the initial cache line request which prompted the speculative burst request, then this initial cache line is considered the "current" line, and the speculative burst request will be abandoned without filling *any* of the speculative lines, regardless of the setting of CCR0[ICSLT]. The filling of the initial cache line *will* be completed, however, as the PLB protocol does not provide for the abandonment of the cache line (non-burst) request type.

Regardless of the value of CCR0[ICSLT], any time that a cache line fill is abandoned such that all of the data for that cache line is not received, the line may still be used to bypass instructions to the instruction unit, but it will not be written into the instruction cache, and it will be overwritten in the ICLFD buffer as soon as instructions for a new line begin arriving from the PLB.

5.2.3 Instruction Cache Coherency

In general, the PPC440GP does not automatically enforce coherency between the instruction cache, data cache, and memory. If the contents of memory location are changed, either within the data cache or within memory itself, and whether by the PPC440GP through the execution of store instructions or by some other mechanism in the system writing to memory, software must use cache management instructions to ensure that the instruction cache is made coherent with these changes. This involves invalidating any obsolete copies of these memory locations within the instruction cache, so that they will be reread from memory the next time they are referenced by program execution.

5.2.3.1 Self-Modifying Code

To illustrate the use of the cache management instructions to enforce instruction cache coherency, consider the example of *self-modifying code*, whereby the program executing on the PPC440GP stores new data to memory, with the intention of later branching to and executing this new "data," which are actually instructions.

The following code example illustrates the required sequence for software to use when writing self-modifying code. This example assumes that *addr1* references a cacheable memory page.

```
stw     regN, addr1     # store the data (an instruction) in regN to addr1 in the data cache
dcbst   addr1           # write the new instruction from the data cache to memory
msync                   # wait until the data actually reaches the memory
icbi    addr1           # invalidate addr1 in the instruction cache if it exists
msync   addr1           # wait for the instruction cache invalidation to take effect
isync                   # flush any prefetched instructions within the ICC and instruction
                        # unit and re-fetch them (an older copy of the instruction at addr1
                        # may have already been fetched)
```

At this point, software may begin executing the instruction at addr1 and be guaranteed that the new instruction will be recognized.

5.2.3.2 Instruction Cache Synonyms

A synonym is a cache line that is associated with the same real address as another cache line that is in the cache array at the same time. Such synonyms can occur when different virtual addresses are mapped to the same real address, and the virtual address is used either as an index to the cache array (a *virtually-indexed* cache) or as the cache line tag (a *virtually-tagged* cache).

The instruction cache on the PPC440GP is real-indexed but virtually-tagged and thus it is possible for synonyms to exist in the cache. (The data cache on the other hand is both real-indexed and real-tagged, and thus cannot have any synonyms.) Because of this, special care must be taken when managing instruction cache coherency and attempting to invalidate lines in the cache.

As explained in ~~Chapter 6, "Memory Management"~~*Memory Management* on page 233, the virtual address (VA) consists of the 32-bit effective address (EA; for instruction fetches, this is the address calculated by the instruction unit and sent to the ICC) combined with the 8-bit Process ID (PID) and the 1-bit address space (MSR[IS] for instruction fetches). As described in ~~Table 5-2 on page 5-3~~*Table 5-2* on page 207, $VA_{27:31}$ chooses the byte offset within the cache line, while $VA_{23:26}$ is used as the *index* to select a set, and then the rest of the virtual address is used as the *tag*. The tag thus consists of $EA_{0:22}$, the PID, and MSR[IS] (for instruction fetches; for cache management instructions such as **icbi**, MSR[DS] is used to specify the address space; see the instruction descriptions for the instruction cache management instructions for more information). The tag portion of the VA is compared against the corresponding tag fields of each cache line within the way selected by $VA_{23:26}$.

Note that the address translation architecture of PowerPC Book-E is such that the low-order address bits 22:31 are always the same for the EA, VA, and real address (RA), because these bits are never translated due to the minimum page size being 1KB (these low-order 10 bits are always used for the byte offset within the page). As the page size increases, more and more low-order bits are used for the byte offset within the page, and thus fewer and fewer bits are translated between the VA and the RA (see ~~Table 6-3," on page 6-10~~*Table 6-3* on page 241). Synonyms only become possible when the system-level memory management software establishes multiple mappings to the same real page, which by definition involves different virtual addresses (either through differences in the higher-order EA bits which make up the VA, or through different process IDs, or different address spaces, or some combination of these three portions of the VA).

A further requirement for synonyms to exist in the instruction cache is for more than one of the virtual pages which map to a given real page to have *execute permission*, and for these pages to be cacheable (cache lines associated with pages without execute permission, or for which the caching inhibited storage attribute is set, cannot be placed in the instruction cache).

If the system-level memory management software permits instruction cache synonyms to be created, then extra care must be taken when attempting to invalidate instruction cache lines associated with a particular address. If software desires to invalidate only the cache line which is associated with a specific VA, then only a single **icbi** instruction need be executed, specifying that VA. If, however, software wishes to invalidate *all* instruction cache lines which are associated with a particular RA, then software must issue an **icbi** instruction for *each* VA which has a mapping to that particular RA and for which a line might exist in the instruction cache. In order to do this, the memory management software must keep track of which mappings to a given RA exist (or ever existed, if a mapping has been removed but cache lines associated with it might still exist), so that **icbi** instructions can be executed using the necessary VAs.

Alternatively, software can execute an **iccci** instruction, which flash invalidates the entire instruction cache without regard to the addresses with which the cache lines are associated.

5.2.4 Instruction Cache Control and Debug

The PPC440GP provides various registers and instructions to control instruction cache operation and to help debug instruction cache problems.

5.2.4.1 Instruction Cache Management and Debug Instruction Summary

For detailed descriptions of the instructions summarized in this section, see ~~Chapter 28, "Instruction Set."~~*Instruction Set* on page 893 Also, see ~~"Instruction Cache Coherency" on page 5-11~~*Instruction Cache Coherency* on page 214 for more information on how these instructions are used to manage coherency in the instruction cache.

In the instruction descriptions, the term "block" describes the unit of storage operated on by the cache block instructions. For the PPC440GP, this is the same as a cache line.

The following instructions are used by software to manage the instruction cache:

icbi Instruction Cache Block Invalidate

Invalidates a cache block.

icbt Instruction Cache Block Touch

Initiates a block fill, enabling a program to begin a cache block fetch before the program needs an instruction in the block. The program can subsequently branch to the instruction address and fetch the instruction without incurring a cache miss.

See ~~"icbt Operation" on page 5-15~~*icbt Operation* on page 218.

iccci Instruction Cache Congruence Class Invalidate

Flash invalidates the entire instruction cache. Execution of this instruction is privileged.

icread Instruction Cache Read

Reads a cache line (tag and data) from a specified index of the instruction cache, into a set of SPRs. Execution of this instruction is privileged.

See ~~"icread Operation" on page 5-15~~*icread Operation* on page 218.

5.2.4.2 Core Configuration Register 0 (CCR0)

The CCR0 register controls the speculative prefetch mechanism and the behavior of the **icbt** instruction. The CCR0 register also controls various other functions within the PPC440GP that are unrelated to the instruction cache. Each of the these functions is discussed in more detail in the related sections of this manual.

~~Figure 5-5~~Figure 5-5 illustrates the fields of the CCR0 register.

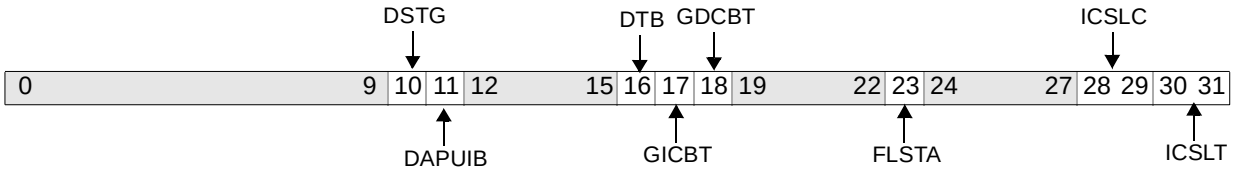


Figure 0-3. Core Configuration Register 0 (CCR0)

0:9		Reserved	
10	DSTG	Disable Store Gathering 0 Enabled; stores to contiguous addresses may be gathered into a single transfer 1 Disabled; all stores to memory will be performed independently	See "Store Gathering" on page 5-21.
11	DAPUIB	Disable APU Instruction Broadcast 0 Enabled. 1 Disabled; instructions not broadcast to APU for decoding	This mechanism is provided as a means of reducing power consumption when an auxilliary processor is not attached and/or is not being used. See Chapter 7, "Reset and Initialization".
12:15		Reserved	
16	DTB	Disable Trace Broadcast 0 Enabled. 1 Disabled; no trace information is broadcast.	This mechanism is provided as a means of reducing power consumption when instruction tracing is not needed. See Chapter 7, "Reset and Initialization".
17	GICBT	Guaranteed Instruction Cache Block Touch 0 **icbt** may be abandoned without having filled cache line if instruction pipeline stalls. 1 **icbt** is guaranteed to fill cache line even if instruction pipeline stalls.	See "icbt Operation" on page 5-15.
18	GDCBT	Guaranteed Data Cache Block Touch 0 **dcbt/dcbtst** may be abandoned without having filled cache line if instruction pipeline stalls. 1 **dcbt/dcbtst** are guaranteed to fill cache line even if instruction pipeline stalls.	See "Data Cache Control and Debug" on page 5-25.
19:22		Reserved	
23	FLSTA	Force Load/Store Alignment 0 No Alignment exception on integer storage access instructions, regardless of alignment 1 An alignment exception occurs on integer storage access instructions if data address is not on an operand boundary.	See "Load and Store Alignment" on page 5-19.
24:27		Reserved	
28:29	ICSLC	Instruction Cache Speculative Line Count	Number of additional lines (0–3) to fill on instruction fetch miss. See "Speculative Prefetch Mechanism" on page 5-10.

30:31	ICSLT	Instruction Cache Speculative Line Threshold	Number of doublewords that must have already been filled in order that the current speculative line fill is *not* abandoned on a redirection of the instruction stream. See "Speculative Prefetch Mechanism" on page 5-10.

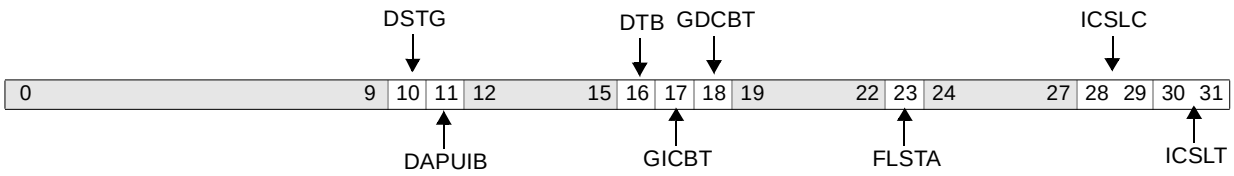


Figure 5-5. Core Configuration Register 0 (CCR0)

0:9		Reserved	
10	DSTG	Disable Store Gathering 0 Enabled; stores to contiguous addresses may be gathered into a single transfer 1 Disabled; all stores to memory will be performed independently	See "Store Gathering" on page -163.
11	DAPUIB	Disable APU Instruction Broadcast 0 Enabled. 1 Disabled; instructions not broadcast to APU for decoding	This mechanism is provided as a means of reducing power consumption when an auxilliary processor is not attached and/or is not being used. See *Reset and Initialization* on page 259.
12:15		Reserved	
16	DTB	Disable Trace Broadcast 0 Enabled. 1 Disabled; no trace information is broadcast.	This mechanism is provided as a means of reducing power consumption when instruction tracing is not needed. See *Reset and Initialization* on page 259.
17	GICBT	Guaranteed Instruction Cache Block Touch 0 **icbt** may be abandoned without having filled cache line if instruction pipeline stalls. 1 **icbt** is guaranteed to fill cache line even if instruction pipeline stalls.	See *icbt Operation* on page 158.
18	GDCBT	Guaranteed Data Cache Block Touch 0 **dcbt/dcbtst** may be abandoned without having filled cache line if load/store pipeline stalls. 1 **dcbt/dcbtst** are guaranteed to fill cache line even if load/store pipeline stalls.	See *Data Cache Control and Debug* on page 168.
19:22		Reserved	

23	FLSTA	Force Load/Store Alignment 0 No Alignment exception on integer storage access instructions, regardless of alignment 1 An alignment exception occurs on integer storage access instructions if data address is not on an operand boundary.	See *Load and Store Alignment* on page 162.
24:27		Reserved	
28:29	ICSLC	Instruction Cache Speculative Line Count	Number of additional lines (0–3) to fill on instruction fetch miss. See *Speculative Prefetch Mechanism* on page 153.
30:31	ICSLT	Instruction Cache Speculative Line Threshold	Number of doublewords that must have already been filled in order that the current speculative line fill is *not* abandoned on a redirection of the instruction stream. See *Speculative Prefetch Mechanism* on page 153.

5.2.4.3 icbt Operation

The **icbt** instruction is typically used as a "hint" to the processor that a particular block of instructions is likely to be executed in the near future. Thus the processor can begin filling that block into the instruction cache, so that when the executing program eventually branches there the instructions will already be present in the cache, thereby improving performance.

Of course, it would not typically be advantageous if the filling of the cache line requested by the **icbt** itself caused a delay in the fetching of instructions needed by the currently executing program. For this reason, the default behavior of the **icbt** instruction is for it to have the lowest priority for sending a request to the PLB. If a subsequent instruction cache miss occurs due to a request from the instruction unit, then the line fill for the **icbt** will be abandoned (if it has not already been acknowledged on the PLB).

On the other hand, the **icbt** instruction can also be used as a convenient mechanism for setting up a fixed, known environment within the instruction cache. This is useful for establishing contents for cache line locking, or for deterministic performance on a particular sequence of code, or even for debugging of low-level hardware and software problems.

When being used for these latter purposes, it is important that the **icbt** instruction deliver a deterministic result, namely the guaranteed establishment in the cache of the specified line. Accordingly, the PPC440GP provides a field in the CCR0 register that can be used to cause the **icbt** instruction to operate in this manner. Specifically, when the CCR0 [GICBT] field is set, the execution of **icbt** is *guaranteed* to establish the specified cache line in the instruction cache (assuming that a TLB entry for the referenced memory page exists and has both read and execute permission, and that the caching inhibited storage attribute is not set). The cache line fill associated with such a guaranteed **icbt** will not be abandoned due to subsequent instruction cache misses.

5.2.4.4 icread Operation

The **icread** instruction can be used to directly read both the tag and instruction information of a specified word in a specified entry of the instruction cache. The instruction information is read into the Instruction Cache Debug Data Register (ICDBDR), while the tag information is read into a pair of SPRs, the Instruction Cache Debug Tag Register High (ICDBTRH) and Instruction Cache Debug Tag Register Low (ICDBTRL). From there, the information can subsequently be moved into GPRs using **mfspr** instructions.

The execution of the **icread** instruction generates the equivalent of an EA, which is then broken down and used to select a specific instruction word from a specific cache line. $EA_{0:16}$ are ignored, $EA_{17:22}$ select the way, $EA_{23:26}$ select the set, and $EA_{27:29}$ select the word.

The EA generated by the **icread** instruction must be word-aligned (that is, $EA_{30:31}$ must be 0); otherwise, it is a programming error and the result is undefined.

Execution of the **icread** instruction is privileged, and is intended for use for debugging purposes only.

Programming Note:

The PPC440GP does not automatically synchronize context between an **icread** instruction and the subsequent **mfspr** instructions which read the results of the **icread** instruction into GPRs. In order to guarantee that the **mfspr** instructions obtain the results of the **icread** instruction, a sequence such as the following must be used:

```
icread      regA,regB      # read cache information (the contents of GPR A and GPR B are
                           # added and the result used to specify a cache line index to be read)
isync                      # ensure icread completes before attempting to read results
mficdbdr    regC           # move instruction information into GPR C
mficdbtrh   regD           # move high portion of tag into GPR D
mficdbtrl   regE           # move low portion of tag into GPR E
```

The following figures illustrate the ICDBDR, ICDBTRH, and ICDBTRL.

Figure 0-4. Instruction Cache Debug Data Register (ICDBDR)

0:31		Instruction machine code from instruction cache

Figure 5-6. Instruction Cache Debug Data Register (ICDBDR)

0:31		Instruction machine code from instruction cache

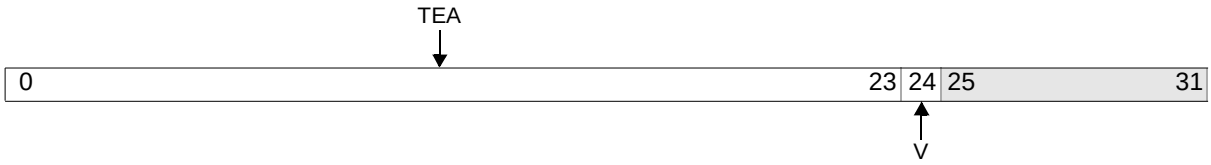


Figure 0-5. Instruction Cache Debug Tag Register High (ICDBTRH)

0:23		Tag Effective Address	Bits 0:23 of the 32-bit effective address associated with the cache line read by **icread**.
24	V	Cache Line Valid 0 Cache line is not valid. 1 Cache line is valid.	The valid indicator for the cache line read by **icread**.
25:31		Reserved	

Figure 5-7. Instruction Cache Debug Tag Register High (ICDBTRH)

0:23		Tag Effective Address	Bits 0:23 of the 32-bit effective address associated with the cache line read by **icread**.
24	V	Cache Line Valid 0 Cache line is not valid. 1 Cache line is valid.	The valid indicator for the cache line read by **icread**.
25:26	TPAR	Tag Parity	The parity bits for the address tag for the cache line read by **icread**, if CCR0[CRPE] is set.
27	DAPAR	Instruction Data parity	The parity bit for the instruction word at the 32-bit effective address specified in the **icread** instruction, if CCR0[CRPE] is set.
28:31		Reserved	

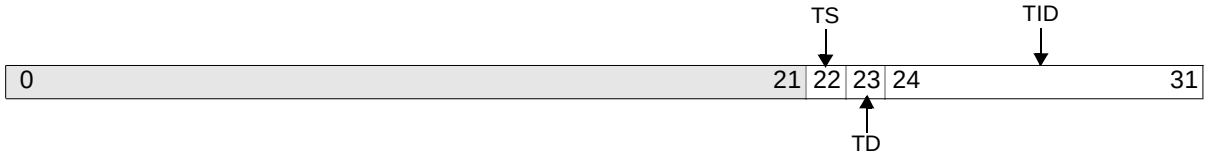


Figure 0-6. Instruction Cache Debug Tag Register Low (ICDBTRL)

0:21		Reserved	
22	TS	Translation Space	The address space portion of the virtual address associated with the cache line read by **icread**.
23	TD	Translation ID (TID) Disable 0 TID enable 1 TID disable	TID Disable field for the memory page associated with the cache line read by **icread**.
24:31	TID	Translation ID	TID field portion of the virtual address associated with the cache line read by **icread**.

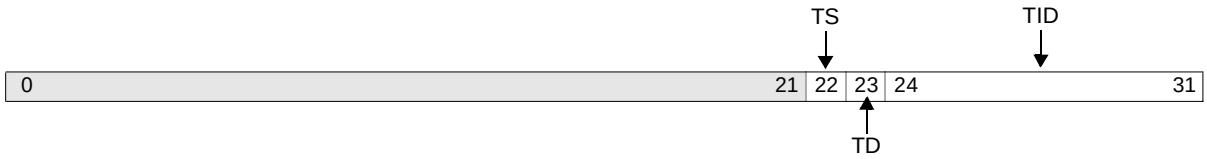


Figure 5-8. Instruction Cache Debug Tag Register Low (ICDBTRL)

0:21		Reserved	
22	TS	Translation Space	The address space portion of the virtual address associated with the cache line read by **icread**.
23	TD	Translation ID (TID) Disable 0 TID enable 1 TID disable	TID Disable field for the memory page associated with the cache line read by **icread**.
24:31	TID	Translation ID	TID field portion of the virtual address associated with the cache line read by **icread**.

5.3 Data Cache Controller

The data cache controller (DCC) handles the execution of the storage access instructions, moving data between memory, the data cache, and the PPC440GP GPR file. The DCC interfaces to the PLB using two independent 128-bit interfaces, one for read operations and one for writes. The DCC handles frequency synchronization between the PPC440GP and the PLB, and can operate at any ratio of *n*:1, *n*:2, and *n*:3, where *n* is an integer greater than the corresponding denominator.

The DCC also handles the execution of the PowerPC data cache management instructions, for touching (prefetching), flushing, invalidating, or zeroing cache lines, or for flash invalidation of the entire cache. Resources for controlling and debugging the data cache operation are also provided.

Extensive load, store, and flush queues are also provided, such that up to three outstanding line fills, up to four outstanding load misses, and up to two outstanding line flushes can be pending, with the DCC continuing to service subsequent load and store hits in an out-of-order fashion.

The rest of this section describes each of these functions in more detail.

5.3.1 DCC Operations

When the DCC executes a load, store, or data cache management instruction, the DCC first translates the effective address specified by the instruction into a real address (see ~~Chapter 6, "Memory Management"~~*Memory Management* on page 233 for more information on address translation). Next, the DCC searches the data cache array for the cache line associated with the real address of the requested data. If the cache line is found in the array (a cache *hit*), that cache line is used to satisfy the request, according to the type of operation (load, store, and so on).

If the cache line is *not* found in the array (a cache *miss*), the next action depends upon the type of instruction being executed, as well as the storage attributes of the memory page containing the data being accessed. For most operations, and assuming the memory page is cacheable (see ~~"Caching Inhibited (I)" on page 6-14~~*Caching Inhibited (I)* on page 245), the DCC will send a request for the entire cache line (32 bytes) to the data read PLB interface. The request to the data read PLB interface is sent using the specific byte address requested by the instruction, so that the memory subsystem may read the cache line *target word first* (if it supports such operation) and supply the specific byte[s] requested before retrieving the rest of the cache line.

While the DCC is waiting for a cache line read to complete, it can continue to process subsequent instructions, and handle those accesses that hit in the data cache. That is, the data cache is completely *non-blocking*.

As the DCC receives each portion of the cache line from the data read PLB interface, it is placed into one of three data cache line fill data (DCLFD) buffers. Data from these buffers may be *bypassed* to the GPR file to satisfy load instructions, without waiting for the entire cache line to be filled. Once the entire cache line has been filled into the buffer, it will be written into the data cache at the first opportunity (either when the data cache is otherwise idle, or when subsequent operations require that the DCLFD buffer be written to the data cache).

If a memory subsystem error (such as an address time-out, invalid address, or some other type of hardware error external to the PPC440GP) occurs during the filling of the cache line, the line will still be written into the data cache, and data from the line may still be delivered to the GPR file for load instructions. However, the

DCC will also report a Data Machine Check exception to the instruction unit of the PPC440GP, and a Machine Check interrupt (if enabled) will result. See ~~"Machine Check Interrupt" on page 10-19~~*Machine Check Interrupt* on page 354 for more information on Machine Check interrupts.

Once a data cache line read request has been made, the entire line read will be performed and the line will be written into the data cache, regardless of whether or not the instruction stream branches (or is interrupted) away from the instruction which prompted the initial line read request. That is, if a data cache line read is initiated speculatively, before knowing whether or not a given instruction execution is really required (for example, on a load instruction which is after an unresolved branch), that line read will be completed, even if it is later determined that the cache line is not really needed. The DCC never aborts any PLB request once it has been made.

In general, the DCC will initiate memory read requests without waiting to determine whether the access is actually required by the *sequential execution model (SEM)*. That is, the request will be initiated *speculatively*, even if the instruction causing the request might be abandoned due to a branch, interrupt, or other change in the instruction flow. Of course, write requests to memory cannot be initiated speculatively, although a line fill request in response to a cacheable store access which misses in the data cache could be.

On the other hand, if the guarded storage attribute is set for the memory page being accessed, then the memory request will *not* be initiated until it is guaranteed that the access is required by the SEM. Once initiated, the access will not be abandoned, and the instruction is guaranteed to complete, *prior* to any change in the instruction stream. That is, if the instruction stream is interrupted, then upon return the instruction execution will resume *after* the instruction which accessed guarded storage, such that the guarded storage access will *not* be re-executed.

See ~~"Guarded (G)" on page 6-15~~*Guarded (G)* on page 246 for more information on accessing guarded storage.

Programming Note:

It is a programming error for a load, store, or **dcbz** instruction to reference a valid cache line in the data cache if the caching inhibited storage attribute is set for the memory page containing the cache line. The result of such an access is undefined. After processor reset, hardware automatically sets the caching inhibited storage attribute for the memory page containing the reset address, and software should flash invalidate the data cache (using **dccci**; see ~~"Data Cache Management and Debug Instruction Summary" on page 5-25~~*Data Cache Management and Debug Instruction Summary* on page 228) before executing any load, store, or **dcbz** instructions. Subsequently, lines will not be placed into the data cache unless they are accessed by reference to a memory page for which the caching inhibited attribute has been turned off. If software subsequently turns on the caching inhibited storage attribute for such a page, software must make sure that no lines from that page remain valid in the data cache (typically by using the **dcbf** instruction), before attempting to access the (now caching inhibited) page with load, store, or **dcbz** instructions.

The only instructions that are permitted to reference a caching inhibited line which is a hit in the data cache are the cache management instructions **dcbst**, **dcbf**, **dcbi**, **dccci**, and **dcread**. The **dcbt** and **dcbtst** instructions have no effect if they reference a caching inhibited address, regardless of whether the line exists in the data cache.

5.3.1.1 Load and Store Alignment

The DCC implements all of the integer load and store instructions defined for 32-bit implementations by the PowerPC Book-E architecture. These include byte, halfword, and word loads and stores, as well as load and store string (0 to 127 bytes) and load and store multiple (1 to 32 registers) instructions. Integer byte, halfword, and word loads and stores are performed with a single access to memory if the entire data operand is contained within an aligned 16-byte (quadword) block of memory, regardless of the actual operand alignment within that block. If the data operand crosses a quadword boundary, the load or store is performed using two accesses to memory.

The load and store string and multiple instructions are performed using one memory access for each four bytes, unless and until an access would cross an aligned quadword boundary. The access that would cross the boundary is shortened to access just the number of bytes left within the current quadword block, and then the accesses are resumed with four bytes per access, starting at the beginning of the next quadword block, until the end of the load or store string or multiple is reached.

The DCC handles all misaligned integer load and store accesses in hardware, without causing an Alignment exception. However, the control bit CCR0[FLSTA] can be set to force all misaligned storage access instructions to cause an Alignment exception (see ~~Figure 5-5 on page 5-14~~*Figure 5-5* on page 216). When this bit is set, all integer storage accesses must be aligned on an operand-size boundary, or an Alignment exception will result. Load and store multiple instructions must be aligned on a 4-byte boundary, while load and store string instructions can be aligned on any boundary (these instructions are considered to reference *byte* strings, and hence the operand size is a byte).

5.3.1.2 Load Operations

Load instructions that reference cacheable memory pages and miss in the data cache result in cache line read requests being presented to the data read PLB interface. Load operations to caching inhibited memory pages, however, will only access the bytes specifically requested, according to the type of load instruction. This behavior (of only accessing the requested bytes) is only architecturally required when the guarded storage attribute is also set, but the DCC will enforce this requirement on any load to a caching inhibited memory page. Subsequent load operations to the same caching inhibited locations will cause new requests to be sent to the data read PLB interface (data from caching inhibited locations will not be reused from the DCLFD buffer).

The DCC includes three DCLFD buffers, such that a total of three independent data cache line fill requests can be in progress at one time. The DCC can continue to process subsequent load and store accesses while these line fills are in progress.

The DCC also includes a 4-entry *load miss queue (LMQ)*, which holds up to four outstanding load instructions that have either missed in the data cache or access caching inhibited memory pages. Collectively, any LMQ entries which reference cacheable memory pages can reference no more than three different cache lines, since there are only three DCLFD buffers. A load instruction in the LMQ remains there until the requested data arrives in the DCLFD buffer, at which time the data is delivered to the register file and the instruction is removed from the LMQ.

5.3.1.3 Store Operations

The processing of store instructions in the DCC is affected by several factors, including the caching inhibited (I), write-through (W), and guarded (G) storage attributes, as well as whether or not the allocation of data cache lines is enabled for cacheable store misses. There are three different behaviors to consider:

- Whether a data cache line is allocated (if the line is not already in the data cache)
- Whether the data is written directly to memory or only into the data cache
- Whether the store data can be *gathered* with store data from previous or subsequent store instructions before being written to memory

Allocation of Data Cache Line on Store Miss

Of course, if the caching inhibited attribute is set for the memory page being referenced by the store instruction, no data cache line will be allocated. For cacheable store accesses, allocation is controlled by one of two mechanisms: either by a "global" control bit in the Memory Management Unit Control Register (MMUCR), which is applied to all cacheable store accesses regardless of address; or by the U2 storage attribute for the memory page being accessed. See ~~"Memory Management Unit Control Register (MMUCR)" on page 6-16~~*Memory Management Unit Control Register (MMUCR)* on page 247 for more information on how store miss cache line allocation is controlled.

Regardless of which mechanism is controlling the allocation, if the corresponding bit is set, the cacheable store miss is handled as a *store without allocate (SWOA)*. That is, if SWOA is indicated, then if the access misses in the data cache, then the line will not be allocated (read from memory), and instead the byte[s] being stored will be written directly to memory. Of course, if the cache line has already been allocated and is being read into a DCLFD buffer (due perhaps to a previous cacheable load access), then the SWOA indication is ignored and the access is treated as if it were a store *with* allocate. Similarly, if SWOA is *not* indicated, the cache line *will* be allocated and the cacheable store miss will result in the cache line being read from memory.

Direct Write to Memory

Of course, if the caching inhibited attribute is set for the memory page being referenced by the store instruction, the data must be written directly to memory. For cacheable store accesses that are also write-through, the store data will also be written directly to memory, regardless of whether the access hits in the data cache, and independent of the SWOA mechanism. For cacheable store accesses that are *not* write through, whether the data is written directly to memory depends on both whether the access hits or misses in the data cache, and the SWOA mechanism. If the access is *either* a hit in the data cache, or if SWOA is *not* indicated, then the data will only be written to the data cache, and not to memory. Conversely, if the cacheable store access is both a miss in the data cache and SWOA is indicated, the access will be treated as if it were caching inhibited and the data will be written directly to memory and not to the data cache (since the data cache line is neither there already ~~and~~ nor will ~~not~~ it be allocated).

Store Gathering

In general, ~~accesses~~ memory write operations caused by separate store instructions that specify locations in either write-through or caching inhibited ~~storage, and which are contiguous and non-overlapping and within the same quadword,~~ storage may be gathered into one simultaneous access to memory. Similarly, store accesses that are handled as if they were caching inhibited (due to their being both a miss in the data cache and being indicated as SWOA) may be gathered. Store accesses that are only written into the data cache do not need to be gathered, because there is no performance penalty associated with the separate accesses to the array.

A given sequence of two store operations may only be gathered together if the targeted bytes are contained within the same aligned quadword of memory, and if they are contiguous with respect to each other. Subsequent store operations may continue to be gathered with the previously gathered sequence, subject to the same two rules (same aligned quadword and contiguous with the collection of previously gathered bytes). For

example, a sequence of three store word operations to addresses 4, 8, and 0 may all be gathered together, as the first two are contiguous with each other, and the third (store word to address 0) is contiguous with the gathered combination of the previous two.

An additional requirement for store gathering applies to stores which target caching inhibited memory pages. Specifically, a given store to a caching inhibited page can only be gathered with previous store operations if the bytes targeted by the given store do not *overlap* with any of the previously gathered bytes. In other words, a store to a caching inhibited page must be both *contiguous* and *non-overlapping* with the previous store operation(s) with which it is being gathered. This ensures that the multiple write operations associated with a sequence of store instructions which each target a common caching inhibited location will each be performed independently on that target location.

~~Store gathering does~~ Finally, a given store operation will *not* ~~occur~~ be gathered with an earlier store operation if it is separated from the earlier store ~~instructions are separated~~ operation by an **msync** or an **mbar** instruction, ~~nor~~ or if ~~a~~ either of the two store ~~instruction references~~ operations reference a memory page which is both guarded ~~memory page~~and caching inhibited, ~~nor~~ or if ~~such~~ store gathering is disabled altogether by CCR0[DSTG] (see ~~Figure 5-5 on page 5-14~~*Figure 5-5* on page 216).

~~Table 5-3~~*Table 5-3* summarizes how the various storage attributes and other circumstances affect the DCC behavior on store accesses.

Table 5-3. Data Cache Behavior on Store Accesses

Store Access Attributes					DCC Actions		
Caching Inhibited (I)	Hit/Miss	SWOA	Write through (W)	Guarded (G)	Write Cache?	Write Memory?	Gather?[1]
0	Hit	—	0	—	Yes	No	N/A
0	Hit	—	1	0	Yes	Yes	Yes
0	Hit	—	1	1	Yes	Yes	No
0	Miss	0	0	—	Yes	No	N/A
0	Miss	0	1	0	Yes	Yes	Yes
0	Miss	0	1	1	Yes	Yes	No
0	Miss	1	—	0	No	Yes	Yes
0	Miss	1	—	1	No	Yes	No
1	—~~1~~[2]	—	—~~2~~[3]	0	No	Yes	~~Yes~~Yes[4]
1	—~~1~~[2]	—	—~~2~~[3]	1	No	Yes	No

Note 1: If store gathering is disabled altogether (by setting CCR0[DSTG] to 1), then such gathering will not occur, regardless of the indication in this table. Furthermore, where this table indicates that store gathering may occur it is presumed that the operations being gathered are targeting the same aligned quadword of memory, and are contiguous with respect to each other.

Note 2: It is a programming error for a data cache hit to occur on a store access to a caching inhibited page. The result of such an access is undefined.

Note 3: It is programming error for the write-through storage attribute to be set for a page which also has the caching inhibited storage attribute set. The result of an access to such a page is undefined.

Note 4:

Note 5: Stores to caching inhibited memory locations may only be gathered with previous store operations if none of the targeted bytes overlap with the bytes targeted by the previous store operations.

5.3.1.4 Line Flush Operations

When a store operation (or the **dcbz** instruction) writes data into the data cache without also writing the data to main memory, the cache line is said to become *dirty*, meaning that the data in the cache is the current value, whereas the value in memory is obsolete. Of course, when such a dirty cache line is replaced (due, for example, to a new cache line fill overwriting the existing line in the cache), the data in the cache line must be copied to memory. Otherwise, the results of the previous store operation[s] that caused the cache line to be marked as dirty would be lost. The operation of copying a dirty cache line to memory is referred to as a *cache line flush*. Cache lines are flushed either due to being replaced when a new cache line is filled, or in response to an explicit software flush request associated with the execution of a **dcbst** or **dcbf** instruction.

The DCC implements four dirty bits per cache line, one for each aligned doubleword within the cache line. Whenever any byte of a given doubleword is stored into a data cache line without also writing that same byte to memory, the corresponding dirty bit for that cache line is set. When a data cache line is flushed, the type of request made to the data write PLB interface depends upon which dirty bits associated with the line are set. If only one dirty bit is set, the request type will be for a single doubleword write. If only two dirty bits are set, and they are in the same quadword, then the request type will be for a 16-byte line write. If two or more dirty bits are set, and they are in different quadwords, the request type will be for an entire 32-byte line write. Regardless of the type of request generated by a cache line flush, the address is always specified as the first byte of the request.

If a store access occurs to a cache line in a memory page for which the write-through storage attribute is set, the dirty bits for that cache line do not get updated, since such a store access will be written directly to memory (and into the data cache as well, if the access is either a hit or if the cache line is allocated upon a miss).

On the other hand, it is permissible for there to exist multiple TLB entries that map to the same real memory page, but specify different values for the write-through storage attribute. In this case, it is possible for a store operation to a virtual page which is marked as non-write-through to have caused the cache line to be marked as dirty, so that a subsequent store operation to a different virtual page mapped to the same real page but marked as write-through encounters a dirty line in the data cache. If this happens, the store to the write-through page will write the data for the store to both the data cache and to memory, but it will not modify the dirty bits for the cache line.

5.3.1.5 Data Read PLB Interface Requests

When a PLB read request results from an access to a cacheable memory location, the request is always for a 32-byte line read, regardless of the type and size of the access that prompted the request. The address presented will be for the first byte of the target of the access.

On the other hand, when a PLB read request results from an access to a caching-inhibited memory location, only the byte[s] specifically accessed will be requested from the PLB, according to the type of instruction prompting the access. The following types of PLB read requests can occur due to caching inhibited requests:

- 1-byte read (any byte address 0–15 within a quadword)
- 2-byte read (any byte address 0–14 within a quadword)
- 3-byte read (any byte address 0–13 within a quadword)
- 4-byte read (any byte address 0–12 within a quadword)
- 8-byte read (any byte address 0–8 within a quadword)
- 16-byte line fill (must be for byte address 0 of a quadword)

5.3.1.6 Data Write PLB Interface Requests

When a PLB write request results from a data cache line flush, the specific type and size of the request is as described in ~~"Line Flush Operations" on page 5-21~~*Line Flush Operations* on page 225.

When a PLB write request results from store operations to caching-inhibited, write-through, and/or store without allocate (SWOA) memory locations, the type and size of the request can be any one of the following (this list includes the possible effects of store gathering; see ~~"Store Gathering" on page 5-21~~*Store Gathering* on page 223):

- 1-byte write request (any byte address 0–15 within a quadword)
- 2-byte write request (any byte address 0–14 within a quadword)
- 3-byte write request (any byte address 0–13 within a quadword)
- 4-byte write request (any byte address 0–12 within a quadword)
- 5-byte write request (any byte address 0–11 within a quadword)

  Only possible due to store gathering
- 6-byte write request (any byte address 0–10 within a quadword)

  Only possible due to store gathering
- 7-byte write request (any byte address 0–9 within a quadword)

  Only possible due to store gathering
- 8-byte write request (any byte address 0–8 within a quadword)

  Only possible due to store gathering
- 9-byte write request (any byte address 0–7 within a quadword)

  Only possible due to store gathering
- 10-byte write request (any byte address 0–6 within a quadword)

  Only possible due to store gathering
- 11-byte write request (any byte address 0–5 within a quadword)

  Only possible due to store gathering
- 12-byte write request (any byte address 0–4 within a quadword)

  Only possible due to store gathering
- 13-byte write request (any byte address 0–3 within a quadword)

  Only possible due to store gathering
- 14-byte write request (any byte address 0–2 within a quadword)

  Only possible due to store gathering
- 15-byte write request (any byte address 0–1 within a quadword)

  Only possible due to store gathering
- 16-byte line write request (must be to byte address 0 of a quadword)

  Only possible due to store gathering

5.3.1.7 Storage Access Ordering

In general, the DCC can perform load and store operations *out-of-order* with respect to the instruction stream. That is, the memory accesses associated with a sequence of load and store instructions may be performed in memory in an order different from that implied by the order of the instructions. For example, loads can be processed ahead of earlier stores, or stores can be processed ahead of earlier loads. Also, later loads and stores that hit in the data cache may be processed before earlier loads and stores that miss in the data cache.

The DCC does enforce the requirements of the SEM, such that the net result of a sequence of load and store operations is the same as that implied by the order of the instructions. This means, for example, that if a later load reads the same address written by an earlier store, the DCC guarantees that the load will use the data written by the store, and not the older "pre-store" data. But the memory subsystem could still see a read access associated with an even later load before it sees the write access associated with the earlier store.

If the DCC needs to make a read request to the data read PLB interface, and this request conflicts with (that is, references one or more of the same bytes as) an earlier write request which is being made to the data write PLB interface, the DCC will withhold the read request from the data read PLB interface until the write request has been acknowledged on the data write PLB interface. Once the earlier write request has been acknowledged, the read request will be presented, and it is the responsibility of the PLB subsystem to ensure that the data returned for the read request reflects the value of the data written by the write operation.

Conversely, if a write request conflicts with an earlier read request, the DCC will withhold the write request until the read request has been acknowledged, at which point it is the responsibility of the PLB subsystem to ensure that the data returned for the read request does *not* reflect the newer data being written by the write request.

The PPC440GP provides storage synchronization instructions to enable software to control the order in which the memory accesses associated with a sequence of instructions are performed. See ~~"Storage Ordering and Synchronization" on page 4-64~~*Storage Ordering and Synchronization* on page 231 for more information on the use of these instructions.

5.3.2 Data Cache Coherency

The PPC440GP does not enforce the coherency of the data cache with respect to alterations of memory performed by entities other than the PPC440GP. Similarly, if entities other than the PPC440GP attempt to read memory locations which currently exist within the PPC440GP data cache and in a modified state, the PPC440GP does not recognize such accesses and thus will not respond to such accesses with the modified data from the cache. In other words, the data cache on the PPC440GP is not a *snooping* data cache, and there is no hardware enforcement of data cache coherency with memory with respect to other entities in the system which access memory.

It is the responsibility of software to manage this coherency through the appropriate use of the caching inhibited storage attribute, the write-through storage attribute, and/or the data cache management instructions.

5.3.3 Data Cache Control and Debug

The PPC440GP provides various registers and instructions to control data cache operation and to help debug data cache problems.

5.3.3.1 Data Cache Management and Debug Instruction Summary

For detailed descriptions of the instructions summarized in this section, see ~~Chapter 28, "Instruction Set."~~*Instruction Set* on page 893

In the instruction descriptions, the term "block" describes the unit of storage operated on by the cache block instructions. For the PPC440GP, this is the same as a cache line.

The following instructions are used by software to manage the data cache.

dcba	Data Cache Block Allocate This instruction is implemented as a nop on the PPC440GP.
dcbf	Data Cache Block Flush Writes a cache block to memory (if the block has been modified) and then invalidates the block.
dcbi	Data Cache Block Invalidate Invalidates a cache block. Any modified data is discarded and not flushed to memory. Execution of this instruction is privileged.
dcbst	Data Cache Block Store Writes a cache block to memory (if the block has been modified) and leaves the block valid but marked as unmodified.
dcbt	Data Cache Block Touch Initiates a cache block fill, enabling the fill to begin prior to the executing program requiring any data in the block. The program can subsequently access the data in the block without incurring a cache miss.
dcbtst	Data Cache Block Touch for Store Implemented identically to the **dcbt** instruction.
dcbz	Data Cache Block Set to Zero Establishes a cache line in the data cache and sets the line to all zeros, without first reading the previous contents of the cache block from memory, thereby improving performance. All four doublewords in the line are marked as dirty.

dccci	Data Cache Congruence Class Invalidate Flash invalidates the entire data cache. Execution of this instruction is privileged.
dcread	Data Cache Read Reads a cache line (tag and data) from a specified index of the ~~instruction~~ data cache, into a GPR and a pair of SPRs. Execution of this instruction is privileged. See ~~"dcread Operation" on page 5-27~~*dcread Operation* on page 231.

5.3.3.2 Core Configuration Register 0 (CCR0)

The CCR0 register controls the behavior of the **dcbt** instruction, the handling of misaligned memory accesses, and the store gathering mechanism. The CCR0 register also controls various other functions within the PPC440GP that are unrelated to the data cache. Each of these functions is discussed in more detail in the related sections of this manual.

~~Figure 5-5 on page 5-14~~*Figure 5-5* on page 216 illustrates the fields of the CCR0 register.

5.3.3.3 dcbt and dcbtst Operation

The **dcbt** instruction is typically used as a "hint" to the processor that a particular block of data is likely to be referenced by the executing program in the near future. Thus the processor can begin filling that block into the data cache, so that when the executing program eventually performs a load from the block it will already be present in the cache, thereby improving performance.

The **dcbtst** instruction is typically used for a similar purpose, but specifically for cases where the executing program is likely to *store* to the referenced block in the near future. The differentiation in the purpose of the **dcbtst** instruction relative to the **dcbt** instruction is only relevant within shared-memory systems with hardware-enforced support for cache coherency. In such systems, the **dcbtst** instruction would attempt to establish the block within the data cache in such a fashion that the processor would most readily be able to subsequently write to the block (for example, in a processor with a *MESI-protocol* cache subsystem, the block might be obtained in *Exclusive* state). However, because the PPC440GP does not provide support for hardware-enforced cache coherency, the **dcbtst** instruction is handled in an identical fashion to the **dcbt** instruction. The rest of this section thus makes reference only to the **dcbt** instruction, but in all cases the information applies to **dcbtst** as well.

Of course, it would not typically be advantageous if the filling of the cache line requested by the **dcbt** itself caused a delay in the reading of data needed by the currently executing program. For this reason, the default behavior of the **dcbt** instruction is for it to be ignored if the filling of the requested cache block cannot be immediately commenced and waiting for such commencement would result in the DCC execution pipeline being stalled. For example, the **dcbt** instruction will be ignored if all three DCLFD buffers are already in use, and execution of subsequent storage access instructions is pending.

On the other hand, the **dcbt** instruction can also be used as a convenient mechanism for setting up a fixed, known environment within the data cache. This is useful for establishing contents for cache line locking, or for deterministic performance on a particular sequence of code, or even for debugging of low-level hardware and software problems.

When being used for these latter purposes, it is important that the **dcbt** instruction deliver a deterministic result, namely the guaranteed establishment in the cache of the specified line. Accordingly, the PPC440GP provides a field in the CCR0 register which can be used to cause the **dcbt** instruction to operate in this manner. Specifically, when the CCR0 [GDCBT] field is set, the execution of **dcbt** is *guaranteed* to establish the specified cache line in the data cache (assuming that a TLB entry for the referenced memory page exists and has read permission, and that the caching inhibited storage attribute is not set). The cache line fill associated with such a guaranteed **dcbt** will occur regardless of any potential instruction execution-stalling circumstances within the DCC.

5.3.3.4 dcread Operation

The **dcread** instruction can be used to directly read both the tag information and a specified data word in a specified entry of the data cache. The data word is read into the target GPR specified in the instruction encoding, while the tag information is read into a pair of SPRs, Data Cache Debug Tag Register High (DCDBTRH) and Data Cache Debug Tag Register Low (DCDBTRL). The tag information can subsequently be moved into GPRs using **mfspr** instructions.

The execution of the **dcread** instruction generates the equivalent of an EA, which is then broken down and used to select a specific ~~instruction~~ data word from a specific cache line. $EA_{0:16}$ are ignored, $EA_{17:22}$ select the way, $EA_{23:26}$ select the set, and $EA_{27:29}$ select the word.

The EA generated by the **dcread** instruction must be word-aligned (that is, $EA_{30:31}$ must be 0); otherwise, it is a programming error and the result is undefined.

Execution of the **dcread** instruction is privileged, and is intended for use for debugging purposes only.

Programming Note:

The PPC440GP does not support the use of the **dcread** instruction when the DCC is still in the process of performing cache operations associated with previously executed instructions (such as line fills and line flushes). Also, the PPC440GP does not automatically synchronize context between a **dcread** instruction and the subsequent **mfspr** instructions that read the results of the **dcread** instruction into GPRs. In order to guarantee that the **dcread** instruction operates correctly, and that the **mfspr** instructions obtain the results of the **dcread** instruction, a sequence such as the following must be used:

```
msync                   # ensure that all previous cache operations have completed
dcread    regT,regA,regB# read cache information; the contents of GPR A and GPR B are
                        # added and the result used to specify a cache line index to be read;
                        # the data word is moved into GPR T and the tag information is read
                        # into DCDBTRH and DCDBTRL
isync                   # ensure dcread completes before attempting to read results
mfdcdbtrh regD          # move high portion of tag into GPR D
mfdcdbtrl regE          # move low portion of tag into GPR E
```

The following figures illustrate the DCDBTRH and DCDBTRL.

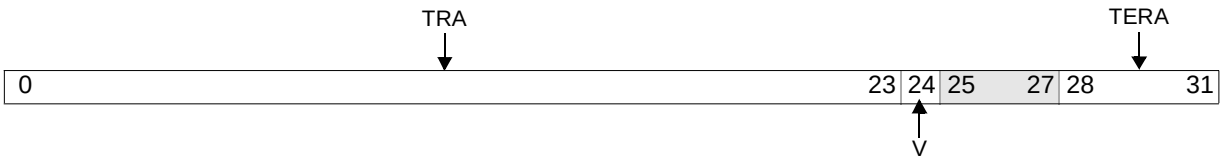


Figure 0-7. Data Cache Debug Tag Register High (DCDBTRH)

0:23	TRA	Tag Real Address	Bits 0:23 of the lower 32 bits of the 36-bit real address associated with the cache line read by **dcread**.
24	V	Cache Line Valid 0 Cache line is not valid. 1 Cache line is valid.	The valid indicator for the cache line read by **dcread**.
25:27		Reserved	
28:31	TERA	Tag Extended Real Address	Upper 4 bits of the 36-bit real address associated with the cache line read by **dcread**.

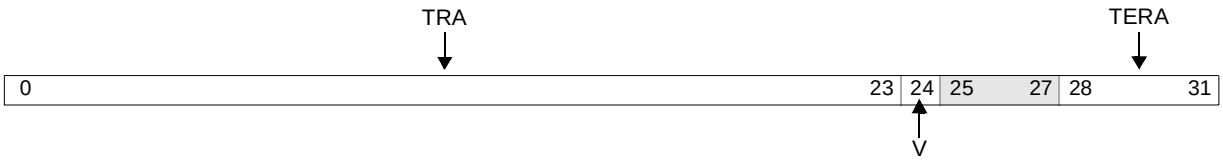


Figure 5-9. Data Cache Debug Tag Register High (DCDBTRH)

0:23	TRA	Tag Real Address	Bits 0:23 of the lower 32 bits of the 36-bit real address associated with the cache line read by **dcread**.
24	V	Cache Line Valid 0 Cache line is not valid. 1 Cache line is valid.	The valid indicator for the cache line read by **dcread**.
25:27		Reserved	
28:31	TERA	Tag Extended Real Address	Upper 4 bits of the 36-bit real address associated with the cache line read by **dcread**.

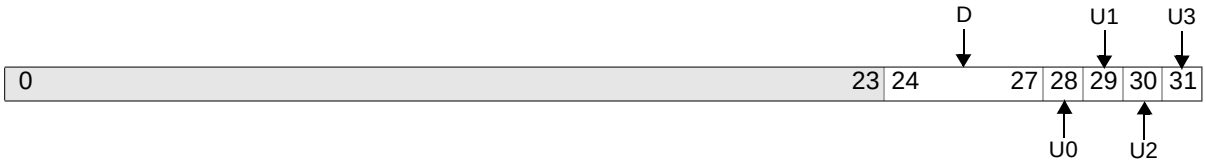


Figure 0-8. Data Cache Debug Tag Register Low (DCDBTRL)

0:23		Reserved

24:27	D	Dirty Indicators	The "dirty" (modified) indicators for each of the four doublewords in the cache line read by **dcread**.
28	U0	U0 Storage Attribute	The U0 storage attribute for the memory page associated with this cache line read by **dcread**.
29	U1	U1 Storage Attribute	The U0 storage attribute for the memory page associated with this cache line read by **dcread**.
30	U2	U2 Storage Attribute	The U0 storage attribute for the memory page associated with this cache line read by **dcread**.
31	U3	U3 Storage Attribute	The U0 storage attribute for the memory page associated with this cache line read by **dcread**.

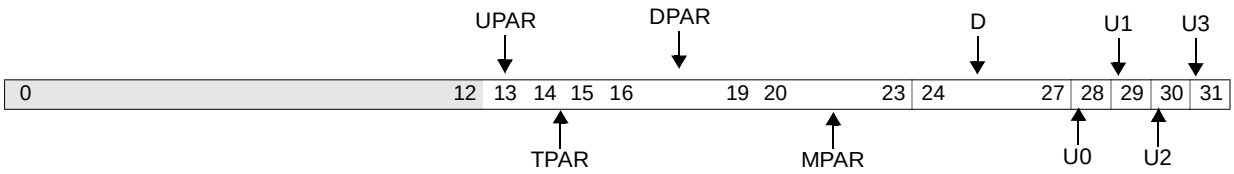


Figure 5-10. Data Cache Debug Tag Register Low (DCDBTRL)

0:12		Reserved	
13	UPAR	U bit parity	The parity for the U0-U3 bits in the cache line read by **dcread**
14:15	TPAR	Tag parity	The parity for the tag bits in the cache line read by **dcread**
16:19	DPAR	Data parity	The parity *check values* for the data bytes in the word read by **dcread**
20:23	MPAR	Modified (dirty) parity	The parity for the modified (dirty) indicators for each of the four doublewords in the cache line read by **dcread**.
24:27	D	Dirty Indicators	The "dirty" (modified) indicators for each of the four doublewords in the cache line read by **dcread**.
28	U0	U0 Storage Attribute	The U0 storage attribute for the memory page associated with this cache line read by **dcread**.
29	U1	U1 Storage Attribute	The U1 storage attribute for the memory page associated with this cache line read by **dcread**.
30	U2	U2 Storage Attribute	The U2 storage attribute for the memory page associated with this cache line read by **dcread**.
31	U3	U3 Storage Attribute	The U3 storage attribute for the memory page associated with this cache line read by **dcread**.

4. Programming Model

The programming model of the PPC440GP describes how the following features and operations of the processor core appear to programmers:

- Storage addressing (including data types and byte ordering), starting on ~~page 4-1~~page 145
- Registers, starting on ~~page 4-10~~page 154
- Instruction classes, starting on ~~page 4-35~~page 175
- Instruction set, starting on ~~page 4-38~~page 178
- Branch processing, starting on ~~page 4-46~~page 185
- Integer processing, starting on ~~page 4-53~~page 192
- Processor control, starting on ~~page 4-57~~page 195
- User and supervisor state, starting on ~~page 4-61~~page 199
- Speculative access, starting on ~~page 4-62~~page 200
- Synchronization, starting on ~~page 4-62~~page 201

4.1 Storage Addressing

As a 32-bit implementation of the Book-E Enhanced PowerPC Architecture, the PPC440GP implements a uniform 32-bit effective address (EA) space. Effective addresses are expanded into virtual addresses and then translated to 36-bit (64GB) real addresses by the memory management unit (see ~~Chapter 6, "Memory Management,"~~*Memory Management* on page 233 for more information on the translation process).

The PPC440GP generates an effective address whenever it executes a storage access, branch, cache management, or translation lookaside buffer (TLB) management instruction, or when it fetches the next sequential instruction.

0.0.1 Physical Address Map

Table 4-1 illustrates the physical address map.

Table 0-1. PPC440GP Address Space

Function	Start Address	End Address	Size
Local Memory/Peripherals[a]	0x000000000	0x0FFFFFFFF	
DDR SDRAM	0x000000000	0x07FFFFFFF	2GB
SRAM	0x080000000	0x080001FFF	8KB
Internal Peripherals (Total)	0x100000000	0x1EFFFFFFF	
EBCO Memory Banks	0x100000000	0x13FFFFFFF	1GB
UART0 Registers	0x140000200	0x140000207	8B
UART1 Registers	0x140000300	0x140000307	8B
IIC0 Registers	0x140000400	0x14000041F	32B
IIC1 Registers	0x140000500	0x14000051F	32B
OPB Arbiter Registers	0x140000600	0x14000063F	64B
GPIO Controller Registers	0x140000700	0x14000077F	128B
Ethernet PHY Interface (ZMII) Registers	0x140000780	0x14000078F	16B
Ethernet MAC0 Registers	0x140000800	0x1400008FF	256B
Ethernet MAC1 Registers	0x140000900	0x1400009FF	256B
General Purpose Timers	0x140000A00	0x140000AFF	256B
Expansion ROM[b]	0x1F0000000	0x1FFDFFFFF	254MB
Boot ROM[2, c]	0x1FFE00000	0x1FFFFFFFF	2MB
PCI (Total)	0x200000000	0xFFFFFFFFF	
PCI I/O	0x208000000	0x2E800FFFF	64KB
PCI External Configuration Registers	0x20EC00000	0x20EC00007	8B
PCI Bridge Configuration Registers	0x20EC80000	0x20EC800FF	256B
PCI Special Cycle	0x20ED00000	0x20EDFFFFF	1MB
PCI Memory	0x20EE00000	0xFFFFFFFFF	55.75GB

a. The local memory/peripheral area of the memory map can be configured for SDRAM or on-chip SRAM.

b. The boot ROM and expansion ROM areas of the memory map are intended for ROM or flash-type devices. While the controller supports volatile memory devices such as SDRAM and SRAM in these areas, it is not recommended that these regions be used for this purpose.

c. When booting from PCI memory, the boot ROM address space begins at 0x2FFFE0000 (size is 128KB).

4.1.1 Physical Address Map

Table 4-1 illustrates the physical address map.

Table 4-1. PPC440GP Address Space

Function	Start Address	End Address	Size
Local Memory/Peripherals[a]	0x000000000	0x0FFFFFFFF	
DDR SDRAM Registers	0x000000000	0x07FFFFFFF	2GB
SRAM Registers	0x080000000	0x080001FFF	8KB
Internal Peripherals (Total)	0x100000000	0x1EFFFFFFF	
EBCO Memory Banks Registers	0x100000000	0x13FFFFFFF	1GB
UART0 Registers	0x140000200	0x140000207	8B
UART1 Registers	0x140000300	0x140000307	8B
IIC0 Registers	0x140000400	0x14000041F	32B
IIC1 Registers	0x140000500	0x14000051F	32B
OPB Arbiter Registers	0x140000600	0x14000063F	64B
GPIO Controller Registers	0x140000700	0x14000077F	128B
Ethernet PHY Interface (ZMII) Registers	0x140000780	0x14000078F	16B
Ethernet MAC0 Registers	0x140000800	0x1400008FF	256B
Ethernet MAC1 Registers	0x140000900	0x1400009FF	256B
General Purpose Timers Registers	0x140000A00	0x140000AFF	256B
Expansion ROM[b]	0x1F0000000	0x1FFDFFFFF	254MB
Boot ROM[2, c]	0x1FFE00000	0x1FFFFFFFF	2MB
PCI (Total)	0x200000000	0xFFFFFFFFF	
PCI I/O	0x208000000	0x20800FFFF	64KB
PCI External Configuration Registers	0x20EC00000	0x20EC00007	8B
PCI Bridge Configuration Registers	0x20EC80000	0x20EC800FF	256B
PCI Special Cycle	0x20ED00000	0x20EDFFFFF	1MB
PCI Memory	0x20EE00000	0xFFFFFFFFF	55.75GB

a.The local memory/peripheral area of the memory map can be configured for SDRAM or on-chip SRAM.

b.The boot ROM and expansion ROM areas of the memory map are intended for ROM or flash-type devices. While the controller supports volatile memory devices such as SDRAM and SRAM in these areas, it is not recommended that these regions be used for this purpose.

c.When booting from PCI memory, the boot ROM address space begins at 0x2FFFE0000 (size is 128KB).

4.1.2 Storage Operands

Bytes in storage are numbered consecutively starting with 0. Each number is the address of the corresponding byte.

Data storage operands accessed by the integer load/store instructions may be bytes, halfwords, words, or—for load/store multiple and string instructions—a sequence of words or bytes, respectively. The address of a storage operand is the address of its first byte (that is, of its lowest-numbered byte). Byte ordering can be either big endian or little endian, as controlled by the endian storage attribute (see ~~"Byte Ordering" on page 4-5~~*Byte Ordering* on page 149; also see ~~"Endian (E)" on page 6-15~~*Endian (E)* on page 246 for more information on the endian storage attribute).

Operand length is implicit for each scalar storage access instruction type (that is, each storage access instruction type other than the load/store multiple and string instructions). The operand of such a scalar storage access instruction has a "natural" alignment boundary equal to the operand length. In other words, the 'natural' address of an operand is an integral multiple of the operand length. A storage operand is said to be *aligned* if it is aligned at its natural boundary: otherwise it is said to be *unaligned*.

Data storage operands for storage access instructions have the following characteristics.

Table 4-2. Data Operand Definitions

Storage Access Instruction Type	Operand Length	Addr[28:31] if aligned
Byte (or String)	8 bits	0bxxxx
Halfword	2 bytes	0bxxx0
Word (or Multiple)	4 bytes	0bxx00
Doubleword (AP only)	8 bytes	0bx000
Quadword (AP only)	16 bytes	0b0000
Note: An "x" in an address bit position indicates that the bit can be 0 or 1 independent of the state of other bits in the address.		

The alignment of the operand effective address of some storage access instructions may affect performance, and in some cases may cause an Alignment exception to occur. For such storage access instructions, the best performance is obtained when the storage operands are aligned. ~~Table 4-3~~*Table 4-3* summarizes the effects of alignment on those storage access instruction types for which such effects exist. If an instruction type is not shown in the table, then there are no alignment effects for that instruction type.

Table 4-3. Alignment Effects for Storage Access Instructions

Storage Access InstructionType	Alignment Effects
Integer load/store halfword	Broken into two byte accesses if crosses 16-byte boundary (EA[28:31] = 0b1111); otherwise no effect
Integer load/store word	Broken into two accesses if crosses 16-byte boundary (EA[28:31] > 0b1100); otherwise no effect
Integer load/store multiple or string	Broken into a series of 4-byte accesses until the last byte is accessed or a 16-byte boundary is reached, whichever occurs first. If bytes remain past a 16-byte boundary, resume accessing 4 bytes at a time until the last byte is accessed or the next 16-byte boundary is reached, whichever occurs first; repeat.

Cache management instructions access *cache block* operands, and for the PPC440GP the cache block size is 32 bytes. However, the effective addresses calculated by cache management instructions are not required to be aligned on cache block boundaries. Instead, the architecture specifies that the associated low-order effective address bits (bits 27:31 for PPC440GP) are ignored during the execution of these instructions.

Similarly, the TLB management instructions access *page* operands, and—as determined by the page size—the associated low-order effective address bits are ignored during the execution of these instructions.

Instruction storage operands, on the other hand, are always four bytes long, and the effective addresses calculated by Branch instructions are therefore always word-aligned.

4.1.3 Effective Address Calculation

For a storage access instruction, if the sum of the effective address and the operand length exceeds the maximum effective address of 2^{32}–1 (that is, the storage operand itself crosses the maximum address boundary), the result of the operation is undefined, as specified by the architecture. The PPC440GP performs the operation as if the storage operand wrapped around from the maximum effective address to effective address 0. Software, however, should not depend upon this behavior, so that it may be ported to other implementations that do not handle this scenario in the same fashion. Accordingly, software should ensure that no data storage operands cross the maximum address boundary.

Note that since instructions are words and since the effective addresses of instructions are always implicitly on word boundaries, it is not possible for an instruction storage operand to cross any word boundary, including the maximum address boundary.

Effective address arithmetic, which calculates the starting address for storage operands, wraps around from the maximum address to address 0, for all effective address computations except next sequential instruction fetching. See ~~"Instruction Storage Addressing Modes" on page 4-5~~*Instruction Storage Addressing Modes on page 148* for more information on next sequential instruction fetching at the maximum address boundary.

4.1.3.1 Data Storage Addressing Modes

There are two data storage addressing modes supported by the PPC440GP:

- Base + displacement (D-mode) addressing mode:

  The 16-bit D field is sign-extended and added to the contents of the GPR designated by RA or to zero if RA = 0; the low-order 32 bits of the sum form the effective address of the data storage operand.

- Base + index (X-mode) addressing mode:

  The contents of the GPR designated by RB (or the value 0 for **lswi** and **stswi**) are added to the contents of the GPR designated by RA, or to 0 if RA = 0; the low-order 32 bits of the sum form the effective address of the data storage operand.

4.1.3.2 Instruction Storage Addressing Modes

There are four instruction storage addressing modes supported by the PPC440GP:

- I-form branch instructions (unconditional):

  The 24-bit LI field is concatenated on the right with 0b00, sign-extended, and then added to either the address of the branch instruction if AA=0, or to 0 if AA=1; the low-order 32 bits of the sum form the effective address of the next instruction.

- Taken B-form branch instructions:

  The 14-bit BD field is concatenated on the right with 0b00, sign-extended, and then added to either the address of the branch instruction if AA=0, or to 0 if AA=1; the low-order 32 bits of the sum form the effective address of the next instruction.

- Taken XL-form branch instructions:

The contents of bits 0:29 of the Link Register (LR) or the Count Register (CTR) are concatenated on the right with 0b00 to form the 32-bit effective address of the next instruction.

- Next sequential instruction fetching (including non-taken branch instructions):

  The value 4 is added to the address of the current instruction to form the 32-bit effective address of the next instruction. If the address of the current instruction is 0xFFFFFFFC, the PPC440GP wraps the next sequential instruction address back to address 0. This behavior is not required by the architecture, which specifies that the next sequential instruction address is undefined under these circumstances. Therefore, software should not depend upon this behavior, so that it may be ported to other implementations that do not handle this scenario in the same fashion. Accordingly, if software wishes to execute across this maximum address boundary and wrap back to address 0, it should place an unconditional branch at the boundary, with a displacement of 4.

  In addition to the above four instruction storage addressing modes, the following behavior applies to branch instructions:

- Any branch instruction with LK=1:

  The value 4 is added to the address of the current instruction and the low-order 32 bits of the result are placed into the LR. As for the similar scenario for next sequential instruction fetching, if the address of the branch instruction is 0xFFFF FFFC, the result placed into the LR is architecturally undefined, although once again the PPC440GP wraps the LR update value back to address 0. Again, however, software should not depend on this behavior, in order that it may be ported to implementations which do not handle this scenario in the same fashion.

4.1.4 Byte Ordering

If scalars (individual data items and instructions) were indivisible, there would be no such concept as "byte ordering." It is meaningless to consider the order of bits or groups of bits within the smallest addressable unit of storage, because nothing can be observed about such order. Only when scalars, which the programmer and processor regard as indivisible quantities, can comprise more than one addressable unit of storage does the question of order arise.

For a machine in which the smallest addressable unit of storage is the 64-bit doubleword, there is no question of the ordering of bytes within doublewords. All transfers of individual scalars between registers and storage are of doublewords, and the address of the byte containing the high-order eight bits of a scalar is no different from the address of a byte containing any other part of the scalar.

For the Book-E Enhanced PowerPC Architecture, as for most current computer architectures, the smallest addressable unit of storage is the 8-bit byte. Many scalars are halfwords, words, or doublewords, which consist of groups of bytes. When a word-length scalar is moved from a register to storage, the scalar occupies four consecutive byte addresses. It thus becomes meaningful to discuss the order of the byte addresses with respect to the value of the scalar: which byte contains the highest-order eight bits of the scalar, which byte contains the next-highest-order eight bits, and so on.

Given a scalar that contains multiple bytes, the choice of byte ordering is essentially arbitrary. There are 4! = 24 ways to specify the ordering of four bytes within a word, but only two of these orderings are sensible:

- The ordering that assigns the lowest address to the highest-order ("left-most") eight bits of the scalar, the next sequential address to the next-highest-order eight bits, and so on.

This ordering is called *big endian* because the "big end" (most-significant end) of the scalar, considered as a binary number, comes first in storage. IBM RISC System/6000, IBM System/390, and Motorola 680x0 are examples of computer architectures using this byte ordering.

- The ordering that assigns the lowest address to the lowest-order ("right-most") eight bits of the scalar, the next sequential address to the next-lowest-order eight bits, and so on.

  This ordering is called *little endian* because the "little end" (least-significant end) of the scalar, considered as a binary number, comes first in storage. The Intel x86 is an example of a processor architecture using this byte ordering.

PowerPC Book-E supports both big endian and little endian byte ordering, for both instruction and data storage accesses. Which byte ordering is used is controlled on a memory page basis by the endian (E) storage attribute, which is a field within the TLB entry for the page. The endian storage attribute is set to 0 for a big endian page, and is set to 1 for a little endian page. See ~~Chapter 6, "Memory Management"~~*Memory Management* on page 233 for more information on memory pages, the TLB, and storage attributes, including the endian storage attribute.

4.1.4.1 Structure Mapping Examples

The following C language structure, *s*, contains an assortment of scalars and a character string. The comments show the value assumed to be in each structure element; these values show how the bytes comprising each structure element are mapped into storage.

```
struct {
        int a;          /* 0x1112_1314 word */
        long long b;    /* 0x2122_2324_2526_2728 doubleword */
        char *c;        /* 0x3132_3334 word */
        char d[7];      /* 'A','B','C','D','E','F','G' array of bytes */
        short e;        /* 0x5152 halfword */
        int f;          /* 0x6162_6364 word */
} s;
```

C structure mapping rules permit the use of padding (skipped bytes) to align scalars on desirable boundaries. The structure mapping examples below show each scalar aligned at its natural boundary. This alignment introduces padding of four bytes between *a* and *b*, one byte between *d* and *e*, and two bytes between *e* and *f*. The same amount of padding is present in both big endian and little endian mappings.

Big Endian Mapping

The big endian mapping of structure *s* follows (the data is highlighted in the structure mappings). Addresses, in hexadecimal, are below the data stored at the address. The contents of each byte, as defined in structure *s*, is shown as a (hexadecimal) number or character (for the string elements). The shaded cells correspond to padded bytes.

11	**12**	**13**	**14**				
0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07
21	**22**	**23**	**24**	**25**	**26**	**27**	**28**
0x08	0x09	0x0A	0x0B	0x0C	0x0D	0x0E	0x0F
31	**32**	**33**	**34**	**'A'**	**'B'**	**'C'**	**'D'**
0x10	0x11	0x12	0x13	0x14	0x15	0x16	0x17
'E'	**'F'**	**'G'**		**51**	**52**		
0x18	0x19	0x1A	0x1B	0x1C	0x1D	0x1E	0x1F
61	**62**	**63**	**64**				
0x20	0x21	0x22	0x23	0x24	0x25	0x26	0x27

Little Endian Mapping

Structure *s* is shown mapped little endian.

14	**13**	**12**	**11**				
0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07
28	**27**	**26**	**25**	**24**	**23**	**22**	**21**
0x08	0x09	0x0A	0x0B	0x0C	0x0D	0x0E	0x0F
34	**33**	**32**	**31**	**'A'**	**'B'**	**'C'**	**'D'**
0x10	0x11	0x12	0x13	0x14	0x15	0x16	0x17
'E'	**'F'**	**'G'**		**52**	**51**		
0x18	0x19	0x1A	0x1B	0x1C	0x1D	0x1E	0x1F
64	**63**	**62**	**61**				
0x20	0x21	0x22	0x23	0x24	0x25	0x26	0x27

4.1.4.2 Instruction Byte Ordering

PowerPC Book-E defines instructions as aligned words (four bytes) in memory. As such, instructions in a big endian program image are arranged with the most-significant byte (MSB) of the instruction word at the lowest-numbered address.

Consider the big endian mapping of instruction *p* at address 0x00, where, for example, *p* = add r7, r7, r4:

MSB			LSB
0x00	0x01	0x02	0x03

On the other hand, in a little endian mapping the same instruction is arranged with the least-significant byte (LSB) of the instruction word at the lowest-numbered address:

LSB			MSB
0x00	0x01	0x02	0x03

By the definition of PowerPC Book-E bit numbering, the most-significant byte of an instruction is the byte containing bits 0:7 of the instruction. As depicted in the instruction format diagrams (see ~~"Instruction Formats" on page 28-3~~*Instruction Formats* on page 894), this most-significant byte is the one which contains the primary opcode field (bits 0:5). Due to this difference in byte orderings, the processor must perform whatever byte reversal is required (depending on the particular byte ordering in use) in order to correctly deliver the opcode field to the instruction decoder. In the PPC440GP, this reversal is performed between the memory interface and the instruction cache, according to the value of the endian storage attribute for each memory page, such that the bytes in the instruction cache are always correctly arranged for delivery directly to the instruction decoder.

If the endian storage attribute for a memory page is reprogrammed from one byte ordering to the other, the contents of the memory page must be reloaded with program and data structures that are in the appropriate byte ordering. Furthermore, anytime the contents of instruction memory change, the instruction cache must be made coherent with the updates by invalidating the instruction cache and refetching the updated memory contents with the new byte ordering.

4.1.4.3 Data Byte Ordering

Unlike instruction fetches, data accesses cannot be byte-reversed between memory and the data cache. Data byte ordering in memory depends upon the data type (byte, halfword, word, and so on) of a specific data item. It is only when moving a data item of a specific type from or to an architected register (as directed by the execution of a particular storage access instruction) that it becomes known what kind of byte reversal may be required due to the byte ordering of the memory page containing the data item. Therefore, byte reversal during load or store accesses is performed between data cache (or memory, on a data cache miss, for example) and the load register target or store register source, depending on the specific type of load or store instruction (that is, byte, halfword, word, and so on).

Comparing the big endian and little endian mappings of structure *s*, as shown in ~~"Structure Mapping Examples" on page 4-6~~*Structure Mapping Examples* on page 150, the differences between the byte locations of any data item in the structure depends upon the size of the particular data item. For example (again referring to the big endian and little endian mappings of structure *s*):

- The word *a* has its four bytes reversed within the word spanning addresses 0x00 – 0x03.
- The halfword *e* has its two bytes reversed within the halfword spanning addresses 0x1C – 0x1D.

Note that the array of bytes d, where each data item is a byte, is not reversed when the big endian and little endian mappings are compared. For example, the character 'A' is located at address 0x14 in both the big endian and little endian mappings.

The size of the data item being loaded or stored must be known before the processor can decide whether, and if so, how to reorder the bytes when moving them between a register and the data cache (or memory).

- For byte loads and stores, including strings, no reordering of bytes occurs, regardless of byte ordering.
- For halfword loads and stores, bytes are reversed within the halfword, for one byte order with respect to the other.

- For word loads and stores (including load/store multiple), bytes are reversed within the word, for one byte order with respect to the other.
- For doubleword loads and stores (AP loads/stores only), bytes are reversed within the doubleword, for one byte order with respect to the other.
- For quadword loads and stores (AP loads/stores only), bytes are reversed within the quadword, for one byte order with respect to the other.

Note that this mechanism applies independent of the alignment of data. In other words, when loading a multi-byte data operand with a scalar load instruction, bytes are accessed from the data cache (or memory) starting with the byte at the calculated effective address and continuing with consecutively higher-numbered bytes until the required number of bytes have been retrieved. Then, the bytes are arranged such that either the byte from the highest-numbered address (for big endian storage regions) or the lowest-numbered address (for little endian storage regions) is placed into the least-significant byte of the register. The rest of the register is filled in corresponding order with the rest of the accessed bytes. An analogous procedure is followed for scalar store instructions.

For load/store multiple instructions, each group of four bytes is transferred between memory and the register according to the procedure for a scalar load word instruction.

For load/store string instructions, the most-significant byte of the first register is transferred to or from memory at the starting (lowest-numbered) effective address, regardless of byte ordering. Subsequent register bytes (from most-significant to least-significant, and then moving into the next register, starting with the most-significant byte, and so on) are transferred to or from memory at sequentially higher-numbered addresses. This behavior for byte strings ensures that if two strings are loaded into registers and then compared, the first bytes of the strings are treated as most significant with respect to the comparison.

4.1.4.4 Byte-Reverse Instructions

PowerPC Book-E defines load/store byte-reverse instructions which can access storage which is specified as being of one byte ordering in the same manner that a regular (that is, non-byte-reverse) load/store instruction would access storage which is specified as being of the opposite byte ordering. In other words, a load/store byte-reverse instruction to a big endian memory page transfers data between the data cache (or memory) and the register in the same manner that a normal load/store would transfer the data to or from a little endian memory page. Similarly, a load/store byte-reverse instruction to a little endian memory page transfers data between the data cache (or memory) and the register in the same manner that a normal load/store would transfer the data to or from a big endian memory page.

The function of the load/store byte-reverse instructions is useful when a particular memory page contains a combination of data with both big endian and little endian byte ordering. In such an environment, the Endian storage attribute for the memory page would be set according to the predominant byte ordering for the page, and the normal load/store instructions would be used to access data operands which used this predominant byte ordering. Conversely, the load/store byte-reverse instructions would be used to access the data operands which were of the other (less prevalent) byte ordering.

Software compilers cannot typically make general use of the load/store byte-reverse instructions, so they are ordinarily used only in special, hand-coded device drivers.

4.2 Registers

This section provides an overview of the register categories and types provided by the PPC440GP. Detailed descriptions of each of the registers are provided within the chapters covering the functions with which they are associated (for example, the cache control and cache debug registers are described in ~~Chapter 5, "Instruction and Data Caches"~~*Instruction and Data Caches* on page 205). An alphabetical summary of all registers, including bit definitions, is provided in ~~Chapter 29, "Register Summary."~~*Register Summary* on page 1093

All registers in the PPC440GP are architected as 32 bits wide, although certain bits in some registers are *reserved* and thus not necessarily implemented. For all registers with fields marked as reserved, these reserved fields should be written as 0 and read as *undefined*. The recommended coding practice is to perform the initial write to a register with reserved fields set to 0, and to perform all subsequent writes to the register using a read-modify-write strategy: read the register; use logical instructions to alter defined fields, leaving reserved fields unmodified; and write the register.

All of the registers are grouped into categories according to the processor functions with which they are associated. In addition, each register is classified as being of a particular *type*, as characterized by the specific instructions which are used to read and write registers of that type. Finally, most of the registers contained within the PPC440GP are defined by the Book-E Enhanced PowerPC Architecture, although some registers are implementation-specific and unique to the PPC440GP.

~~Figure 4-1 on page 4-11~~*Figure 4-1* on page 155 illustrates the PPC440GP registers contained in the user programming model, that is, those registers to which access is non-privileged and which are available to both user and supervisor programs. ~~Figure 4-2 on page 4-12~~*Figure 4-2* on page 156 illustrates the PPC440GP registers contained in the supervisor programming model, to which access is privileged and which are available to supervisor programs only. See ~~"User and Supervisor Modes" on page 4-61~~*User and Supervisor Modes* on page 199 for more information on privileged instructions and register access, and the user and supervisor programming models.

~~Table 4-4, on page 4-13~~*Table 4-4* on page 157, lists each register category and the registers that belong to each category, along with their types and a cross-reference to the section of this document which describes them more fully. Registers that are not part of PowerPC Book-E, and are thus specific to the PPC440GP, are shown in italics in ~~Table 4-4~~*Table 4-4*. Unless otherwise indicated, all registers have read/write access.

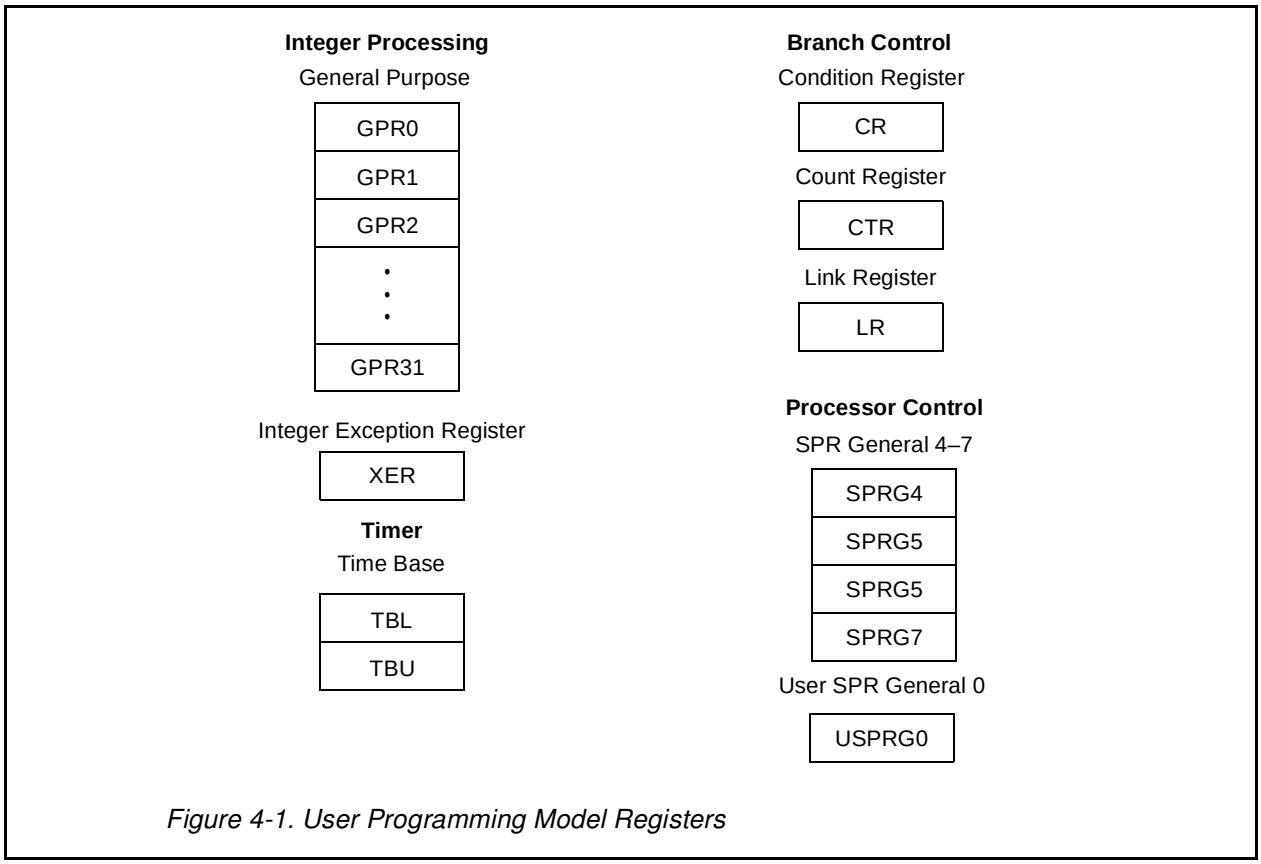


Figure 4-1. User Programming Model Registers

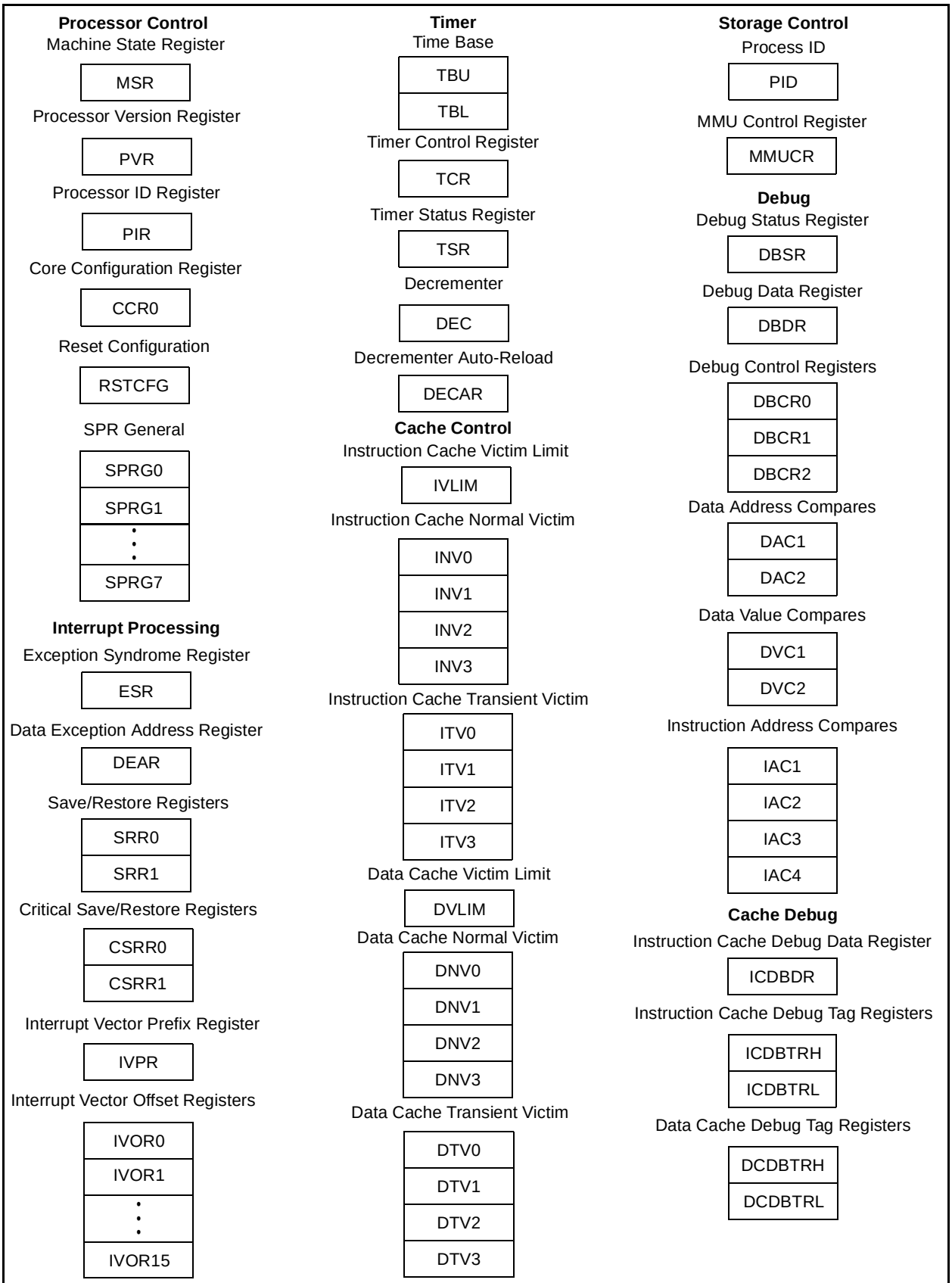


Figure 4-2. Supervisor Programming Model Registers

Table 0-2. Register Categories

Register Category	Register(s)	Model and Access	Type	Page
Branch Control	CR	User	CR	4-50
	CTR	User	SPR	4-49
	LR	User	SPR	4-49
Cache Control	*DNV0–DNV3*	Supervisor	SPR	5-2
	DTV0–DTV3	Supervisor	SPR	5-2
	DVLIM	Supervisor	SPR	5-4
	INV0–INV3	Supervisor	SPR	5-2
	ITV0–ITV3	Supervisor	SPR	5-2
	IVLIM	Supervisor	SPR	5-4
Cache Debug	*DCDBTRH, DCDBTRL*	Supervisor, read-only	SPR	5-27
	ICDBDR, ICDBTRH, ICDBTRL	Supervisor, read-only	SPR	5-15
Debug	DAC1–DAC2	Supervisor	SPR	15-30
	DBCR0–DBCR2	Supervisor	SPR	15-23
	DBDR	Supervisor	SPR	15-31
	DBSR	Supervisor	SPR	15-28
	DVC1–DVC2	Supervisor	SPR	15-31
	IAC1–IAC4	Supervisor	SPR	15-30
Device Control	Implemented outside core	Supervisor	DCR	4-16
Integer Processing	GPR0–GPR31	User	GPR	4-53
	XER	User	SPR	4-53
Interrupt Processing	CSRR0–CSRR1	Supervisor	SPR	10-10
	DEAR	Supervisor	SPR	10-11
	ESR	Supervisor	SPR	10-13
	IVOR0–IVOR15	Supervisor	SPR	10-11
	IVPR	Supervisor	SPR	10-12
	SRR0–SRR1	Supervisor	SPR	10-9

Table 0-2. Register Categories

Register Category	Register(s)	Model and Access	Type	Page
Processor Control	*CCR0*	Supervisor	SPR	5-13
	MSR	Supervisor	MSR	10-7
	PIR, PVR	Supervisor, read-only	SPR	4-58
	RSTCFG	Supervisor, read-only	SPR	4-60
	SPRG0–SPRG3	Supervisor	SPR	4-57
	SPRG4–SPRG7	User, read-only; Supervisor	SPR	4-57
	USPRG0	User, read-only; Supervisor	SPR	4-57
Storage Control	*MMUCR*	Supervisor	SPR	6-16
	PID	Supervisor	SPR	6-20
Timer	DEC	Supervisor	SPR	11-3
	DECAR	Supervisor, write-only	SPR	11-3
	TBL, TBU	User read, Supervisor write	SPR	11-2
	TCR	Supervisor	SPR	11-8
	TSR	Supervisor	SPR	11-9

Table 4-4. Register Categories

Register Category	Register(s)	Model and Access	Type	Page
Branch Control	CR	User	CR	189
	CTR	User	SPR	188
	LR	User	SPR	188
Cache Control	*DNV0–DNV3*	Supervisor	SPR	206
	DTV0–DTV3	Supervisor	SPR	206
	DVLIM	Supervisor	SPR	208
	INV0–INV3	Supervisor	SPR	206
	ITV0–ITV3	Supervisor	SPR	206
	IVLIM	Supervisor	SPR	208
Cache Debug	*DCDBTRH, DCDBTRL*	Supervisor, read-only	SPR	231
	ICDBDR, ICDBTRH, ICDBTRL	Supervisor, read-only	SPR	218
Debug	DAC1–DAC2	Supervisor	SPR	458
	DBCR0–DBCR2	Supervisor	SPR	451
	DBDR	Supervisor	SPR	459
	DBSR	Supervisor	SPR	456
	DVC1–DVC2	Supervisor	SPR	458
	IAC1–IAC4	Supervisor	SPR	457
Device Control	Implemented outside core	Supervisor	DCR	159
Integer Processing	GPR0–GPR31	User	GPR	192
	XER	User	SPR	192
Interrupt Processing	CSRR0–CSRR1	Supervisor	SPR	346
	DEAR	Supervisor	SPR	347
	ESR	Supervisor	SPR	349
	IVOR0–IVOR15	Supervisor	SPR	347
	IVPR	Supervisor	SPR	348
	SRR0–SRR1	Supervisor	SPR	345
Processor Control	*CCR0*	Supervisor	SPR	216
	MSR	Supervisor	MSR	343
	PIR, PVR	Supervisor, read-only	SPR	196
	RSTCFG	Supervisor, read-only	SPR	198
	SPRG0–SPRG3	Supervisor	SPR	195
	SPRG4–SPRG7	User, read-only; Supervisor	SPR	195
	USPRG0	User	SPR	195
Storage Control	*MMUCR*	Supervisor	SPR	247
	PID	Supervisor	SPR	251
Timer	DEC	Supervisor	SPR	385
	DECAR	Supervisor, write-only	SPR	385
	TBL, TBU	User read, Supervisor write	SPR	384
	TCR	Supervisor	SPR	389
	TSR	Supervisor	SPR	390

4.2.1 Register Types

There are five register types contained within and/or supported by the PPC440GP. Each register type is characterized by the instructions which are used to read and write the registers of that type. The following subsections provide an overview of each of the register types and the instructions associated with them.

4.2.1.1 General Purpose Registers

The PPC440GP contains 32 integer general purpose registers (GPRs); each contains 32 bits. Data from the data cache or memory can be loaded into GPRs using integer load instructions; the contents of GPRs can be stored to the data cache or memory using integer store instructions. Most of the integer instructions reference GPRs. The GPRs are also used as targets and sources for most of the instructions which read and write the other register types.

~~"Integer Processing" on page 4-53~~*Integer Processing* on page 192 provides more information on integer operations and the use of GPRs.

4.2.1.2 Special Purpose Registers

Special Purpose Registers (SPRs) are directly accessed using the **mtspr** and **mfspr** instructions. In addition, certain SPRs may be updated as a side-effect of the execution of various instructions. For example, the Integer Exception Register (XER) (see ~~"Integer Exception Register (XER)" on page 4-53~~*Integer Exception Register (XER)* on page 192) is an SPR which is updated with arithmetic status (such as carry and overflow) upon execution of certain forms of integer arithmetic instructions.

SPRs control the use of the debug facilities, timers, interrupts, memory management, caches, and other architected processor resources. ~~Table 29-2, on page 29-4~~*Table 29-2* on page 1095 shows the mnemonic, name, and number for each SPR, in order by SPR number. Each of the SPRs is described in more detail within the section or chapter covering the function with which it is associated. See ~~Table 4-4, on page 4-13~~*Table 4-4* on page 157 for a cross-reference to the associated document section for each register.

4.2.1.3 Condition Register

The Condition Register (CR) is a 32-bit register of its own unique type and is divided up into eight, independent 4-bit fields (CR0–CR7). The CR may be used to record certain conditional results of various arithmetic and logical operations. Subsequently, conditional branch instructions may designate a bit of the CR as one of the branch conditions (see ~~"Branch Processing" on page 4-46~~*Branch Processing* on page 185). Instructions are also provided for performing logical bit operations and for moving fields within the CR.

See ~~"Condition Register (CR)" on page 4-50~~*Condition Register (CR)* on page 189 for more information on the various instructions which can update the CR.

4.2.1.4 Machine State Register

The Machine State Register (MSR) is a register of its own unique type that controls important chip functions, such as the enabling or disabling of various interrupt types.

The MSR can be written from a GPR using the **mtmsr** instruction. The contents of the MSR can be read into a GPR using the **mfmsr** instruction. The MSR[EE] bit can be set or cleared atomically using the **wrtee** or **wrteei** instructions. The MSR contents are also automatically saved, altered, and restored by the interrupt-handling mechanism. See ~~"Machine State Register (MSR)" on page 10-7~~*Machine State Register (MSR)* on page 343 for more detailed information on the MSR and the function of each of its bits.

4.2.1.5 Device Control Registers

Device Control Registers (DCRs) are on-chip registers that exist architecturally and physically outside the processor core, and thus are not specified by the Book-E Enhanced PowerPC Architecture. Rather, PowerPC Book-E simply defines the existence of the DCR address space and the instructions that access the DCRs, and does not define any particular DCRs. The DCR access instructions are **mtdcr** (move to device control register) and **mfdcr** (move from device control register), which move data between GPRs and the DCRs.

DCRs may be used to control various on-chip system functions, such as the operation of on-chip buses, peripherals, and certain processor core behaviors.

Some DCRs are directly accessed, that is, they are accessed using their DCR numbers. Other DCRs are indirectly accessed. Such DCRs are accessed by writing an offset to a directly accessed DCR and then reading the data at the offset in another directly accessed DCR. Table 4-5 lists the directly accessed DCRs. The indirectly accessed DCRs are described immediately following Table 4-5.

Table 0-3. Directly Accessed DCRs

Register	DCR Number	Access	Description
DCRs Used for Indirect Access			
SDRAM0_CFGADDR	0x010	R/W	DDR-SDRAM Address Register
SDRAM0_CFGDATA	0x011	R/W	DDR-SDRAM Data Register
EBC0_CFGADDR	0x012	R/W	EBCO Address Register
EBC0_CFGDATA	0x013	R/W	EBCO Data Register
EBM0_CFGADDR	0x014	R/W	EBMI Address Register
EBM0_CFGDATA	0x015	R/W	EBMI Data Register
PPM0_CFGADDR	0x016	R/W	PPM Address Register
PPM0_CFGDATA	0x017	R/W	PPM Data Register
Internal SRAM Controller			
SRAM0_SB0CR	0x020	R/W	SRAM Bank Configuration Register 0
SRAM0_SB1CR	0x021	R/W	SRAM Bank Configuration Register 1
SRAM0_SB2CR	0x022	R/W	SRAM Bank Configuration Register 2
SRAM0_SB3CR	0x023	R/W	SRAM Bank Configuration Register 3
SRAM0_BEAR	0x024	R/W	SRAM Bus Error Address Register
SRAM0_BESR0	0x025	R/W	SRAM Bus Error Status Register 0
SRAM0_BESR1	0x026	R/W	SRAM Bus Error Status Register 1

Table 0-3. Directly Accessed DCRs

Register	DCR Number	Access	Description
SRAM0_PMEG	0x027	R/W	SRAM Power Management
SRAM0_CID	0x028	R/W	SRAM Bus Core ID Register 1
SRAM0_REVID	0x029	R/W	SRAM Bus Revision ID Register
SRAM0_DPC	0x02A	R/W	SRAM Data Parity Check Register
On-Chip Buses			
PLB0_REVID	0x082	R	PLB Arbiter Revision ID
PLB0_ACR	0x083	R/W	PLB Arbiter Control Register
PLB0_BESR	0x084	R/Clear	PLB Bus Error Status Register
PLB0_BEARL	0x086	R	PLB Bus Error Address Register
PLB0_BEARH	0x087	R	PLB Bus Error Address Register
POB0_BESR0	0x090	R/Clear	PLB to OPB Bridge Error Status Register 0
POB0_BEARL	0x092	R	PLB to OPB Bridge Error Address Register
POB0_BEARH	0x093	R	PLB to OPB Bridge Error Address Register
POB0_BESR1	0x094	R/Clear	PLB to OPB Bridge Error Status Register 1
POB0_CONFG	0x096	R/Clear	PLB to OPB Bridge Error Configuration Register
POB0_LATENCY	0x098	R/Clear	PLB to OPB Bridge Burst Latency Timer
POB0_REVID	0x09A	R	PLB to OPB Bridge Revision ID Register
OPB0_BCTRL	0x0A8	R/W	OPB to PLB Bridge Control Register
OPB0_BSTAT	0x0A9	R/C	OPB to PLB Bridge Status Register
OPB0_BEARL	0x0AA	R/C	OPB to PLB Bridge Error Address Register Low
OPB0_BEARH	0x0AB	R/C	OPB to PLB Bridge Error Address Register High
OPB0_REVID	0x0AC	R/C	OPB to PLB Bridge Revision ID Register
Clocking, Power Management, and Chip Control			
CPC0_SR	0x0B0	R/W	CPM Status Register
CPC0_ER	0x0B1	R/W	CPM Enable Register
CPC0_FR	0x0B2	R/W	CPM Force Register
CPC0_SYS0	0x0E0	R/W	System Configuration Register 0
CPC0_SYS1	0x0E1	R/W	System Configuration Register 1
CPC0_CUST0	0x0E2	R/W	Customer Configuration Register 0
CPC0_CUST1	0x0E3	R/W	Customer Configuration Register 1
CPC0_STRP0	0x0E4	R	Power-on Configuration Register 0
CPC0_STRP1	0x0E5	R	Power-on Configuration Register 1
CPC0_STRP2	0x0E6	R	Power-on Configuration Register 3
CPC0_STRP3	0x0E7	R	Power-on Configuration Register 4
CPC0_GPIO	0x0E8	R/W	GPIO Configuration Register
CPC0_PLB	0x0E9	R/W	PLB Configuration Register
CPC0_CR1	0x0EA	R/W	CPU Master Priority Configuration Register 1
CPC0_CR0	0x0EB	R/W	UART and Timer Configuration Register 0
CPC0_MIRQ0	0x0EC	R/W	Master Interrupt Request Register 0
CPC0_MIRQ1	0x0ED	R/W	Master Interrupt Request Register 1

Table 0-3. Directly Accessed DCRs

Register	DCR Number	Access	Description
CPC0_JTAGID	0x0EF	R	JTAG ID Register
Universal Interrupt Controllers			
UIC0_SR	0x0C0	R/Clear	UIC 0 Status Register
UIC0_ER	0x0C2	R/W	UIC 0 Enable Register
UIC0_CR	0x0C3	R/W	UIC 0 Critical Register
UIC0_PR	0x0C4	R/W	UIC 0 Polarity Register
UIC0_TR	0x0C5	R/W	UIC 0 Triggering Register
UIC0_MSR	0x0C6	R	UIC 0 Masked Status Register
UIC0_VR	0x0C7	R	UIC 0 Vector Register
UIC0_VCR	0x0C8	W	UIC 0 Vector Configuration Register
UIC1_SR	0x0D0	R/Clear	UIC 1 Status Register
UIC1_ER	0x0D2	R/W	UIC 1 Enable Register
UIC1_CR	0x0D3	R/W	UIC 1 Critical Register
UIC1_PR	0x0D4	R/W	UIC 1 Polarity Register
UIC1_TR	0x0D5	R/W	UIC 1 Triggering Register
UIC1_MSR	0x0D6	R	UIC 1 Masked Status Register
UIC1_VR	0x0D7	R	UIC 1 Vector Register
UIC1_VCR	0x0D8	W	UIC 1 Vector Configuration Register
Direct Memory Access			
DMA0_CR0	0x100	R/W	DMA Channel Control Register 0
DMA0_CT0	0x101	R/W	DMA Count Register 0
DMA0_SAH0	0x102	R/W	DMA Source Address High Register 0
DMA0_SAL0	0x103	R/W	DMA Source Address Low Register 0
DMA0_DAH0	0x104	R/W	DMA Destination Address High Register 0
DMA0_DAL0	0x105	R/W	DMA Destination Address Low Register 0
DMA0_SGH0	0x106	R/W	DMA Scatter/Gather Descriptor Address High Register 0
DMA0_SGL0	0x107	R/W	DMA Scatter/Gather Descriptor Address Low Register 0
DMA0_CR1	0x108	R/W	DMA Channel Control Register 1
DMA0_CT1	0x109	R/W	DMA Count Register 1
DMA0_SAH1	0x10A	R/W	DMA Source Address High Register 1
DMA0_SAL1	0x10B	R/W	DMA Source Address Low Register 1
DMA0_DAH1	0x10C	R/W	DMA Destination Address High Register 1
DMA0_DAL1	0x10D	R/W	DMA Destination Address Low Register 1
DMA0_SGH1	0x10E	R/W	DMA Scatter/Gather Descriptor Address High Register 1
DMA0_SGL1	0x10F	R/W	DMA Scatter/Gather Descriptor Address Low Register 1
DMA0_CR2	0x110	R/W	DMA Channel Control Register 2
DMA0_CT2	0x111	R/W	DMA Count Register 2

Table 0-3. Directly Accessed DCRs

Register	DCR Number	Access	Description
DMA0_SAH2	0x112	R/W	DMA Source Address High Register 2
DMA0_SAL2	0x113	R/W	DMA Source Address Low Register 2
DMA0_DAH2	0x114	R/W	DMA Destination Address High Register 2
DMA0_DAL2	0x115	R/W	DMA Destination Address Low Register 2
DMA0_SGH2	0x116	R/W	DMA Scatter/Gather Descriptor Address High Register 2
DMA0_SGL2	0x117	R/W	DMA Scatter/Gather Descriptor Address Low Register 2
DMA0_CR3	0x118	R/W	DMA Channel Control Register 3
DMA0_CT3	0x119	R/W	DMA Count Register 3
DMA0_SAH3	0x11A	R/W	DMA Source Address High Register 3
DMA0_SAL3	0x11B	R/W	DMA Source Address Low Register 3
DMA0_DAH3	0x11C	R/W	DMA Destination Address High Register 3
DMA0_DAL3	0x11D	R/W	DMA Destination Address Low Register 3
DMA0_SGH3	0x11E	R/W	DMA Scatter/Gather Descriptor Address High Register 3
DMA0_SGL3	0x11F	R/W	DMA Scatter/Gather Descriptor Address Low Register 3
DMA0_SR	0x120	R/Clear	DMA Status Register
DMA0_SGC	0x123	R/W	DMA Scatter/Gather Command Register
DMA0_SLP	0x125	R/W	DMA Sleep Mode Register
DMA0_POL	0x126	R/W	DMA Polarity Configuration Register
Memory Access Layer			
MAL0_CFG	0x180	R/W	MAL Configuration Register
MAL0_ESR	0x181	R/Clear	MAL Error Status Register
MAL0_IER	0x182	R/W	MAL Interrupt Enable Register
MAL0_TXCASR	0x184	R/W	MAL Tx Channel Active Register (Set)
MAL0_TXCARR	0x185	R/W	MAL Tx Channel Active Register (Reset)
MAL0_TXEOBISR	0x186	R/Clear	MAL Tx End of Buffer Interrupt Status Register
MAL0_TXDEIR	0x187	R/Clear	MAL Tx Descriptor Error Interrupt Register
MAL0_TXTATTRR	0x188	R/W	MAL Tx PLB Attribute Register
MAL0_TXBADDR	0x189	R/W	MAL Tx Descriptor Base Address Register
MAL0_RXCASR	0x190	R/W	MAL Rx Channel Active Register (Set)
MAL0_RXCARR	0x191	R/W	MAL Rx Channel Active Register (Reset)
MAL0_RXEOBISR	0x192	R/Clear	MAL Rx End of Buffer Interrupt Status Register
MAL0_RXDEIR	0x193	R/Clear	MAL Rx Descriptor Error Interrupt Register
MAL0_RXTATTRR	0x194	R/W	MAL Rx PLB Attribute Register
MAL0_RXBADDR	0x195	R/W	MAL Rx Descriptor Base Address Register
MAL0_TXCTP0R	0x1A0	R/W	MAL TX Channel 0 Table Pointer Register
MAL0_TXCTP1R	0x1A1	R/W	MAL TX Channel 1 Table Pointer Register
MAL0_TXCTP2R	0x1A2	R/W	MAL TX Channel 2 Table Pointer Register

Table 0-3. Directly Accessed DCRs

Register	DCR Number	Access	Description
MAL0_TXCTP3R	0x1A3	R/W	MAL TX Channel 3 Table Pointer Register
MAL0_RXCTP0R	0x1C0	R/W	MAL RX Channel 0 Table Pointer Register
MAL0_RXCTP1R	0x1C1	R/W	MAL RX Channel 1 Table Pointer Register
MAL0_RCBS0	0x1E0	R/W	MAL RX Channel 0 Buffer Size Register
MAL0_RCBS1	0x1E1	R/W	MAL RX Channel 1 Buffer Size Register

Table 4-5. Directly Accessed DCRs

Register	DCR Number	Access	Description
DCRs Used for Indirect Access			
SDRAM0_CFGADDR	0x010	R/W	DDR-SDRAM Address Register
SDRAM0_CFGDATA	0x011	R/W	DDR-SDRAM Data Register
EBC0_CFGADDR	0x012	R/W	EBCO Address Register
EBC0_CFGDATA	0x013	R/W	EBCO Data Register
EBM0_CFGADDR	0x014	R/W	EBMI Address Register
EBM0_CFGDATA	0x015	R/W	EBMI Data Register
PPM0_CFGADDR	0x016	R/W	PPM Address Register
PPM0_CFGDATA	0x017	R/W	PPM Data Register
Internal SRAM Controller			
SRAM0_SB0CR	0x020	R/W	SRAM Bank Configuration Register 0
SRAM0_SB1CR	0x021	R/W	SRAM Bank Configuration Register 1
SRAM0_SB2CR	0x022	R/W	SRAM Bank Configuration Register 2
SRAM0_SB3CR	0x023	R/W	SRAM Bank Configuration Register 3
SRAM0_BEAR	0x024	R/W	SRAM Bus Error Address Register
SRAM0_BESR0	0x025	R/W	SRAM Bus Error Status Register 0
SRAM0_BESR1	0x026	R/W	SRAM Bus Error Status Register 1
SRAM0_PMEG	0x027	R/W	SRAM Power Management
SRAM0_CID	0x028	R/W	SRAM Bus Core ID Register 1
SRAM0_REVID	0x029	R/W	SRAM Bus Revision ID Register
SRAM0_DPC	0x02A	R/W	SRAM Data Parity Check Register
On-Chip Buses			
PLB0_REVID	0x082	R	PLB Arbiter Revision ID
PLB0_ACR	0x083	R/W	PLB Arbiter Control Register
PLB0_BESR	0x084	R/Clear	PLB Bus Error Status Register
PLB0_BEARL	0x086	R	PLB Bus Error Address Register
PLB0_BEARH	0x087	R	PLB Bus Error Address Register
POB0_BESR0	0x090	R/Clear	PLB to OPB Bridge Error Status Register 0

Table 4-5. Directly Accessed DCRs

Register	DCR Number	Access	Description
POB0_BEARL	0x092	R	PLB to OPB Bridge Error Address Register
POB0_BEARH	0x093	R	PLB to OPB Bridge Error Address Register
POB0_BESR1	0x094	R/Clear	PLB to OPB Bridge Error Status Register 1
POB0_CONFG	0x096	R/Clear	PLB to OPB Bridge Error Configuration Register
POB0_LATENCY	0x098	R/Clear	PLB to OPB Bridge Burst Latency Timer
POB0_REVID	0x09A	R	PLB to OPB Bridge Revision ID Register
OPB0_BCTRL	0x0A8	R/W	OPB to PLB Bridge Control Register
OPB0_BSTAT	0x0A9	R/C	OPB to PLB Bridge Status Register
OPB0_BEARL	0x0AA	R/C	OPB to PLB Bridge Error Address Register Low
OPB0_BEARH	0x0AB	R/C	OPB to PLB Bridge Error Address Register High
OPB0_REVID	0x0AC	R/C	OPB to PLB Bridge Revision ID Register
Clocking, Power Management, and Chip Control			
CPC0_SR	0x0B0	R/W	CPM Status Register
CPC0_ER	0x0B1	R/W	CPM Enable Register
CPC0_FR	0x0B2	R/W	CPM Force Register
CPC0_SYS0	0x0E0	R/W	System Configuration Register 0
CPC0_SYS1	0x0E1	R/W	System Configuration Register 1
CPC0_CUST0	0x0E2	R/W	Customer Configuration Register 0
CPC0_CUST1	0x0E3	R/W	Customer Configuration Register 1
CPC0_STRP0	0x0E4	R	Power-on Configuration Register 0
CPC0_STRP1	0x0E5	R	Power-on Configuration Register 1
CPC0_STRP2	0x0E6	R	Power-on Configuration Register 3
CPC0_STRP3	0x0E7	R	Power-on Configuration Register 4
CPC0_GPIO	0x0E8	R/W	GPIO Configuration Register
CPC0_PLB	0x0E9	R/W	PLB Configuration Register
CPC0_CR1	0x0EA	R/W	CPU Master Priority Configuration Register 1
CPC0_CR0	0x0EB	R/W	UART and Timer Configuration Register 0
CPC0_MIRQ0	0x0EC	R/W	Master Interrupt Request Register 0
CPC0_MIRQ1	0x0ED	R/W	Master Interrupt Request Register 1
CPC0_JTAGID	0x0EF	R	JTAG ID Register
Universal Interrupt Controllers			
UIC0_SR	0x0C0	R/Clear	UIC 0 Status Register
UIC0_ER	0x0C2	R/W	UIC 0 Enable Register
UIC0_CR	0x0C3	R/W	UIC 0 Critical Register
UIC0_PR	0x0C4	R/W	UIC 0 Polarity Register
UIC0_TR	0x0C5	R/W	UIC 0 Triggering Register

Table 4-5. Directly Accessed DCRs

Register	DCR Number	Access	Description
UIC0_MSR	0x0C6	R	UIC 0 Masked Status Register
UIC0_VR	0x0C7	R	UIC 0 Vector Register
UIC0_VCR	0x0C8	W	UIC 0 Vector Configuration Register
UIC1_SR	0x0D0	R/Clear	UIC 1 Status Register
UIC1_ER	0x0D2	R/W	UIC 1 Enable Register
UIC1_CR	0x0D3	R/W	UIC 1 Critical Register
UIC1_PR	0x0D4	R/W	UIC 1 Polarity Register
UIC1_TR	0x0D5	R/W	UIC 1 Triggering Register
UIC1_MSR	0x0D6	R	UIC 1 Masked Status Register
UIC1_VR	0x0D7	R	UIC 1 Vector Register
UIC1_VCR	0x0D8	W	UIC 1 Vector Configuration Register
Direct Memory Access			
DMA0_CR0	0x100	R/W	DMA Channel Control Register 0
DMA0_CT0	0x101	R/W	DMA Count Register 0
DMA0_SAH0	0x102	R/W	DMA Source Address High Register 0
DMA0_SAL0	0x103	R/W	DMA Source Address Low Register 0
DMA0_DAH0	0x104	R/W	DMA Destination Address High Register 0
DMA0_DAL0	0x105	R/W	DMA Destination Address Low Register 0
DMA0_SGH0	0x106	R/W	DMA Scatter/Gather Descriptor Address High Register 0
DMA0_SGL0	0x107	R/W	DMA Scatter/Gather Descriptor Address Low Register 0
DMA0_CR1	0x108	R/W	DMA Channel Control Register 1
DMA0_CT1	0x109	R/W	DMA Count Register 1
DMA0_SAH1	0x10A	R/W	DMA Source Address High Register 1
DMA0_SAL1	0x10B	R/W	DMA Source Address Low Register 1
DMA0_DAH1	0x10C	R/W	DMA Destination Address High Register 1
DMA0_DAL1	0x10D	R/W	DMA Destination Address Low Register 1
DMA0_SGH1	0x10E	R/W	DMA Scatter/Gather Descriptor Address High Register 1
DMA0_SGL1	0x10F	R/W	DMA Scatter/Gather Descriptor Address Low Register 1
DMA0_CR2	0x110	R/W	DMA Channel Control Register 2
DMA0_CT2	0x111	R/W	DMA Count Register 2
DMA0_SAH2	0x112	R/W	DMA Source Address High Register 2
DMA0_SAL2	0x113	R/W	DMA Source Address Low Register 2
DMA0_DAH2	0x114	R/W	DMA Destination Address High Register 2
DMA0_DAL2	0x115	R/W	DMA Destination Address Low Register 2
DMA0_SGH2	0x116	R/W	DMA Scatter/Gather Descriptor Address High Register 2
DMA0_SGL2	0x117	R/W	DMA Scatter/Gather Descriptor Address Low Register 2

Table 4-5. Directly Accessed DCRs

Register	DCR Number	Access	Description
DMA0_CR3	0x118	R/W	DMA Channel Control Register 3
DMA0_CT3	0x119	R/W	DMA Count Register 3
DMA0_SAH3	0x11A	R/W	DMA Source Address High Register 3
DMA0_SAL3	0x11B	R/W	DMA Source Address Low Register 3
DMA0_DAH3	0x11C	R/W	DMA Destination Address High Register 3
DMA0_DAL3	0x11D	R/W	DMA Destination Address Low Register 3
DMA0_SGH3	0x11E	R/W	DMA Scatter/Gather Descriptor Address High Register 3
DMA0_SGL3	0x11F	R/W	DMA Scatter/Gather Descriptor Address Low Register 3
DMA0_SR	0x120	R/Clear	DMA Status Register
DMA0_SGC	0x123	R/W	DMA Scatter/Gather Command Register
DMA0_SLP	0x125	R/W	DMA Sleep Mode Register
DMA0_POL	0x126	R/W	DMA Polarity Configuration Register
Memory Access Layer			
MAL0_CFG	0x180	R/W	MAL Configuration Register
MAL0_ESR	0x181	R/Clear	MAL Error Status Register
MAL0_IER	0x182	R/W	MAL Interrupt Enable Register
MAL0_TXCASR	0x184	R/W	MAL Tx Channel Active Register (Set)
MAL0_TXCARR	0x185	R/W	MAL Tx Channel Active Register (Reset)
MAL0_TXEOBISR	0x186	R/Clear	MAL Tx End of Buffer Interrupt Status Register
MAL0_TXDEIR	0x187	R/Clear	MAL Tx Descriptor Error Interrupt Register
MAL0_TXTATTRR	0x188	R/W	MAL Tx PLB Attribute Register
MAL0_TXBADDR	0x189	R/W	MAL Tx Descriptor Base Address Register
MAL0_RXCASR	0x190	R/W	MAL Rx Channel Active Register (Set)
MAL0_RXCARR	0x191	R/W	MAL Rx Channel Active Register (Reset)
MAL0_RXEOBISR	0x192	R/Clear	MAL Rx End of Buffer Interrupt Status Register
MAL0_RXDEIR	0x193	R/Clear	MAL Rx Descriptor Error Interrupt Register
MAL0_RXTATTRR	0x194	R/W	MAL Rx PLB Attribute Register
MAL0_RXBADDR	0x195	R/W	MAL Rx Descriptor Base Address Register
MAL0_TXCTP0R	0x1A0	R/W	MAL TX Channel 0 Table Pointer Register
MAL0_TXCTP1R	0x1A1	R/W	MAL TX Channel 1 Table Pointer Register
MAL0_TXCTP2R	0x1A2	R/W	MAL TX Channel 2 Table Pointer Register
MAL0_TXCTP3R	0x1A3	R/W	MAL TX Channel 3 Table Pointer Register
MAL0_RXCTP0R	0x1C0	R/W	MAL RX Channel 0 Table Pointer Register
MAL0_RXCTP1R	0x1C1	R/W	MAL RX Channel 1 Table Pointer Register
MAL0_RCBS0	0x1E0	R/W	MAL RX Channel 0 Buffer Size Register
MAL0_RCBS1	0x1E1	R/W	MAL RX Channel 1 Buffer Size Register

~~Indirectly Accessed DCRs~~

~~The DCRs for the DDR-SDRAM controller, external bus controller (EBCO), and external bus master interface (EBMI) are indirectly accessed.~~

~~Indirect Access of DDR-SDRAM Controller DCRs~~

~~The following procedure accesses the DDR-SDRAM controller DCRs listed in Table 4-6.~~

~~1. Write the offset from Table 4-7 to the DDR-SDRAM Address Register (SDRAM0_CFGADDR).~~

~~2. Read data from or write data to the DDR-SDRAM Data Register (SDRAM0_CFGDATA).~~

Table 0-4. SDRAM Controller DCR Usage

Register	DCR Number	Access	Description
SDRAM0_CFGADDR	0x010	R/W	DDR-SDRAM Address Register
SDRAM0_CFGDATA	0x011	R/W	DDR-SDRAM Data Register

Table 0-5. Offsets for SDRAM Controller Registers

Register	Offset	R/W	Description
SDRAM0_BESR0	0x00	R/W	DDR-SDRAM Bus Error Syndrome Register 0
SDRAM0_BESR1	0x08	R/W	DDR-SDRAM Bus Error Syndrome Register 1
SDRAM0_BEAR	0x10	R	DDR-SDRAM Bus Error Address Register
SDRAM0_MIRQ	0x11	R/W	DDR-SDRAM Master Write Interrupt
SDRAM0_SLIO	0x18	R/W	DDR-SDRAM Slave Interface Options
SDRAM0_CFG0	0x20	R/W	DDR-SDRAM Options 0
SDRAM0_CFG1	0x21	R/W	DDR-SDRAM Options 1
SDRAM0_DEVOPT	0x22	R/W	DDR-SDRAM Device Options
SDRAM0_MCSTS	0x24	R	DDR-SDRAM Memory Controller Status
SDRAM0_RTR	0x30	R/W	DDR-SDRAM Refresh Timer Register
SDRAM0_PMIT	0x34	R/W	DDR-SDRAM Power Management Idle Timer
SDRAM0_UABBA	0x38	R/W	DDR-SDRAM PLB UA Bus Base Address
SDRAM0_B0CR	0x40	R/W	DDR-SDRAM Bank 0 Configuration Register
SDRAM0_B1CR	0x44	R/W	DDR-SDRAM Bank 1 Configuration Register
SDRAM0_B2CR	0x48	R/W	DDR-SDRAM Bank 2 Configuration Register
SDRAM0_B3CR	0x4C	R/W	DDR-SDRAM Bank 3 Configuration Register
SDRAM0_TR0	0x80	R/W	DDR-SDRAM Timing Register 0

Table 0-5. Offsets for SDRAM Controller Registers (continued)

Register	Offset	R/W	Description
SDRAM0_TR1	0x81	R/W	DDR-SDRAM Timing Register 1
SDRAM0_CLKTR	0x82	R/W	DDR-SDRAM Clock Timing Register
SDRAM0_WDDCTR	0x83	R/W	DDR-SDRAM Write Data, DQS, DM Clock Timing Register
SDRAM0_DLYCAL	0x84	R/W	DDR-SDRAM Delay Line Calibration Register
SDRAM0_ECCESR	0x98	R/W	DDR-SDRAM ECC Error Status Register
SDRAM0_CID	0xA4	R	DDR-SDRAM Core ID Register
SDRAM0_RID	0xA8	R	DDR-SDRAM Revision ID Register

~~Indirect Access of External Bus Controller DCRs~~

~~The following procedure accesses the EBCO DCRs listed in Table 4-8.~~

~~1. Write the offset from Table 4-9 to the EBCO Address Register (EBC0_CFGADDR).~~

~~2. Read data from or write data to the EBCO Data Register (EBC0_CFGDATA).~~

Table 0-6. External Bus Controller DCR Usage

Register	DCR Number	Access	Description
EBC0_CFGADDR	0x012	R/W	EBCO Controller Address Register
EBC0_CFGDATA	0x013	R/W	EBCO Controller Data Register

Indirect Access of External Bus Controller DCRs

Table 0-7. Offsets for External Bus Controller Registers

Register	Offset	Access	Description
EBC0_B0CR	0x00	R/W	EBCO Bank 0 Configuration Register
EBC0_B1CR	0x01	R/W	EBCO Bank 1 Configuration Register
EBC0_B2CR	0x02	R/W	EBCO Bank 2 Configuration Register
EBC0_B3CR	0x03	R/W	EBCO Bank 3 Configuration Register
EBC0_B4CR	0x04	R/W	EBCO Bank 4 Configuration Register
EBC0_B5CR	0x05	R/W	EBCO Bank 5 Configuration Register
EBC0_B6CR	0x06	R/W	EBCO Bank 6 Configuration Register
EBC0_B7CR	0x07	R/W	EBCO Bank 7 Configuration Register
EBC0_B0AP	0x10	R/W	EBCO Bank 0 Access Parameters
EBC0_B1AP	0x11	R/W	EBCO Bank 1 Access Parameters
EBC0_B2AP	0x12	R/W	EBCO Bank 2 Access Parameters
EBC0_B3AP	0x13	R/W	EBCO Bank 3 Access Parameters
EBC0_B4AP	0x14	R/W	EBCO Bank 4 Access Parameters
EBC0_B5AP	0x15	R/W	EBCO Bank 5 Access Parameters
EBC0_B6AP	0x16	R/W	EBCO Bank 6 Access Parameters
EBC0_B7AP	0x17	R/W	EBCO Bank 7 Access Parameters
EBC0_BEAR	0x20	R/W	EBCO Bus Error Address Register
EBC0_BESR	0x21	R/W	EBCO Bus Error Status Register 0
EBC0_CFG	0x23	R/W	EBCO Peripheral Control Register
EBC0_CID	0x24	R/W	EBCO Peripheral Core ID Register

Indirect Access of External Bus Master DCRs

The following procedure accesses the EBMI DCRs listed in Table 4-10.

1. Write the offset from Table 4-11 to the EBMI Address Register (EBM0_CFGADDR).
2. Read data from or write data to the EBMI Data Register (EBM0_CFGDATA).

Table 0-8. External Bus Master DCR Usage

Register	DCR Number	Access	Description
EBM0_CFGADDR	0x014	R/W	EBMI Controller Address Register
EBM0_CFGDATA	0x015	R/W	EBMI Controller Data Register

Table 0-9. Offsets for External Bus Master Registers

Register	Offset	Access	Description
EBM0_CTL	0x00	R/W	EBMI Control Register
EBM0_LCNT	0x01	R/W	EBMI OPB Latency Count Register
EBM0_BEAR	0x02	R/W	EBMI Bus Error Address Register
EBM0_BESR	0x03	R/W	EBMI Bus Error Status Register
EBM0_BEMR	0x04	R/W	EBMI Bus Error Mask Register
EBM0_UAR	0x05	R/W	EBMI OPB Upper Address Register
EBM0_UAM	0x06	R/W	EBMI OPB Upper Address Mask
EBM0_SLPMD	0x07	R/W	EBMI Sleep Mode Register
EBM0_FAIR	0x08	R/W	EBMI Fairness Control Register
EBM0_CID	0x11	R	EBMI Core ID Register

~~Indirect Access of PLB Performance Monitor DCRs~~

~~The following procedure accesses the PPM DCRs listed in Table 4-12.~~

~~1. Write the offset from Table 4-13 to the PPM Address Register (PPM0_CFGADDR).~~

~~2. Read data from or write data to the PPM Data Register (PPM0_CFGDATA).~~

Table 0-10. PLB Performance Monitor DCR Usage

Register	DCR Number	Access	Description
PPM0_CFGADDR	0x016	R/W	PPM Address Register
PPM0_CFGDATA	0x017	R/W	PPM Data Register

Table 0-11. Offsets for PLB Performance Monitor Registers

Register	Offset	Access	Description
PPM0_ISR	0x0	Read	Interrupt Status Register
PPM0_CR	0x2	R/W	Control Register
PPM0_CCR	0x3	R/W	Cycle Control Register
PPM0_UAR	0x4	R/W	Upper Address Register
PPM0_LAR	0x5	R/W	Lower Address Register
PPM0_UAMR	0x6	R/W	Upper Address Mask Register
PPM0_LAMR	0x7	R/W	Lower Address Mask Register
PPM0_RIDR	0x8	R	Revision ID Register
PPM0_MCSR0	0x9	R/W	Master Event Counter Selection Register 0

Table 0-11. Offsets for PLB Performance Monitor Registers (continued)

Register	Offset	Access	Description
PPM0_MCSR1	0xA	R/W	Master Event Counter Selection Register 1
PPM0_MCSR2	0xB	R/W	Master Event Counter Selection Register 2
PPM0_MCSR3	0xC	R/W	Master Event Counter Selection Register 3
PPM0_SCSR0	0x11	R/W	Slave Event Counter Selection Register 0
PPM0_SCSR1	0x12	R/W	Slave Event Counter Selection Register 1
PPM0_SCSR2	0x13	R/W	Slave Event Counter Selection Register 2
PPM0_SCSR3	0x14	R/W	Slave Event Counter Selection Register 3
PPM0_GCSR0	0x19	R/W	Generic Event Counter Selection Register 0
PPM0_GCSR1	0x1A	R/W	Generic Event Counter Selection Register 1
PPM0_GCSR2	0x1B	R/W	Generic Event Counter Selection Register 2
PPM0_GCSR3	0x1C	R/W	Generic Event Counter Selection Register 3
PPM0_MCR0	0x1D	R/W	Master Event Counter Register 0
PPM0_MCR1	0x1E	R/W	Master Event Counter Register 1
PPM0_MCR2	0x1F	R/W	Master Event Counter Register 2
PPM0_MCR3	0x20	R/W	Master Event Counter Register 3
PPM0_SCR0	0x25	R/W	Slave Event Counter Register 0
PPM0_SCR1	0x26	R/W	Slave Event Counter Register 1
PPM0_SCR2	0x27	R/W	Slave Event Counter Register 2
PPM0_SCR3	0x28	R/W	Slave Event Counter Register 3
PPM0_GCR0	0x2D	R/W	Generic Pipeline Event Counter Register 0
PPM0_GCR1	0x2E	R/W	Generic Pipeline Event Counter Register 1
PPM0_GCR2	0x2F	R/W	Generic Pipeline Event Counter Register 2
PPM0_GCR3	0x30	R/W	Generic Pipeline Event Counter Register 3
PPM0_DCSR0	0x31	R/W	Duration Counter Selection Register 0
PPM0_DCSR1	0x32	R/W	Duration Counter Selection Register 1
PPM0_DCMXR0	0x33	R/W	Duration Counter Max Register 0
PPM0_DCMXR1	0x34	R/W	Duration Counter Max Register 1
PPM0_DCMNR0	0x35	R/W	Duration Counter Min Register 0
PPM0_DCMNR1	0x36	R/W	Duration Counter Minimum Register 1
PPM0_DCTVR0	0x37	R/W	Duration Counter Total Value Register 0
PPM0_DCTVR1	0x38	R/W	Duration Counter Total Value Register 1
PPM0_DCOTR0	0x39	R/W	Duration Counter Occurrence Total Register 0
PPM0_DCOTR1	0x3A	R/W	Duration Counter Occurrence Total Register 1

Indirectly Accessed DCRs

The DCRs for the DDR-SDRAM controller, external bus controller (EBCO), and external bus master interface (EBMI) are indirectly accessed.

Indirect Access of DDR-SDRAM Controller DCRs

The following procedure accesses the DDR-SDRAM controller DCRs listed in Table 4-6.

1. Write the offset from Table 4-7 to the DDR-SDRAM Address Register (SDRAM0_CFGADDR).
2. Read data from or write data to the DDR-SDRAM Data Register (SDRAM0_CFGDATA).

Table 4-6. SDRAM Controller DCR Usage

Register	DCR Number	Access	Description
SDRAM0_CFGADDR	0x010	R/W	DDR-SDRAM Address Register
SDRAM0_CFGDATA	0x011	R/W	DDR-SDRAM Data Register

Table 4-7. Offsets for SDRAM Controller Registers

Register	Offset	R/W	Description
SDRAM0_BESR0	0x00	R/W	DDR-SDRAM Bus Error Syndrome Register 0
SDRAM0_BESR1	0x08	R/W	DDR-SDRAM Bus Error Syndrome Register 1
SDRAM0_BEAR	0x10	R	DDR-SDRAM Bus Error Address Register
SDRAM0_MIRQ	0x11	R/W	DDR-SDRAM Master Write Interrupt
SDRAM0_SLIO	0x18	R/W	DDR-SDRAM Slave Interface Options
SDRAM0_CFG0	0x20	R/W	DDR-SDRAM Options 0
SDRAM0_CFG1	0x21	R/W	DDR-SDRAM Options 1
SDRAM0_DEVOPT	0x22	R/W	DDR-SDRAM Device Options
SDRAM0_MCSTS	0x24	R	DDR-SDRAM Memory Controller Status
SDRAM0_RTR	0x30	R/W	DDR-SDRAM Refresh Timer Register
SDRAM0_PMIT	0x34	R/W	DDR-SDRAM Power Management Idle Timer
SDRAM0_UABBA	0x38	R/W	DDR-SDRAM PLB UA Bus Base Address
SDRAM0_B0CR	0x40	R/W	DDR-SDRAM Bank 0 Configuration Register
SDRAM0_B1CR	0x44	R/W	DDR-SDRAM Bank 1 Configuration Register
SDRAM0_B2CR	0x48	R/W	DDR-SDRAM Bank 2 Configuration Register
SDRAM0_B3CR	0x4C	R/W	DDR-SDRAM Bank 3 Configuration Register
SDRAM0_TR0	0x80	R/W	DDR-SDRAM Timing Register 0
SDRAM0_TR1	0x81	R/W	DDR-SDRAM Timing Register 1
SDRAM0_CLKTR	0x82	R/W	DDR-SDRAM Clock Timing Register
SDRAM0_WDDCTR	0x83	R/W	DDR-SDRAM Write Data, DQS, DM Clock Timing Register
SDRAM0_DLYCAL	0x84	R/W	DDR-SDRAM Delay Line Calibration Register

Table 4-7. Offsets for SDRAM Controller Registers

Register	Offset	R/W	Description
SDRAM0_ECCESR	0x98	R/W	DDR-SDRAM ECC Error Status Register
SDRAM0_CID	0xA4	R	DDR-SDRAM Core ID Register
SDRAM0_RID	0xA8	R	DDR-SDRAM Revision ID Register

Indirect Access of External Bus Controller DCRs

The following procedure accesses the EBCO DCRs listed in Table 4-8.

1. Write the offset from Table 4-9 to the EBCO Address Register (EBC0_CFGADDR).
2. Read data from or write data to the EBCO Data Register (EBC0_CFGDATA).

Table 4-8. External Bus Controller DCR Usage

Register	DCR Number	Access	Description
EBC0_CFGADDR	0x012	R/W	EBCO Controller Address Register
EBC0_CFGDATA	0x013	R/W	EBCO Controller Data Register

Indirect Access of External Bus Controller DCRs

Table 4-9. Offsets for External Bus Controller Registers

Register	Offset	Access	Description
EBC0_B0CR	0x00	R/W	EBCO Bank 0 Configuration Register
EBC0_B1CR	0x01	R/W	EBCO Bank 1 Configuration Register
EBC0_B2CR	0x02	R/W	EBCO Bank 2 Configuration Register
EBC0_B3CR	0x03	R/W	EBCO Bank 3 Configuration Register
EBC0_B4CR	0x04	R/W	EBCO Bank 4 Configuration Register
EBC0_B5CR	0x05	R/W	EBCO Bank 5 Configuration Register
EBC0_B6CR	0x06	R/W	EBCO Bank 6 Configuration Register
EBC0_B7CR	0x07	R/W	EBCO Bank 7 Configuration Register
EBC0_B0AP	0x10	R/W	EBCO Bank 0 Access Parameters
EBC0_B1AP	0x11	R/W	EBCO Bank 1 Access Parameters
EBC0_B2AP	0x12	R/W	EBCO Bank 2 Access Parameters
EBC0_B3AP	0x13	R/W	EBCO Bank 3 Access Parameters
EBC0_B4AP	0x14	R/W	EBCO Bank 4 Access Parameters
EBC0_B5AP	0x15	R/W	EBCO Bank 5 Access Parameters
EBC0_B6AP	0x16	R/W	EBCO Bank 6 Access Parameters
EBC0_B7AP	0x17	R/W	EBCO Bank 7 Access Parameters
EBC0_BEAR	0x20	R/W	EBCO Bus Error Address Register
EBC0_BESR	0x21	R/W	EBCO Bus Error Status Register 0
EBC0_CFG	0x23	R/W	EBCO Peripheral Control Register
EBC0_CID	0x24	R/W	EBCO Peripheral Core ID Register

Indirect Access of External Bus Master DCRs

The following procedure accesses the EBMI DCRs listed in Table 4-10.

1. Write the offset from Table 4-11 to the EBMI Address Register (EBM0_CFGADDR).
2. Read data from or write data to the EBMI Data Register (EBM0_CFGDATA).

Table 4-10. External Bus Master DCR Usage

Register	DCR Number	Access	Description
EBM0_CFGADDR	0x014	R/W	EBMI Controller Address Register
EBM0_CFGDATA	0x015	R/W	EBMI Controller Data Register

Table 4-11. Offsets for External Bus Master Registers

Register	Offset	Access	Description
EBM0_CTL	0x00	R/W	EBMI Control Register
EBM0_LCNT	0x01	R/W	EBMI OPB Latency Count Register
EBM0_BEAR	0x02	R/W	EBMI Bus Error Address Register
EBM0_BESR	0x03	R/W	EBMI Bus Error Status Register
EBM0_BEMR	0x04	R/W	EBMI Bus Error Mask Register
EBM0_UAR	0x05	R/W	EBMI OPB Upper Address Register
EBM0_UAM	0x06	R/W	EBMI OPB Upper Address Mask
EBM0_SLPMD	0x07	R/W	EBMI Sleep Mode Register
EBM0_FAIR	0x08	R/W	EBMI Fairness Control Register
EBM0_CID	0x11	R	EBMI Core ID Register

Indirect Access of PLB Performance Monitor DCRs

The following procedure accesses the PPM DCRs listed in Table 4-12.

1. Write the offset from Table 4-13 to the PPM Address Register (PPM0_CFGADDR).
2. Read data from or write data to the PPM Data Register (PPM0_CFGDATA).

Table 4-12. PLB Performance Monitor DCR Usage

Register	DCR Number	Access	Description
PPM0_CFGADDR	0x016	R/W	PPM Address Register
PPM0_CFGDATA	0x017	R/W	PPM Data Register

Table 4-13. Offsets for PLB Performance Monitor Registers

Register	Offset	Access	Description
PPM0_ISR	0x0	Read	Interrupt Status Register
PPM0_CR	0x2	R/W	Control Register
PPM0_CCR	0x3	R/W	Cycle Control Register
PPM0_UAR	0x4	R/W	Upper Address Register
PPM0_LAR	0x5	R/W	Lower Address Register
PPM0_UAMR	0x6	R/W	Upper Address Mask Register
PPM0_LAMR	0x7	R/W	Lower Address Mask Register
PPM0_RIDR	0x8	R	Revision ID Register
PPM0_MCSR0	0x9	R/W	Master Event Counter Selection Register 0
PPM0_MCSR1	0xA	R/W	Master Event Counter Selection Register 1
PPM0_MCSR2	0xB	R/W	Master Event Counter Selection Register 2
PPM0_MCSR3	0xC	R/W	Master Event Counter Selection Register 3
PPM0_SCSR0	0x11	R/W	Slave Event Counter Selection Register 0

Table 4-13. Offsets for PLB Performance Monitor Registers

Register	Offset	Access	Description
PPM0_SCSR1	0x12	R/W	Slave Event Counter Selection Register 1
PPM0_SCSR2	0x13	R/W	Slave Event Counter Selection Register 2
PPM0_SCSR3	0x14	R/W	Slave Event Counter Selection Register 3
PPM0_GCSR0	0x19	R/W	Generic Event Counter Selection Register 0
PPM0_GCSR1	0x1A	R/W	Generic Event Counter Selection Register 1
PPM0_GCSR2	0x1B	R/W	Generic Event Counter Selection Register 2
PPM0_GCSR3	0x1C	R/W	Generic Event Counter Selection Register 3
PPM0_MCR0	0x1D	R/W	Master Event Counter Register 0
PPM0_MCR1	0x1E	R/W	Master Event Counter Register 1
PPM0_MCR2	0x1F	R/W	Master Event Counter Register 2
PPM0_MCR3	0x20	R/W	Master Event Counter Register 3
PPM0_SCR0	0x25	R/W	Slave Event Counter Register 0
PPM0_SCR1	0x26	R/W	Slave Event Counter Register 1
PPM0_SCR2	0x27	R/W	Slave Event Counter Register 2
PPM0_SCR3	0x28	R/W	Slave Event Counter Register 3
PPM0_GCR0	0x2D	R/W	Generic Pipeline Event Counter Register 0
PPM0_GCR1	0x2E	R/W	Generic Pipeline Event Counter Register 1
PPM0_GCR2	0x2F	R/W	Generic Pipeline Event Counter Register 2
PPM0_GCR3	0x30	R/W	Generic Pipeline Event Counter Register 3
PPM0_DCSR0	0x31	R/W	Duration Counter Selection Register 0
PPM0_DCSR1	0x32	R/W	Duration Counter Selection Register 1
PPM0_DCMXR0	0x33	R/W	Duration Counter Max Register 0
PPM0_DCMXR1	0x34	R/W	Duration Counter Max Register 1
PPM0_DCMNR0	0x35	R/W	Duration Counter Min Register 0
PPM0_DCMNR1	0x36	R/W	Duration Counter Minimum Register 1
PPM0_DCTVR0	0x37	R/W	Duration Counter Total Value Register 0
PPM0_DCTVR1	0x38	R/W	Duration Counter Total Value Register 1
PPM0_DCOTR0	0x39	R/W	Duration Counter Occurrence Total Register 0
PPM0_DCOTR1	0x3A	R/W	Duration Counter Occurrence Total Register 1

4.2.1.6 Memory-Mapped Input/Output Registers

Some registers associated with on-chip peripherals are memory-mapped input/output (MMIO) registers. Such registers are mapped into the system memory space and are accessed using load/store instructions that contain the register addresses. The tables that follow list all MMIO registers.

Table 0-12. MMIO Registers

Register	Address	Access	Description
Serial Port 0			
UART0_RBR	0x1 4000 0200	R	UART 0 Receiver Buffer Register Note: Set UART0_LCR[DLAB] = 0 to access.
UART0_THR		W	UART 0 Transmitter Holding Register Note: Set UART0_LCR[DLAB] = 0 to access.
UART0_DLL		R/W	UART 0 Baud-rate Divisor Latch LSB Note: Set UART0_LCR[DLAB] = 1 to access.
UART0_IER	0x1 4000 0201	R/W	UART 0 Interrupt Enable Register Note: Set UART0_LCR[DLAB] = 0 to access.
UART0_DLM		R/W	UART 0 Baud-rate Divisor Latch MSB Note: Set UART0_LCR[DLAB] = 1 to access.
UART0_IIR	0x1 4000 0202	R	UART 0 Interrupt Identification Register
UART0_FCR	0x1 4000 0202	W	UART 0 FIFO Control Register
UART0_LCR	0x1 4000 0203	R/W	UART 0 Line Control Register
UART0_MCR	0x1 4000 0204	R/W	UART 0 Modem Control Register
UART0_LSR	0x1 4000 0205	R/W	UART 0 Line Status Register
UART0_MSR	0x1 4000 0206	R/W	UART 0 Modem Status Register
UART0_SCR	0x1 4000 0207	R/W	UART 0 Scratch Register
Serial Port 1			
UART1_RBR	0x1 4000 0300	R	UART 1 Receiver Buffer Register Note: Set UART1_LCR[DLAB] = 0 to access.
UART1_THR		W	UART 1 Transmitter Holding Register Note: Set UART1_LCR[DLAB] = 0 to access.
UART1_DLL		R/W	UART 1 Baud-rate Divisor Latch LSB Note: Set UART1_LCR[DLAB] = 1 to access.
UART1_IER	0x1 4000 0301	R/W	UART 1 Interrupt Enable Register Note: Set UART1_LCR[DLAB] = 0 to access.
UART1_DLM		R/W	UART 1 Baud-rate Divisor Latch MSB Note: Set UART1_LCR[DLAB] = 1 to access.
UART1_IIR	0x1 4000 0302	R	UART 1 Interrupt Identification Register
UART1_FCR	0x1 4000 0302	W	UART 1 FIFO Control Register

Table 0-12. MMIO Registers (continued)

Register	Address	Access	Description
UART1_LCR	0x1 40000303	R/W	UART 1 Line Control Register
UART1_MCR	0x1 40000304	R/W	UART 1 Modem Control Register
UART1_LSR	0x1 40000305	R/W	UART 1 Line Status Register
UART1_MSR	0x1 40000306	R/W	UART 1 Modem Status Register
UART1_SCR	0x1 40000307	R/W	UART 1 Scratch Register
Inter-Integrated Circuit 0			
IIC0_MDBUF	0x1 40000400	R/W	IIC 0 Master Data Buffer
IIC0_SDBUF	0x1 40000402	R/W	IIC 0 Slave Data Buffer
IIC0_LMADR	0x1 40000404	R/W	IIC 0 Low Master Address
IIC0_HMADR	0x1 40000405	R/W	IIC 0 High Master Address
IIC0_CNTL	0x1 40000406	R/W	IIC 0 Control
IIC0_MDCNTL	0x1 40000407	R/W	IIC 0 Mode Control
IIC0_STS	0x1 40000408	R/W	IIC 0 Status
IIC0_EXTSTS	0x1 40000409	R/W	IIC 0 Extended Status
IIC0_LSADR	0x1 4000040A	R/W	IIC 0 Low Slave Address
IIC0_HSADR	0x1 4000040B	R/W	IIC 0 High Slave Address
IIC0_CLKDIV	0x1 4000040C	R/W	IIC 0 Clock Divide
IIC0_INTRMSK	0x1 4000040D	R/W	IIC 0 Interrupt Mask
IIC0_XFRCNT	0x1 4000040E	R/W	IIC 0 Transfer Count
IIC0_XTCNTLSS	0x1 4000040F	R/W	IIC 0 Extended Control and Slave Status
IIC0_DIRECTCNTL	0x1 40000410	R/W	IIC 0 Direct Control
Inter-Integrated Circuit 1			
IIC1_MDBUF	0x1 40000500	R/W	IIC 1 Master Data Buffer
IIC1_SDBUF	0x1 40000502	R/W	IIC 1 Slave Data Buffer
IIC1_LMADR	0x1 40000504	R/W	IIC 1 Low Master Address
IIC1_HMADR	0x1 40000505	R/W	IIC 1 High Master Address
IIC1_CNTL	0x1 40000506	R/W	IIC 1 Control
IIC1_MDCNTL	0x1 40000507	R/W	IIC 1 Mode Control
IIC1_STS	0x1 40000508	R/W	IIC 1 Status
IIC1_EXTSTS	0x1 40000509	R/W	IIC 1 Extended Status
IIC1_LSADR	0x1 4000050A	R/W	IIC 1 Low Slave Address
IIC1_HSADR	0x1 4000050B	R/W	IIC 1 High Slave Address

Table 0-12. MMIO Registers (continued)

Register	Address	Access	Description
IIC1_CLKDIV	0x1 4000050C	R/W	IIC 1 Clock Divide
IIC1_INTRMSK	0x1 4000050D	R/W	IIC 1 Interrupt Mask
IIC1_XFRCNT	0x1 4000050E	R/W	IIC 1 Transfer Count
IIC1_XTCNTLSS	0x1 4000050F	R/W	IIC 1 Extended Control and Slave Status
IIC1_DIRECTCNTL	0x1 40000510	R/W	IIC 1 Direct Control
OPB Arbiter			
OPBA0_PR	0x1 40000600	R/W	OPB Arbiter Priority Register
OPBA0_CR	0x1 40000601	R/W	OPB Arbiter Control Register
General-Purpose I/O			
GPIO0_OR	0x1 40000700	R/W	GPIO Output Register
GPIO0_TCR	0x1 40000704	R/W	GPIO Three-State Control Register
GPIO0_ODR	0x1 40000718	R/W	GPIO Open Drain Register
GPIO0_IR	0x1 4000071C	R	GPIO Input Register
ZMII			
ZMII0_FER	0x1 40000780	R/W	ZMII Function Enable Register
ZMII0_SSR	0x1 40000784	R/W	ZMII Speed Select Register
ZMII0_SMIISR	0x1 40000788	R/W	ZMII SMII Status Register
Ethernet MAC 0			
EMAC0_MR0	0x1 40000800	R/W	EMAC 0 Mode Register 0
EMAC0_MR1	0x1 40000804	R/W	EMAC 0 Mode Register 1
EMAC0_TMR0	0x1 40000808	R/W	EMAC 0 Transmit Mode Register 0
EMAC0_TMR1	0x1 4000080C	R/W	EMAC 0 Transmit Mode Register 1
EMAC0_RMR	0x1 40000810	R/W	EMAC 0 Receive Mode Register
EMAC0_ISR	0x1 40000814	R/W	EMAC 0 Interrupt Status Register
EMAC0_ISER	0x1 40000818	R/W	EMAC 0 Interrupt Status Enable Register
EMAC0_IAHR	0x1 4000081C	R/W	EMAC 0 Individual Address High
EMAC0_IALR	0x1 40000820	R/W	EMAC 0 Individual Address Low
EMAC0_VTPID	0x1 40000824	R/W	EMAC 0 VLAN TPID Register
EMAC0_VTCI	0x1 40000828	R/W	EMAC 0 VLAN TCI Register

Table 0-12. MMIO Registers (continued)

Register	Address	Access	Description
EMAC0_PTR	0x1 4000082C	R/W	EMAC 0 Pause Timer Register
EMAC0_IAHT1	0x1 40000830	R/W	EMAC 0 Individual Address Hash Table 1
EMAC0_IAHT2	0x1 40000834	R/W	EMAC 0 Individual Address Hash Table 2
EMAC0_IAHT3	0x1 40000838	R/W	EMAC 0 Individual Address Hash Table 3
EMAC0_IAHT4	0x1 4000083C	R/W	EMAC 0 Individual Address Hash Table 4
EMAC0_GAHT1	0x1 40000840	R/W	EMAC 0 Group Address Hash Table 1
EMAC0_GAHT2	0x1 40000844	R/W	EMAC 0 Group Address Hash Table 2
EMAC0_GAHT3	0x1 40000848	R/W	EMAC 0 Group Address Hash Table 3
EMAC0_GAHT4	0x1 4000084C	R/W	EMAC 0 Group Address Hash Table 4
EMAC0_LSAH	0x1 40000850	R	EMAC 0 Last Source Address Low
EMAC0_LSAL	0x1 40000854	R	EMAC 0 Last Source Address High
EMAC0_IPGVR	0x1 40000858	R/W	EMAC 0 Inter-Packet Gap Value Register
EMAC0_STACR	0x1 4000085C	R/W	EMAC 0 STA Control Register
EMAC0_TRTR	0x1 40000860	R/W	EMAC 0 Transmit Request Threshold Register
EMAC0_RWMR	0x1 40000864	R/W	EMAC 0 Receive Low/High Water Mark Register
EMAC0_OCTX	0x1 40000868	R	EMAC 0 Number of Octets Transimitted
EMAC0_OCRX	0x1 4000086C	R	EMAC 0 Number of Octets Received
Ethernet MAC 1			
EMAC1_MR0	0x1 40000900	R/W	EMAC 1 Mode Register 0
EMAC1_MR1	0x1 40000904	R/W	EMAC 1 Mode Register 1
EMAC1_TMR0	0x1 40000908	R/W	EMAC 1 Transmit Mode Register 0
EMAC1_TMR1	0x1 4000090C	R/W	EMAC 1 Transmit Mode Register 1
EMAC1_RMR	0x1 40000910	R/W	EMAC 1 Receive Mode Register
EMAC1_ISR	0x1 40000914	R/W	EMAC 1 Interrupt Status Register
EMAC1_ISER	0x1 40000918	R/W	EMAC 1 Interrupt Status Enable Register
EMAC1_IAHR	0x1 4000091C	R/W	EMAC 1 Individual Address High
EMAC1_IALR	0x1 40000920	R/W	EMAC 1 Individual Address Low
EMAC1_VTPID	0x1 40000924	R/W	EMAC 1 VLAN TPID Register

Table 0-12. MMIO Registers (continued)

Register	Address	Access	Description
EMAC1_VTCI	0x1 40000928	R/W	EMAC 1 VLAN TCI Register
EMAC1_PTR	0x1 4000092C	R/W	EMAC 1 Pause Timer Register
EMAC1_IAHT1	0x1 40000930	R/W	EMAC 1 Individual Address Hash Table 1
EMAC1_IAHT2	0x1 40000934	R/W	EMAC 1 Individual Address Hash Table 2
EMAC1_IAHT3	0x1 40000938	R/W	EMAC 1 Individual Address Hash Table 3
EMAC1_IAHT4	0x1 4000093C	R/W	EMAC 1 Individual Address Hash Table 4
EMAC1_GAHT1	0x1 40000940	R/W	EMAC 1 Group Address Hash Table 1
EMAC1_GAHT2	0x1 40000944	R/W	EMAC 1 Group Address Hash Table 2
EMAC1_GAHT3	0x1 40000948	R/W	EMAC 1 Group Address Hash Table 3
EMAC1_GAHT4	0x1 4000094C	R/W	EMAC 1 Group Address Hash Table 4
EMAC1_LSAH	0x1 40000950	R	EMAC 1 Last Source Address Low
EMAC1_LSAL	0x1 40000954	R	EMAC 1 Last Source Address High
EMAC1_IPGVR	0x1 40000958	R/W	EMAC 1 Inter-Packet Gap Value Register
EMAC1_STACR	0x1 4000095C	R/W	EMAC 1 STA Control Register
EMAC1_TRTR	0x1 40000960	R/W	EMAC 1 Transmit Request Threshold Register
EMAC1_RWMR	0x1 40000964	R/W	EMAC 1 Receive Low/High Water Mark Register
EMAC1_OCTX	0x1 40000968	R	EMAC 1 Number of Octets Transimitted
EMAC1_OCRX	0x1 4000096C	R	EMAC 1 Number of Octets Received
General Purpose Timer			
GPT0_TBC	0x1 40000A00	R/W	GPT Time Base Counter
GPT0_OE	0x1 40000A10	R/W	GPT Output Enable
GPT0_OL	0x1 40000A14	R/W	GPT Output Level
GPT0_IM	0x1 40000A18	R/W	GPT Interrupt Mask
GPT0_ISS	0x1 40000A1C	R/W	GPT Interrupt Status (Set bits if write 1)
GPT0_ISC	0x1 40000A20	R/W	GPT Interrupt Status (Clear bits if write 1)

Table 0-12. MMIO Registers (continued)

Register	Address	Access	Description
GPT0_IE	0x1 40000A24	R/W	GPT Interrupt Enable
GPT0_COMP0	0x1 40000A80	R/W	GPT Compare Timer 0
GPT0_COMP1	0x1 40000A84	R/W	GPT Compare Timer 1
GPT0_COMP2	0x1 40000A88	R/W	GPT Compare Timer 2
GPT0_COMP3	0x1 40000A8C	R/W	GPT Compare Timer 3
GPT0_COMP4	0x1 40000A90	R/W	GPT Compare Timer 4
GPT0_MASK0	0x1 40000AC0	R/W	GPT Compare Mask 0
GPT0_MASK1	0x1 40000AC4	R/W	GPT Compare Mask 1
GPT0_MASK2	0x1 40000AC8	R/W	GPT Compare Mask 2
GPT0_MASK3	0x1 40000ACC	R/W	GPT Compare Mask 3
GPT0_MASK4	0x1 40000AD0	R/W	GPT Compare Mask 4

~~The two registers listed in Table 4-15 are used to access the configuration registers of external PCI-X devices only.~~

Table 0-13. External PCI-X Device Configuration Register Access

Register	Address	Access	Description
PCIX0_CFGADDR	0x2 0EC00000	R/W	PCI-X Configuration Address Register
PCIX0_CFGDATA	0x2 0EC00004	R/W	PCI-X Configuration Data Register

~~Registers listed in Table 4-16 can be accessed in two ways:~~

1. By the processor using the PLB address
2. By an external PCI-X device using configuration cycles to the proper offset.

Table 0-14. Internal PCI-X Configuration Registers

Register	PLB		PCI-X Configuration		Description
	Address	Access	Offset	Access	
PCIX0_VENDID	0x2 0EC80000	R/W	0x01–0x00	R	PCI-X Vendor ID
PCIX0_DEVID	0x2 0EC80002	R/W	0x03–0x02	R	PCI-X Device ID
PCIX0_CMD	0x2 0EC80004	R/W	0x05–0x04	R/W	PCI-X Command Register
PCIX0_STATUS	0x2 0EC80006	R/W	0x07–0x06	R/W	PCI-X Status Register
PCIX0_REVID	0x2 0EC80008	R/W	0x08	R	PCI-X Revision ID
PCIX0_CLS	0x2 0EC80009	R/W	0x0B–0x09	R	PCI-X Class Register
PCIX0_CACHELS	0x2 0EC8000C	R/W	0x0C	R/W	PCI-X Cache Line Size
PCIX0_LATTIM	0x2 0EC8000D	R/W	0x0D	R/W	PCI-X Latency Timer
PCIX0_HDTYPE	0x2 0EC8000E	R	0x0E	R	PCI-X Header Type
PCIX0_BIST	0x2 0EC8000F	R	0x0F	R	PCI-X Built In Self Test Control
PCIX0_ BAR0L	0x2 0EC80010	R/W	0x13–0x10	R/W	PCI-X BAR 0 Low
PCIX0_ BAR0H	0x2 0EC80014	R/W	0x17–0x14	R/W	PCI-X BAR 0 High
PCIX0_BAR1	0x2 0EC80018	R/W	0x1B–0x18	R/W	PCI-X BAR 1
PCIX0_ BAR2L	0x2 0EC8001C	R/W	0x1F–0x1C	R/W	PCI-X BAR 2 Low
PCIX0_ BAR2H	0x2 0EC80020	R/W	0x23–0x20	R/W	PCI-X BAR 2 High
PCIX0_ BAR3	0x2 0EC80024	R	0x27–0x24	R	Unused BAR 3
PCIX0_CISPTR	0x2 0EC80028	R	0x2B–0x28	R	Unused Cardbus CIS Pointer
PCIX0_SBSYSVID	0x2 0EC8002C	R/W	0x2D–0x2C	R	PCI-X Subsystem Vendor ID
PCIX0_SBSYSID	0x2 0EC8002E	R/W	0x2F–0x2E	R	PCI-X Subsystem ID
PCIX0_EROMBA	0x2 0EC80030	R/W	0x33–0x30	R/W	PCI-X Expansion ROM Base Address
PCIX0_CAP	0x2 0EC80034	R	0x34	R	PCI-X Capabilities Pointer
PCIX0_RES0	0x2 0EC80035	R	0x35	R	Reserved
PCIX0_RES1	0x2 0EC80036	R	0x37–0x6	R	Reserved
PCIX0_RES2	0x2 0EC80038	R	0x3B–0x8	R	Reserved

Table 0-14. Internal PCI-X Configuration Registers (continued)

Register	PLB		PCI-X Configuration		Description
	Address	Access	Offset	Access	
PCIX0_INTLN	0x2 0EC8003C	R/W	0x3C	R/W	PCI-X Interrupt Line
PCIX0_INTPN	0x2 0EC8003D	R	0x3D	R	PCI-X Interrupt Pin
PCIX0_MINGNT	0x2 0EC8003E	R	0x3E	R	PCI-X Minimum Grant
PCIX0_MAXLTNCY	0x2 0EC8003F	R	0x3F	R	PCI-X Maximum Latency
PCIX0_BRDGOPT1	0x2 0EC80040	R/W	0x43–0x40	R/W	PCI-X Bridge Options 1
PCIX0_BRDGOPT2	0x2 0EC80044	R/W	0x47–0x44	R/W	PCI-X Bridge Options 2
PCIX0_ERREN	0x2 0EC80050	R/W	0x53–0x50	R/W	PCI-X Error Enable
PCIX0_ERRSTS	0x2 0EC80054	R/W	0x57–0x54	R/W	PCI-X Error Status
PCIX0_PLBBESR	0x2 0EC80058	R	0x5B–0x58	R	PCI-X PLB Slave Error Syndrome Register
PCIX0_PLBBEARL	0x2 0EC8005C	R	0x5F–0x5C	R	PCI-X PLB Slave Error Address Register Low
PCIX0_PLBBEARH	0x2 0EC80060	R	0x63–0x60	R	PCI-X PLB Slave Error Address Register High
PCIX0_POM0LAL	0x2 0EC80068	R/W	0x6B–0x68	R/W	PCI-X POM0 Local Address Low
PCIX0_POM0LAH	0x2 0EC8006C	R/W	0x6F–0x6C	R/W	PCI-X POM0 Local Address High
PCIX0_POM0SA	0x2 0EC80070	R/W	0x73–0x70	R/W	PCI-X POM0 Size Attribute
PCIX0_POM0PCIAL	0x2 0EC80074	R/W	0x77–0x74	R/W	PCI-X POM0 PCI Address Low
PCIX0_POM0PCIAH	0x2 0EC80078	R/W	0x7B–0x78	R/W	PCI-X POM0 PCI Address High
PCIX0_POM1LAL	0x2 0EC8007C	R/W	0x7F–0x7C	R/W	PCI-X POM1 Local Address Low
PCIX0_POM1LAH	0x2 0EC80080	R/W	0x83–0x80	R/W	PCI-X POM1 Local Address High
PCIX0_POM1SA	0x2 0EC80084	R/W	0x87–0x84	R/W	PCI-X POM1 Size Attribute
PCIX0_POM1PCIAL	0x2 0EC80088	R/W	0x8B–0x88	R/W	PCI-X POM1 PCI Address Low
PCIX0_POM1PCIAH	0x2 0EC8008C	R/W	0x8F–0x8C	R/W	PCI-X POM1 PCI Address High
PCIX0_POM2SA	0x2 0EC80090	R/W	0x93–0x90	R/W	PCI-X POM2 Size Attribute
PCIX0_PIM0SA	0x2 0EC80098	R/W	0x9B–0x98	R/W	PCI-X PIM0 Size/Attribute

Table 0-14. Internal PCI-X Configuration Registers (continued)

Register	PLB		PCI-X Configuration		Description
	Address	Access	Offset	Access	
PCIX0_PIM0LAL	0x2 0EC8009C	R/W	0x9F–0x9C	R/W	PCI-X PIM0 Local Address Low
PCIX0_PIM0LAH	0x2 0EC800A0	R/W	0xA3–0xA0	R/W	PCI-X PIM0 Local Address High
PCIX0_PIM1SA	0x2 0EC800A4	R/W	0xA7–0xA4	R/W	PCI-X PIM1 Size/Attribute
PCIX0_PIM1LAL	0x2 0EC800A8	R/W	0xAB–0xA8	R/W	PCI-X PIM1 Local Address Low
PCIX0_PIM1LAH	0x2 0EC800AC	R/W	0xAF–0xAC	R/W	PCI-X PIM1 Local Address High
PCIX0_PIM2SA	0x2 0EC800B0	R/W	0xB3–0xB0	R/W	PCI-X PIM2 Size/Attribute
PCIX0_PIM2LAL	0x2 0EC800B4	R/W	0xB7–0xB4	R/W	PCI-X PIM2 Local Address Low
PCIX0_PIM2LAH	0x2 0EC800B8	R/W	0xBB–0xB8	R/W	PCI-X PIM2 Local Address High
PCIX0_OMCAPID	0x2 0EC800C0	R	0xC0	R	PCI-X Outbound MSI Capability Identifier
PCIX0_OMNIPTR	0x2 0EC800C1	R/W	0xC1	R	PCI-X Outbound MSI Next Item Pointer
PCIX0_OMMC	0x2 0EC800C2	R/W	0xC3–0xC2	R/W	PCI-X Outbound MSI Message Control
PCIX0_OMMA	0x2 0EC800C4	R/W	0xC7–0xC4	R/W	PCI-X Outbound MSI Message Address
PCIX0_OMMUA	0x2 0EC800C8	R/W	0xCB–0xC8	R/W	PCI-X Outbound MSI Message Upper Address
PCIX0_OMMDATA	0x2 0EC800CC	R/W	0xCD–0xCC	R/W	PCI-X Outbound MSI Message Data
PCIX0_OMMEOI	0x2 0EC800CE	R/W	0xCE	R/W	PCI-X Outbound MSI Message End Of Interrupt
PCIX0_PMCAPID	0x2 0EC800D0	R/W	0xD0	R	PCI-X PMC Capability Identifier
PCIX0_PMNIPTR	0x2 0EC800D1	R/W	0xD1	R	PCI-X PMC Next Item Pointer
PCIX0_PMC	0x2 0EC800D2	R	0xD3–0xD2	R	PCI-X Power Management Capabilities
PCIX0_PMCSR	0x2 0EC800D4	R/W	0xD5–0xD4	R/W	PCI-X Power Management Control Status

Table 0-14. Internal PCI-X Configuration Registers (continued)

Register	PLB		PCI-X Configuration		Description
	Address	Access	Offset	Access	
PCIX0_PMCSRBSE	0x2 0EC800D6	R	0xD6	R	PCI-X PMCSR PCI to PCI Bridge Support Extensions
PCIX0_PMDATA	0x2 0EC800D7	R	0xD7	R	PCI-X PMC Unused Data Register
PCIX0_PMSCRR	0x2 0EC800D8	R/W	0xD8	R/W	PCI-X Power Management State Change Request Register
PCIX0_PCIXCAPID	0x2 0EC800DC	R	0xDC	R	PCI-X Capability Identifier
PCIX0_PCIXNIPTR	0x2 0EC800DD	R	0xDD	R	PCI-X Next Item Pointer
PCIX0_PCIXCMD	0x2 0EC800DE	R/W	0xDF–0xDE	R/W	PCI-X Command
PCIX0_PCIXSTS	0x2 0EC800E0	R/W	0xE3–0xE0	R/W	PCI-X Status
PCIX0_PCIXIDR	0x2 0EC800E4	R/W	0xE7–0xE4	R/W	PCI-X Internal Debug Register
PCIX0_PCIXCID	0x2 0EC800E8	R	0xEB–0xE8	R	PCI-X Internal Core Device ID
PCIX0_PCIXRID	0x2 0EC800EC	R	0xEF–0xEC	R	PCI-X Internal Core Revision ID

~~Registers listed in Table 4-17 can be accessed in two ways:~~

~~1. By the processor using the PLB address~~

~~2. By an external PCI-X device using memory cycles to the proper address.~~

Table 0-15. PCI-X SImple Message Passing and Inbound MSI Registers

Register	PLB		PCI-X Memory		Description
	Address	Access	Address	Access	
PCIX0_MSGIL	0x2 0EC80100	R/W	**(PCIX0_BAR0H** || PCIX0_BAR0L) + (0x03–0x00)	R/W	PCI-X Message In Low

Table 0-15. PCI-X SImple Message Passing and Inbound MSI Registers (continued)

Register	PLB		PCI-X Memory		Description
	Address	Access	Address	Access	
PCIX0_MSGIH	0x2 0EC80104	R/W	(PCIX0_BAR0H || PCIX0_BAR0L) + (0x07–0x04)	R/W	PCI-X Message In High
PCIX0_MSGOL	0x2 0EC80108	R/W	(PCIX0_BAR0H || PCIX0_BAR0L) + (0x0B–0x08)	R/W	PCI-X Message Out Low
PCIX0_MSGOH	0x2 0EC8010C	R/W	(PCIX0_BAR0H || PCIX0_BAR0L) + (0x0F–0x0C)	R/W	PCI-X Message Out High
PCIX0_IM	0x2 0EC801F8	W	(PCIX0_BAR0H || PCIX0_BAR0L) + (0xFB–0xF8)	W	PCI-X Inbound MSI

Some registers associated with on-chip peripherals are memory-mapped input/output (MMIO) registers. Such registers are mapped into the system memory space and are accessed using load/store instructions that contain the register addresses. The tables that follow list all MMIO registers.

Table 4-14. MMIO Registers

Register	Address	Access	Description
Serial Port 0			
UART0_RBR	0x1 4000 0200	R	UART 0 Receiver Buffer Register **Note:** Set UART0_LCR[DLAB] = 0 to access.
UART0_THR		W	UART 0 Transmitter Holding Register **Note:** Set UART0_LCR[DLAB] = 0 to access.
UART0_DLL		R/W	UART 0 Baud-rate Divisor Latch LSB **Note:** Set UART0_LCR[DLAB] = 1 to access.
UART0_IER	0x1 4000 0201	R/W	UART 0 Interrupt Enable Register **Note:** Set UART0_LCR[DLAB] = 0 to access.
UART0_DLM		R/W	UART 0 Baud-rate Divisor Latch MSB **Note:** Set UART0_LCR[DLAB] = 1 to access.
UART0_IIR	0x1 4000 0202	R	UART 0 Interrupt Identification Register
UART0_FCR	0x1 4000 0202	W	UART 0 FIFO Control Register
UART0_LCR	0x1 4000 0203	R/W	UART 0 Line Control Register
UART0_MCR	0x1 4000 0204	R/W	UART 0 Modem Control Register
UART0_LSR	0x1 4000 0205	R/W	UART 0 Line Status Register
UART0_MSR	0x1 4000 0206	R/W	UART 0 Modem Status Register
UART0_SCR	0x1 4000 0207	R/W	UART 0 Scratch Register

Table 4-14. MMIO Registers

Register	Address	Access	Description
Serial Port 1			
UART1_RBR	0x1 4000 0300	R	UART 1 Receiver Buffer Register **Note:** Set UART1_LCR[DLAB] = 0 to access.
UART1_THR		W	UART 1 Transmitter Holding Register **Note:** Set UART1_LCR[DLAB] = 0 to access.
UART1_DLL		R/W	UART 1 Baud-rate Divisor Latch LSB **Note:** Set UART1_LCR[DLAB] = 1 to access.
UART1_IER	0x1 4000 0301	R/W	UART 1 Interrupt Enable Register **Note:** Set UART1_LCR[DLAB] = 0 to access.
UART1_DLM		R/W	UART 1 Baud-rate Divisor Latch MSB **Note:** Set UART1_LCR[DLAB] = 1 to access.
UART1_IIR	0x1 4000 0302	R	UART 1 Interrupt Identification Register
UART1_FCR	0x1 4000 0302	W	UART 1 FIFO Control Register
UART1_LCR	0x1 4000 0303	R/W	UART 1 Line Control Register
UART1_MCR	0x1 4000 0304	R/W	UART 1 Modem Control Register
UART1_LSR	0x1 4000 0305	R/W	UART 1 Line Status Register
UART1_MSR	0x1 4000 0306	R/W	UART 1 Modem Status Register
UART1_SCR	0x1 4000 0307	R/W	UART 1 Scratch Register
Inter-Integrated Circuit 0			
IIC0_MDBUF	0x1 40000400	R/W	IIC 0 Master Data Buffer
IIC0_SDBUF	0x1 40000402	R/W	IIC 0 Slave Data Buffer
IIC0_LMADR	0x1 40000404	R/W	IIC 0 Low Master Address
IIC0_HMADR	0x1 40000405	R/W	IIC 0 High Master Address
IIC0_CNTL	0x1 40000406	R/W	IIC 0 Control
IIC0_MDCNTL	0x1 40000407	R/W	IIC 0 Mode Control
IIC0_STS	0x1 40000408	R/W	IIC 0 Status
IIC0_EXTSTS	0x1 40000409	R/W	IIC 0 Extended Status
IIC0_LSADR	0x1 4000040A	R/W	IIC 0 Low Slave Address
IIC0_HSADR	0x1 4000040B	R/W	IIC 0 High Slave Address
IIC0_CLKDIV	0x1 4000040C	R/W	IIC 0 Clock Divide
IIC0_INTRMSK	0x1 4000040D	R/W	IIC 0 Interrupt Mask
IIC0_XFRCNT	0x1 4000040E	R/W	IIC 0 Transfer Count
IIC0_XTCNTLSS	0x1 4000040F	R/W	IIC 0 Extended Control and Slave Status
IIC0_DIRECTCNTL	0x1 40000410	R/W	IIC 0 Direct Control
Inter-Integrated Circuit 1			
IIC1_MDBUF	0x1 40000500	R/W	IIC 1 Master Data Buffer
IIC1_SDBUF	0x1 40000502	R/W	IIC 1 Slave Data Buffer
IIC1_LMADR	0x1 40000504	R/W	IIC 1 Low Master Address
IIC1_HMADR	0x1 40000505	R/W	IIC 1 High Master Address

Table 4-14. MMIO Registers

Register	Address	Access	Description
IIC1_CNTL	0x1 40000506	R/W	IIC 1 Control
IIC1_MDCNTL	0x1 40000507	R/W	IIC 1 Mode Control
IIC1_STS	0x1 40000508	R/W	IIC 1 Status
IIC1_EXTSTS	0x1 40000509	R/W	IIC 1 Extended Status
IIC1_LSADR	0x1 4000050A	R/W	IIC 1 Low Slave Address
IIC1_HSADR	0x1 4000050B	R/W	IIC 1 High Slave Address
IIC1_CLKDIV	0x1 4000050C	R/W	IIC 1 Clock Divide
IIC1_INTRMSK	0x1 4000050D	R/W	IIC 1 Interrupt Mask
IIC1_XFRCNT	0x1 4000050E	R/W	IIC 1 Transfer Count
IIC1_XTCNTLSS	0x1 4000050F	R/W	IIC 1 Extended Control and Slave Status
IIC1_DIRECTCNTL	0x1 40000510	R/W	IIC 1 Direct Control
OPB Arbiter			
OPBA0_PR	0x1 40000600	R/W	OPB Arbiter Priority Register
OPBA0_CR	0x1 40000601	R/W	OPB Arbiter Control Register
General-Purpose I/O			
GPIO0_OR	0x1 40000700	R/W	GPIO Output Register
GPIO0_TCR	0x1 40000704	R/W	GPIO Three-State Control Register
GPIO0_ODR	0x1 40000718	R/W	GPIO Open Drain Register
GPIO0_IR	0x1 4000071C	R	GPIO Input Register
Ethernet to PHY Bridge (ZMII)			
ZMII0_FER	0x1 40000780	R/W	ZMII Function Enable Register
ZMII0_SSR	0x1 40000784	R/W	ZMII Speed Select Register
ZMII0_SMIISR	0x1 40000788	R/W	ZMII SMII Status Register
Ethernet MAC 0			
EMAC0_MR0	0x1 40000800	R/W	EMAC 0 Mode Register 0
EMAC0_MR1	0x1 40000804	R/W	EMAC 0 Mode Register 1
EMAC0_TMR0	0x1 40000808	R/W	EMAC 0 Transmit Mode Register 0
EMAC0_TMR1	0x1 4000080C	R/W	EMAC 0 Transmit Mode Register 1
EMAC0_RMR	0x1 40000810	R/W	EMAC 0 Receive Mode Register
EMAC0_ISR	0x1 40000814	R/W	EMAC 0 Interrupt Status Register
EMAC0_ISER	0x1 40000818	R/W	EMAC 0 Interrupt Status Enable Register
EMAC0_IAHR	0x1 4000081C	R/W	EMAC 0 Individual Address High
EMAC0_IALR	0x1 40000820	R/W	EMAC 0 Individual Address Low
EMAC0_VTPID	0x1 40000824	R/W	EMAC 0 VLAN TPID Register
EMAC0_VTCI	0x1 40000828	R/W	EMAC 0 VLAN TCI Register
EMAC0_PTR	0x1 4000082C	R/W	EMAC 0 Pause Timer Register
EMAC0_IAHT1	0x1 40000830	R/W	EMAC 0 Individual Address Hash Table 1

Table 4-14. MMIO Registers

Register	Address	Access	Description
EMAC0_IAHT2	0x1 40000834	R/W	EMAC 0 Individual Address Hash Table 2
EMAC0_IAHT3	0x1 40000838	R/W	EMAC 0 Individual Address Hash Table 3
EMAC0_IAHT4	0x1 4000083C	R/W	EMAC 0 Individual Address Hash Table 4
EMAC0_GAHT1	0x1 40000840	R/W	EMAC 0 Group Address Hash Table 1
EMAC0_GAHT2	0x1 40000844	R/W	EMAC 0 Group Address Hash Table 2
EMAC0_GAHT3	0x1 40000848	R/W	EMAC 0 Group Address Hash Table 3
EMAC0_GAHT4	0x1 4000084C	R/W	EMAC 0 Group Address Hash Table 4
EMAC0_LSAH	0x1 40000850	R	EMAC 0 Last Source Address Low
EMAC0_LSAL	0x1 40000854	R	EMAC 0 Last Source Address High
EMAC0_IPGVR	0x1 40000858	R/W	EMAC 0 Inter-Packet Gap Value Register
EMAC0_STACR	0x1 4000085C	R/W	EMAC 0 STA Control Register
EMAC0_TRTR	0x1 40000860	R/W	EMAC 0 Transmit Request Threshold Register
EMAC0_RWMR	0x1 40000864	R/W	EMAC 0 Receive Low/High Water Mark Register
EMAC0_OCTX	0x1 40000868	R	EMAC 0 Number of Octets Transimitted
EMAC0_OCRX	0x1 4000086C	R	EMAC 0 Number of Octets Received
Ethernet MAC 1			
EMAC1_MR0	0x1 40000900	R/W	EMAC 1 Mode Register 0
EMAC1_MR1	0x1 40000904	R/W	EMAC 1 Mode Register 1
EMAC1_TMR0	0x1 40000908	R/W	EMAC 1 Transmit Mode Register 0
EMAC1_TMR1	0x1 4000090C	R/W	EMAC 1 Transmit Mode Register 1
EMAC1_RMR	0x1 40000910	R/W	EMAC 1 Receive Mode Register
EMAC1_ISR	0x1 40000914	R/W	EMAC 1 Interrupt Status Register
EMAC1_ISER	0x1 40000918	R/W	EMAC 1 Interrupt Status Enable Register
EMAC1_IAHR	0x1 4000091C	R/W	EMAC 1 Individual Address High
EMAC1_IALR	0x1 40000920	R/W	EMAC 1 Individual Address Low
EMAC1_VTPID	0x1 40000924	R/W	EMAC 1 VLAN TPID Register
EMAC1_VTCI	0x1 40000928	R/W	EMAC 1 VLAN TCI Register
EMAC1_PTR	0x1 4000092C	R/W	EMAC 1 Pause Timer Register
EMAC1_IAHT1	0x1 40000930	R/W	EMAC 1 Individual Address Hash Table 1
EMAC1_IAHT2	0x1 40000934	R/W	EMAC 1 Individual Address Hash Table 2
EMAC1_IAHT3	0x1 40000938	R/W	EMAC 1 Individual Address Hash Table 3
EMAC1_IAHT4	0x1 4000093C	R/W	EMAC 1 Individual Address Hash Table 4
EMAC1_GAHT1	0x1 40000940	R/W	EMAC 1 Group Address Hash Table 1
EMAC1_GAHT2	0x1 40000944	R/W	EMAC 1 Group Address Hash Table 2
EMAC1_GAHT3	0x1 40000948	R/W	EMAC 1 Group Address Hash Table 3
EMAC1_GAHT4	0x1 4000094C	R/W	EMAC 1 Group Address Hash Table 4
EMAC1_LSAH	0x1 40000950	R	EMAC 1 Last Source Address Low

Table 4-14. MMIO Registers

Register	Address	Access	Description
EMAC1_LSAL	0x1 40000954	R	EMAC 1 Last Source Address High
EMAC1_IPGVR	0x1 40000958	R/W	EMAC 1 Inter-Packet Gap Value Register
EMAC1_STACR	0x1 4000095C	R/W	EMAC 1 STA Control Register
EMAC1_TRTR	0x1 40000960	R/W	EMAC 1 Transmit Request Threshold Register
EMAC1_RWMR	0x1 40000964	R/W	EMAC 1 Receive Low/High Water Mark Register
EMAC1_OCTX	0x1 40000968	R	EMAC 1 Number of Octets Transimitted
EMAC1_OCRX	0x1 4000096C	R	EMAC 1 Number of Octets Received
General Purpose Timer			
GPT0_TBC	0x1 40000A00	R/W	GPT Time Base Counter
GPT0_OE	0x1 40000A10	R/W	GPT Output Enable
GPT0_OL	0x1 40000A14	R/W	GPT Output Level
GPT0_IM	0x1 40000A18	R/W	GPT Interrupt Mask
GPT0_ISS	0x1 40000A1C	R/W	GPT Interrupt Status (Set bits if write 1)
GPT0_ISC	0x1 40000A20	R/W	GPT Interrupt Status (Clear bits if write 1)
GPT0_IE	0x1 40000A24	R/W	GPT Interrupt Enable
GPT0_COMP0	0x1 40000A80	R/W	GPT Compare Timer 0
GPT0_COMP1	0x1 40000A84	R/W	GPT Compare Timer 1
GPT0_COMP2	0x1 40000A88	R/W	GPT Compare Timer 2
GPT0_COMP3	0x1 40000A8C	R/W	GPT Compare Timer 3
GPT0_COMP4	0x1 40000A90	R/W	GPT Compare Timer 4
GPT0_MASK0	0x1 40000AC0	R/W	GPT Compare Mask 0
GPT0_MASK1	0x1 40000AC4	R/W	GPT Compare Mask 1
GPT0_MASK2	0x1 40000AC8	R/W	GPT Compare Mask 2
GPT0_MASK3	0x1 40000ACC	R/W	GPT Compare Mask 3
GPT0_MASK4	0x1 40000AD0	R/W	GPT Compare Mask 4

The two registers listed in Table 4-15 are used to access the configuration registers of external PCI-X devices only.

Table 4-15. External PCI-X Device Configuration Register Access

Register	Address	Access	Description
PCIX0_CFGADDR	0x2 0EC00000	R/W	PCI-X Configuration Address Register
PCIX0_CFGDATA	0x2 0EC00004	R/W	PCI-X Configuration Data Register

Registers listed in Table 4-16 can be accessed in two ways:

1. By the processor using the PLB address

2. By an external PCI-X device using configuration cycles to the proper offset.

Table 4-16. Internal PCI-X Configuration Registers

Register	PLB		PCI-X Configuration		Description
	Address	Access	Offset	Access	
PCIX0_VENDID	0x2 0EC80000	R/W	0x01–0x00	R	PCI-X Vendor ID
PCIX0_DEVID	0x2 0EC80002	R/W	0x03–0x02	R	PCI-X Device ID
PCIX0_CMD	0x2 0EC80004	R/W	0x05–0x04	R/W	PCI-X Command Register
PCIX0_STATUS	0x2 0EC80006	R/W	0x07–0x06	R/W	PCI-X Status Register
PCIX0_REVID	0x2 0EC80008	R/W	0x08	R	PCI-X Revision ID
PCIX0_CLS	0x2 0EC80009	R/W	0x0B–0x09	R	PCI-X Class Register
PCIX0_CACHELS	0x2 0EC8000C	R/W	0x0C	R/W	PCI-X Cache Line Size
PCIX0_LATTIM	0x2 0EC8000D	R/W	0x0D	R/W	PCI-X Latency Timer
PCIX0_HDTYPE	0x2 0EC8000E	R	0x0E	R	PCI-X Header Type
PCIX0_BIST	0x2 0EC8000F	R	0x0F	R	PCI-X Built In Self Test Control
PCIX0_ BAR0L	0x2 0EC80010	R/W	0x13–0x10	R/W	PCI-X BAR 0 Low
PCIX0_ BAR0H	0x2 0EC80014	R/W	0x17–0x14	R/W	PCI-X BAR 0 High
PCIX0_BAR1	0x2 0EC80018	R/W	0x1B–0x18	R/W	PCI-X BAR 1
PCIX0_ BAR2L	0x2 0EC8001C	R/W	0x1F–0x1C	R/W	PCI-X BAR 2 Low
PCIX0_ BAR2H	0x2 0EC80020	R/W	0x23–0x20	R/W	PCI-X BAR 2 High
PCIX0_ BAR3	0x2 0EC80024	R	0x27–0x24	R	Unused BAR 3
PCIX0_CISPTR	0x2 0EC80028	R	0x2B–0x28	R	Unused Cardbus CIS Pointer
PCIX0_SBSYSVID	0x2 0EC8002C	R/W	0x2D–0x2C	R	PCI-X Subsystem Vendor ID
PCIX0_SBSYSID	0x2 0EC8002E	R/W	0x2F–0x2E	R	PCI-X Subsystem ID
PCIX0_EROMBA	0x2 0EC80030	R/W	0x33–0x30	R/W	PCI-X Expansion ROM Base Address
PCIX0_CAP	0x2 0EC80034	R	0x34	R	PCI-X Capabilities Pointer
PCIX0_RES0	0x2 0EC80035	R	0x35	R	Reserved
PCIX0_RES1	0x2 0EC80036	R	0x37–0x6	R	Reserved
PCIX0_RES2	0x2 0EC80038	R	0x3B–0x8	R	Reserved
PCIX0_INTLN	0x2 0EC8003C	R/W	0x3C	R/W	PCI-X Interrupt Line
PCIX0_INTPN	0x2 0EC8003D	R	0x3D	R	PCI-X Interrupt Pin
PCIX0_MINGNT	0x2 0EC8003E	R	0x3E	R	PCI-X Minimum Grant
PCIX0_MAXLTNCY	0x2 0EC8003F	R	0x3F	R	PCI-X Maximum Latency
PCIX0_BRDGOPT1	0x2 0EC80040	R/W	0x43–0x40	R/W	PCI-X Bridge Options 1
PCIX0_BRDGOPT2	0x2 0EC80044	R/W	0x47–0x44	R/W	PCI-X Bridge Options 2
PCIX0_ERREN	0x2 0EC80050	R/W	0x53–0x50	R/W	PCI-X Error Enable
PCIX0_ERRSTS	0x2 0EC80054	R/W	0x57–0x54	R/W	PCI-X Error Status
PCIX0_PLBBESR	0x2 0EC80058	R	0x5B–0x58	R	PCI-X PLB Slave Error Syndrome Register

Table 4-16. Internal PCI-X Configuration Registers

Register	PLB		PCI-X Configuration		Description
	Address	Access	Offset	Access	
PCIX0_PLBBEARL	0x2 0EC8005C	R	0x5F–0x5C	R	PCI-X PLB Slave Error Address Register Low
PCIX0_PLBBEARH	0x2 0EC80060	R	0x63–0x60	R	PCI-X PLB Slave Error Address Register High
PCIX0_POM0LAL	0x2 0EC80068	R/W	0x6B–0x68	R/W	PCI-X POM0 Local Address Low
PCIX0_POM0LAH	0x2 0EC8006C	R/W	0x6F–0x6C	R/W	PCI-X POM0 Local Address High
PCIX0_POM0SA	0x2 0EC80070	R/W	0x73–0x70	R/W	PCI-X POM0 Size Attribute
PCIX0_POM0PCIAL	0x2 0EC80074	R/W	0x77–0x74	R/W	PCI-X POM0 PCI Address Low
PCIX0_POM0PCIAH	0x2 0EC80078	R/W	0x7B–0x78	R/W	PCI-X POM0 PCI Address High
PCIX0_POM1LAL	0x2 0EC8007C	R/W	0x7F–0x7C	R/W	PCI-X POM1 Local Address Low
PCIX0_POM1LAH	0x2 0EC80080	R/W	0x83–0x80	R/W	PCI-X POM1 Local Address High
PCIX0_POM1SA	0x2 0EC80084	R/W	0x87–0x84	R/W	PCI-X POM1 Size Attribute
PCIX0_POM1PCIAL	0x2 0EC80088	R/W	0x8B–0x88	R/W	PCI-X POM1 PCI Address Low
PCIX0_POM1PCIAH	0x2 0EC8008C	R/W	0x8F–0x8C	R/W	PCI-X POM1 PCI Address High
PCIX0_POM2SA	0x2 0EC80090	R/W	0x93–0x90	R/W	PCI-X POM2 Size Attribute
PCIX0_PIM0SA	0x2 0EC80098	R/W	0x9B–0x98	R/W	PCI-X PIM0 Size/Attribute
PCIX0_PIM0LAL	0x2 0EC8009C	R/W	0x9F–0x9C	R/W	PCI-X PIM0 Local Address Low
PCIX0_PIM0LAH	0x2 0EC800A0	R/W	0xA3–0xA0	R/W	PCI-X PIM0 Local Address High
PCIX0_PIM1SA	0x2 0EC800A4	R/W	0xA7–0xA4	R/W	PCI-X PIM1 Size/Attribute
PCIX0_PIM1LAL	0x2 0EC800A8	R/W	0xAB–0xA8	R/W	PCI-X PIM1 Local Address Low
PCIX0_PIM1LAH	0x2 0EC800AC	R/W	0xAF–0xAC	R/W	PCI-X PIM1 Local Address High
PCIX0_PIM2SA	0x2 0EC800B0	R/W	0xB3–0xB0	R/W	PCI-X PIM2 Size/Attribute
PCIX0_PIM2LAL	0x2 0EC800B4	R/W	0xB7–0xB4	R/W	PCI-X PIM2 Local Address Low
PCIX0_PIM2LAH	0x2 0EC800B8	R/W	0xBB–0xB8	R/W	PCI-X PIM2 Local Address High
PCIX0_OMCAPID	0x2 0EC800C0	R	0xC0	R	PCI-X Outbound MSI Capability Identifier
PCIX0_OMNIPTR	0x2 0EC800C1	R/W	0xC1	R	PCI-X Outbound MSI Next Item Pointer
PCIX0_OMMC	0x2 0EC800C2	R/W	0xC3–0xC2	R/W	PCI-X Outbound MSI Message Control
PCIX0_OMMA	0x2 0EC800C4	R/W	0xC7–0xC4	R/W	PCI-X Outbound MSI Message Address
PCIX0_OMMUA	0x2 0EC800C8	R/W	0xCB–0xC8	R/W	PCI-X Outbound MSI Message Upper Address
PCIX0_OMMDATA	0x2 0EC800CC	R/W	0xCD–0xCC	R/W	PCI-X Outbound MSI Message Data
PCIX0_OMMEOI	0x2 0EC800CE	R/W	0xCE	R/W	PCI-X Outbound MSI Message End Of Interrupt
PCIX0_PMCAPID	0x2 0EC800D0	R/W	0xD0	R	PCI-X PMC Capability Identifier
PCIX0_PMNIPTR	0x2 0EC800D1	R/W	0xD1	R	PCI-X PMC Next Item Pointer
PCIX0_PMC	0x2 0EC800D2	R	0xD3–0xD2	R	PCI-X Power Management Capabilities

Table 4-16. Internal PCI-X Configuration Registers

Register	PLB		PCI-X Configuration		Description
	Address	Access	Offset	Access	
PCIX0_PMCSR	0x2 0EC800D4	R/W	0xD5–0xD4	R/W	PCI-X Power Management Control Status
PCIX0_PMCSRBSE	0x2 0EC800D6	R	0xD6	R	PCI-X PMCSR PCI to PCI Bridge Support Extensions
PCIX0_PMDATA	0x2 0EC800D7	R	0xD7	R	PCI-X PMC Unused Data Register
PCIX0_PMSCRR	0x2 0EC800D8	R/W	0xD8	R/W	PCI-X Power Management State Change Request Register
PCIX0_PCIXCAPID	0x2 0EC800DC	R	0xDC	R	PCI-X Capability Identifier
PCIX0_PCIXNIPTR	0x2 0EC800DD	R	0xDD	R	PCI-X Next Item Pointer
PCIX0_PCIXCMD	0x2 0EC800DE	R/W	0xDF–0xDE	R/W	PCI-X Command
PCIX0_PCIXSTS	0x2 0EC800E0	R/W	0xE3–0xE0	R/W	PCI-X Status
PCIX0_PCIXIDR	0x2 0EC800E4	R/W	0xE7–0xE4	R/W	PCI-X Internal Debug Register
PCIX0_PCIXCID	0x2 0EC800E8	R	0xEB–0xE8	R	PCI-X Internal Core Device ID
PCIX0_PCIXRID	0x2 0EC800EC	R	0xEF–0xEC	R	PCI-X Internal Core Revision ID

Registers listed in Table 4-17 can be accessed in two ways:

1. By the processor using the PLB address
2. By an external PCI-X device using memory cycles to the proper address.

Table 4-17. PCI-X SImple Message Passing and Inbound MSI Registers

Register	PLB		PCI-X Memory		Description
	Address	Access	Address	Access	
PCIX0_MSGIL	0x2 0EC80100	R/W	(PCIX0_BAR0H || PCIX0_BAR0L) + (0x03–0x00)	R/W	PCI-X Message In Low
PCIX0_MSGIH	0x2 0EC80104	R/W	(PCIX0_BAR0H || PCIX0_BAR0L) + (0x07–0x04)	R/W	PCI-X Message In High
PCIX0_MSGOL	0x2 0EC80108	R/W	(PCIX0_BAR0H || PCIX0_BAR0L) + (0x0B–0x08)	R/W	PCI-X Message Out Low
PCIX0_MSGOH	0x2 0EC8010C	R/W	(PCIX0_BAR0H || PCIX0_BAR0L) + (0x0F–0x0C)	R/W	PCI-X Message Out High
PCIX0_IM	0x2 0EC801F8	W	(PCIX0_BAR0H || PCIX0_BAR0L) + (0xFB–0xF8)	W	PCI-X Inbound MSI

4.3 Instruction Classes

PowerPC Book-E architecture defines all instructions as falling into exactly one of the following four classes, as determined by the primary opcode (and the extended opcode, if any):

1. Defined
2. Allocated
3. Preserved
4. Reserved (-illegal or -nop)

4.3.1 Defined Instruction Class

This class of instructions consists of all the instructions defined in PowerPC Book-E. In general, defined instructions are guaranteed to be supported within a PowerPC Book-E system as specified by the architecture, either within the processor implementation itself or within emulation software supported by the system operating software.

One exception to this is that, for implementations (such as the PPC440GP) that only provide the 32-bit subset of PowerPC Book-E, it is not expected (and likely not even possible) that emulation of the 64-bit behavior of the defined instructions will be provided by the system.

As defined by PowerPC Book-E, any attempt to execute a defined instruction will:

- cause an Illegal Instruction exception type Program interrupt, if the instruction is not recognized by the implementation; or
- cause an Unimplemented Instruction exception type Program interrupt, if the instruction is recognized by the implementation and is not a floating-point instruction, but is not supported by the implementation; or
- cause a Floating-Point Unavailable interrupt if the instruction is recognized as a floating-point instruction, but floating-point processing is disabled; or
- cause an Unimplemented Instruction exception type Program interrupt, if the instruction is recognized as a floating-point instruction and floating-point processing is enabled, but the instruction is not supported by the implementation; or
- perform the actions described in the rest of this document, if the instruction is recognized and supported by the implementation. The architected behavior may cause other exceptions.

The PPC440GP recognizes and fully supports all of the instructions in the defined class, with a few exceptions. First, because the PPC440GP is a 32-bit implementation, those operations which are defined specifically for 64-bit operation are not supported at all, and will always cause an Illegal Instruction exception type Program interrupt.

Second, instructions that are defined for floating-point processing are not supported within the PPC440GP.

Finally, there are two other defined instructions which are not supported within the PPC440GP. One is a TLB management instruction (**tlbiva**, TLB Invalidate Virtual Address) that is specifically intended for coherent multiprocessor systems. The other is **mfapidi** (Move From Auxiliary Processor ID Indirect), which is a special instruction intended to assist with identification of the auxiliary processors which may be attached to a particular processor implementation. Since the PPC440GP does not support **mfapidi**, the means of identifying the auxiliary processors in a PPC440GP-based system are implementation-dependent. Execution of either **tlbiva** or **mfapidi** will cause an Illegal Instruction exception type Program interrupt.

4.3.2 Allocated Instruction Class

This class of instructions contains a set of primary opcodes, as well as extended opcodes for certain primary opcodes. The specific opcodes are listed in ~~"Allocated Instruction Opcodes," on page A-40~~ *Appendix A.3* on page 1575.

Allocated instructions are provided for purposes that are outside the scope of PowerPC Book-E, and are for implementation-dependent and application-specific use.

PowerPC Book-E declares that any attempt to execute an allocated instruction results in one of the following effects:

- Causes an Illegal Instruction exception type Program interrupt, if the instruction is not recognized by the implementation
- Causes an Auxiliary Processor Unavailable interrupt if the instruction is recognized by the implementation, but allocated instruction processing is disabled
- Causes an Unimplemented Instruction exception type Program interrupt, if the instruction is recognized and allocated instruction processing is enabled, but the instruction is not supported by the implementation
- Perform the actions described for the particular implementation of the allocated instruction. The implementation-dependent behavior may cause other exceptions.

In addition to supporting the defined instructions of PowerPC Book-E, the PPC440GP also implements a number of instructions which use the allocated instruction opcodes, and thus are not part of the PowerPC Book-E architecture. ~~Table 4-34, on page 4-45~~*Table 4-34* on page 185 identifies the allocated instructions that are implemented within the PPC440GP. All of these instructions are always enabled and supported, and thus they always perform the functions defined for them within this document, and never cause Illegal Instruction, Auxiliary Processor Unavailable, nor Unimplemented Instruction exceptions.

The PPC440GP also supports the use of *any* of the allocated opcodes by an attached auxiliary processor, except for those allocated opcodes which have been implemented within the PPC440GP, as mentioned above. Also, there is one other allocated opcode (primary opcode 31, secondary opcode 262) that has been implemented within the PPC440GP and is thus not available for use by an attached auxiliary processor. This is the opcode which was used on previous PowerPC 400 Series embedded controllers for the **icbt** (Instruction Cache Block Touch) instruction. The **icbt** instruction is now part of the defined instruction class for PowerPC Book-E, and uses a new opcode (primary opcode 31, secondary opcode 22). The PPC440GP implements the new defined opcode, but also continues to support the previous opcode, in order to support legacy software written for earlier PowerPC 400 Series implementations. The **icbt** instruction description in ~~Chapter 28, "Instruction Set,"~~ *Instruction Set* on page 893 only identifies the defined opcode, although Appendix A, "Instruction Summary," includes both the defined and the allocated opcode in the table which lists all the instructions by opcode. In order to ensure portability between the PPC440GP and future PowerPC Book-E implementations, software should take care to only use the defined opcode for **icbt**, and avoid usage of the previous opcode which is now in the allocated class.

4.3.3 Preserved Instruction Class

The preserved instruction class is provided to support backward compatibility with the PowerPC Architecture, and/or earlier versions of the PowerPC Book-E architecture. This instruction class includes opcodes which were defined for these previous architectures, but which are no longer defined for PowerPC Book-E.

Any attempt to execute a preserved instruction results in one of the following effects:

- Performs the actions described in the previous version of the architecture, if the instruction is recognized by the implementation
- Causes an Illegal Instruction exception type Program interrupt, if the instruction is not recognized by the implementation.

The only preserved instruction recognized and supported by the PPC440GP is the **mftb** (Move From Time Base) opcode. This instruction was used in the the PowerPC Architecture to read the Time Base Upper (TBU) and Time Base Lower (TBL) registers. PowerPC Book-E architecture instead defines TBU and TBL as Special Purpose Registers (SPRs), and thus the **mfspr** (Move From Special Purpose Register) instruction is used to read them. In order to enable legacy time base management software to be run on the PPC440GP, the processor core also supports the preserved opcode of **mftb**. However, the **mftb** instruction is not included in the various sections of this document that describe the implemented instructions, and software should take care to use the currently architected mechanism of **mfspr** to read the time base registers, in order to guarantee portability between the PPC440GP and future implementations of PowerPC Book-E.

On the other hand, Appendix A, "Instruction Summary," does identify the **mftb** instruction as an implemented preserved opcode in the table which lists all the instructions by opcode.

4.3.4 Reserved Instruction Class

This class of instructions consists of all instruction primary opcodes (and associated extended opcodes, if applicable) which do not belong to either the defined, allocated, or preserved instruction classes.

Reserved instructions are available for future versions of PowerPC Book-E architecture. That is, future versions of PowerPC Book-E may define any of these instructions to perform new functions or make them available for implementation-dependent use as allocated instructions. There are two types of reserved instructions: reserved-illegal and reserved-nop.

Any attempt to execute a reserved-illegal instruction will cause an Illegal Instruction exception type Program interrupt. Reserved-illegal instructions are, therefore, available for future extensions to PowerPC Book-E that would affect architected state. Such extensions might include new forms of integer or floating-point arithmetic instructions, or new forms of load or store instructions that affect architected registers or the contents of memory.

Any attempt to execute a reserved-nop instruction, on the other hand, either has no effect (that is, is treated as a no-operation instruction), or causes an Illegal Instruction exception type Program interrupt. Because implementations are typically expected to treat reserved-nop instructions as true no-ops, these instruction opcodes are thus available for future extensions to PowerPC Book-E which have no effect on architected state. Such extensions might include performance-enhancing hints, such as new forms of cache touch instructions. Software would be able to take advantage of the functionality offered by the new instructions, and still remain backwards-compatible with implementations of previous versions of PowerPC Book-E.

The PPC440GP implements all of the reserved-nop instruction opcodes as true no-ops. The specific reserved-nop opcodes are listed in ~~"Reserved Instruction Opcodes," on page A-41~~ *Appendix A.5* on page 1576

4.4 Implemented Instruction Set Summary

This section provides an overview of the various types and categories of instructions implemented within the PPC440GP. In addition, ~~Chapter 28, "Instruction Set,"~~*Instruction Set* on page 893 provides a complete alphabetical listing of every implemented instruction, including its register transfer language (RTL) and a detailed description of its operation. Also, Appendix A, "Instruction Summary," lists each implemented instruction alphabetically (and by opcode) along with a short-form description and its extended mnemonic(s).

~~Table 4-18~~*Table 4-18* summarizes the PPC440GP instruction set by category. Instructions within each category are described in subsequent sections.

Table 0-16. Instruction Categories

Category	Subcategory	Instruction Types
Integer	Integer Storage Access	load, store
	Integer Arithmetic	add, subtract, multiply, divide, negate
	Integer Logical	and, andc, or, orc, xor, nand, nor, xnor, extend sign, count leading zeros
	Integer Compare	compare, compare logical
	Integer Trap	trap
	Integer Rotate	rotate and insert, rotate and mask
	Integer Shift	shift left, shift right, shift right algebraic
Branch		branch, branch conditional, branch to link, branch to count
Processor Control	Condition Register Logical	crand, crandc, cror, crorc, crnand, crnor, crxor, crxnor
	Register Management	move to/from SPR, move to/from DCR, move to/from MSR, write to external interrupt enable bit, move to/from CR
	System Linkage	system call, return from interrupt, return from critical interrupt
	Processor Synchronization	instruction synchronize
Storage Control	Cache Management	data allocate, data invalidate, data touch, data zero, data flush, data store, instruction invalidate, instruction touch
	TLB Management	read, write, search, synchronize
	Storage Synchronization	memory synchronize, memory barrier
Allocated	Allocated Arithmetic	multiply-accumulate, negative multiply-accumulate, multiply halfword
	Allocated Logical	detect left-most zero byte
	Allocated Cache Management	data congruence-class invalidate, instruction congruence-class invalidate
	Allocated Cache Debug	data read, instruction read

Table 4-18. Instruction Categories

Category	Subcategory	Instruction Types
Integer	Integer Storage Access	load, store
	Integer Arithmetic	add, subtract, multiply, divide, negate
	Integer Logical	and, andc, or, orc, xor, nand, nor, xnor, extend sign, count leading zeros
	Integer Compare	compare, compare logical
	Integer Trap	trap
	Integer Rotate	rotate and insert, rotate and mask
	Integer Shift	shift left, shift right, shift right algebraic
Branch		branch, branch conditional, branch to link, branch to count
Processor Control	Condition Register Logical	crand, crandc, cror, crorc, crnand, crnor, crxor, crxnor
	Register Management	move to/from SPR, move to/from DCR, move to/from MSR, write to external interrupt enable bit, move to/from CR
	System Linkage	system call, return from interrupt, return from critical interrupt
	Processor Synchronization	instruction synchronize
Storage Control	Cache Management	data allocate, data invalidate, data touch, data zero, data flush, data store, instruction invalidate, instruction touch
	TLB Management	read, write, search, synchronize
	Storage Synchronization	memory synchronize, memory barrier
Allocated	Allocated Arithmetic	multiply-accumulate, negative multiply-accumulate, multiply halfword
	Allocated Logical	detect left-most zero byte
	Allocated Cache Management	data congruence-class invalidate, instruction congruence-class invalidate
	Allocated Cache Debug	data read, instruction read

4.4.1 Integer Instructions

Integer instructions transfer data between memory and the GPRs, and perform various operations on the GPRs. This category of instructions is further divided into seven sub-categories, described below.

4.4.1.1 Integer Storage Access Instructions

Integer storage access instructions load and store data between memory and the GPRs. These instructions operate on bytes, halfwords, and words. Integer storage access instructions also support loading and storing multiple registers, character strings, and byte-reversed data, and loading data with sign-extension.

~~Table 4-19~~*Table 4-19* shows the integer storage access instructions in the PPC440GP. In the table, the syntax "[**u**]" indicates that the instruction has both an "update" form (in which the RA addressing register is updated with the calculated address) and a "non-update" form. Similarly, the syntax "[**x**]" indicates that the instruction has both an "indexed" form (in which the address is formed by adding the contents of the RA and

RB GPRs) and a "base + displacement" form (in which the address is formed by adding a 16-bit signed immediate value (specified as part of the instruction) to the contents of GPR RA. See the detailed instruction descriptions in ~~Chapter 28~~*Instruction Set* on page 893.

Table 4-19. Integer Storage Access Instructions

Loads				Stores			
Byte	Halfword	Word	Multiple/String	Byte	Halfword	Word	Multiple/String
lbz[**u**][**x**]	**lha**[**u**][**x**] **lhbrx** **lhz**[**u**][**x**]	**lwarx** **lwbrx** **lwz**[**u**][**x**]	**lmw** **lswi** **lswx**	**stb**[**u**][**x**]	**sth**[**u**][**x**] **sthbrx**	**stw**[**u**][**x**] **stwbrx** **stwcx.**	**stmw** **stswi** **stswx**

4.4.1.2 Integer Arithmetic Instructions

Arithmetic operations are performed on integer or ordinal operands stored in registers. Instructions that perform operations on two operands are defined in a three-operand format; an operation is performed on the operands, which are stored in two registers. The result is placed in a third register. Instructions that perform operations on one operand are defined in a two-operand format; the operation is performed on the operand in a register and the result is placed in another register. Several instructions also have immediate formats in which one of the source operands is a field in the instruction.

Most integer arithmetic instructions have versions that can update CR[CR0] and/or XER[SO, OV] (Summary Overflow, Overflow), based on the result of the instruction. Some integer arithmetic instructions also update XER[CA] (Carry) implicitly. See ~~"Integer Processing" on page 4-53~~*Integer Processing* on page 192 for more information on how these instructions update the CR and/or the XER.

~~Table 4-20~~*Table 4-20* lists the integer arithmetic instructions in the PPC440GP. In the table, the syntax "[**o**]" indicates that the instruction has both an "o" form (which updates the XER[SO,OV] fields) and a "non-o" form. Similarly, the syntax "[**.**]" indicates that the instruction has both a "record" form (which updates CR[CR0]) and a "non-record" form.

Table 4-20. Integer Arithmetic Instructions

Add	Subtract	Multiply	Divide	Negate
add[**o**][**.**] **addc**[**o**][**.**] **adde**[**o**][**.**] **addi** **addic**[.] **addis** **addme**[**o**][**.**] **addze**[**o**][**.**]	**subf**[**o**][**.**] **subfc**[**o**][**.**] **subfe**[**o**][**.**] **subfic** **subfme**[**o**][**.**] **subfze**[**o**][**.**]	**mulhw**[**.**] **mulhwu**[**.**] **mulli** **mullw**[**o**][**.**]	**divw**[**o**][**.**] **divwu**[**o**][**.**]	**neg**[**o**][**.**]

4.4.1.3 Integer Logical Instructions

Table 4-21 *Table 4-21* lists the integer logical instructions in the PPC440GP. See ~~"Integer Arithmetic Instructions" on page 4-39~~ *Integer Arithmetic Instructions* on page 180 for an explanation of the "**[.]**" syntax.

Table 4-21. Integer Logical Instructions

And	And with complement	Nand	Or	Or with complement	Nor	Xor	Equivalence	Extend sign	Count leading zeros
and[.] **andi.** **andis.**	**andc**[.]	**nand**[.]	**or**[.] **ori** **oris**	**orc**[.]	**nor**[.]	**xor**[.] **xori** **xoris**	**eqv**[.]	**extsb**[.] **extsh**[.]	**cntlzw**[.]

4.4.1.4 Integer Compare Instructions

These instructions perform arithmetic or logical comparisons between two operands and update the CR with the result of the comparison.

Table 4-22 *Table 4-22* lists the integer compare instructions in the PPC440GP.

Table 4-22. Integer Compare Instructions

Arithmetic	Logical
cmp **cmpi**	**cmpl** **cmpli**

4.4.1.5 Integer Trap Instructions

Table 4-23 *Table 4-23* lists the integer trap instructions in the PPC440GP.

Table 4-23. Integer Trap Instructions

Trap
tw **twi**

4.4.1.6 Integer Rotate Instructions

These instructions rotate operands stored in the GPRs. Rotate instructions can also mask rotated operands.

Table 4-24 *Table 4-24* lists the rotate instructions in the PPC440GP. See ~~"Integer Arithmetic Instructions" on page 4-39~~ *Integer Arithmetic Instructions* on page 180 for an explanation of the "**[.]**" syntax.

Table 4-24. Integer Rotate Instructions

Rotate and Insert	Rotate and Mask
rlwimi[.]	**rlwinm**[.] **rlwnm**[.]

4.4.1.7 Integer Shift Instructions

Table 4-25 *Table 4-25* lists the integer shift instructions in the PPC440GP. Note that the shift right algebraic insructions implicitly update the XER[CA] field. See "Integer Arithmetic Instructions" on page 4-39 *Integer Arithmetic Instructions* on page 180 for an explanation of the "**[.]**" syntax.

Table 4-25. Integer Shift Instructions

Shift Left	Shift Right	Shift Right Algebraic
slw[**.**]	**srw**[**.**]	**sraw**[**.**] **srawi**[**.**]

4.4.2 Branch Instructions

These instructions unconditionally or conditionally branch to an address. Conditional branch instructions can test condition codes set in the CR by a previous instruction and branch accordingly. Conditional branch instructions can also decrement and test the Count Register (CTR) as part of branch determination, and can save the return address in the Link Register (LR).The target address for a branch can be a displacement from the current instruction address or an absolute address, or contained in the LR or CTR.

See "Branch Processing" on page 4-46 *Branch Processing* on page 185 for more information on branch operations.

Table 4-26 *Table 4-26* lists the branch instructions in the PPC440GP. In the table, the syntax "[**l**]" indicates that the instruction has both a "link update" form (which updates LR with the address of the instruction after the branch) and a "non-link update" form. Similarly, the syntax "[**a**]" indicates that the instruction has both an "absolute address" form (in which the target address is formed directly using the immediate field specified as part of the instruction) and a "relative" form (in which the target address is formed by adding the specified immediate field to the address of the branch instruction).

Table 4-26. Branch Instructions

Branch
b[**l**][**a**] **bc**[**l**][**a**] **bcctr**[**l**] **bclr**[**l**]

4.4.3 Processor Control Instructions

Processor control instructions manipulate system registers, perform system software linkage, and synchronize processor operations. The instructions in these three sub-categories of processor control instructions are described below.

4.4.3.1 Condition Register Logical Instructions

These instructions perform logical operations on a specified pair of bits in the CR, placing the result in another specified bit. The benefit of these instructions is that they can logically combine the results of several comparison operations without incurring the overhead of conditional branching between each one. Software performance can significantly improve if multiple conditions are tested at once as part of a branch decision.

~~Table 4-27~~*Table 4-27* lists the condition register logical instructions in the PPC440GP.

Table 4-27. Condition Register Logical Instructions

crand	**crnor**
crandc	**cror**
creqv	**crorc**
crnand	**crxor**

4.4.3.2 Register Management Instructions

These instructions move data between the GPRs and control registers in the PPC440GP.

~~Table 4-28~~*Table 4-28* lists the register management instructions in the PPC440GP.

Table 4-28. Register Management Instructions

CR	DCR	MSR	SPR
mcrf		**mfmsr**	
mcrxr	**mfdcr**	**mtmsr**	**mfspr**
mfcr	**mtdcr**	**wrtee**	**mtspr**
mtcrf		**wrteei**	

4.4.3.3 System Linkage Instructions

These instructions invoke supervisor software level for system services, and return from interrupts.

~~Table 4-29~~*Table 4-29* lists the system linkage instructions in the PPC440GP.

Table 4-29. System Linkage Instructions

rfi
rfci
sc

4.4.3.4 Processor Synchronization Instruction

Tne processor synchronization instruction, **isync**, forces the processor to complete all instructions preceding the **isync** before allowing any context changes as a result of any instructions that follow the **isync**. Additionally, all instructions that follow the **isync** will execute within the context established by the completion of all the instructions that precede the **isync**. See ~~"Synchronization" on page 4-62~~*Synchronization* on page 201 for more information on the synchronizing effect of **isync**.

~~Table 4-30~~*Table 4-30* shows the processor synchronization instruction in the PPC440GP.

Table 4-30. Processor Synchronization Instruction

isync

4.4.4 Storage Control Instructions

These instructions manage the instruction and data caches and the TLB of the PPC440GP. Instructions are also provided to synchronize and order storage accesses. The instructions in these three sub-categories of storage control instructions are described below.

4.4.4.1 Cache Management Instructions

These instructions control the operation of the data and instruction caches. Instructions are provided to fill, flush, invalidate, or zero data cache blocks, where a block is defined as a 32-byte cache line. instructions are also provided to fill or invalidate instruction cache blocks.

~~Table 4-31~~*Table 4-31* lists the cache management instructions in the PPC440GP.

Table 4-31. Cache Management Instructions

Data Cache	Instruction Cache
dcba **dcbf** **dcbi** **dcbst** **dcbt** **dcbtst** **dcbz**	**icbi** **icbt**

4.4.4.2 TLB Management Instructions

The TLB management instructions read and write entries of the TLB array, and search the TLB array for an entry which will translate a given virtual address. There is also an instruction for synchronizing TLB updates with other processors, but since the PPC440GP is intended for use in uni-processor environments, this instruction performs no operation on the PPC440GP.

~~Table 4-32~~*Table 4-32* lists the TLB management instructions in the PPC440GP. See ~~"Integer Arithmetic Instructions" on page 4-39~~*Integer Arithmetic Instructions* on page 180 for an explanation of the "**[.]**" syntax.

Table 4-32. TLB Management Instructions

tlbre
tlbsx[**.**]
tlbsync
tlbwe

4.4.4.3 Storage Synchronization Instructions

The storage synchronization instructions allow software to enforce ordering amongst the storage accesses caused by load and store instructions, which by default are "weakly-ordered" by the processor. "Weakly-ordered" means that the processor is architecturally permitted to perform loads and stores generally out-of-order with respect to their sequence within the instruction stream, with some exceptions. However, if a storage synchronization instruction is executed, then all storage accesses prompted by instructions preceding the synchronizing instruction must be performed before any storage accesses prompted by instructions which come after the synchronizing instruction. See ~~"Synchronization" on page 4-62~~*Synchronization* on page 201 for more information on storage synchronization.

~~Table 4-30~~*Table 4-30* shows the storage synchronization instructions in the PPC440GP.

Table 4-33. Storage Synchronization Instructions

msync
mbar

4.4.5 Allocated Instructions

These instructions are not part of the PowerPC Book-E architecture, but they are included as part of the PPC440GP. Architecturally, they are considered allocated instructions, as they use opcodes which are within the allocated class of instructions, which the PowerPC Book-E architecture identifies as being available for implementation-dependent and/or application-specific purposes. However, all of the allocated instructions which are implemented within the PPC440GP are "standard" for IBM's family of PowerPC embedded controllers, and are not unique to the PPC440GP.

The allocated instructions implemented within the PPC440GP are divided into four sub-categories, and are shown in ~~Table 4-34~~*Table 4-34*. See ~~"Integer Arithmetic Instructions" on page 4-39~~*Integer Arithmetic Instructions* on page 180 for an explanation of the "**[.]**" and "**[o]**" syntax.

Table 4-34. Allocated Instructions

Arithmetic			Logical	Cache Management	Cache Debug
Multiply-Accumulate	Negative Multiply-Accumulate	Multiply Halfword			
macchw[**o**][**.**] **macchws**[**o**][**.**] **macchwsu**[**o**][**.**] **macchwu**[**o**][**.**] **machhw**[**o**][**.**] **machhws**[**o**][**.**] **machhwsu**[**o**][**.**] **machhwu**[**o**][**.**] **maclhw**[**o**][**.**] **maclhws**[**o**][**.**] **maclhwsu**[**o**][**.**] **maclhwu**[**o**][**.**]	**nmacchw**[**o**][**.**] **nmacchws**[**o**][**.**] **nmachhw**[**o**][**.**] **nmachhws**[**o**][**.**] **nmaclhw**[**o**][**.**] **nmaclhws**[**o**][**.**]	**mulchw**[**.**] **mulchwu**[**.**] **mulhhw**[**.**] **mulhhwu**[**.**] **mullhw**[**.**] **mullhwu**[**.**]	**dlmzb**[**.**]	**dccci** **iccci**	**dcread** **icread**

4.5 Branch Processing

The four branch instructions provided by PPC440GP are summarized in ~~Table 4.4.2~~*Table 4.4.2* on Page 181. In addition, each of these instructions is described in detail in ~~Chapter 28, "Instruction Set"~~*Instruction Set* on page 893. The following sections provide additional information on branch addressing, instruction fields, prediction, and registers.

4.5.1 Branch Addressing

The branch instruction (**b**[**l**][**a**]) specifies the displacement of the branch target address as a 26-bit value (the 24-bit LI field right-extended with 0b00). This displacement is regarded as a signed 26-bit number covering an address range of ±32MB. Similarly, the branch conditional instruction (**bc**[**l**][**a**]) specifies the displacement as a 16-bit value (the 14-bit BD field right-extended with 0b00). This displacement covers an address range of ±32KB.

For the relative form of the branch and branch conditional instructions (**b**[**l**] and **bc**[**l**], with instruction field AA = 0), the target address is the address of the branch instruction itself (the Current Instruction Address, or CIA) plus the signed displacement. This address calculation is defined to "wrap around" from the maximum effective address (0xFFFFFFFF) to 0x0000 0000, and vice-versa.

For the absolute form of the branch and branch conditional instructions (**ba**[**l**] and **bca**[**l**], with instruction field AA = 1), the target address is the sign-extended displacement. This means that with absolute forms of the branch and branch conditional instructions, the branch target can be within the first or last 32MB or 32KB of the address space, respectively.

The other two branch instructions, **bclr** (branch conditional to LR) and **bcctr** (branch conditional to CTR), do not use absolute nor relative addressing. Instead, they use *indirect* addressing, in which the target of the branch is specified indirectly as the contents of the LR or CTR.

4.5.2 Branch Instruction BI Field

Conditional branch instructions can optionally test one bit of the CR, as indicated by instruction field BO[0] (see BO field description below). The value of instruction field BI specifies the CR bit to be tested (0-31). The BI field is ignored if BO[0] = 1. The branch (**b**[**l**][**a**]) instruction is by definition unconditional, and hence does not have a BI instruction field. Instead, the position of this field is part of the LI displacement field.

4.5.3 Branch Instruction BO Field

The BO field specifies the condition under which a conditional branch is taken, and whether the branch decrements the CTR. The branch (**b**[**l**][**a**]) instruction is by definition unconditional, and hence does not have a BO instruction field. Instead, the position of this field is part of the LI displacement field.

Conditional branch instructions can optionally test one bit in the CR. This option is selected when BO[0] = 0; if BO[0] = 1, the CR does not participate in the branch condition test. If the CR condition option is selected, the condition is satisfied (branch can occur) if the CR bit selected by the BI instruction field matches BO[1].

Conditional branch instructions can also optionally decrement the CTR by one, and test whether the decremented value is 0. This option is selected when BO[2] = 0; if BO[2] = 1, the CTR is not decremented and does not participate in the branch condition test. If CTR decrement option is selected, BO[3] specifies the condition that must be satisfied to allow the branch to be taken. If BO[3] = 0, CTR ≠ 0 is required for the branch to occur. If BO[3] = 1, CTR = 0 is required for the branch to occur.

~~Table 4-35~~*Table 4-35* summarizes the usage of the bits of the BO field. BO[4] is further discussed in ~~"Branch Prediction."~~*Branch Prediction* on page 187

Table 4-35. BO Field Definition

BO Bit	Description
BO[0]	CR Test Control 0 Test CR bit specified by BI field for value specified by BO[1] 1 Do not test CR
BO[1]	CR Test Value 0 If BO[0] = 0, test for CR[BI] = 0. 1 If BO[0] = 0, test for CR[BI] = 1.
BO[2]	CTR Decrement and Test Control 0 Decrement CTR by one and test whether the decremented CTR satisfies the condition specified by BO[3]. 1 Do not decrement CTR, do not test CTR.
BO[3]	CTR Test Value 0 If BO[2] = 0, test for decremented CTR ≠ 0. 1 If BO[2] = 0, test for decremented CTR = 0.
BO[4]	Branch Prediction Reversal 0 Apply standard branch prediction. 1 Reverse the standard branch prediction.

~~Table 4-36~~*Table 4-36* lists specific BO field contents, and the resulting actions; *z* represents a mandatory value of zero, and *y* is a branch prediction option discussed in ~~"Branch Prediction."~~*Branch Prediction* on page 187

Table 4-36. BO Field Examples

BO Value	Description
0000*y*	Decrement the CTR, then branch if the decremented CTR ≠ 0 and CR[BI]=0.
0001*y*	Decrement the CTR, then branch if the decremented CTR = 0 and CR[BI] = 0.
001*zy*	Branch if CR[BI] = 0.
0100*y*	Decrement the CTR, then branch if the decremented CTR ≠ 0 and CR[BI] = 1.
0101*y*	Decrement the CTR, then branch if the decremented CTR=0 and CR[BI] = 1.
011*zy*	Branch if CR[BI] = 1.
1*z*00*y*	Decrement the CTR, then branch if the decremented CTR ≠ 0.
1*z*01*y*	Decrement the CTR, then branch if the decremented CTR = 0.
1*z*1*zz*	Branch always.

4.5.4 Branch Prediction

Conditional branches might be taken or not taken; if taken, instruction fetching is re-directed to the target address. If the branch is not taken, instruction fetching simply falls through to the next sequential instruction. The PPC440GP attempts to predict whether or not a branch is taken before all information necessary to determine the branch direction is available. This action is called *branch prediction*. The processor core can then prefetch instructions down the predicted path. If the prediction is correct, performance is improved because the branch target instruction is available immediately, instead of having to wait until the branch

conditions are resolved. If the prediction is incorrect, then the prefetched instructions (which were fetched from addresses down the "wrong" path of the branch) must be discarded, and new instructions fetched from the correct path.

The PPC440GP combines the static prediction mechanism defined by PowerPC Book-E, together with a dynamic branch prediction mechanism, in order to provide correct branch prediction as often as possible. The dynamic branch prediction mechanism is an implementation optimization, and is not part of the architecture, nor is it visible to the programming model. Appendix B, "PPC440GP Compiler Optimizations," provides additional information on the dynamic branch prediction mechanism.

The static branch prediction mechanism enables software to designate the "preferred" branch prediction via bits in the instruction encoding. The "default" static branch prediction for conditional branches is as follows:

Predict that the branch is to be taken if ((BO[0] $\wedge$ BO[2]) $\vee$ *s*) = 1

where *s* is bit 16 of the instruction (the sign bit of the displacement for all **bc** forms, and zero for all **bclr** and **bcctr** forms). In other words, conditional branches are predicted taken if they are "unconditional" (i.e., they do not test the CR nor the CTR decrement, and are always taken), or if their branch displacement is "negative" (i.e., the branch is branching "backwards" from the current instruction address). The standard prediction for this case derives from considering the relative form of **bc**, often used at the end of loops to control the number of times that a loop is executed. The branch is taken each time the loop is executed except the last, so it is best if the branch is predicted taken. The branch target is the beginning of the loop, so the branch displacement is negative and *s* = 1. Because this situation is most common, a branch is taken if *s* = 1.

If branch displacements are positive, *s* = 0, then the branch is predicted not taken. Also, if the branch instruction is any form of **bclr** or **bcctr** except the "unconditional" form, then *s* = 0, and the branch is predicted not taken.

There is a peculiar consequence of this prediction algorithm for the absolute forms of **bc** (**bca** and **bcla**). As described in ~~"Branch Addressing" on page 4-46~~*Branch Addressing* on page 185, if *s* = 1, the branch target is in high memory. If *s* = 0, the branch target is in low memory. Because these are absolute-addressing forms, there is no reason to treat high and low memory differently. Nevertheless, for the high memory case the standard prediction is taken, and for the low memory case the standard prediction is not taken.

Another bit in the BO field allows software further control over branch prediction. Specifically, BO[4] is the *prediction reversal bit*. If BO[4] = 0, the default prediction is applied. If BO[4] = 1, the reverse of the default prediction is applied. For the cases in ~~Table 4-36~~*Table 4-36* where BO[4] = *y*, software can reverse the default prediction by setting *y* to 1. This should only be done when the default prediction is likely to be wrong. Note that for the "branch always" condition, reversal of the default prediction is not allowed, as BO[4] is designated as *z* for this case, meaning the bit must be set to 0 or the instruction form is invalid.

4.5.5 Branch Control Registers

There are three registers in the PPC440GP which are associated with branch processing, and they are described in the following sections.

4.5.5.1 Link Register (LR)

The LR is written from a GPR using **mtspr**, and can be read into a GPR using **mfspr**. The LR can also be updated by the "link update" form of branch instructions (instruction field LK = 1). Such branch instructions load the LR with the address of the instruction following the branch instruction (4 + address of the branch

instruction). Thus, the LR contents can be used as a return address for a subroutine that was entered using a link update form of branch. The **bclr** instruction uses the LR in this fashion, enabling indirect branching to any address.

When being used as a return address by a **bclr** instruction, bits 30:31 of the LR are ignored, since all instruction addresses are on word boundaries.
Access to the LR is non-privileged.

Figure 0-1. Link Register (LR)

0:31		Link Register contents	Target address of **bclr** instruction

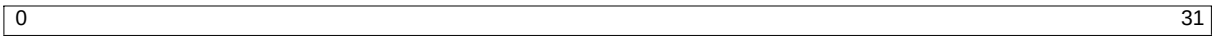


Figure 4-3. Link Register (LR)

0:31		Link Register contents	Target address of **bclr** instruction

4.5.5.2 Count Register (CTR)

The CTR is written from a GPR using **mtspr**, and can be read into a GPR using **mfspr**. The CTR contents can be used as a loop count that gets decremented and tested by conditional branch instructions that specify count decrement as one of their branch conditions (instruction field BO[2] = 0). Alternatively, the CTR contents can specify a target address for the **bcctr** instruction, enabling indirect branching to any address.

Access to the CTR is non-privileged.

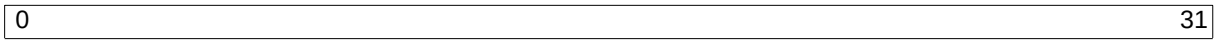


Figure 0-2. Count Register (CTR)

0:31		Count	Used as count for branch conditional with decrement instructions, or as target address for **bcctr** instructions

Figure 4-4. Count Register (CTR)

0:31		Count	Used as count for branch conditional with decrement instructions, or as target address for **bcctr** instructions

4.5.5.3 Condition Register (CR)

The CR is used to record certain information ("conditions") related to the results of the various instructions which are enabled to update the CR. A bit in the CR may also be selected to be tested as part of the condition of a conditional branch instruction.

The CR is organized into eight 4-bit fields (CR0–CR7), as shown in ~~Figure 4-5~~Figure 4-5. Table 4-37 lists the instructions which update the CR.

Access to the CR is non-privileged.

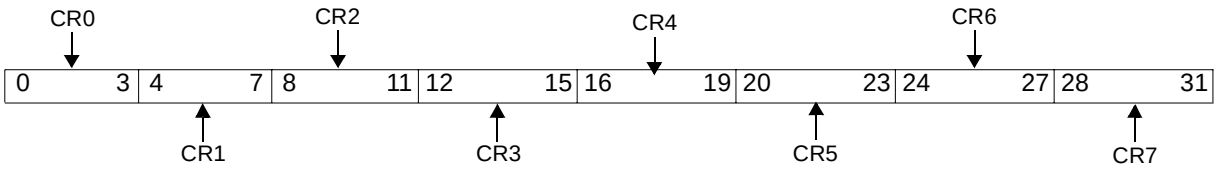


Figure 0-3. Condition Register (CR)

0:3	CR0	Condition Register Field 0
4:7	CR1	Condition Register Field 1
8:11	CR2	Condition Register Field 2
12:15	CR3	Condition Register Field 3
16:19	CR4	Condition Register Field 4
20:23	CR5	Condition Register Field 5
24:27	CR6	Condition Register Field 6
28:31	CR7	Condition Register Field 7

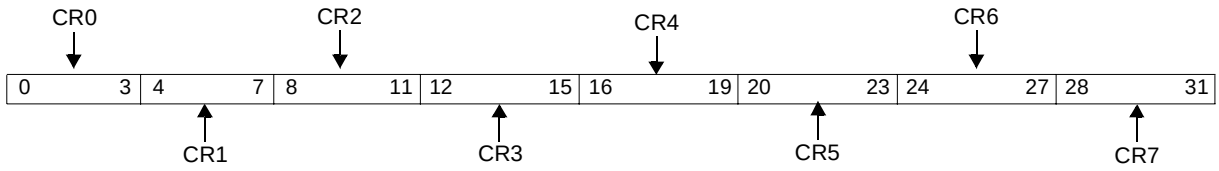


Figure 4-5. Condition Register (CR)

0:3	CR0	Condition Register Field 0
4:7	CR1	Condition Register Field 1
8:11	CR2	Condition Register Field 2
12:15	CR3	Condition Register Field 3
16:19	CR4	Condition Register Field 4
20:23	CR5	Condition Register Field 5
24:27	CR6	Condition Register Field 6
28:31	CR7	Condition Register Field 7

Table 4-37. CR Updating Instructions

Integer						Processor Control	Storage Control	
Storage Access	Arithmetic	Logical	Compare	Rotate	Shift	CR-Logical and Register Management	TLB Mgmt.	Multiply-Accumulate and Logical
stwcx.	**add.**[**o**] **addc.**[**o**] **adde.**[**o**] **addic.** **addme.**[**o**] **addze.**[**o**] **subf.**[**o**] **subfc.**[**o**] **subfe.**[**o**] **subfme.**[**o**] **subfze.**[**o**] **mulhw.** **mulhwu.** **mullw.**[**o**] **divw.**[**o**] **divwu.**[**o**] **neg.**[**o**]	**and.** **andi.** **andis.** **andc.** **nand.** **or.** **orc.** **nor.** **xor.** **eqv.** **extsb.** **extsh.** **cntlzw.**	**cmp** **cmpi** **cmpl** **cmpli**	**rlwimi.** **rlwinm.** **rlwnm.**	**slw.** **srw.** **sraw.** **srawi.**	**crand** **crandc** **creqv** **crnand** **crnor** **cror** **crorc** **crxor** **mcrf** **mcrxr** ~~**mfcr**~~ **mtcrf**	**tlbsx.**	**macchw.**[**o**] **macchws.**[**o**] **macchwsu.**[**o**] **macchwu.**[**o**] **machhw.**[**o**] **machhws.**[**o**] **machhwsu.**[**o**] **machhwu.**[**o**] **maclhw.**[**o**] **maclhws.**[**o**] **maclhwsu.**[**o**] **maclhwu.**[**o**] **nmacchw.**[**o**] **nmacchws.**[**o**] **nmachhw.**[**o**] **nmachhws.**[**o**] **nmaclhw.**[**o**] **nmaclhws.**[**o**] **mulchw.** **mulchwu.** **mulhhw.** **mulhhwu.** **mullhw.** **mullhwu.** **dlmzb.**

~~Chapter 28, "Instruction Set"~~*Instruction Set* on page 893, provides detailed information on how each of these instructions updates the CR. To summarize, the CR can be accessed in any of the following ways:

- **mfcr** reads the CR into a GPR. Note that this instruction does not *update* the CR and is therefore not listed in Table 4-37.
- Conditional branch instructions can designate a CR bit to be used as a branch condition. Note that these instructions do not *update* the CR and are therefore not listed in Table 4-37.
- **mtcrf** sets specified CR fields by writing to the CR from a GPR, under control of a mask field specified as part of the instruction.
- **mcrf** updates a specified CR field by copying another specified CR field into it.
- **mcrxr** copies certain bits of the XER into a specified CR field, and clears the corresponding XER bits.
- Integer compare instructions update a specified CR field.
- CR-logical instructions update a specified CR bit with the result of any one of eight logical operations on a specified pair of CR bits.

- Certain forms of various integer instructions (the "." forms) implicitly update CR[CR0], as do certain forms of the auxiliary processor instructions implemented within the PPC440GP.

CR[CR0] Implicit Update By Integer Instructions

Most of the CR-updating instructions listed in Table 4-37 implicitly update the CR0 field. These are the various "dot-form" instructions, indicated by a "." in the instruction mnemonic. Most of these instructions update CR[CR0] according to an arithmetic comparison of 0 with the 32-bit result which the instruction writes to the GPR file. That is, after performing the operation defined for the instruction, the 32-bit result which is written to the GPR file is compared to 0 using a signed comparison, independent of whether the actual operation being performed by the instruction is considered "signed" or not. For example, logical instructions such as **and.**, **or.**, and **nor.** update CR[CR0] according to this signed comparison to 0, even though the result of such a logical operation is not typically interpreted as a signed value. For each of these dot-form instructions, the individual bits in CR[CR0] are updated as follows:

$CR[CR0]_0$ — LT	Less than 0; set if the most-significant bit of the 32-bit result is 1.
$CR[CR0]_1$ — GT	Greater than 0; set if the 32-bit result is non-zero and the most-significant bit of the result is 0.
$CR[CR0]_2$ — EQ	Equal to 0; set if the 32-bit result is 0.
$CR[CR0]_3$ — SO	Summary overflow; a copy of XER[SO] at the completion of the instruction (including any XER[SO] update being performed the instruction itself.

Note that if an arithmetic overflow occurs, the "sign" of an instruction result indicated in CR[CR0] might not represent the "true" (infinitely precise) algebraic result of the instruction that set CR0. For example, if an **add.** instruction adds two large positive numbers and the magnitude of the result cannot be represented as a twos-complement number in a 32-bit register, an overflow occurs and $CR[CR0]_0$ is set, even though the infinitely precise result of the add is positive.

Similarly, adding the largest 32-bit twos-complement negative number (0x8000 0000) to itself results in an arithmetic overflow and 0x0000 0000 is recorded in the target register. $CR[CR0]_2$ is set, indicating a result of 0, but the infinitely precise result is negative.

$CR[CR0]_3$ is a copy of XER[SO] at the completion of the instruction, whether or not the instruction which is updating CR[CR0] is also updating XER[SO]. Note that if an instruction causes an arithmetic overflow but is not of the form which actually updates XER[SO], then the value placed in $CR[CR0]_3$ does not reflect the arithmetic overflow which occurred on the instruction (it is merely a copy of the value of XER[SO] which was already in the XER before the execution of the instruction updating CR[CR0]).

There are a few dot-form instructions which do not update CR[CR0] in the fashion described above. These instructions are: **stwcx.**, **tlbsx.**, and **dlmzb.** See the instruction descriptions in ~~Chapter 28, "Instruction Set,"~~ *Instruction Set* on page 893 for details on how these instructions update CR[CR0].

CR Update By Integer Compare Instructions

Integer compare instructions update a specified CR field with the result of a comparison of two 32-bit numbers, the first of which is from a GPR and the second of which is either an immediate value or from another GPR. There are two types of integer compare instructions, *arithmetic* and *logical*, and they are distinguished by the interpretation given to the 32-bit numbers being compared. For *arithmetic* compares, the numbers are considered to be signed, whereas for *logical* compares, the numbers are considered to be unsigned. As an example, consider the comparison of 0 with 0xFFFF FFFF. In an *arithmetic* compare, 0 is larger; in a *logical* compare, 0xFFFF FFFF is larger.

A compare instruction can direct its result to any CR field. The BF field (bits 6:8) of the instruction specifies the CR field to be updated. After a compare, the specified CR field is interpreted as follows:

$CR[(BF)]_0$ — LT	The first operand is less than the second operand.
$CR[(BF)]_1$ — GT	The first operand is greater than the second operand.
$CR[(BF)]_2$ — EQ	The first operand is equal to the second operand.
$CR[(BF)]_3$ — SO	Summary overflow; a copy of XER[SO].

4.6 Integer Processing

Integer processing includes loading and storing data between memory and GPRs, as well as performing various operations on the values in GPRs and other registers (the categories of integer instructions are summarized in ~~Table 4-18 on page 4-38~~*Table 4-18* on page 178). The sections which follow describe the registers which are used for integer processing, and how they are updated by various instructions. In addition, ~~"Condition Register (CR)" on page 4-50~~*Condition Register (CR)* on page 189 provides more information on the CR updates caused by integer instructions. Finally, ~~Chapter 28, "Instruction Set"~~*Instruction Set* on page 893 also provides details on the various register updates performed by integer instructions.

4.6.1 General Purpose Registers (GPRs)

The PPC440GP contains 32 GPRs. The contents of these registers can be transferred to and from memory using integer storage access instructions. Operations are performed on GPRs by most other instructions.

Access to the GPRs is non-privileged.

Figure 0-4. General Purpose Registers (R0-R31)

0:31		General Purpose Register data

Figure 4-6. General Purpose Registers (R0-R31)

0:31		General Purpose Register data

4.6.2 Integer Exception Register (XER)

The XER records overflow and carry indications from integer arithmetic and shift instructions. It also provides a byte count for string indexed integer storage access instructions (**lswx** and **stswx**). Note that the term *exception* in the name of this register does not refer to exceptions as they relate to interrupts, but rather to the *arithmetic* exceptions of carry and overflow.

~~Figure 4-7~~Figure 4-7 illustrates the fields of the XER, while Tables 4-38 and 4-39 list the instructions which update XER[SO,OV] and the XER[CA] fields, respectively. The sections which follow the figure and tables describe the fields of the XER in more detail.

Access to the XER is non-privileged.

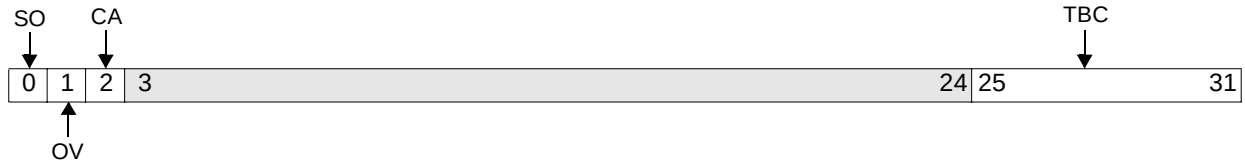


Figure 0-5. Integer Exception Register (XER)

0	SO	Summary Overflow 0 No overflow has occurred. 1 Overflow has occurred.	Can be *set* by **mtspr** or by integer or auxiliary processor instructions with the [**o**] option; can be *reset* by **mtspr** or by **mcrxr**.
1	OV	Overflow 0 No overflow has occurred. 0 Overflow has occurred.	Can be *set* by **mtspr** or by integer or allocated structions with the [**o**] option; can be *reset* by **mtspr**, by **mcrxr**, or by integer or allocated instructions with the [**o**] option.
2	CA	Carry 0 Carry has not occurred. 1 Carry has occurred.	Can be *set* by **mtspr** or by certain integer arithmetic and shift instructions; can be *reset* by **mtspr**, by **mcrxr**, or by certain integer arithmetic and shift instructions.
3:24		Reserved	
25:31	TBC	Transfer Byte Count	Used as a byte count by **lswx** and **stswx**; written by **dlmzb**[**.**] and by **mtspr.**

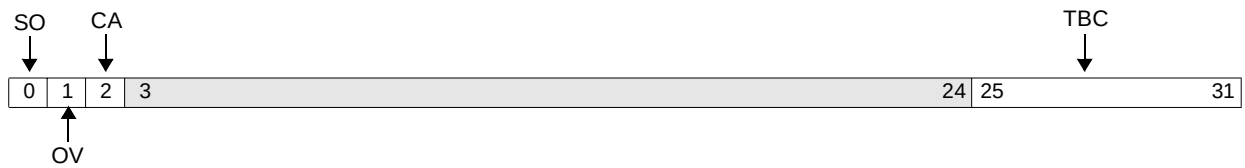


Figure 4-7. Integer Exception Register (XER)

0	SO	Summary Overflow 0 No overflow has occurred. 1 Overflow has occurred.	Can be *set* by **mtspr** or by integer or auxiliary processor instructions with the [**o**] option; can be *reset* by **mtspr** or by **mcrxr**.

1	OV	Overflow 0 No overflow has occurred. 1 Overflow has occurred.	Can be *set* by **mtspr** or by integer or allocated instructions with the [**o**] option; can be *reset* by **mtspr**, by **mcrxr**, or by integer or allocated instructions with the [**o**] option.
2	CA	Carry 0 Carry has not occurred. 1 Carry has occurred.	Can be *set* by **mtspr** or by certain integer arithmetic and shift instructions; can be *reset* by **mtspr**, by **mcrxr**, or by certain integer arithmetic and shift instructions.
3:24		Reserved	
25:31	TBC	Transfer Byte Count	Used as a byte count by **lswx** and **stswx**; written by **dlmzb**[**.**] and by **mtspr.**

Table 4-38. XER[SO,OV] Updating Instructions

Integer Arithmetic							Processor Control
Add	Subtract	Multiply	Divide	Negate	Multiply-Accumulate	Negative Multiply- Accumulate	Register Management
addo[**.**] **addco**[**.**] **addeo**[**.**] **addmeo**[**.**] **addzeo**[**.**]	**subfo**[**.**] **subfco**[**.**] **subfeo**[**.**] **subfmeo**[**.**] **subfzeo**[**.**]	**mullwo**[**.**]	**divwo**[**.**] **divwuo**[**.**]	**nego**[**.**]	**macchwo**[**.**] **macchwso**[**.**] **macchwsuo**[**.**] **macchwuo**[**.**] **machhwo**[**.**] **machhwso**[**.**] **machhwsuo**[**.**] **machhwuo**[**.**] **maclhwo**[**.**] **maclhwso**[**.**] **maclhwsuo**[**.**] **maclhwuo**[**.**]	**nmacchwo**[**.**] **nmacchwso**[**.**] **nmachhwo**[**.**] **nmachhwso**[**.**] **nmaclhwo**[**.**] **nmaclhwso**[**.**]	**mtspr** **mcrxr**

Table 4-39. XER[CA] Updating Instructions

Integer Arithmetic		Integer Shift	Processor Control
Add	Subtract	Shift Right Algebraic	Register Management
addc[**o**][**.**] **adde**[**o**][**.**] **addic**[**.**] **addme**[**o**][**.**] **addze**[**o**][**.**]	**subfc**[**o**][**.**] **subfe**[**o**][**.**] **subfic** **subfme**[**o**][**.**] **subfze**[**o**][**.**]	**sraw**[**.**] **srawi**[**.**]	**mtspr** **mcrxr**

4.6.2.1 Summary Overflow (SO) Field

This field is set to 1 when an instruction is executed that causes XER[OV] to be set to 1, except for the case of **mtspr**(XER), which writes XER[SO,OV] with the values in $(RS)_{0:1}$, respectively. Once set, XER[SO] is not reset until either an **mtspr**(XER) is executed with data that explicitly writes 0 to XER[SO], or until an **mcrxr** instruction is executed. The **mcrxr** instruction sets XER[SO] (as well as XER[OV,CA]) to 0 after copying all three fields into $CR[CR0]_{0:2}$ (and setting $CR[CR0]_3$ to 0).

Given this behavior, XER[SO] does not necessarily indicate that an overflow occurred on the most recent integer arithmetic operation, but rather that one occurred at some time subsequent to the last clearing of XER[SO] by **mtspr**(XER) or **mcrxr**.

XER[SO] is read (along with the rest of the XER) into a GPR by **mfspr**(XER). In addition, various integer instructions copy XER[SO] into $CR[CR0]_3$ (see ~~"Condition Register (CR)" on page 4-50~~*Condition Register (CR)* on page 189).

4.6.2.2 Overflow (OV) Field

This field is updated by certain integer arithmetic instructions to indicate whether the infinitely precise result of the operation can be represented in 32 bits. For those integer arithmetic instructions that update XER[OV] and produce *signed* results, XER[OV] = 1 if the result is greater than $2^{31} – 1$ or less than $–2^{31}$; otherwise, XER[OV] = 0. For those integer arithmetic instructions that update XER[OV] and produce *unsigned* results (certain integer divide instructions and multiply-accumulate instructions), XER[OV] = 1 if the result is greater than $2^{32}–1$; otherwise, XER[OV] = 0. See the instruction descriptions in ~~Chapter 28, "Instruction Set"~~*Instruction Set* on page 893 for more details on the conditions under which the integer divide instructions set XER[OV] to 1.

The **mtspr**(XER) and **mcrxr** instructions also update XER[OV]. Specifically, **mcrxr** sets XER[OV] (and XER[SO,CA]) to 0 after copying all three fields into $CR[CR0]_{0:2}$ (and setting $CR[CR0]_3$ to 0), while **mtspr**(XER) writes XER[OV] with the value in $(RS)_1$.

XER[OV] is read (along with the rest of the XER) into a GPR by **mfspr**(XER).

4.6.2.3 Carry (CA) Field

This field is updated by certain integer arithmetic instructions (the "carrying" and "extended" versions of add and subract) to indicate whether or not there is a carry-out of the most-significant bit of the 32-bit result. XER[CA] = 1 indicates a carry. The integer shift right algebraic instructions update XER[CA] to indicate whether or not any 1-bits were shifted out of the least significant bit of the result, if the source operand was negative (see the instruction descriptions in ~~Chapter 28, "Instruction Set"~~*Instruction Set* on page 893 for more details).

The **mtspr**(XER) and **mcrxr** instructions also update XER[CA]. Specifically, **mcrxr** sets XER[CA] (as well as XER[SO,OV]) to 0 after copying all three fields into $CR[CR0]_{0:2}$ (and setting $CR[CR0]_3$ to 0), while **mtspr**(XER) writes XER[CA] with the value in $(RS)_2$.

XER[CA] is read (along with the rest of the XER) into a GPR by **mfspr**(XER). In addition, the "extended" versions of the add and subtract integer arithmetic instructions use XER[CA] as a source operand for their arithmetic operations.

Transfer Byte Count (TBC) Field

The TBC field is used by the string indexed integer storage access instructions (**lswx** and **stswx**) as a byte count. The TBC field is updated by the **dlmzb[.]** instruction with a value indicating the number of bytes up to and including the zero byte detected by the instruction (see the instruction description for **dlmzb** in ~~Chapter 28, "Instruction Set"~~*Instruction Set* on page 893 for more details). The TBC field is also written by **mtspr**(XER) with the value in $(RS)_{25:31}$.

XER[TBC] is read (along with the rest of the XER) into a GPR by **mfspr**(XER).

4.7 Processor Control

The PPC440GP provides several registers for general processor control and status. These include:

- Machine State Register (MSR)

  Controls interrupts and other processor functions
- Special Purpose Registers General (SPRGs)

  SPRs for general purpose software use
- Processor Version Register (PVR)

  Indicates the specific implementation of a processor
- Processor Identification Register (PIR)

  Indicates the specific instance of a processor in a multi-processor system
- Core Configuration Register 0 (CCR0)

  Controls specific processor functions, such as instruction prefetch
- Reset Configuration (RSTCFG)

  Reports the values of certain fields of the TLB as supplied at reset

Except for the MSR, each of these registers is described in more detail in the following sections. The MSR is described in more detail in ~~Chapter 10, "Interrupts and Exceptions"~~*Interrupts and Exceptions* on page 337.

4.7.1 Special Purpose Registers General (USPRG0, SPRG0–SPRG7)

USPRG0 and SPRG0–SPRG7 are provided for general purpose, system-dependent software use. One common system usage of these registers is as temporary storage locations. For example, a routine might save the contents of a GPR to an SPRG, and later restore the GPR from it. This is faster than a save/restore to a memory location. These registers are written using **mtspr** and read using **mfspr**.

Access to USPRG0 is non-privileged for both read and write.

Access to SPRG4–SPRG7 is non-privileged for read but privileged for write, and hence different SPR numbers are used for reading than for writing.

Access to SPRG0–SPRG3 is privileged for both read and write.

Figure 0-6. Special Purpose Registers General (USPRG0, SPRG0–SPRG7)

0:31		General data	Software value; no hardware usage.

Figure 4-8. Special Purpose Registers General (USPRG0, SPRG0–SPRG7)

0:31		General data	Software value; no hardware usage.

4.7.2 Processor Version Register (PVR)

The PVR is a read-only register typically used to identify a specific processor core and chip implementation. Software can read the PVR to determine processor core and chip hardware features. The PVR can be read into a GPR using **mfspr**.

Refer to *PowerPC 440GP Embedded Processor Data Sheet* for the PVR value.

Access to the PVR is privileged.

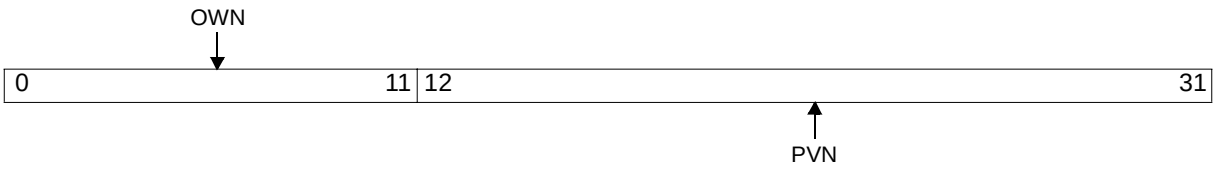


Figure 0-7. Processor Version Register (PVR)

0:31		Processor Version	Refer to *PowerPC 440GP Embedded Processor Data Sheet* for the PVR value.

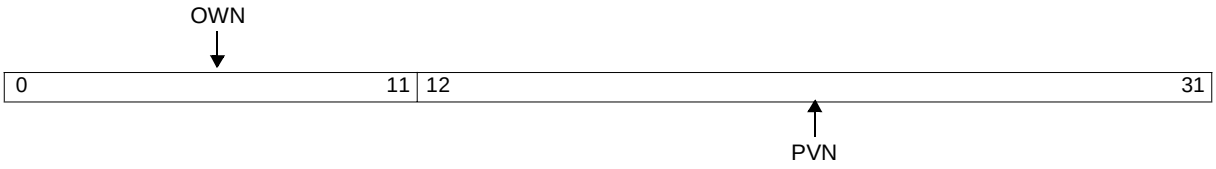


Figure 4-9. Processor Version Register (PVR)

0:31		Processor Version	Refer to *PowerPC 440GP Embedded Processor Data Sheet* for the PVR value.

4.7.3 Processor Identification Register (PIR)

The PIR is a read-only register that uniquely identifies a specific instance of a processor core, within a multiprocessor configuration, enabling software to determine exactly which processor it is running on. This capability is important for operating system software within multiprocessor configurations. The PIR can be read into a GPR using **mfspr**.

Because the PPC440GP is a uniprocessor, PIR[PIN] = 0b0000.

Access to the PIR is privileged.

Figure 0-8. Processor Identification Register (PIR)

0:27		Reserved
28:31	PIN	Processor Identification Number (PIN)

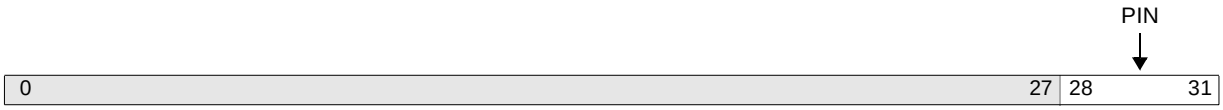


Figure 4-10. Processor Identification Register (PIR)

0:27		Reserved
28:31	PIN	Processor Identification Number (PIN)

4.7.4 Core Configuration Register 0 (CCR0)

The CCR0 controls a number of special chip functions, including data cache and auxiliary processor operation, speculative instruction fetching, trace, and the operation of the cache block touch instructions. The CCR0 is written from a GPR using **mtspr**, and can be read into a GPR using **mfspr**. ~~Figure 4-11~~Figure 4-11 illustrates the fields of the CCR0, and gives a brief description of their functions. A cross reference after the ~~figure~~ bit-field description indicates the ~~sections~~ section of this document ~~in~~ which describes each ~~of the fields is described~~ field in more detail.

Access to the CCR0 is privileged.

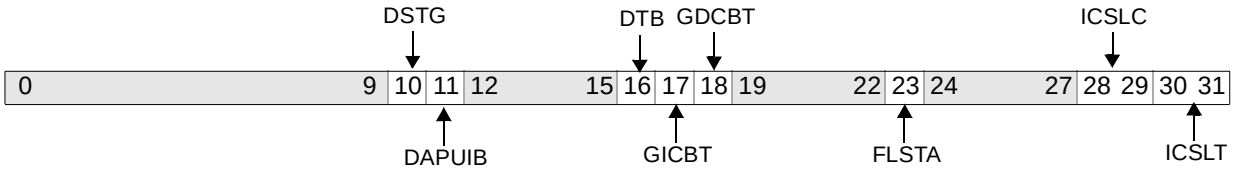


Figure 0-9. Core Configuration Register 0 (CCR0)

0:9		Reserved	
10	DSTG	Disable Store Gathering 0 Enabled; stores to contiguous addresses may be gathered into a single transfer 1 Disabled; all stores to memory will be performed independently	See "Store Gathering" on page 5-21.
11	DAPUIB	Disable APU Instruction Broadcast 0 Enabled. 1 Disabled; instructions not broadcast to APU for decoding	This mechanism is provided as a means of reducing power consumption when an auxilliary processor is not attached and/or is not being used. See Chapter 7, "Reset and Initialization".
12:15		Reserved	
16	DTB	Disable Trace Broadcast 0 Enabled. 1 Disabled; no trace information is broadcast.	This mechanism is provided as a means of reducing power consumption when instruction tracing is not needed. See Chapter 7, "Reset and Initialization".
17	GICBT	Guaranteed Instruction Cache Block Touch 0 **icbt** may be abandoned without having filled cache line if instruction pipeline stalls. 1 **icbt** is guaranteed to fill cache line even if instruction pipeline stalls.	See "icbt Operation" on page 5-15.
18	GDCBT	Guaranteed Data Cache Block Touch 0 **dcbt/dcbtst** may be abandoned without having filled cache line if instruction pipeline stalls. 1 **dcbt/dcbtst** are guaranteed to fill cache line even if instruction pipeline stalls.	See "Data Cache Control and Debug" on page 5-25.
19:22		Reserved	
23	FLSTA	Force Load/Store Alignment 0 No Alignment exception on integer storage access instructions, regardless of alignment 1 An alignment exception occurs on integer storage access instructions if data address is not on an operand boundary.	See "Load and Store Alignment" on page 5-19.
24:27		Reserved	

28:29	ICSLC	Instruction Cache Speculative Line Count	Number of additional lines (0–3) to fill on instruction fetch miss. See "Speculative Prefetch Mechanism" on page 5-10.
30:31	ICSLT	Instruction Cache Speculative Line Threshold	Number of doublewords that must have already been filled in order that the current speculative line fill is *not* abandoned on a redirection of the instruction stream. See "Speculative Prefetch Mechanism" on page 5-10.

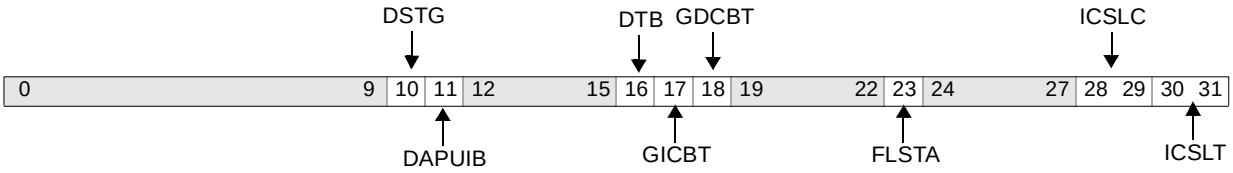


Figure 4-11. Core Configuration Register 0 (CCR0)

0:9		Reserved	
10	DSTG	Disable Store Gathering 0 Enabled; stores to contiguous addresses may be gathered into a single transfer 1 Disabled; all stores to memory will be performed independently	See "Store Gathering" on page -223.
11	DAPUIB	Disable APU Instruction Broadcast 0 Enabled. 1 Disabled; instructions not broadcast to APU for decoding	This mechanism is provided as a means of reducing power consumption when an auxilliary processor is not attached and/or is not being used. See *Reset and Initialization* on page 259.
12:15		Reserved	
16	DTB	Disable Trace Broadcast 0 Enabled. 1 Disabled; no trace information is broadcast.	This mechanism is provided as a means of reducing power consumption when instruction tracing is not needed. See *Reset and Initialization* on page 259.
17	GICBT	Guaranteed Instruction Cache Block Touch 0 **icbt** may be abandoned without having filled cache line if instruction pipeline stalls. 1 **icbt** is guaranteed to fill cache line even if instruction pipeline stalls.	See *icbt Operation* on page 218.
18	GDCBT	Guaranteed Data Cache Block Touch 0 **dcbt/dcbtst** may be abandoned without having filled cache line if load/store pipeline stalls. 1 **dcbt/dcbtst** are guaranteed to fill cache line even if load/store pipeline stalls.	See *Data Cache Control and Debug* on page 228.
19:22		Reserved	

23	FLSTA	Force Load/Store Alignment 0 No Alignment exception on integer storage access instructions, regardless of alignment 1 An alignment exception occurs on integer storage access instructions if data address is not on an operand boundary.	See *Load and Store Alignment* on page 222.
24:27		Reserved	
28:29	ICSLC	Instruction Cache Speculative Line Count	Number of additional lines (0–3) to fill on instruction fetch miss. See *Speculative Prefetch Mechanism* on page 213.
30:31	ICSLT	Instruction Cache Speculative Line Threshold	Number of doublewords that must have already been filled in order that the current speculative line fill is *not* abandoned on a redirection of the instruction stream. See *Speculative Prefetch Mechanism* on page 213.

4.7.5 Reset Configuration (RSTCFG)

The read-only RSTCFG register reports the values of certain fields of TLB as supplied at reset.

Access to RSTCFG is privileged.

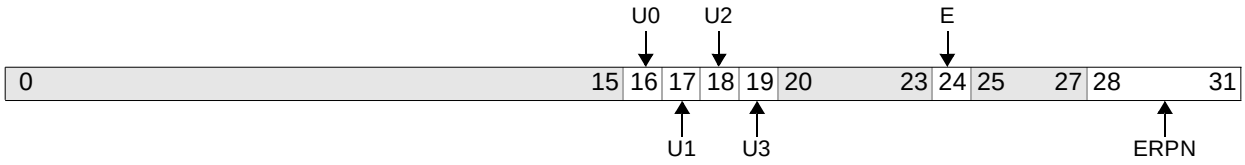


Figure 0-10. Reset Configuration

0:15		Reserved	
16	U0	U0 Storage Attribute 0 U0 storage attribute is disabled 1 U0 storage attribute is enabled	U0 has no effect in the PPC440GP.
17	U1	U1 Storage Attribute 0 Memory page contains normal instructions and data 1 Memory page contains transient instructions or data	
18	U2	U2 Storage Attribute 0 A storage miss does not cause a line to be allocated in the data cache 1 A storage miss causes a line to be allocated in the data cache	

19	U3	U3 Storage Attribute 0 U3 storage attribute is disabled 1 U3 storage attribute is enabled	U3 has no effect in the PPC440GP.
20:23		Reserved	
24	E	E Storage Attribute 0 Accesses to the page are big endian. 1 Accesses to the page are little endian.	
25:27		Reserved	
28:31	ERPN	Extended Real Page Number	f ROM is connected to EBCO, ERPN is 0b0001. If ROM is connected to PCI, ERPN is 0b0010.

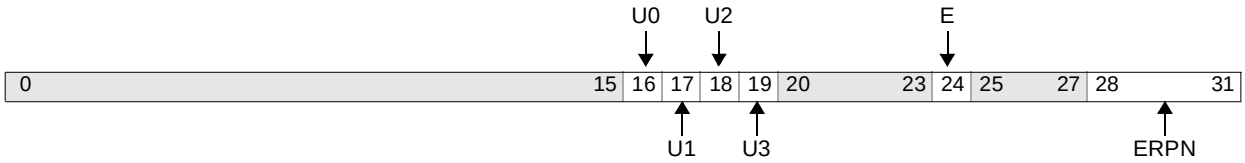


Figure 4-12. Reset Configuration

0:15		Reserved	
16	U0	U0 Storage Attribute 0 U0 storage attribute is disabled 1 U0 storage attribute is enabled	U0 has no effect in the PPC440GP.
17	U1	U1 Storage Attribute 0 Memory page contains normal instructions and data 1 Memory page contains transient instructions or data	
18	U2	U2 Storage Attribute 0 A storage miss does not cause a line to be allocated in the data cache 1 A storage miss causes a line to be allocated in the data cache	
19	U3	U3 Storage Attribute 0 U3 storage attribute is disabled 1 U3 storage attribute is enabled	U3 has no effect in the PPC440GP.
20:23		Reserved	
24	E	E Storage Attribute 0 Accesses to the page are big endian. 1 Accesses to the page are little endian.	
25:27		Reserved	
28:31	ERPN	Extended Real Page Number	f ROM is connected to EBCO, ERPN is 0b0001. If ROM is connected to PCI, ERPN is 0b0010.

4.8 User and Supervisor Modes

PowerPC Book-E architecture defines two operating "states" or "modes," supervisor (privileged), and user (non-privileged). Which mode the processor is operating in is controlled by MSR[PR]. When MSR[PR] is 0, the processor is in supervisor mode, and can execute all instructions and access all registers, including privileged ones. When MSR[PR] is 1, the processor is in user mode, and can only execute non-privileged instructions and access non-privileged registers. An attempt to execute a privileged instruction or to access a privileged register while in user mode causes a Privileged Instruction exception type Program interrupt to occur.

Note that the name "PR" for the MSR field refers to an historical alternative name for user mode, which is "problem state." Hence the value 1 in the field indicates "problem state," and not "privileged" as one might expect.

4.8.1 Privileged Instructions

The following instructions are privileged and cannot be executed in user mode:

Table 0-17. Privileged Instructions

dcbi	
dccci	
dcread	
iccci	
icread	
mfdcr	
mfmsr	
mfspr	For any SPR Number with $SPRN_5$ = 1. See "Privileged SPRs" on page 4-61.
mtdcr	
mtmsr	
mtspr	For any SPR Number with $SPRN_5$ = 1. See "Privileged SPRs" on page 4-61.
rfci	
rfi	
wrtee	
wrteei	

Table 4-40. Privileged Instructions

dcbi	
dccci	
dcread	
iccci	
icread	
mfdcr	
mfmsr	

Table 4-40. Privileged Instructions (continued)

mfspr	For any SPR Number with $SPRN_5$ = 1. See *Privileged SPRs* on page 200.
mtdcr	
mtmsr	
mtspr	For any SPR Number with $SPRN_5$ = 1. See *Privileged SPRs* on page 200.
rfci	
rfi	
tlbre	
tlbsx	
tlbsync	
tlbwe	
wrtee	
wrteei	

4.8.2 Privileged SPRs

Most SPRs are privileged. The only defined non-privileged SPRs are the LR, CTR, XER, USPRG0, SPRG4–7 (read access only), TBU (read access only), and TBL (read access only). The PPC440GP also treats all SPR numbers with a 1 in bit 5 of the SPRN field as privileged, whether the particular SPR number is defined or not. Thus the processor core ~~cause~~ causes a Privileged Instruction exception type Program interrupt on any attempt to access such an SPR number while in user mode. In addition, the processor ~~corecore~~ core causes an Illegal Instruction exception type Program interrupt on any attempt to access while in user mode an undefined SPR number with a 0 in $SPRN_5$. On the other hand, the result of attempting to access an undefined SPR number in supervisor mode is undefined, regardless of the value in $SPRN_5$.

4.9 Speculative Accesses

The PowerPC Book-E Architecture permits implementations to perform speculative accesses to memory, either for instruction fetching, or for data loads. A speculative access is defined as any access that is not required by the sequential execution model (SEM).

For example, the PPC440GP speculatively prefetches instructions down the predicted path of a conditional branch; if the branch is later determined to not go in the predicted direction, the fetching of the instructions from the predicted path is not required by the SEM and thus is speculative. Similarly, the PPC440GP executes load instructions out-of-order, and may read data from memory for a load instruction that is past an undetermined branch.

Sometimes speculative accesses are inappropriate, however. For example, attempting to access data at addresses to which I/O devices are mapped can cause problems. If the I/O device is a serial port, reading it speculatively could cause data to be lost.

The architecture provides two mechanisms for protecting against errant accesses to such "non-well-behaved" memory addresses. The first is the guarded (G) storage attribute, and protects against speculative data accesses. The second is the execute permission mechanism, and protects against speculative instruction fetches. Both of these mechanisms are described in ~~Chapter 6, "Memory Management."~~ *Memory Management* on page 233

4.10 Synchronization

The PPC440GP supports the synchronization operations of the PowerPC Book-E architecture. There are three kinds of synchronization defined by the architecture, each of which is described in the following sections.

4.10.1 Context Synchronization

The context of a program is the environment in which the program executes. For example, the mode (user or supervisor) is part of the context, as are the address translation space and storage attributes of the memory pages being accessed by the program. Context is controlled by the contents of certain registers and other resources, such as the MSR and the translation lookaside buffer (TLB).

Under certain circumstances, it is necessary for the hardware or software to force the synchronization of a program's context. Context synchronizing operations include all interrupts except Machine Check, as well as the **isync**, **sc**, **rfi**, and **rfci** instructions. Context synchronizing operations satisfy the following requirements:

1. The operation is not initiated until all instructions preceding the operation have completed to the point at which they have reported any and all exceptions that they will cause.
2. All instructions *preceding* the operation must complete in the context in which they were initiated. That is, they must not be affected by any context changes caused by the context synchronizing operation, or any instructions *after* the context synchronizing operation.
3. If the operation is the **sc** instruction (which causes a System Call interrupt) or is itself an interrupt, then the operation is not initiated until no higher priority interrupt is pending (see ~~"Interrupts and Exceptions" on page 10-1~~*Interrupts and Exceptions* on page 337).
4. All instructions that *follow* the operation must be re-fetched and executed in the context that is established by the completion of the context synchronizing operation and all of the instructions which *preceded* it.

Note that context synchronizing operations do not force the completion of storage accesses, nor do they enforce any ordering amongst accesses before and/or after the context synchronizing operation. If such behavior is required, then a storage synchronizing instruction must be used (see ~~"Storage Ordering and Synchronization" on page 4-64~~*Storage Ordering and Synchronization* on page 203).

Also note that architecturally Machine Check interrupts are not context synchronizing. Therefore, an instruction that *precedes* a context synchronizing operation can cause a Machine Check interrupt *after* the context synchronizing operation occurs and additional instructions have completed. For the PPC440GP, this can only occur with Data Machine Check exceptions, and not Instruction Machine Check exceptions.

The following scenarios use ~~psuedocode~~ pseudocode examples to illustrate the effects of context synchronization. Subsequent text explains how software can further guarantee "storage ordering."

1. Consider the following self-modifying code instruction sequence:

   ```
   stw XYZ    Store to caching inhibited address XYZ
   isync
   XYZ        fetch and execute the instruction at address XYZ
   ```

   In this sequence, the **isync** instruction does not guarantee that the XYZ instruction is fetched after the store has occurred to memory. There is no guarantee which XYZ instruction will execute; either the old version or the new (stored) version might.

2. Now consider the required self-modifying code sequence:

   stw Write new instruction to data cache

   dcbst Push the new instruction from the data cache to memory

   msync Guarantee that **dcbst** completes before subsequent instructions begin

   icbi invalidate old copy of instruction in instruction cache

   msync Guarantee that **icbi** completes before subsequent instructions begin

   isync force context synchronization, discard ed instructions and re-fetch, fetch of

PVR	OWN	System-dependent	PVR[OWN] value (after reset and otherwise) is specified by core input signals
	PVN	System-dependent	PVR[PVN] value (after reset and otherwise) is specified by core input signals

   stored instruction guaranteed to get new value

3. This final example illustrates the use of **isync** with context changes to the debug facilities

   mtdbcr0 Enable the instruction address compare (IAC) debug event

   isync Wait for the new Debug Control Register 0 (DBCR0) context to be established

   XYZ This instruction is at the IAC address; an **isync** is necessary to guarantee that the IAC event is recognized on the execution of this instruction; without the **isync**, the XYZ instruction may be prefetched and dispatched to execution before recognizing that the IAC event has been enabled.

4.10.2 Execution Synchronization

Execution synchronization is a subset of context synchronization. An execution synchronizing operation satisfies the first two requirements of context synchronizing operations, but not the latter two. That is, execution synchronizing operations guarantee that preceding instructions execute in the "old" context, but do not guarantee that subsequent instructions operate in the "new" context. An example of a scenario requiring execution synchronization would be just before the execution of a TLB-updating instructions (such as **tlbwe**). An execution synchronizing instruction should be executed to guarantee that all preceding storage access instructions have performed their address translations before executing **tlbwe** to invalidate an entry which might be used by those preceding instructions.

There are four execution synchronizing instructions: **mtmsr**, **wrtee**, **wrteei**, and **msync**. Of course, all context synchronizing instruction are also implicitly execution synchronizing, since context synchronization is a superset of execution synchronization.

Note that PowerPC Book-E imposes additional requirements on updates to MSR[EE] (the external interrupt enable bit). Specifically, if a **mtmsr**, **wrtee**, or **wrteei** instruction sets MSR[EE] = 1, and an External Input, Decrementer, or Fixed Interval Timer exception is pending, the interrupt must be taken before the instruction that follows the MSR[EE]-updating is executed. In this sense, these MSR[EE]-updating instructions can be thought of as being context synchronizing with respect to the MSR[EE] bit, in that it guarantees that subsequent instructions execute (or are prevented from executing and an interrupt taken) according to the new context of MSR[EE].

4.10.3 Storage Ordering and Synchronization

Storage synchronization enforces ordering between storage access instructions executed by the PPC440GP. There are two storage synchronizing instructions: **msync** and **mbar**. PowerPC Book-E architecture defines different ordering requirements for these two instructions, but the PPC440GP implements them in an identical fashion. Architecturally, **msync** is the "stronger" of the two, and is also execution synchronizing, whereas **mbar** is not.

mbar acts as a "barrier" between all storage access instructions executed before the **mbar** and all those executed after the **mbar**. That is, **mbar** ensures that all of the storage accesses initiated by instructions before the **mbar** are performed with respect to the memory subsystem before any of the accesses initiated by instructions after the **mbar**. However, **mbar** does not prevent subsequent instructions from executing (nor even from completing) before the completion of the storage accesses initiated by instructions before the **mbar**.

msync, on the other hand, does guarantee that all preceding storage accesses have actually been performed with respect to the memory subsystem before the execution of any instruction after the **msync**. Note that this requirement goes beyond the requirements of mere execution synchronization, in that execution synchronization doesn't require the completion of preceding storage accesses.

The following two examples illustrate the distinctive use of **mbar** vs. **msync**.

```
stw     Store data to an I/O device
msync   Wait for store to actually complete
mtdcr   Reconfigure the I/O device
```

In this example, the **mtdcr** is reconfiguring the I/O device in a manner which would cause the preceding store instruction to fail, were the **mtdcr** to change the device before the completion of the store. Since **mtdcr** is not a storage access instruction, the use of **mbar** instead of **msync** would not guarantee that the store is performed before letting the **mtdcr** reconfigure the device. It only guarantees that subsequent storage accesses are not performed to memory or any device before the earlier store.

Now consider this next example:

```
stb X   Store data to an I/O device at address X, causing a status bit at address Y to be reset
mbar    Guarantee preceding store is performed to the device before any subequent
        storage accesses are performed
lbz Y   Load status from the I/O device at address Y
```

Here, **mbar** is appropriate instead of **msync**, because all that is required is that the store to the I/O device happens before the load does, but not that other instructions subsequent to the **mbar** won't get executed before the store.

6. Memory Management

The PPC440GP supports a uniform, 4 gigabyte (GB) effective address (EA) space, and a 64GB (36-bit) real address (RA) space. The PPC440GP memory management unit (MMU) performs address translation between effective and real addresses, as well as protection functions. With appropriate system software, the MMU supports:

- Translation of effective addresses into real addresses
- Software control of the page replacement strategy
- Page-level access control for instruction and data accesses
- Page-level storage attribute control

6.1 MMU Overview

The PPC440GP generates effective addresses for instruction fetches and data accesses. An effective address is a 32-bit address formed by adding an index or displacement to a base address (see ~~"Effective Address Calculation" on page 4-4~~*Effective Address Calculation* on page 148). Instruction effective addresses are for sequential instruction fetches, and for fetches caused by changes in program flow (branches and interrupts). Data effective addresses are for load, store and cache management instructions. The MMU expands effective addresses into virtual addresses (VAs) and then translates them into real addresses (RAs); the instruction and data caches use real addresses to access memory.

The PPC440GP MMU supports demand-paged virtual memory and other management schemes that depend on precise control of effective to real address mapping and flexible memory protection. Translation misses and protection faults cause precise interrupts. The hardware provides sufficient information to correct the fault and restart the faulting instruction.

The MMU divides storage into pages. The page represents the granularity of address translation, access control, and storage attribute control. PowerPC Book-E architecture defines sixteen page sizes, of which the PPC440GP MMU supports eight. These eight page sizes (1KB, 4KB, 16KB, 64KB, 256KB, 1MB, 16MB, and 256MB) are simultaneously supported. A valid entry for a page referenced by an effective address must be in the translation lookaside buffer (TLB) in order for the address to be accessed. An attempt to access an address for which no TLB entry exists causes an Instruction or Data TLB Error interrupt, depending on the type of access (instruction or data). See ~~Chapter 10, "Interrupts and Exceptions"~~ *Interrupts and Exceptions* on page 337 for more information on these and other interrupt types.

6.1.1 Support for PowerPC Book-E MMU Architecture

The Book-E Enhanced PowerPC Architecture defines specific requirements for MMU implementations, but also leaves the details of several features implementation-dependent. The PPC440GP is fully compliant with the required MMU mechanisms defined by PowerPC Book-E, but a few optional mechanisms are not supported. These are:

- Memory coherence required (M) storage attribute

  Because the PPC440GP does not provide hardware support for multiprocessor coherence, the memory coherence required storage attribute has no effect. If a TLB entry is created with M=1, then any memory transactions for the page associated with that TLB entry will be indicated as being memory coherence

required via a corresponding transfer attribute interface signal, but the setting will have no effect on the operation within the PPC440GP.

- TLB Invalidate virtual address (**tlbiva**)instruction

  The **tlbiva** instruction is used to support the invalidation of TLB entries in a multiprocessor environment with hardware-enforced coherency, which is not supported by the PPC440GP. Consequently, the attempted execution of this instruction will cause an Illegal Instruction exception type Program interrupt. The **tlbwe** instruction may be used to invalidate TLB entries in a uniprocessor environment.

- TLB Synchronize (**tlbsync**) instruction

  The **tlbsync** instruction is used to synchronize software TLB management operations in a multiprocessor environment with hardware-enforced coherency, which is not supported by the PPC440GP. Consequently, this instruction is treated as a no-op.

- Page Sizes

  PowerPC Book-E defines sixteen different page sizes, but does not require that an implementation support all of them. Furthermore, some of the page sizes are only applicable to 64-bit implementations, as they are larger than a 32-bit effective address space can support (4GB). Accordingly, the PPC440GP supports eight of the sixteen page sizes, from 1KB up to 256MB, as mentioned above and as listed in ~~Table 6-2, "Page Size and Effective Address to EPN Comparison," on page 6-9~~*Table 6-2 Page Size and Effective Address to EPN Comparison* on page 240.

- Address Space

  Since the PPC440GP is a 32-bit implementation of the 64-bit PowerPC Book-E architecture, there are differences in the sizes of some of the TLB fields. First, the Effective Page Number (EPN) field varies from 4 to 22 bits, depending on page size. Second, the page number portion of the real address is made up of a concatenation of two TLB fields, rather than a single Real Page Number (RPN) field as described in PowerPC Book-E. These fields are the RPN field (which can vary from 4 to 22 bits, depending on page size), and the Extended Real Page Number (ERPN) field, which is 4 bits, for a total of 36 bits of real address, when combined with the page offset portion of the real address. See ~~"Address Translation" on page 6-9~~*Address Translation* on page 240 for a more detailed explanation of these fields and the formation of the real address.

6.2 Translation Lookaside Buffer

The Translation Lookaside Buffer (TLB) is the hardware resource that controls translation, protection, and storage attributes. A single unified 64-entry, fully-associative TLB is used for both instruction and data accesses. In addition, the PPC440GP implements two separate, smaller "shadow" TLB arrays, one for instruction fetch accesses and one for data accesses. These shadow TLBs improve performance by lowering the latency for address translation, and by reducing contention for the main unified TLB between instruction fetching and data storage accesses. See ~~"Shadow TLB Arrays" on page 6-20~~*Shadow TLB Arrays* on page 251 for additional information on the operation of the shadow TLB arrays.

Maintenance of TLB entries is under software control. System software determines the TLB entry replacement strategy and the format and use of any page table information. A TLB entry contains all of the information required to identify the page, to specify the address translation, to control the access permissions, and to designate the storage attributes.

A TLB entry is written by copying information from a GPR and the ~~Process Identification (PID) register~~MMUCR[STID] field, using a series of three **tlbwe** instructions. A TLB entry is read by copying the information ~~to~~ into a GPR and the ~~PID~~MMUCR[STID] field, using a series of three **tlbre** instructions. Software can also search for specific TLB entries using the **tlbsx[.]** instruction. See ~~"TLB Management Instructions" on page 6-21~~*TLB Management Instructions* on page 252 for more information on these instructions.

Each TLB entry identifies a page and defines its translation, access controls, and storage attributes. Accordingly, fields in the TLB entry fall into four categories:

- Page identification fields (information required to identify the page to the hardware translation mechanism).
- Address translation fields
- Access control fields
- Storage attribute fields

~~Table 6-1~~*Table 6-1* summarizes the TLB entry fields for each of the categories.

Table 0-1. TLB Entry Fields

TLB Word	Bit	Field	Description
		Page Identification Fields	
0	0:21	EPN	**Effective Page Number** (variable size, from 4 - 22 bits) Bits 0:n–1 of the EPN field are compared to bits 0:n–1 of the effective address (EA) of the storage access (where n = 32–$\log_2$(page size in bytes) and page size is specified by the SIZE field of the TLB entry). See Table 6-2 on page 6-9.
0	22	V	**Valid** (1 bit) This bit indicates that this TLB entry is valid and may be used for translation. The Valid bit for a given entry can be set or cleared with a **tlbwe** instruction.
0	23	TS	Translation Address Space (1 bit) This bit indicates the address space this TLB entry is associated with. For instruction storage accesses, MSR[IS] must match the value of TS in the TLB entry for that TLB entry to provide the translation. Likewise, for data storage accesses, MSR[DS] must match the value of TS in the TLB entry. For the **tlbsx[.]** instruction, the MMUCR[STS] field must match the value of TS.
0	24:27	SIZE	**Page Size** (4 bits) The SIZE field specifies the size of the page associated with the TLB entry as 4^{SIZE}KB, where SIZE $\in$ {0, 1, 2, 3, 4, 5, 7, 9}. See Table 6-2 on page 6-9.
0	32:39	TID	**Translation ID** (8 bits) Field used to identify a globally shared page (TID=0) or the process ID of the owner of a private page (TID<>0). See "Page Identification" on page 6-6.

Table 0-1. TLB Entry Fields (continued)

TLB Word	Bit	Field	Description
		Address Translation Fields	
1	0:21	RPN	**Real Page Number** (variable size, from 4 - 22 bits) Bits 0:n–1 of the RPN field are used to replace bits 0:n–1 of the effective address to produce a portion of the real address for the storage access (where n = 32–$\log_2$(page size in bytes) and page size is specified by the SIZE field of the TLB entry). Software must set unused low-order RPN bits (that is, bits n:21) to 0. See "Address Translation" on page 6-9 and Table 6-3 on page 6-10.
1	28:31	ERPN	**Extended Real Page Number** (4 bits) The 4-bit ERPN field are prepended to the rest of the translated address to form a total of a 36-bit (64 GB) real address. See "Address Translation" on page 6-9 and Table 6-3 on page 6-10.
		Storage Attribute Fields	
2	16	U0	**User-Definable Storage Attribute 0** (1 bit) See "User-Definable (U0–U3)" on page 6-15. Specifies the U0 storage attribute for the page associated with the TLB entry. The function of this storage attribute is system-dependent, and has no effect within the PPC440GP.
2	17	U1	**User-Definable Storage Attribute 1** (1 bit) See "User-Definable (U0–U3)" on page 6-15. Specifies the U1 storage attribute for the page associated with the TLB entry. The function of this storage attribute is system-dependent, but the PPC440GP can be programmed to use this attribute to designate a memory page as containing *transient* data and/or instructions (see Chapter 5, "Instruction and Data Caches").
2	18	U2	**User-Definable Storage Attribute 2** (1 bit) See "User-Definable (U0–U3)" on page 6-15. Specifies the U2 storage attribute for the page associated with the TLB entry. The function of this storage attribute is system-dependent, but the PPC440GP can be programmed to use this attribute to specify whether or not stores that miss in the data cache should allocate the line in the data cache (see Chapter 5, "Instruction and Data Caches").
2	19	U3	**User-Definable Storage Attribute 3** (1 bit) See "User-Definable (U0–U3)" on page 6-15. Specifies the U3 storage attribute for the page associated with the TLB entry. The function of this storage attribute is system-dependent, and has no effect within the PPC440GP.
2	20	W	**Write-Through** (1 bit) See "Write-Through (W)" on page 6-14.
			0 — The page is not write-through (that is, the page is copy-back).
			1 — The page is write-through.
2	21	I	**Caching Inhibited** (1 bit) See "Caching Inhibited (I)" on page 6-14.
			0 — The page is not caching inhibited (that is, the page is cacheable).
			1 — The page is caching inhibited.

Table 0-1. TLB Entry Fields (continued)

TLB Word	Bit	Field	Description	
2	22	M	**Memory Coherence Required** (1 bit) See "Memory Coherence Required (M)" on page 6-14.	
			0	The page is not memory coherence required.
			1	The page is memory coherence required.
			Note that the PPC440GP does not support multiprocessing, and thus all storage accesses will behave as if M=0. Setting M=1 in a TLB entry has no effect other than to cause any system interface transactions to the corresponding page to be indicated as memory coherence required via the "transfer attributes" interface signals.	
2	23	G	**Guarded** (1 bit) See "Guarded (G)" on page 6-15.	
			0	The page is not guarded.
			1	The page is guarded.
2	24	E	**Endian** (1 bit) See "Endian (E)" on page 6-15.	
			0	All accesses to the page are performed with big-endian byte ordering, which means that the byte at the effective address is considered the most-significant byte of a multi-byte scalar (see "Byte Ordering" on page 4-5).
			1	All accesses to the page are performed with little-endian byte ordering, which means that the byte at the effective address is considered the least-significant byte of a multi-byte scalar (see "Byte Ordering" on page 4-5).
		Access Control Fields		
2	26	UX	**User State Execute Enable** (1 bit) See "Execute Access" on page 6-11.	
			0	Instruction fetch is not permitted from this page while MSR[PR]=1 and the attempt to execute an instruction from this page while MSR[PR] =1 will cause an Execute Access Control exception type Instruction Storage interrupt.
			1	Instruction fetch and execution is permitted from this page while MSR[PR]=1.
2	27	UW	**User State Write Enable** (1 bit) See "Write Access" on page 6-11.	
			0	Store operations and the **dcbz** instruction are not permitted to this page when MSR[PR]=1 and will cause a Write Access Control exception type Data Storage interrupt.
			1	Store operations and the **dcbz** instruction are permitted to this page when MSR[PR]=1.

Table 0-1. TLB Entry Fields (continued)

TLB Word	Bit	Field	Description	
2	28	UR	**User State Read Enable** (1 bit) See "Read Access" on page 6-12.	
			0	Load operations and the **dcbt**, **dcbtst**, **dcbst**, **dcbf**, **icbt**, and **icbi** instructions are not permitted from this page when MSR[PR]=1 and will cause a Read Access Control exception. Except for the **dcbt**, **dcbtst**, and **icbt** instructions, a Data Storage interrupt will occur (see Table 6-4 on page 6-13).
			1	Load operations and the **dcbt**, **dcbtst**, **dcbst**, **dcbf**, **icbt**, and **icbi** instructions are permitted from this page when MSR[PR]=1.
2	29	SX	**Supervisor State Execute Enable** (1 bit) See "Execute Access" on page 6-11.	
			0	Instruction fetch is not permitted from this page while MSR[PR]=0 and the attempt to execute an instruction from this page while MSR[PR] =0 will cause an Execute Access Control exception type Instruction Storage interrupt.
			1	Instruction fetch and execution is permitted from this page while MSR[PR]=0.
2	30	SW	**Supervisor State Write Enable** (1 bit) See "Write Access" on page 6-11.	
			0	Store operations and the **dcbz** and **dcbi** instructions are not permitted to this page when MSR[PR]=0 and will cause a Write Access Control exception type Data Storage interrupt.
			1	Store operations and the **dcbz** and **dcbi** instructions are permitted to this page when MSR[PR]=0.
2	31	SR	**Supervisor State Read Enable** (1 bit) See "Read Access" on page 6-12.	
			0	Load operations and the **dcbt**, **dcbtst**, **dcbst**, **dcbf**, **icbt**, and **icbi** instructions are not permitted from this page when MSR[PR]=0 and will cause a Read Access Control exception. Except for the **dcbt**, **dcbtst**, and **icbt** instructions, a Data Storage interrupt will occur (see Table 6-4 on page 6-13).
			1	Load operations and the **dcbt**, **dcbtst**, **dcbst**, **dcbf**, **icbt**, and **icbi** instructions are permitted from this page when MSR[PR]=0.

Table 6-1. TLB Entry Fields

TLB Word	Bit	Field	Description
		Page Identification Fields	
0	0:21	EPN	**Effective Page Number** (variable size, from 4 - 22 bits) Bits 0:n–1 of the EPN field are compared to bits 0:n–1 of the effective address (EA) of the storage access (where n = 32–$\log_2$(page size in bytes) and page size is specified by the SIZE field of the TLB entry). See *Table 6-2* on page 240.
0	22	V	**Valid** (1 bit) This bit indicates that this TLB entry is valid and may be used for translation. The Valid bit for a given entry can be set or cleared with a **tlbwe** instruction.

Table 6-1. TLB Entry Fields (continued)

TLB Word	Bit	Field	Description
0	23	TS	Translation Address Space (1 bit) This bit indicates the address space this TLB entry is associated with. For instruction fetch accesses, MSR[IS] must match the value of TS in the TLB entry for that TLB entry to provide the translation. Likewise, for data storage accesses (including instruction cache management operations), MSR[DS] must match the value of TS in the TLB entry. For the **tlbsx[.]** instruction, the MMUCR[STS] field must match the value of TS.
0	24:27	SIZE	**Page Size** (4 bits) The SIZE field specifies the size of the page associated with the TLB entry as 4^{SIZE}KB, where SIZE ∈ {0, 1, 2, 3, 4, 5, 7, 9}. See *Table 6-2* on page 240.
0	32:39	TID	**Translation ID** (8 bits) Field used to identify a globally shared page (TID=0) or the process ID of the owner of a private page (TID<>0). See *Page Identification* on page 237.
		Address Translation Fields	
1	0:21	RPN	**Real Page Number** (variable size, from 4 - 22 bits) Bits 0:n–1 of the RPN field are used to replace bits 0:n–1 of the effective address to produce a portion of the real address for the storage access (where n = 32–$\log_2$(page size in bytes) and page size is specified by the SIZE field of the TLB entry). Software must set unused low-order RPN bits (that is, bits n:21) to 0. See *Address Translation* on page 240 and *Table 6-3* on page 241.
1	28:31	ERPN	**Extended Real Page Number** (4 bits) The 4-bit ERPN field are prepended to the rest of the translated address to form a total of a 36-bit (64 GB) real address. See *Address Translation* on page 240 and *Table 6-3* on page 241.
		Storage Attribute Fields	
2	16	U0	**User-Definable Storage Attribute 0** (1 bit) See *User-Definable (U0–U3)* on page 246. Specifies the U0 storage attribute for the page associated with the TLB entry. The function of this storage attribute is system-dependent, and has no effect within the PPC440GP.
2	17	U1	**User-Definable Storage Attribute 1** (1 bit) See *User-Definable (U0–U3)* on page 246. Specifies the U1 storage attribute for the page associated with the TLB entry. The function of this storage attribute is system-dependent, but the PPC440GP can be programmed to use this attribute to designate a memory page as containing *transient* data and/or instructions (see *Instruction and Data Caches* on page 205).
2	18	U2	**User-Definable Storage Attribute 2** (1 bit) See *User-Definable (U0–U3)* on page 246. Specifies the U2 storage attribute for the page associated with the TLB entry. The function of this storage attribute is system-dependent, but the PPC440GP can be programmed to use this attribute to specify whether or not stores that miss in the data cache should allocate the line in the data cache (see *Instruction and Data Caches* on page 205).
2	19	U3	**User-Definable Storage Attribute 3** (1 bit) See *User-Definable (U0–U3)* on page 246. Specifies the U3 storage attribute for the page associated with the TLB entry. The function of this storage attribute is system-dependent, and has no effect within the PPC440GP.
2	20	W	**Write-Through** (1 bit) See *Write-Through (W)* on page 245. 0 — The page is not write-through (that is, the page is copy-back). 1 — The page is write-through.
2	21	I	**Caching Inhibited** (1 bit) See *Caching Inhibited (I)* on page 245. 0 — The page is not caching inhibited (that is, the page is cacheable). 1 — The page is caching inhibited.

Table 6-1. TLB Entry Fields (continued)

TLB Word	Bit	Field	Description	
2	22	M	**Memory Coherence Required** (1 bit) See *Memory Coherence Required (M)* on page 245.	
			0	The page is not memory coherence required.
			1	The page is memory coherence required.
			Note that the PPC440GP does not support multiprocessing, and thus all storage accesses will behave as if M=0. Setting M=1 in a TLB entry has no effect other than to cause any system interface transactions to the corresponding page to be indicated as memory coherence required via the "transfer attributes" interface signals.	
2	23	G	**Guarded** (1 bit) See *Guarded (G)* on page 246.	
			0	The page is not guarded.
			1	The page is guarded.
2	24	E	**Endian** (1 bit) See *Endian (E)* on page 246.	
			0	All accesses to the page are performed with big-endian byte ordering, which means that the byte at the effective address is considered the most-significant byte of a multi-byte scalar (see *Byte Ordering* on page 149).
			1	All accesses to the page are performed with little-endian byte ordering, which means that the byte at the effective address is considered the least-significant byte of a multi-byte scalar (see *Byte Ordering* on page 149).
		Access Control Fields		
2	26	UX	**User State Execute Enable** (1 bit) See *Execute Access* on page 242.	
			0	Instruction fetch is not permitted from this page while MSR[PR]=1 and the attempt to execute an instruction from this page while MSR[PR] =1 will cause an Execute Access Control exception type Instruction Storage interrupt.
			1	Instruction fetch and execution is permitted from this page while MSR[PR]=1.
2	27	UW	**User State Write Enable** (1 bit) See *Write Access* on page 242.	
			0	Store operations and the **dcbz** instruction are not permitted to this page when MSR[PR]=1 and will cause a Write Access Control exception type Data Storage interrupt.
			1	Store operations and the **dcbz** instruction are permitted to this page when MSR[PR]=1.
2	28	UR	**User State Read Enable** (1 bit) See *Read Access* on page 243.	
			0	Load operations and the **dcbt**, **dcbtst**, **dcbst**, **dcbf**, **icbt**, and **icbi** instructions are not permitted from this page when MSR[PR]=1 and will cause a Read Access Control exception. Except for the **dcbt**, **dcbtst**, and **icbt** instructions, a Data Storage interrupt will occur (see *Table 6-4* on page 244).
			1	Load operations and the **dcbt**, **dcbtst**, **dcbst**, **dcbf**, **icbt**, and **icbi** instructions are permitted from this page when MSR[PR]=1.
2	29	SX	**Supervisor State Execute Enable** (1 bit) See *Execute Access* on page 242.	
			0	Instruction fetch is not permitted from this page while MSR[PR]=0 and the attempt to execute an instruction from this page while MSR[PR] =0 will cause an Execute Access Control exception type Instruction Storage interrupt.
			1	Instruction fetch and execution is permitted from this page while MSR[PR]=0.
2	30	SW	**Supervisor State Write Enable** (1 bit) See *Write Access* on page 242.	
			0	Store operations and the **dcbz** and **dcbi** instructions are not permitted to this page when MSR[PR]=0 and will cause a Write Access Control exception type Data Storage interrupt.
			1	Store operations and the **dcbz** and **dcbi** instructions are permitted to this page when MSR[PR]=0.

Table 6-1. TLB Entry Fields (continued)

TLB Word	Bit	Field	Description	
2	31	SR	**Supervisor State Read Enable** (1 bit) See *Read Access* on page 243.	
			0	Load operations and the **dcbt**, **dcbtst**, **dcbst**, **dcbf**, **icbt**, and **icbi** instructions are not permitted from this page when MSR[PR]=0 and will cause a Read Access Control exception. Except for the **dcbt**, **dcbtst**, and **icbt** instructions, a Data Storage interrupt will occur (see *Table 6-4* on page 244).
			1	Load operations and the **dcbt**, **dcbtst**, **dcbst**, **dcbf**, **icbt**, and **icbi** instructions are permitted from this page when MSR[PR]=0.

6.3 Page Identification

The Valid (V), Effective Page Number (EPN), Translation Space Identifier (TS), Page Size (SIZE), and Translation ID (TID) fields of a particular TLB entry identify the page associated with that TLB entry. Except as noted, all comparisons must succeed to validate this entry for subsequent translation and access control processing. Failure to locate a matching TLB entry based on this criteria for instruction fetches will result in a TLB Miss exception type Instruction TLB Error interrupt. Failure to locate a matching TLB entry based on this criteria for data storage accesses will result in a TLB Miss exception which may result in a Data TLB Error interrupt, depending on the type of data storage access (certain cache management instructions do not result in an interrupt if they cause an exception; they simply no-op).

6.3.1 Virtual ~~Address~~Address Formation

The first step in page identification is the expansion of the effective address into a virtual address. Again, the effective address is the 32-bit address calculated by a load, store, or cache management instruction, or as part of an instruction fetch. The virtual address is formed by prepending the effective address with a 1-bit address space identifier and an 8-bit process identifier. The process identifier is contained in the Process ID (PID) register. The address space identifier is provided by MSR[IS] for instruction fetches, and by MSR[DS] for data ~~accesses~~storage accesses and cache management operations, including instruction cache management operations. The resulting 41-bit value forms the virtual address, which is then compared to the virtual addresses contained in the TLB entries.

Note that the **tlbsx**[**.**] instruction also forms a virtual address, for software controlled search of the TLB. This instruction calculates the effective address in the same manner as a data access instruction, but the process identifier and address space identifier are provided by fields in the MMUCR, rather than by the PID and MSR, respectively (see ~~"TLB Search Instruction (tlbsx[.])" on page 6-22~~*TLB Search Instruction (tlbsx[.])* on page 253).

6.3.2 Address Space ~~Identifier~~Identifier Convention

The address space identifier differentiates between two distinct virtual address spaces, one generally associated with interrupt-handling and other system-level code and/or data, and the other generally associated with application-level code and/or data.

Typically, user mode programs will run with MSR[IS,DS] both set to 1, allowing access to application-level code and data memory pages. Then, on an interrupt, MSR[IS,DS] are both automatically cleared to 0, so that the interrupt handler code and data areas may be accessed using system-level TLB entries (that is, TLB entries with the TS field = 0). It is also possible that an operating system could set up certain system-level

code and data areas (and corresponding TLB entries with the TS field = 1) in the application-level address space, allowing user mode programs running with MSR[IS,DS] set to 1 to access them (system library routines, for example, which may be shared by multiple user mode and/or supervisor mode programs). System-level code wishing to use these areas would have to first set the corresponding MSR[IS,DS] field in order to use the application-level TLB entries, or there would have to be alternative system-level TLB entries set up.

The net of this is that the notion of application-level code running with MSR[IS,DS] set to 1 and using corresponding TLB entries with the TS=1, and conversely system-level code running with MSR[IS,DS] set to 0 and using corresponding TLB entries with TS=0, is by convention. It is possible to run in user mode with MSR[IS,DS] set to 0, and conversely to run in supervisor mode with MSR[IS,DS] set to 1, with the corresponding TLB entries being used. The only fixed requirement in this regard is the fact that MSR[IS,DS] are cleared on an interrupt, and thus there *must* be a TLB entry for the system-level interrupt handler code with TS=0 in order to be able to fetch and execute the interrupt handler itself. Whether or not other system-level code switches MSR[IS,DS] and creates corresponding system-level TLB entries ~~is entirely~~ depends upon the operating system ~~dependent~~environment.

Programming Note: Software must ensure that there is always a valid TLB entry with TS=0 and with supervisor mode execute access permission (SX=1) corresponding to the effective address of the interrupt handlers. Otherwise, an Instruction TLB Error interrupt could result upon the fetch of the interrupt handler for some other interrupt type, and the registers holding the state of the routine which was executing at the time of the original interrupt (SRR0/SRR1) could be corrupted. See ~~Chapter 10, "Interrupts and Exceptions"~~ *Interrupts and Exceptions* on page 337 for more information.

6.3.3 TLB Match Process

This virtual address is used to locate the associated entry in the TLB. The address space identifier, the process identifier, and a portion of the effective address of the storage access are compared to the TS, TID, and EPN fields, respectively, of each TLB entry.

The virtual address matches a TLB entry if:

- The valid (V) field of the TLB entry is 1, and
- The value of the address ~~specifier~~ space identifier is equal to the value of the TS field of the TLB entry, and
- Either the value of the process identifier is equal to the value of the TID field of the TLB entry (private page), or the value of the TID field is 0 (globally shared page), and
- The value of bits 0:n–1 of the effective address is equal to the value of bits 0:n-1 of the EPN field of the TLB entry (where n = 32–$\log_2$ (page size in bytes) and page size is specified by the value of the SIZE field of the TLB entry). See ~~Table 6-2, "Page Size and Effective Address to EPN Comparison," on page 6-9~~ *Table 6-2 Page Size and Effective Address to EPN Comparison* on page 240.

A TLB Miss exception occurs if there is no matching entry in the TLB for the page specified by the virtual address (except for the **tlbsx[.]** instruction, which simply returns an undefined value to the GPR file and (for **tlbsx.**) sets $CR[CR0]_2$ to 0). See ~~"TLB Search Instruction (tlbsx[.])" on page 6-22~~ *TLB Search Instruction (tlbsx[.])* on page 253.

Programming Note: Although it is possible for software to create multiple TLB entries that match the same virtual address, doing so is a programming error and the results are undefined.

~~Figure 6-1~~Figure 6-1 illustrates the criteria for a virtual address to match a specific TLB entry, while ~~Table 6-2~~*Table 6-2* defines the page sizes associated with each SIZE field value, and the associated comparison of the effective address to the EPN field~~.~~.

Figure 6-1. Virtual Address to TLB Entry Match Process

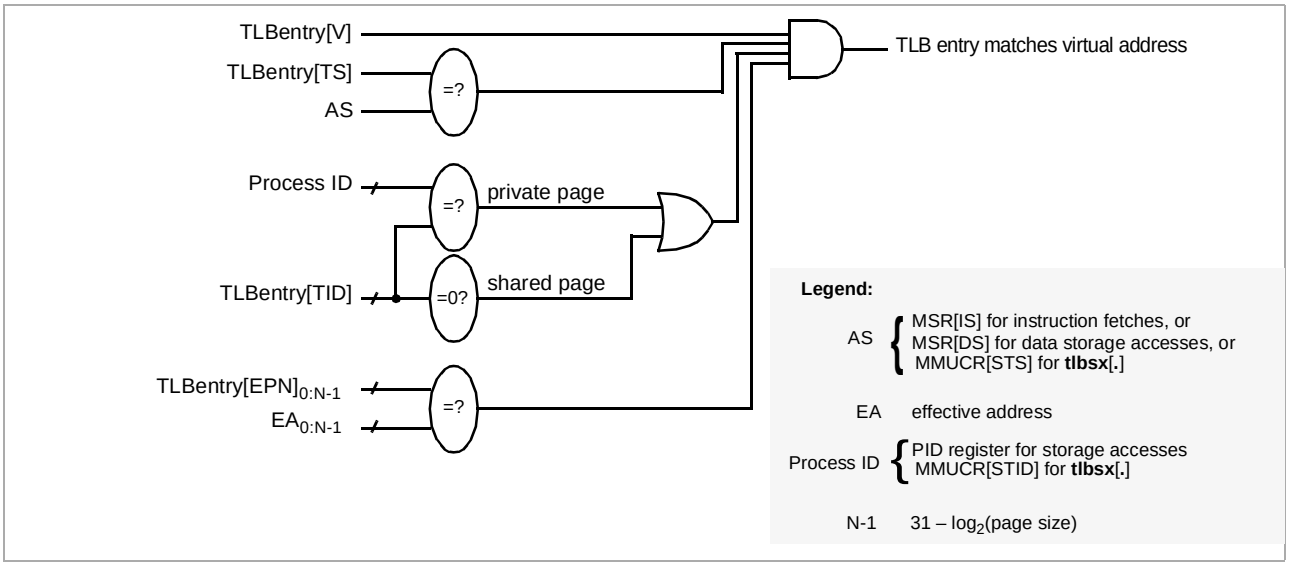


Figure 0-1. Virtual Address to TLB Entry Match Process

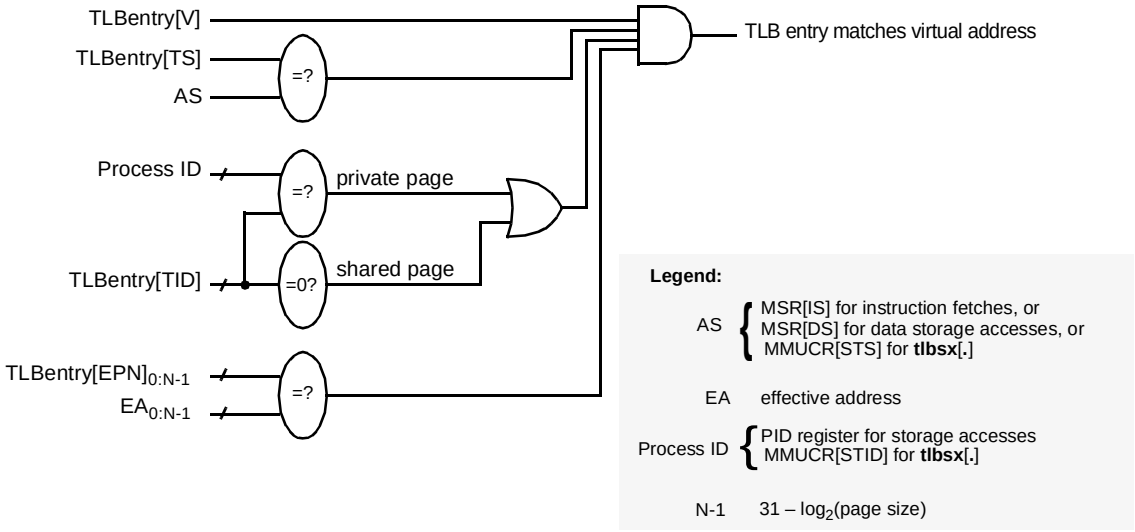


Table 0-2. Page Size and Effective Address to EPN Comparison

SIZE	Page Size	EA to EPN Comparison
0b0000	1KB	$EPN_{0:21}$ =? $EA_{0:21}$
0b0001	4KB	$EPN_{0:19}$ =? $EA_{0:19}$
0b0010	16KB	$EPN_{0:17}$ =? $EA_{0:17}$
0b0011	64KB	$EPN_{0:15}$ =? $EA_{0:15}$
0b0100	256KB	$EPN_{0:13}$ =? $EA_{0:13}$
0b0101	1MB	$EPN_{0:11}$ =? $EA_{0:11}$
0b0110	not supported	not supported
0b0111	16MB	$EPN_{0:7}$ =? $EA_{0:7}$
0b1000	not supported	not supported
0b1001	256MB	$EPN_{0:3}$ =? $EA_{0:3}$
0b1010	not supported	not supported
0b1011	not supported	not supported
0b1100	not supported	not supported
0b1101	not supported	not supported
0b1110	not supported	not supported
0b1111	not supported	not supported

Table 6-2. Page Size and Effective Address to EPN Comparison

SIZE	Page Size	EA to EPN Comparison
0b0000	1KB	$EPN_{0:21}$ =? $EA_{0:21}$
0b0001	4KB	$EPN_{0:19}$ =? $EA_{0:19}$
0b0010	16KB	$EPN_{0:17}$ =? $EA_{0:17}$
0b0011	64KB	$EPN_{0:15}$ =? $EA_{0:15}$
0b0100	256KB	$EPN_{0:13}$ =? $EA_{0:13}$
0b0101	1MB	$EPN_{0:11}$ =? $EA_{0:11}$
0b0110	not supported	not supported
0b0111	16MB	$EPN_{0:7}$ =? $EA_{0:7}$
0b1000	not supported	not supported
0b1001	256MB	$EPN_{0:3}$ =? $EA_{0:3}$
0b1010	not supported	not supported
0b1011	not supported	not supported
0b1100	not supported	not supported
0b1101	not supported	not supported
0b1110	not supported	not supported
0b1111	not supported	not supported

6.4 Address Translation

Once a TLB entry is found which matches the virtual address associated with a given storage access, as described in ~~"Page Identification" on page 6-6~~*Page Identification* on page 237, the virtual address is translated to a real address according to the procedures described in this section.

The Real Page Number (RPN) and Extended Real Page Number (ERPN) fields of the matching TLB entry provide the page number portion of the real address. Let *n*=32–$\log_2$(*page size* in bytes) where *page size* is specified by the SIZE field of the matching TLB entry. Bits *n*:31 of the effective address (the "page offset") are appended to bits 0:*n*–1 of the RPN field, and bits 0:3 of the ERPN field are prepended to this value to produce the 36-bit real address (that is, RA = $ERPN_{0:3}$ || $RPN_{0:n-1}$ || $EA_{n:31}$).

~~Figure 6-2~~Figure 6-2 illustrates the address translation process, while ~~Table 6-3~~*Table 6-3* defines the relationship between the different page sizes and the real address formation~~.~~.

Figure 6-2. Effective-to-Real Address Translation Flow

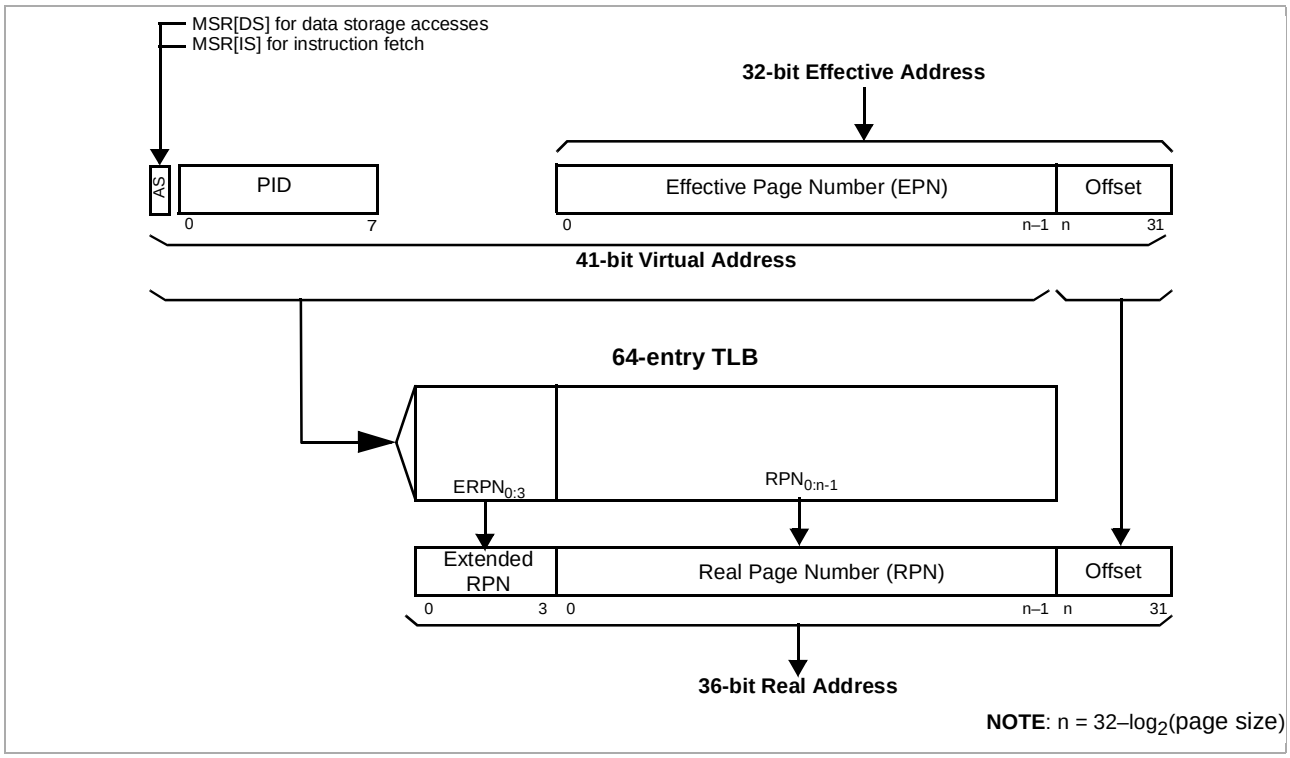
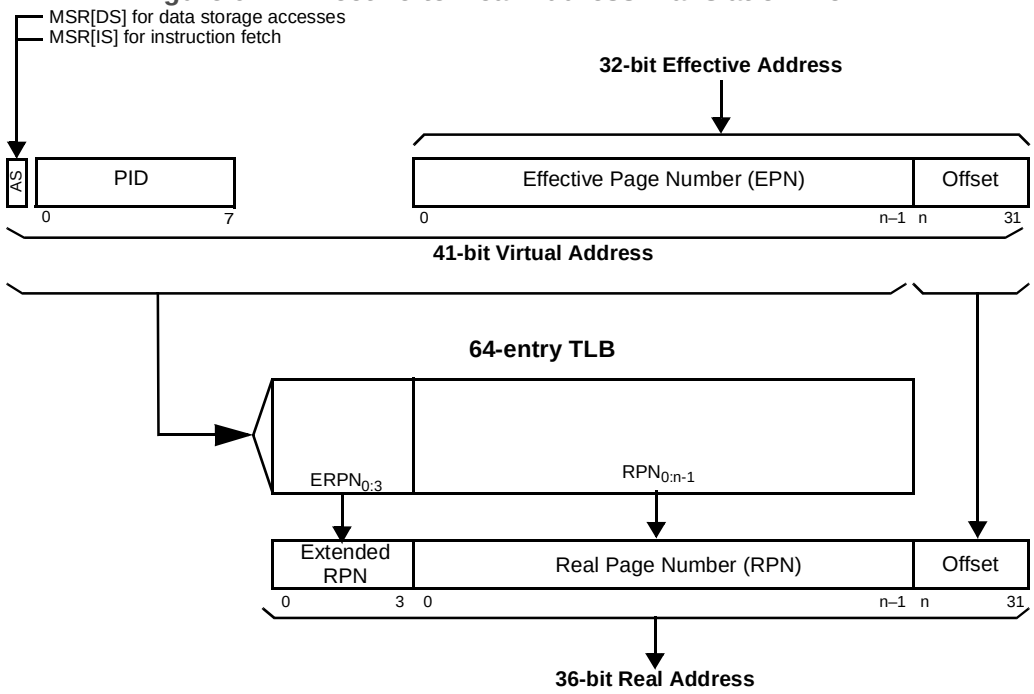


Figure 0-2. Effective-to-Real Address Translation Flow



NOTE: n = 32–$\log_2$(page size)

Table 0-3. Page Size and Real Address Formation

SIZE	Page Size	RPN bits required to be 0	Real Address
0b0000	1KB	none	$RPN_{0:21}$ || $EA_{22:31}$
0b0001	4KB	$RPN_{20:21}$=0	$RPN_{0:19}$ || $EA_{20:31}$
0b0010	16KB	$RPN_{18:21}$=0	$RPN_{0:17}$ || $EA_{18:31}$
0b0011	64KB	$RPN_{16:21}$=0	$RPN_{0:15}$ || $EA_{16:31}$
0b0100	256KB	$RPN_{14:21}$=0	$RPN_{0:13}$ || $EA_{14:31}$
0b0101	1MB	$RPN_{12:21}$=0	$RPN_{0:11}$ || $EA_{12:31}$
0b0110	not supported	not supported	not supported
0b0111	16MB	$RPN_{8:21}$=0	$RPN_{0:7}$ || $EA_{8:31}$
0b1000	not supported	not supported	not supported
0b1001	256MB	$RPN_{4:21}$=0	$RPN_{0:3}$ || $EA_{4:31}$
0b1010	not supported	not supported	not supported
0b1011	not supported	not supported	not supported
0b1100	not supported	not supported	not supported
0b1101	not supported	not supported	not supported
0b1110	not supported	not supported	not supported
0b1111	not supported	not supported	not supported

Table 6-3. Page Size and Real Address Formation

SIZE	Page Size	RPN bits required to be 0	Real Address
0b0000	1KB	none	$RPN_{0:21}$ || $EA_{22:31}$
0b0001	4KB	$RPN_{20:21}$=0	$RPN_{0:19}$ || $EA_{20:31}$
0b0010	16KB	$RPN_{18:21}$=0	$RPN_{0:17}$ || $EA_{18:31}$
0b0011	64KB	$RPN_{16:21}$=0	$RPN_{0:15}$ || $EA_{16:31}$
0b0100	256KB	$RPN_{14:21}$=0	$RPN_{0:13}$ || $EA_{14:31}$
0b0101	1MB	$RPN_{12:21}$=0	$RPN_{0:11}$ || $EA_{12:31}$
0b0110	not supported	not supported	not supported
0b0111	16MB	$RPN_{8:21}$=0	$RPN_{0:7}$ || $EA_{8:31}$
0b1000	not supported	not supported	not supported
0b1001	256MB	$RPN_{4:21}$=0	$RPN_{0:3}$ || $EA_{4:31}$
0b1010	not supported	not supported	not supported
0b1011	not supported	not supported	not supported
0b1100	not supported	not supported	not supported
0b1101	not supported	not supported	not supported
0b1110	not supported	not supported	not supported
0b1111	not supported	not supported	not supported

6.5 Access Control

Once a matching TLB entry has been identified and the address has been translated, the access control mechanism determines whether the program has execute, read, and/or write access to the page referenced by the address, as described in the following sections.

6.5.1 Execute Access

The UX or SX bit of a TLB entry controls *execute* access to a page of storage, depending on the operating mode (user or supervisor) of the processor.

- User mode (MSR[PR] = 1)

Instructions may be fetched and executed from a page in storage while in user mode if the UX access control bit for that page is equal to 1. If the UX access control bit is equal to 0, then instructions from that page will not be fetched, and will not be placed into the instruction cache as the result of a fetch request to that page while in user mode.

Furthermore, if the sequential execution model calls for the execution in user mode of an instruction from a page that is not enabled for execution in user mode (that is, UX=0 when MSR[PR]=1), an Execute Access Control exception type Instruction Storage interrupt is taken (See "Interrupts and Exceptions" on page -337 for more information).

- Supervisor mode (MSR[PR] = 0)

Instructions may be fetched and executed from a page in storage while in supervisor mode if the SX access control bit for that page is equal to 1. If the SX access control bit is equal to 0, then instructions from that page will not be fetched, and will not be placed into any cache as the result of a fetch request to that page while in supervisor mode.

Furthermore, if the sequential execution model calls for the execution in supervisor mode of an instruction from a page that is not enabled for execution in supervisor mode (that is, SX=0 when MSR[PR]=0), an Execute Access Control exception type Instruction Storage interrupt is taken (See "Interrupts and Exceptions" on page -337 for more information).

6.5.2 Write Access

The UW or SW bit of aTLB entry controls *write* access to a page, depending on the operating mode (user or supervisor) of the processor.

- User mode (MSR[PR] = 1)

  Store operations (including the store-class cache management instruction **dcbz**) are permitted to a page in storage while in user mode if the UW access control bit for that page is equal to 1. If execution of a store operation is attempted in user mode to a page for which the UW access control bit is 0, then a Write Access Control exception occurs. If the instruction is an **stswx** with string length 0, then no interrupt is taken and no operation is performed (see "Access Control Applied to Cache Management Instructions" on page -243). For all other store operations, execution of the instruction is suppressed and a Data Storage interrupt is taken.

  Note that although the **dcbi** cache management instruction is a store-class instruction, its execution is privileged and thus will not cause a Data Storage interrupt if execution of it is attempted in user mode (a Privileged Instruction exception type Program interrupt will occur instead).

- Supervisor mode (MSR[PR] = 0)

  Store operations (including the store-class cache management instructions **dcbz** and **dcbi**) are permitted to a page in storage while in supervisor mode if the SW access control bit for that page is equal to 1. If execution of a store operation is attemped in supervisor mode to a page for which the SW access control bit is 0, then a Write Access Control exception occurs. If the instruction is an **stswx** with string length 0, then no interrupt is taken and no operation is performed (see ~~"Access Control Applied to Cache Management Instructions" on page 6-12~~*Access Control Applied to Cache Management Instructions* on page 243). For all other store operations, execution of the instruction is suppressed and a Data Storage interrupt is taken.

6.5.3 Read Access

The UR or SR bit of a TLB entry controls *read* access to a page, depending on the operating mode (user or supervisor) of the processor.

- User mode (MSR[PR] = 1)

  Load operations (including the load-class cache management instructions **dcbst**, **dcbf**, **dcbt**, **dcbtst**, **icbi**, and **icbt**) are permitted from a page in storage while in user mode if the UR access control bit for that page is equal to 1. If execution of a load operation is attempted in user mode to a page for which the UR access control bit is 0, then a Read Access Control exception occurs. If the instruction is a load (not including **lswx** with string length 0) or is a **dcbst**, **dcbf**, or **icbi**, then execution of the instruction is suppressed and a Data Storage interrupt is taken. On the other hand, if the instruction is an **lswx** with string length 0, or is a **dcbt**, **dcbtst**, or **icbt**, then no interrupt is taken and no operation is performed (see ~~"Access Control Applied to Cache Management Instructions"~~*Access Control Applied to Cache Management Instructions* below).

- Supervisor mode (MSR[PR] = 0)

  Load operations (including the load-class cache management instructions **dcbst**, **dcbf**, **dcbt**, **dcbtst**, **icbi**, and **icbt**) are permitted from a page in storage while in supervisor mode if the SR access control bit for that page is equal to 1. If execution of a load operation is attempted in supervisor mode to a page for which the SR access control bit is 0, then a Read Access Control exception occurs. If the instruction is a load (not including **lswx** with string length 0) or is a **dcbst**, **dcbf**, or **icbi**, then execution of the instruction

is suppressed and a Data Storage interrupt is taken. On the other hand, if the instruction is an **lswx** with string length 0, or is a **dcbt**, **dcbtst**, or **icbt**, then no interrupt is taken and no operation is performed (see ~~"Access Control Applied to Cache Management Instructions"~~*Access Control Applied to Cache Management Instructions* below).

6.5.4 Access Control Applied to Cache Management Instructions

This section summarizes how each of the cache management instructions is affected by the access control mechanism.

- **dcbz** instructions are treated as *stores* with respect to access control since they actually change the data in a cache block. As such, they can cause Write Access Control exception type Data Storage interrupts.
- **dcbi** instructions are treated as *stores* with respect to access control since they can change the value of a storage location by invalidating the "current" copy of the location in the data cache, effectively "restoring" the value of the location to the "former" value which is contained in memory. As such, they can cause Write Access Control exception type Data Storage interrupts.
- **dcba** instructions are treated as no-ops by the PPC440GP under all circumstances, and thus can not cause any form of Data Storage interrupt.
- **icbi** instructions are treated as *loads* with respect to access control. As such, they can cause Read Access Control exception type Data Storage interrupts. Note that this instruction may cause a *Data* Storage interrupt (and not an *Instruction* Storage interrupt), even though it otherwise would perform its operation on the *instruction* cache. Instruction storage interrupts are associated with exceptions which occur upon the *fetch* of an instruction, whereas Data storage interrupts are associated with exceptions which occur upon the *execution* of a storage access or cache management instruction.
- **dcbt**, **dcbtst**, and **icbt** instructions are treated as *loads* with respect to access control. As such, they can cause Read Access Control exceptions. However, because these instructions are intended to act merely as "hints" that the specified cache block will likely be accessed by the processor in the near future, such exceptions will not result in a Data Storage interrupt. Instead, if a Read Access Control exception occurs, the instruction is treated as a no-op.
- **dcbf** and **dcbst** instructions are treated as *loads* with respect to access control. As such, they can cause Read Access Control exception type Data Storage interrupts. Flushing or storing a dirty line from the cache is not considered a store since an earlier store operation has already updated the cache line, and the **dcbf** or **dcbst** instruction is simply causing the results of that earlier store operation to be propagated to memory.
- **dccci** and **iccci** instructions do not even generate an address, nor are they affected by the access control mechanism. They are privileged instructions, and if executed in supervisor mode they will flash invalidate the entire associated cache.

~~Table 6-4~~*Table 6-4* summarizes the effect of access control on each of the cache management instructions.

Table 6-4. Access Control Applied to Cache Management Instructions

Instruction	Read Protection Violation Exception?	Write Protection Violation Exception?
dcba	No	No
dcbf	Yes	No
dcbi	No	Yes

Table 6-4. Access Control Applied to Cache Management Instructions

Instruction	Read Protection Violation Exception?	Write Protection Violation Exception?
dcbst	Yes	No
dcbt	Yes[1]	No
dcbtst	Yes[1]	No
dcbz	No	Yes
dccci	No	No
icbi	Yes	No
icbt	Yes[1]	No
iccci	No	No

1. **dcbt**, **dcbtst**, or **icbt** may cause a Read Access Control exception but will not result in a Data Storage interrupt

6.6 Storage Attributes

Each TLB entry specifies a number of storage attributes for the memory page with which it is associated. Storage attributes affect the manner in which storage accesses to a given page are performed. The storage attributes (and their corresponding TLB entry fields) are:

- Write-through (W)
- Caching inhibited (I)
- Memory coherence required (M)
- Guarded (G)
- Endianness (E)
- User-definable (U0, U1, U2, U3)

All combinations of these attributes are supported except combinations which simultaneously specify a region as write-through and caching inhibited.

6.6.1 Write-Through (W)

If a memory page is marked as write-through (W=1), then the data for all store operations to that page are written to memory, as opposed to only being written into the data cache. If the referenced line also exists in the data cache (that is, the store operation is a "hit"), then the data will also be written into the data cache, although the cache line will *not* be marked as having been modified (that is, the "dirty" bit(s) will not be set).

See ~~Chapter 5, "Instruction and Data Caches"~~*Instruction and Data Caches* on page 205 for more information on the handling of accesses to write-through storage.

6.6.2 Caching Inhibited (I)

If a memory page is marked as caching inhibited (I=1), then all load, store, and instruction fetch operations perform their access in memory, as opposed to in the respective cache. If I=0, then the page is cacheable and the operations may be performed in the cache.

It is a programming error for the target location of a load, store, **dcbz**, or fetch access to caching inhibited storage to be in the respective cache; the results of such an access are undefined. It is *not* a programming error for the target locations of the other cache management instructions to be in the cache when the caching inhibited storage attribute is set. The behavior of these instructions is defined for both I=0 and I=1 storage. See the instruction descriptions in ~~Chapter 28~~*Section 28 Instruction Set* on page 893 for more information.

See ~~Chapter 5, "Instruction and Data Caches"~~*Instruction and Data Caches* on page 205 for more information on the handling of accesses to caching inhibited storage.

6.6.3 Memory Coherence Required (M)

The memory coherence required (M) storage attribute is defined by the architecture to support cache and memory coherency within multiprocessor shared memory systems. Because the PPC440GP does not provide hardware support for multiprocessor coherence, the memory coherence required storage attribute has no effect. If a TLB entry is created with M = 1, any storage accesses to the page associated with that TLB entry are indicated, using the corresponding internal transfer attribute interface signal, as being memory coherence required, but the setting has no effect on the operation within the PPC440GP.

6.6.4 Guarded (G)

The guarded storage attribute is provided to control "speculative" access to "non-well-behaved" memory locations. Storage is said to be "well-behaved" if the corresponding real storage exists and is not defective, and if the effects of a single access to it are indistinguishable from the effects of multiple identical accesses to it. As such, data and instructions can be fetched out-of-order from well-behaved storage without causing undesired side effects.

In general, storage that is not well-behaved should be marked as guarded. Because such storage may represent a control register on an I/O device or may include locations that do not exist, an out-of-order access to such storage may cause an I/O device to perform unintended operations or may result in a Machine Check exception. For example, if the input buffer of a serial I/O device is memory-mapped, then an out-of-order or speculative access to that location could result in the loss of an item of data from the input buffer, if the instruction execution is interrupted and later re-attempted.

A data access to a guarded storage location is performed only if either the access is caused by an instruction that is known to be required by the sequential execution model, or the access is a load and the storage location is already in the data cache. Once a guarded data storage access is initiated, if the storage is also caching inhibited then only the bytes specifically requested are accessed in memory, according to the operand size for the instruction type. Data storage accesses to guarded storage which is marked as cacheable may access the entire cache block, either in the cache itself or in memory.

Instruction fetch is not affected by guarded storage. While the architecture does not prohibit instruction fetching from guarded storage, system software should generally prevent such instruction fetching by marking all guarded pages as "no-execute" (UX/SX = 0). Then, if an instruction fetch is attempted from such a page, the memory access will not occur and an Execute Access Control exception type Instruction Storage interrupt will result if and when execution is attempted for an instruction at any address within the page.

See ~~Chapter 5, "Instruction and Data Caches"~~*Section 5 Instruction and Data Caches* on page 205 for more information on the handling of accesses to guarded storage. Also see ~~"Partially Executed Instructions" on page 10-6~~*Partially Executed Instructions* on page 341 for information on the relationship between the guarded storage attribute and instruction restart and partially executed instructions.

6.6.5 Endian (E)

The endian (E) storage attribute controls the *byte ordering* with which load, store, and fetch operations are performed. Byte ordering refers to the order in which the individual bytes of a multiple-byte scalar operand are arranged in memory. The operands in a memory page with E=0 are arranged with *big-endian* byte ordering, which means that the bytes are arranged with the *most*-significant byte at the lowest-numbered memory address. The operands in a memory page with E=1 are arranged with *little-endian* byte ordering, which means that the bytes are arranged with the *least*-significant byte at the lowest-numbered address.

See ~~"Byte Ordering" on page 4-5~~*Byte Ordering* on page 149 for a more detailed explanation of big-endian and little-endian byte ordering.

6.6.6 User-Definable (U0–U3)

The PPC440GP provides four user-definable (U0–U3) storage attributes which can be used to control system-dependent behavior of the storage system. By default, these storage attributes do not have any effect on the operation of the PPC440GP, although all storage accesses indicate to the memory subsystem the values of U0–U3 using the corresponding transfer attribute interface signals. The specific system design may then take advantage of these attributes to control some system-level behaviors. As an example, one of the user-definable storage attributes could be used to enable code compession using the IBM CodePack core, if this function is included within a specific implementation incorporating the PPC440GP.

On the other hand, the PPC440GP can be programmed to make specific use of two of the four user-definable storage attributes. Specifically, by enabling the function using a control bit in the MMUCR (see ~~"Memory Management Unit Control Register (MMUCR)"~~*Memory Management Unit Control Register (MMUCR)* on page 247), the U1 storage attribute can be used to designate whether storage accesses to the associated memory page should use the "normal" or "transient" region of the respective cache. Similarly, another control bit in the MMUCR can be set to enable the U2 storage attribute to be used to control whether or not store accesses to the associated memory page which miss in the data cache should allocate the line in the cache. The U1 or U2 storage attributes do not affect PPC440GP operation unless they are enabled using the MMUCR to perform these specific functions. See ~~Chapter 5, "Instruction and Data Caches"~~*Instruction and Data Caches* on page 205 for more information on the mechanisms that can be controlled by the U1 and U2 storage attributes.

The U0 and U3 storage attributes have no such mechanism that enables them to control any specific function within the PPC440GP.

6.6.7 Supported Storage Attribute Combinations

Storage modes where both W = 1 and I = 1 (which would represent write-through but caching inhibited storage) are not supported. For all supported combinations of the W and I storage attributes, the G, E, and U0-U3 storage attributes may used in any combination.

6.7 Storage Control Registers

In addition to the two registers described below, the MSR[IS,DS] bits specify which of the two address spaces the respective instruction or data storage accesses are directed towards. Also, the MSR[PR] bit is used by the access control mechanism. See ~~"Machine State Register (MSR)" on page 10-7~~*Machine State Register (MSR)* on page 343 for more detailed information on the MSR and the function of each of its bits.

6.7.1 Memory Management Unit Control Register (MMUCR)

The MMUCR is written from a GPR using **mtspr**, and can be read into a GPR using **mfspr**. In addition, the MMUCR[STID] is updated with the TID field of the selected TLB entry when a **tlbre** instruction is executed. Conversely, the TID field of the selected TLB entry is updated with the value of the MMUCR[STID] field when a **tlbwe** instruction is executed. Other functions associated with the STID and other fields of the MMUCR are described in more detail in the sections that follow.

The following figure illustrates the MMUCR.

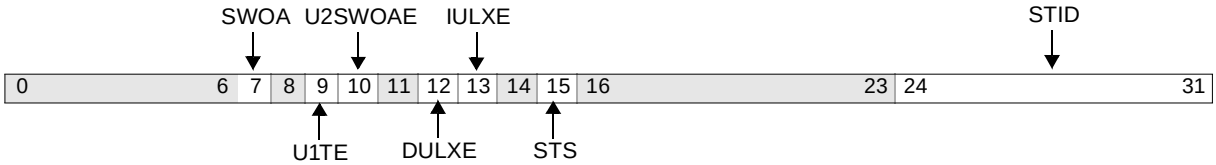


Figure 6-3. Memory Management Unit Control Register (MMUCR)

0:6		Reserved	
7	SWOA	Store Without Allocate 0 Cacheable store misses allocate a line in the data cache. 1 Cacheable store misses do not allocate a line in the data cache.	If MMUCR[U2SWOAE] = 1, this field is ignored.
8		Reserved	
9	U1TE	U1 Transient Enable 0 Disable U1 storage attribute as transient storage attribute. 1 Enable U1 storage attribute as transient storage attribute.	
10	U2SWOAE	U2 Store without Allocate Enable 0 Disable U2 storage attribute control of store without allocate. 1 Enable U2 storage attribute control of store without allocate.	If MMUCR[U2SWOAE] = 1, the U2 storage attribute overrides MMUCR[SWOA].
11		Reserved	
12	DULXE	Data Cache Unlock Exception Enable 0 Data cache unlock exception is disabled. 1 Data cache unlock exception is enabled.	**dcbf** in user mode will cause Cache Locking exception type Data Storage interrupt when MMUCR[DULXE] is 1.

13	IULXE	Instruction Cache Unlock Exception Enable 0 Instruction cache unlock exception is disabled. 1 Instruction cache unlock exception is enabled.	**icbi** in user mode will cause Cache Locking exception type Data Storage interrupt when MMUCR[IULXE] is 1.
14		Reserved	
15	STS	Search Translation Space	Specifies the value of the translation space (TS) field for the **tlbsx[.]** instruction
16:23		Reserved	
24:31	STID	Search Translation ID	Specifies the value of the process identifier to be compared against the TLB entry's TID field for the **tlbsx[.]** instruction; also used to transfer a TLB entry's TID value for the **tlbre** and **tlbwe** instructions.

Store Without Allocate (SWOA) Field

Performance for certain applications can be affected by the allocation of cache lines on store misses. If the store accesses for a particular application are distributed sparsely in memory, and if the data is typically not re-used after having been stored, then performance may be improved by avoiding the latency and bus bandwidth associated with filling the entire cache line containing the bytes being stored. On the other hand, if an application typically stores to contiguous locations, or tends to store repeatedly to the same locations or to re-access data after it has been stored, then performance would likely be improved by allocating the line in the cache upon the first miss so that subsequent accesses will hit in the cache.

The SWOA field is one of two MMUCR fields which can control the allocation of cache lines upon store misses. The other is the U2SWOAE field, and if U2SWOAE is 1 then the U2 storage attribute controls the allocation and the SWOA field is ignored (see ~~"User-Definable (U0–U3)" on page 6-15~~*User-Definable (U0–U3)* on page 246). However, if the U2SWOAE field is 0, then the SWOA field controls cache line allocation for all cacheable store misses. Specifically, if a cacheable store access misses in the data cache, then if SWOA is 0, then the cache line will be filled into the data cache, and the store data will be written into the cache (as well as to memory if the associated memory page is also marked as write-through; see ~~"Write-Through (W)" on page 6-14~~*Write-Through (W)* on page 245). Conversely, if SWOA is 1, then cacheable store misses will *not* allocate the line in the data cache, and the store data will be written to memory only, whether or not the write-through attribute is set.

See ~~Chapter 5, "Instruction and Data Caches"~~*Instruction and Data Caches* on page 205 for more information on cache line allocation on store misses.

U1 Transient Enable (U1TE) Field

When U1TE is 1, then the U1 storage attribute is enabled to control the *transient* mechanism of the instruction and data caches (see "User-Definable (U0–U3)" on page -246). If the U1 field of the TLB entry for the memory page being accessed is 0, then the access will use the *normal* portion of the cache. If the U1 field is 1, then the transient portion of cache will be used.

If the U1TE field is 0, then the transient cache mechanism is disabled and all accesses use the normal portion of the cache.

See Chapter 5, "Instruction and Data Caches" for more information on the transient cache mechanism.

U2 Store Without Allocate Enable (U2SWOAE) Field

An explanation of the allocation of cache lines on store misses is provided in the section on the SWOA field above. The U2SWOAE field is the other mechanism which can control such allocation. If U2SWOAE is 0, then the SWOA field determines whether or not a cache line is allocated on a store miss.

When U2SWOAE is 1, then the U2 storage attribute is enabled to control the allocation on a memory page basis, and the SWOA field is ignored (see "User-Definable (U0–U3)" on page -246). If the U2 field of the TLB entry for the memory page containing the bytes being stored is 0, then the cache line will be allocated in the data cache on a store miss. If the U2 field is 0, then the cache line will *not* be allocated.

See Chapter 5, "Instruction and Data Caches" for more information on cache line allocation on store misses.

Data Cache Unlock Exception Enable (DULXE) Field

The DULXE field can be used to force a Cache Locking exception type Data Storage interrupt to occur if a **dcbf** instruction is executed in user mode (MSR[PR]=1). Since **dcbf** can be executed in user mode and since it causes a cache line to be flushed from the data cache, it has the potential for allowing an application program to remove a locked line from the cache. The locking and unlocking of cache lines is generally a supervisor mode function, as the supervisor has access to the various mechanisms which control the cache locking mechanism (e.g., the Data Cache Victim Limit (DVLIM) and Instruction Cache Victim Limit (IVLIM) registers, and the MMUCR). Therefore, the DULXE field provides a means to prevent any **dcbf** instructions executed while in user mode from flushing any cache lines.

Note that with the PPC440GP, the Cache Locking exception occurs independent of whether the target line is truly locked or not. This behavior is necessary because the instruction execution pipeline is such that the exception determination must be made before it is determined whether or not the target line is actually locked (or whether it is even a hit).

Software at the Data Storage interrupt handler can determine whether the target line is locked, and if so whether or not the application should be allowed to unlock it.

If DULXE is 0, or if **dcbf** is executed while in supervisor mode, then the instruction execution is allowed to proceed and flush the target line, independent of whether it is locked or not.

See Chapter 5, "Instruction and Data Caches" for more information on cache locking.

Instruction Cache Unlock Exception Enable (IULXE) Field

The IULXE field can be used to force a Cache Locking exception type Data Storage interrupt to occur if an **icbi** instruction is executed in user mode (MSR[PR]=1). Since **icbi** can be executed in user mode and since it causes a cache line to be removed from the instruction cache, it has the potential for allowing an application program to remove a locked line from the cache. The locking and unlocking of cache lines is generally a supervisor mode function, as the supervisor has access to the various mechanisms which control the cache locking mechanism (e.g., the DVLIM and IVLIM registers, and the MMUCR). Therefore, the IULXE field provides a means to prevent any **icbi** instructions executed while in user mode from flushing any cache lines.

Note that with the PPC440GP, the Cache Locking exception occurs independent of whether the target line is truly locked or not. This behavior is necessary because the instruction execution pipeline is such that the exception determination must be made before it is determined whether or not the target line is actually locked (or whether it is even a hit).

Software at the Data Storage interrupt handler can determine whether the target line is locked, and if so whether or not the application should be allowed to unlock it.

If IULXE is 0, or if **icbi** is executed while in supervisor mode, then the instruction execution is allowed to proceed and flush the target line, independent of whether it is locked or not.

See Chapter 5, "Instruction and Data Caches" for more information on cache locking.

Search Translation Space (STS) Field

The STS field is used by the **tlbsx[.]** instruction to designate the value against which the TS field of the TLB entries is to be matched. For instruction fetch and data storage accesses, the TS field of the TLB entries is compared with the MSR[IS] bit or the MSR[DS] bit, respectively. For **tlbsx[.]** however, the MMUCR[STS] field is used, allowing the TLB to be searched for entries with a TS field which is references an address space other than the one being used by the currently executing process.

See "Address Space Identifier Convention" on page -238 for more information on the TLB entry TS field.

Search Translation ID (STID) Field

The STID field is used by the **tlbsx[.]** instruction to designate the process identifier value ~~against which~~ to be compared with the TID field of the TLB ~~entries is to be matched~~entries. For instruction fetch and data storage ~~accesses~~accesses and cache management operations, the TID field of the TLB entries is compared with the value in the PID register (see "Process ID (PID)" on page -251). For **tlbsx[.]** however, the MMUCR[STID] field is used, allowing the TLB to be searched for entries with a TID field which does not match the Process ID of the currently executing process.

The MMUCR[STID] field is also used to transfer the TLB entry's TID field on **tlbre** and **tlbwe** instructions which target TLB word 0, as there are not enough bits in the GPR used for transferring the other fields such that it could hold this field as well.

See "TLB Match Process" on page -239 for more information on the TLB entry TID field and the address matching process. Also see "TLB Read/Write Instructions (tlbre/tlbwe)" on page -253 for more information on how the MMUCR[STID] field is used by these instructions.

6.7.2 Process ID (PID)

The Process ID (PID) is a 32-bit register, although only the lower 8 bits are defined in the PPC440GP. The 8-bit PID value is used as a portion of the virtual address for accessing storage (see "Virtual Address Formation" on page -238). The PID value is compared against the TID field of a TLB entry to determine whether or not the entry corresponds to a given virtual address. If an entry's TID field is 0 (signifying that the entry defines a "global" as opposed to "private" page), then the PID value is ignored when determining whether the entry corresponds to a given virtual address. See "TLB Match Process" on page -239 for a more detailed description of the use of the PID value in the TLB match process.

The PID is written from a GPR using **mtspr**, and can be read into a GPR using **mfspr**. The following figure illustrates the PID.

Figure 6-4. Process ID (PID)

0:23		Reserved
24:31		Process ID

6.8 Shadow TLB Arrays

The PPC440GP implements two shadow TLB arrays, one for instruction fetches and one for data accesses. These arrays "shadow" the value of a subset of the entries in the main, unified TLB (the UTLB in the context of this discussion). The purpose of the shadow TLB arrays is to reduce the latency of the address translation operation, and to avoid contention for the UTLB array between instruction fetches and data accesses.

The instruction shadow TLB (ITLB) contains four entries, while the data shadow TLB (DTLB) contains eight. There is no latency associated with accessing the shadow TLB arrays, and instruction execution continues in a pipelined fashion as long as the requested address is found in the shadow TLB. If the requested address is not found in the shadow TLB, the instruction fetch or data storage access is automatically stalled while the address is looked up in the UTLB. If the address is found in the UTLB, the penalty associated with the miss in the shadow array is three cycles. If the address is also a miss in the UTLB, then an Instruction or Data TLB Miss exception is reported.

The replacement of entries in the shadow TLB's is managed by hardware, in a round-robin fashion. Upon a shadow TLB miss which leads to a UTLB hit, the hardware will automatically cast-out the oldest entry in the shadow TLB and replace it with the new translation.

The hardware will also automatically invalidate all of the entries in both of the shadow TLB's upon any context synchronization (see "Context Synchronization" on page -201). Context synchronizing operations include the following:

- Any interrupt (including Machine Check)
- Execution of **isync**
- Execution of **rfi** or **rfci**
- Execution of **sc**

Note that there are other "context changing" operations which do not cause automatic context synchronization in the hardware. For example, execution of a **tlbwe** instruction changes the UTLB contents but does not cause a context synchronization and thus does not invalidate or otherwise update the shadow TLB entries. In order for changes to the entries in the UTLB (or to other address-related resources such as the PID) to be reflected in the shadow TLB's, software must ensure that a context synchronizing operation occurs prior to any attempt to use any address associated with the updated UTLB entries (either the old or new contents of

those entries). By invalidating the shadow TLB arrays, a context synchronizing operation forces the hardware to refresh the shadow TLB entries with the updated information in the UTLB as each memory page is accessed.

Note: Of the items in the preceding list of shadow TLB invalidating operations, the Machine Check interrupt is not architecturally required to be context synchronizing, and thus is not guaranteed to cause invalidation of any shadow TLB arrays on implementations other than the PPC440GP. Consequently, software which is intended to be portable to other implementations should not depend on this behavior, and should insert the appropriate architecturally-defined context synchronizing operation as necessary for desired operation.

6.9 TLB Management Instructions

The processor does not imply any format for the page tables or the page table entries. Software has significant flexibility in organizing the size, location, and format of the page table, and in implementing a custom TLB entry replacement strategy. For example, software can "lock" TLB entries that correspond to frequently used storage, so that those entries are never cast out of the TLB, and TLB Miss exceptions to those pages never occur.

In order to enable software to manage the TLB, a set of TLB management instructions is implemented within the PPC440GP. These instructions are described briefly in the sections which follow, and in detail in Chapter 28, "Instruction Set." In addition, the interrupt mechanism provides resources to assist with software handling of TLB-related exceptions. One such resource is Save/Restore Register 0 (SRR0), which provides the exception-causing address for Instruction TLB Error and Instruction Storage interrupts. Another resource is the Data Exception Address Register (DEAR), which provides the exception-causing address for Data TLB Error and Data Storage interrupts. Finally, the Exception Syndrome Register (ESR) provides bits to differentiate amongst the various exception types which may cause a particular interrupt type. See Chapter 10, "Interrupts and Exceptions." for more information on these mechanisms.

All of the TLB management instructions are privileged, in order to prevent user mode programs from affecting the address translation and access control mechanisms.

6.9.1 TLB Search Instruction (tlbsx[.])

The **tlbsx[.]** instruction can be used to locate an entry in the TLB which is associated with a particular virtual address. This instruction forms an effective address for which the TLB is to be searched, in the same manner by which data storage access instructions perform their address calculation, by adding the contents of registers RA (or the value 0 if RA=0) and RB together. The MMUCR[STID] and MMUCR[STS] fields then provide the process ID and address space portions of the virtual address, respectively. Next, the TLB is searched for this virtual address, with the searching process including the notion of disabling the comparison to the process ID if the TID field of a given TLB entry is 0 (see "TLB Match Process" on page -239). Finally, the TLB index of the matching entry is written into the target register (RT). This index value can then serve as the source value for a subsequent **tlbre** or **tlbwe** instruction, to read or update the entry. If no matching entry is found, then the target register contents are undefined.

The "record form" of the instruction (**tlbsx.**) updates $CR[CR0]_2$ with the result of the search: if a match is found, then $CR[CR0]_2$ is set to 1; otherwise it is set to 0.

6.9.2 TLB Read/Write Instructions (tlbre/tlbwe)

TLB entries can be read and written by the **tlbre** and **tlbwe** instructions, respectively. Since a TLB entry contains more than 32 bits, multiple **tlbre/tlbwe** instructions must be executed in order to transfer all of the TLB entry information. A TLB entry is divided into three portions, TLB word 0, TLB word 1, and TLB word 2. The RA field of the **tlbre** and **tlbwe** instructions designates a GPR from which the low-order six bits are used to specify the TLB index of the TLB entry to be read or written~~, using the low-order six bits of RA~~. An immediate field (WS) designates which word of the TLB entry is to be transferred (that is, WS=0 specifies TLB word 0, and so on). Finally, the contents of the selected TLB word are transferred to or from a designated target or source ~~GPR~~GPR (and the MMUCR[STID] field, for TLB word 0; see below), respectively.

The fields in each TLB ~~Word~~ word are illustrated in ~~Figure 6-5~~Figure 6-5. The bit numbers indicate which bits of the target/source GPR correspond to each TLB field. Note that the TID field of TLB ~~Word~~ word 0 is transferred to/from the MMUCR[STID] field, rather than to/from the target/source GPR.

~~Figure 0-3. TLB Entry Word Definitions~~

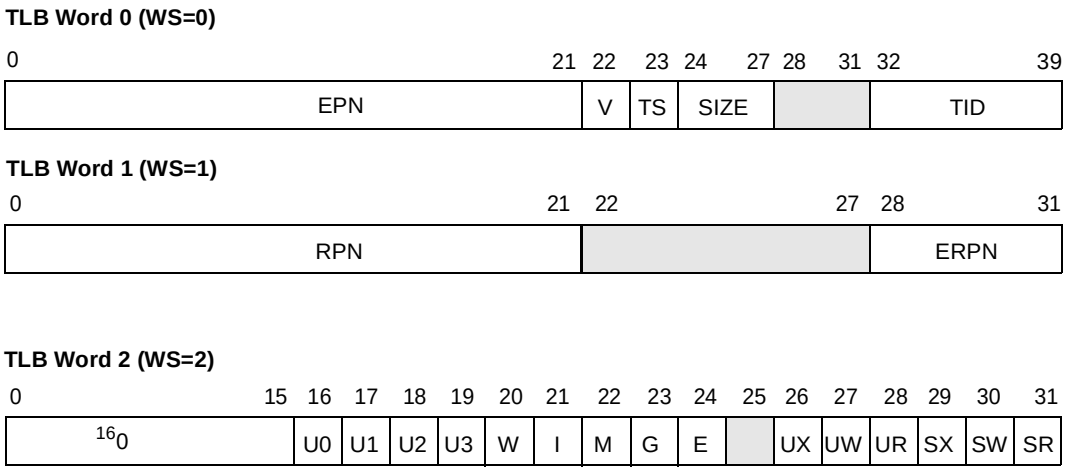
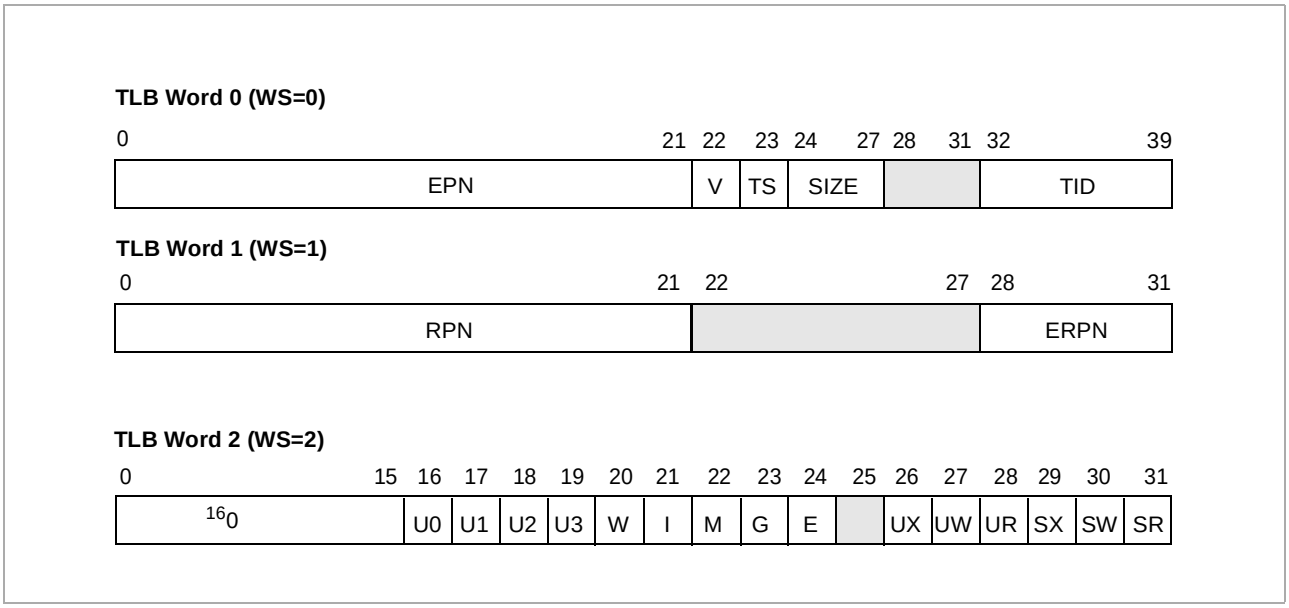


Figure 6-5. TLB Entry Word Definitions



6.9.3 TLB Sync Instruction (tlbsync)

The **tlbsync** instruction is used to synchronize software TLB management operations in a multiprocessor environment with hardware-enforced coherency, which is not supported by the PPC440GP. Consequently, this instruction is treated as a no-op. It is provided in support of software compatibility between PowerPC-based systems.

6.10 Page Reference and Change Status Management

When performing page management, it is useful to know whether a given memory page has been referenced, and whether its contents have been modified. Note that this may be more involved than determining whether a given TLB entry has been used to reference or change memory, since multiple TLB entries may translate to the same memory page. If it is necessary to replace the contents of some memory page with other contents, a page which has been referenced (accessed for any purpose) is more likely to be maintained than a page which has never been referenced. If the contents of a given memory page are to be replaced and the contents of that page have been changed, then the current contents of that page must be written to backup physical storage (such as a hard disk) before replacement.

Similarly, when performing TLB management, it is useful to know whether a given TLB entry has been referenced. When making a decision about which entry of the TLB to replace in order to make room for a new entry, an entry which has never been referenced is a more likely candidate to be replaced.

The PPC440GP does not automatically record references or changes to a page or TLB entry. Instead, the interrupt mechanism may be used by system software to maintain reference and change information for TLB entries and their associated pages, respectively.

Execute, Read and Write Access Control exceptions may be used to allow software to maintain reference and change information for a TLB entry and for its associated memory page. The following description explains one way in which system software can maintain such reference and change information.

The TLB entry is originally written into the TLB with its access control bits (UX, SX, UR, SR, UW, and SW) off. The first attempt of application code to use the page will therefore cause an Access Control exception and a corresponding Instruction or Data Storage interrupt. The interrupt handler records the reference to the TLB entry and to the associated memory page in a software table, and then turns on the appropriate access control bit, thereby indicating that the particular TLB entry has been referenced. An initial read from the page is handled by only turning on the appropriate UR or SR access control bit, leaving the page "read-only". Subsequent read accesses to the page via that TLB entry will proceed normally.

If a write access is later attempted, a Write Access Control exception type Data Storage interrupt will occur. The interrupt handler records the change status to the memory page in a software table, and then turns on the appropriate UW or SW access control bit, thereby indicating that the memory page associated with the particular TLB entry has been changed. Subsequent write accesses to the page via that TLB entry will proceed normally.

Part III. PPC440GP System Operations

7. Reset and Initialization

This chapter describes the initial state of the PPC440GP after a hardware reset, and contains a description of the initialization software required to complete initialization so that the PPC440GP can begin executing application code. Initialization of other on-chip and off-chip system components is described in the appropriate chapter.

7.1 Reset Signals

The PPC440GP provides two reset signals, SysReset and ExtReset. SysReset is bidirectional and ExtReset is an output.

When the SysReset signal is an input (asserted by an off-chip device), such as during power on reset (POR), the chip responds by performing a system reset as described in a following section. An external assertion of SysReset is not extended by an assertion of the open drain bidirectional SysReset driver.

As an output, the PPC440GP asserts the SysReset signal when a system reset request is detected. The SysReset open drain driver is activated and the signal is driven low for 8192 SysClk periods. This enables the PPC440GP to reset itself and other devices attached to the same reset network.

The ExtReset signal is used by synchronous peripheral devices served by the external bus clock, such as ROM and external masters. During chip and system resets, ExtReset is asserted until the EXT_BUS signal is stable and all internal resets are released.

7.2 Reset Types

Three types of reset, each with different scope, are possible in the PPC440GP. A core reset affects only the processor core. Chip resets affect the processor core and all on-chip peripherals. System resets affect the processor core, all on-chip peripherals, and any off-chip devices connected to the PPC440GP SysReset signal. Refer to chapters describing the on-chip peripherals for detailed information about their reset behavior.

The effects of system, chip and core resets on the processor core are identical. To determine which reset type occurred, the most-recent reset (MRR) field of the Debug Status Register (DBSR) can be examined.

7.2.1 Core Reset

A core reset results in a reset of the processor core. No other on-chip logic is affected. Clocking logic, outside the processor core, detects the core reset request and asserts the reset input to the processor core for a period of 32 system clock cycles.

7.2.2 Chip Reset

A chip reset results in the reset of the processor core and on-chip peripherals. Clocking logic detects the request for a chip reset and asserts the reset input to the processor core and all on-chip peripherals for an extended period while the PLL is allowed to relock. PLL locking is performed as described for a system reset.

During chip reset, the ExtReset signal is driven low to ensure the reset of synchronous devices that use the external bus clock signal, PerClk.

7.2.3 System Reset

A system reset results in a reset of all PPC440GP logic, including the processor core, internal phase-locked loop (PLL), and on-chip peripherals. A system reset can be initiated externally or internally. External system resets are initiated by the assertion of the $\overline{\text{SysReset}}$ signal for at least 16 SysClk cycles. Internal system resets are initiated by either the processor core or PCI power management logic.

When a system reset is requested internally, the bidirectional open drain $\overline{\text{SysReset}}$ signal is asserted to enable other chips to be reset at the same time. In this case, the $\overline{\text{SysReset}}$ signal is driven low for 8192 SysClk cycles, resulting in a System reset of the PPC440GP chip and all other devices attached to the reset network connected to $\overline{\text{SysReset}}$. During this time all internal clocks and EXT_BUS clock toggle.

After the $\overline{\text{SysReset}}$ signal is deasserted, the PLL begins its locking process, which requires about 8192 SysClk cycles when CPU feedback is selected and 16384 SysClk cycles when EXT_BUS feedback is selected. During this time all internal clocks and EXT_BUS clock toggle. When the PLL lock timer expires, internal resets are released, and the processor core begins its initial instruction fetch.

During system reset, the $\overline{\text{ExtReset}}$ signal is driven low to ensure the reset of synchronous devices that use the external bus clock signal, EXT_BUS.

7.3 Power-On Reset Details

Figure 7-1 demonstrates the power-on reset process in PPC440GP from the deassertion of SYS_RST until the beginning of system operation. Note that the horizontal timeline is relative and that precise durations of events are not implied.

RESET STATE is a state machine implemented in PPC440GP which automatically transitions through each of the indicated states. Several of the states can be arbitrarily long, including "system reset," "serial rom load," "pci xcapable detection," because exiting these states is dependant upon logic and signalling that occurs outside of the PPC440GP. The PLL locking state is always 8192 system clock periods long.

The "flip check" state is used to check whether the CPU and PLB clocks have proper rising edge alignment after the PLL has locked. Since correct alignment between these clocks is not guaranteed for all clock ratios, a special circuit is used to sometimes perform a single "flip" to correct the alignment. If CPU feedback is chosen, no additional PLL locking time is needed because the affected clocks are not in the feedback path. For predictability, though, when EXT bus feedback is chosen, a second PLL locking time is always used to permit the PLL time to recover from a momentary perturbation of the feedback clock. This second PLL locking is also 8192 system clock periods long, and occurs when EXT bus feedback is chosen regardless of whether a flip was required.

The deassertion edge of the SYS_RST signal causes the initial transition of RESET STATE from "system reset" to the next state. During the entire reset process the EXT_RST signal is asserted, and it only deasserts after the system clocking becomes stable and operational.

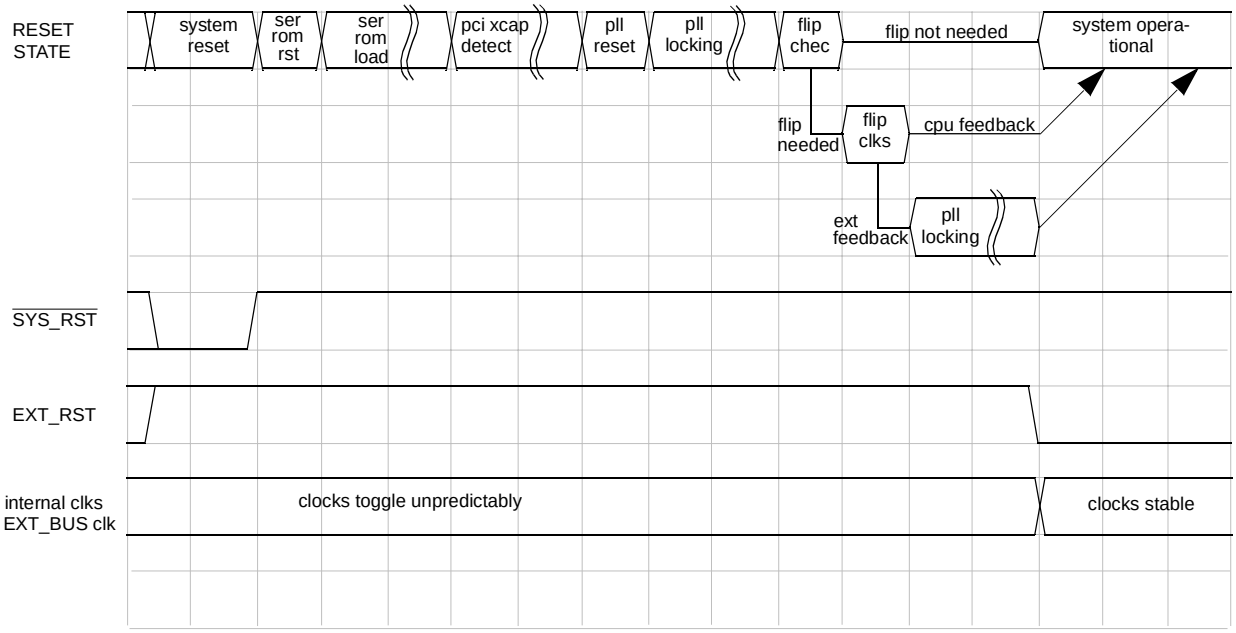


Figure 7-1. PPC440GP Power-on Reset Process

Note: The internal clocks, including CPU, PLB, OPB, EXT and Serial, as well as the externally visible EXT_BUS clock and DDR DQS may all toggle unpredictably during the reset process. They are only guaranteed to be valid and stable after RESET STATE enters the 'system operational' state.

7.4 Processor Core State After Reset

In general, the contents of registers and other facilities within the processor core are undefined after a hardware reset. Reset is defined to initialize only the minimal resources required such that instructions can be fetched and executed from the initial program memory page, and so that repeatable, deterministic behavior can be guaranteed provided that the proper software initialization sequence is followed. System software must fully configure the rest of the PPC440GP resources, as well as the other facilities within the chip and/or system.

The following list summarizes the requirements of the Book-E Enhanced PowerPC Architecture with regards to the processor state after reset, prior to any additional initialization by software.

- All fields of the MSR are set to 0, disabling all asynchonous interrupts, placing the processor in supervisor mode, and specifying that instruction and data accesses are to the system (as opposed to application) address space.
- DBCR0[RST] is set to 0, thereby ending any previous software-initiated reset operation.
- DBSR[MRR] records the type of the just ended reset operation (core, chip, or system; see ~~"Reset Types" on page 7-1~~*Reset Types* on page 259).

- TCR[WRC] is set to 0, thereby disabling the Watchdog timer reset operation.
- TSR[WRS] records the type of the just ended reset operation, if the reset was initiated by the Watchdog Timer (otherwise this field is unchanged from its pre-reset value).
- The PVR is defined, after reset and otherwise, to contain a value that indicates the specific processor implementation.
- The program counter (PC) is set to 0xFFFFFFFC, the effective address (EA) of the last word of the address space.

The memory management resources are set to values such that the processor is able to successfully fetch and execute instructions and read ~~and write~~ (but not write) data within the 4KB program memory page located at the end of the 32-bit effective address space. Exactly how this is accomplished is implementation-dependent. For example, it may or may not be the case that a TLB entry is established in a manner which is visible to software using the TLB management instructions. Regardless of how the implementation enables access to the initial program memory page, instruction execution ~~will start with~~ starts at the effective adddress of 0xFFFFFFFC, the last word of the effective address space. The instruction at this address must be an unconditional branch backwards to the start of the initialization sequence, which must lie somewhere within the initial 4KB program memory page. The real address to which the initial effective address will be translated is also implementation- or system-dependent, as are the various storage attributes of the initial program memory page such as the caching inhibited and endian attributes.

Note: In the PPC440GP~~,~~ a single entry is established in the instruction shadow TLB (ITLB) and data shadow TLB (DTLB) at reset with the properties described in Table 7-1. It is required that initialization software insert an entry into the UTLB to cover this same memory region before performing any context synchronizing ~~operation~~operation (including causing any exceptions which would lead to an interrupt), since a context synchronizing operation will invalidate the shadow TLB entries.

Initialization software should consider all other resources within the PPC440GP to be undefined after reset, in order for the initialization sequence to be compatible with other PowerPC implementations. There are, however, additional core resources which are initialized by reset, in order to guarantee correct and deterministic operation of the processor during the initialization sequence. Table 7-1 shows the reset state of all PPC440GP resources which are defined to be initialized by reset. While certain other register fields and other facilities within the PPC440GP may be affected by reset, this is not an architectural nor hardware requirement, and software must treat those resources as undefined. Likewise, even those resources which are included in Table 7-1 but which are not identified in the previous list as being architecturally required, should be treated as undefined by the initialization software.

Table 0-1. Reset Values of Registers and Other PPC440GP Core Facilities

Resource	Field	Reset Value	Comment
CCR0	DAPUIB	0	Enable broadcast of instruction data to auxiliary processor interface
	DTB	0	Enable broadcast of trace information

Table 0-1. Reset Values of Registers and Other PPC440GP Core Facilities

Resource	Field	Reset Value	Comment
DBCR0	EDM	0	External Debug mode disabled
	RST	0b00	Software-initiated debug reset disabled
	ICMP	0	Instruction completion debug events disabled
	BRT	0	Branch taken debug events disabled
	IAC1	0	Instruction Address Compare 1 (IAC1) debug events disabled
	IAC2	0	IAC2 debug events disabled
	IAC3	0	IAC3 debug events disabled
	IAC4	0	IAC4 debug events disabled
DBSR	UDE	0	Unconditional debug event has not occurred
	MRR	Reset-dependent	Indicates most recent type of reset as follows: 00 No reset has occurred since this field last cleared by software 01 Core reset 10 Chip reset 11 System reset
	ICMP	0	Instruction completion debug event has not occurred
	BRT	0	Branch taken debug event has not occurred
	IRPT	0	Interrupt debug event has not occurred
	TRAP	0	Trap debug event has not occurred
	IAC1	0	IAC1 debug event has not occurred
	IAC2	0	IAC2 debug event has not occurred
	IAC3	0	IAC3 debug event has not occurred
	IAC4	0	IAC4 debug event has not occurred
	DAC1R	0	Data address compare 1 (DAC1) read debug event has not occurred
	DAC1W	0	DAC1 write debug event has not occurred
	DAC2R	0	DAC2 read debug event has not occurred
	DAC2W	0	DAC2 write debug event has not occurred
	RET	0	Return debug event has not occurred
ESR	MCI	0	Instruction Machine Check exception has not occurred

Table 0-1. Reset Values of Registers and Other PPC440GP Core Facilities

Resource	Field	Reset Value	Comment
MSR	WE	0	Wait state disabled
	CE	0	Asynchronous critical interrupts disabled
	EE	0	Asynchronous non-critical interrupts disabled
	PR	0	Processor in supervisor mode
	FP	0	Floating-point Unavailable interrupts disabledStorage
	ME	0	Machine Check interrupts disabled
	FE0	0	Floating-point Enabled interrupts disabled
	DWE	0	Debug Wait mode disabled
	DE	0	Debug interrupts disabled
	FE1	0	Floating-point Enabled interrupts disabled
	IS	0	Instruction fetch access is to system-level virtual address space
	DS	0	Data access is to system level virtual address space
PC		0xFFFFFFFC	Initial reset instruction fetched from last word of effective addess space
PVR	0:31	System-dependent	Refer to *PowerPC 440GP Embedded Processor Data Sheet* for the PVR value
RSTCFG	U0	0	The U0 storage attribute has no effect in the PPC440GP.
	U1	0	Memory page contain normal instructions or data
	U2	0	Storage misses do not cause a line to allocated in the data cache
	U3	0	The U3 storage attribute has no effect in the PPC440GP.
	E	0	Memory pages are big endian
	EPRN	System-dependent	If ROM is connected to EBCO, ERPN is 0b0001. If ROM is connected to PCI, ERPN is 0b0010.
TCR	WRC	0b00	Watchdog Timer reset disabled

Table 0-1. Reset Values of Registers and Other PPC440GP Core Facilities

Resource	Field	Reset Value	Comment
TLBentry[1]	$EPN_{0:19}$	0xFFFFF000	Match EA of initial reset instruction ($EPN_{20:21}$ are undefined, as they are not compared to the EA because the page size is 4KB).
	V	1	Translation table entry for the initial program memory page is valid.
	TS	0	Initial program memory page is in system-level virtual address space.
	SIZE	0b0001	Initial program memory page size is 4KB.
	TID	0x00	Initial program memory page is globally shared; no match required against PID register.
	$RPN_{0:21}$	0xFFFFF || 0b00	Initial program memory page mapped effective=real.
	ERPN	0b0001 0b0010	Extended real page number of the initial program memory page is specified by core input signals. Shadow TLB entry at reset is 0x1FFFFF000 if PPC440GP is strapped to boot from ROM connected to the EBCO and 0x2FFFFF000 if PPC440GP is strapped to boot from PCI memory.
	U0	0	The U0 storage attribute has no effect in the PPC440GP.
	U1	0	Memory page contains normal intstructions or data.
	U2	0	Storage misses do not cause a line to be allocated in the data cache.
	U3	0	The U3 storage attribute has no effect in the PPC440GP.
	W	0	Write-through storage attribute disabled.
	I	1	Caching inhibited storage attribute enabled.
	M	0	Memory coherent storage attribute disabled.
	G	1	Guarded storage attribute enabled.
	E	0	Reset value of endian storage attribute is specified by a core input signal.
	UX	1	User mode execution access enabled.
	UW	1	User mode write access enabled.
	UR	1	User mode read access enabled.
	SX	1	Supervisor mode execution access enabled.
	SW	1	Supervisor mode write access enabled.
	SR	1	Supervisor mode read access enabled.

Table 0-1. Reset Values of Registers and Other PPC440GP Core Facilities

Resource	Field	Reset Value	Comment
TSR	WRS	Copy of TCR[WRC]	If reset caused by Watchdog Timer
		Unchanged	If reset not caused by Watchdog Timer
		Undefined	After power-up

Note 1: "TLBentry" refers to an entry in the shadow instruction and data TLB arrays that is automatically configured by the PPC440GP to enable fetching, reading, and writing from the initial program memory page. This entry is not architecturally visible to software, and is invalidated upon any context synchronizing operation. Software must initialize a corresponding entry in the main unified TLB array before executing any operation which could lead to a context synchronization. See "Initialization Software Requirements" on page 7-22 for more information.

Table 7-1. Reset Values of Registers and Other PPC440GP Core Facilities

Resource	Field	Reset Value	Comment
CCR0	DAPUIB	0	Enable broadcast of instruction data to auxiliary processor interface
	DTB	0	Enable broadcast of trace information
DBCR0	EDM	0	External Debug mode disabled
	RST	0b00	Software-initiated debug reset disabled
	ICMP	0	Instruction completion debug events disabled
	BRT	0	Branch taken debug events disabled
	IAC1	0	Instruction Address Compare 1 (IAC1) debug events disabled
	IAC2	0	IAC2 debug events disabled
	IAC3	0	IAC3 debug events disabled
	IAC4	0	IAC4 debug events disabled
DBSR	UDE	0	Unconditional debug event has not occurred
	MRR	Reset-dependent	Indicates most recent type of reset as follows: 00 No reset has occurred since this field last cleared by software 01 Core reset 10 Chip reset 11 System reset
	ICMP	0	Instruction completion debug event has not occurred
	BRT	0	Branch taken debug event has not occurred
	IRPT	0	Interrupt debug event has not occurred
	TRAP	0	Trap debug event has not occurred
	IAC1	0	IAC1 debug event has not occurred
	IAC2	0	IAC2 debug event has not occurred
	IAC3	0	IAC3 debug event has not occurred
	IAC4	0	IAC4 debug event has not occurred
	DAC1R	0	Data address compare 1 (DAC1) read debug event has not occurred

Table 7-1. Reset Values of Registers and Other PPC440GP Core Facilities

Resource	Field	Reset Value	Comment
ESR	MCI	0	Instruction Machine Check exception has not occurred
MSR	WE	0	Wait state disabled
	CE	0	Asynchronous critical interrupts disabled
	EE	0	Asynchronous non-critical interrupts disabled
	PR	0	Processor in supervisor mode
	FP	0	Floating-point Unavailable interrupts disabledStorage
	ME	0	Machine Check interrupts disabled
	FE0	0	Floating-point Enabled interrupts disabled
	DWE	0	Debug Wait mode disabled
	DE	0	Debug interrupts disabled
	FE1	0	Floating-point Enabled interrupts disabled
	IS	0	Instruction fetch access is to system-level virtual address space
	DS	0	Data access is to system level virtual address space
PC		0xFFFFFFFC	Initial reset instruction fetched from last word of effective addess space
PVR	0:31	System-dependent	Refer to *PowerPC 440GP Embedded Processor Data Sheet* for the PVR value
RSTCFG	U0	0	The U0 storage attribute has no effect in the PPC440GP.
	U1	0	Memory page contain normal instructions or data
	U2	0	Storage misses do not cause a line to allocated in the data cache
	U3	0	The U3 storage attribute has no effect in the PPC440GP.
	E	0	Memory pages are big endian
	EPRN	System-dependent	If ROM is connected to EBCO, ERPN is 0b0001. If ROM is connected to PCI, ERPN is 0b0010.
TCR	WRC	0b00	Watchdog Timer reset disabled

Table 7-1. Reset Values of Registers and Other PPC440GP Core Facilities

Resource	Field	Reset Value	Comment
TLBentry[1]	$EPN_{0:19}$	0xFFFFF000	Match EA of initial reset instruction ($EPN_{20:21}$ are undefined, as they are not compared to the EA because the page size is 4KB).
	V	1	Translation table entry for the initial program memory page is valid.
	TS	0	Initial program memory page is in system-level virtual address space.
	SIZE	0b0001	Initial program memory page size is 4KB.
	TID	0x00	Initial program memory page is globally shared; no match required against PID register.
	$RPN_{0:21}$	0xFFFFF || 0b00	Initial program memory page mapped effective=real.
	ERPN	0b0001 0b0010	Extended real page number of the initial program memory page is specified by core input signals. Shadow TLB entry at reset is 0x1FFFFF000 if PPC440GP is strapped to boot from ROM connected to the EBCO and 0x2FFFFF000 if PPC440GP is strapped to boot from PCI memory.
	U0	0	The U0 storage attribute has no effect in the PPC440GP.
	U1	0	Memory page contains normal instructions or data.
	U2	0	Storage misses do not cause a line to be allocated in the data cache.
	U3	0	The U3 storage attribute has no effect in the PPC440GP.
	W	0	Write-through storage attribute disabled.
	I	1	Caching inhibited storage attribute enabled.
	M	0	Memory coherent storage attribute disabled.
	G	1	Guarded storage attribute enabled.
	E	0	Reset value of endian storage attribute is specified by a core input signal.
	SX	1	Supervisor mode execution access enabled.
	SW	0	Supervisor mode write access disabled.
	SR	1	Supervisor mode read access enabled.
TSR	WRS	Copy of TCR[WRC]	If reset caused by Watchdog Timer
		Unchanged	If reset not caused by Watchdog Timer
		Undefined	After power-up

Note 1: "TLBentry" refers to an entry in the shadow instruction and data TLB arrays that is automatically configured by the PPC440GP to enable fetching and reading (but not writing) from the initial program memory page. This entry is not architecturally visible to software, and is invalidated upon any context synchronizing operation. Software must initialize a corresponding entry in the main unified TLB array before executing any operation which could lead to a context synchronization. See "Initialization Software Requirements" on page -278 for more information.

0.1 DCR Contents after Reset

DCR reset values are unaffected by core resets and are generally identical for chip and system resets.

Table 0-2. DCR Contents After Reset

Register	Bits	Reset Value	Comment
Chip Control			
CPC0_SYS0	0:31		Reset according to values received from serial ROM logic
CPC0_SYS1	0:31		Reset according to values received from serial ROM logic
CPC0_CUST0	0:31		Reset according to values received from serial ROM logic
CPC0_CUST1	0:31		Reset according to values received from serial ROM logic
CPC0_STRP0	0:31		Reset according to values received from serial ROM logic
CPC0_STRP1	0:31		Reset according to values received from serial ROM logic
CPC0_STRP2	0:31		Reset according to values received from serial ROM logic
CPC0_STRP3	0:31		Reset according to values received from serial ROM logic
CPC0_GPIO	0:31	0x00002000	
CPC0_PLB	0:31	0x000000F8	
CPC0_CR0	0:31	0x001E01B0	
CPC0_CR1	0:31	0xEAAEA012	
CPC0_MIRQ0	0:31	0x00000000	
CPC0_MIRQ1	0:31	0x00000000	
CPC0_JTAGID	0:31		Refer to *PPC440GP Embedded Processor Data Sheet* for the value of this read-only register.
Clock and Power Management (CPM)			
CPC0_ER	0:31	0x00000000	$CPC0\_ER_{0:16}$ return 1, $CPC0\_ER_{17:31}$ return 0.
CPC0_FR	0:31	0x00000000	
CPC0_SR	0:31	0xFFFFFFFF	
Direct Memory Access (DMA)			
DMA0_CR0	0:31	0x00000000	
DMA0_CR1	0:31	0x00000000	
DMA0_CR2	0:31	0x00000000	
DMA0_CR3	0:31	0x00000000	
DMA0_CT0	0:31	0x00000000	
DMA0_CT1	0:31	0x00000000	
DMA0_CT2	0:31	0x00000000	
DMA0_CT3	0:31	0x00000000	

Table 0-2. DCR Contents After Reset (continued)

Register	Bits	Reset Value	Comment
DMA0_DAH0	0:31	0x00000000	
DMA0_DAH1	0:31	0x00000000	
DMA0_DAH2	0:31	0x00000000	
DMA0_DAH3	0:31	0x00000000	
DMA0_DAL0	0:31	0x00000000	
DMA0_DAL1	0:31	0x00000000	
DMA0_DAL2	0:31	0x00000000	
DMA0_DAL3	0:31	0x00000000	
DMA0_POL	0:31	0x00000000	
DMA0_SAH0	0:31	0x00000000	
DMA0_SAH1	0:31	0x00000000	
DMA0_SAH2	0:31	0x00000000	
DMA0_SAH3	0:31	0x00000000	
DMA0_SAL0	0:31	0x00000000	
DMA0_SAL1	0:31	0x00000000	
DMA0_SAL2	0:31	0x00000000	
DMA0_SAL3	0:31	0x00000000	
DMA0_SGH0	0:31	0x00000000	
DMA0_SGH1	0:31	0x00000000	
DMA0_SGH2	0:31	0x00000000	
DMA0_SGH3	0:31	0x00000000	
DMA0_SGL0	0:31	0x00000000	
DMA0_SGL1	0:31	0x00000000	
DMA0_SGL2	0:31	0x00000000	
DMA0_SGL3	0:31	0x00000000	
DMA0_SGC	0:31	0x00000000	
DMA0_SLP	0:31	0x07C00000	
DMA0_SR	0:31	0x00000000	
External Bus Controller 0 (EBC0)			
EBC0_B0AP	0:31	0x7F8FFE80	Slowest possible bus timings.
EBC0_B0CR	0:31	0xFFE28000	2MB read-only bank.
EBC0_B1AP	0:31	0x00000000	

Table 0-2. DCR Contents After Reset (continued)

Register	Bits	Reset Value	Comment
EBC0_B1CR	0:31	0x00000000	
EBC0_B2AP	0:31	0x00000000	
EBC0_B2CR	0:31	0x00000000	
EBC0_B3AP	0:31	0x00000000	
EBC0_B3CR	0:31	0x00000000	
EBC0_B4AP	0:31	0x00000000	
EBC0_B4CR	0:31	0x00000000	
EBC0_B5AP	0:31	0x00000000	
EBC0_B5CR	0:31	0x00000000	
EBC0_B6AP	0:31	0x00000000	
EBC0_B6CR	0:31	0x00000000	
EBC0_B7AP	0:31	0x00000000	
EBC0_B7CR	0:31	0x00000000	
EBC0_BEAR	0:31	0x00000000	
EBC0_BESR	0:31	0x00000000	
EBC0_CFG	0:31	0x00000000	
EBC0_CID	0:31	0x00000000	
External Bus Master Interface 0 (EBM0)			
EBM0_BEAR	0:31	0x00000000	
EBM0_BEMR	0:31	0x00000000	
EBM0_BESR	0:31	0x00000000	
EBM0_CID	0:31	undefined	
EBM0_CTL	0:31	0x88400000	
EBM0_FAIR	0:31	0xFFF00000	
EBM0_LCNT	0:31	0x00000000	
EBM0_MISCSTS	0:31	undefined	
EBM0_SLPMD	0:31	0x07C00000	
EBM0_UAM	0:31	0x00000000	
EBM0_UAR	0:31	0x00000000	
Indirect Addressing Registers			
EBC0_CFGADDR	0:31	undefined	
EBC0_CFGDATA	0:31	undefined	

Table 0-2. DCR Contents After Reset (continued)

Register	Bits	Reset Value	Comment
EBM0_CFGADDR	0:31	undefined	
EBM0_CFGDATA	0:31	undefined	
PPM0_CFGADDR	0:31	undefined	
PPM0_CFGDATA	0:31	undefined	
SDRAM0_CFGADDR	0:31	undefined	
SDRAM0_CFGDATA	0:31	undefined	
Memory Access Layer (MAL)			
MAL0_CFG	0:31	0x00004080	
MAL0_ESR	0:31	0x00000000	
MAL0_IER	0:31	0x00000000	
MAL0_RXBADDR	0:31	0x00000000	
MAL0_RCBS0	0:31	Undefined	
MAL0_RCBS1	0:31	Undefined	
MAL0_RXCARR	0:31	0x00000000	
MAL0_RXCASR	0:31	0x00000000	
MAL0_RXCTP0R	0:31	Undefined	
MAL0_RXCTP1R	0:31	Undefined	
MAL0_RXDEIR	0:31	0x00000000	
MAL0_RXEOBISR	0:31	0x00000000	
MAL0_RXTATTRR	0:31	0x00000000	
MAL0_TXBADDR	0:31	0x00000000	
MAL0_TXCARR	0:31	0x00000000	
MAL0_TXCASR	0:31	0x00000000	
MAL0_TXCTP0R	0:31	Undefined	
MAL0_TXCTP1R	0:31	Undefined	
MAL0_TXCTP2R	0:31	Undefined	
MAL0_TXCTP3R	0:31	Undefined	
MAL0_TXDEIR	0:31	0x00000000	
MAL0_TXEOBISR	0:31	0x00000000	
MAL0_TXTATTRR	0:31	0x00000000	
On-Chip Buses			
PLB0_ACR	0:31	0x00000000	

Table 0-2. DCR Contents After Reset (continued)

Register	Bits	Reset Value	Comment
PLB0_BEARH	0:31	Undefined	
PLB0_BEARL	0:31	Undefined	
PLB0_BESR	0:31	0x00000000	
PLB0_REVID	0:31	Undefined	
POB0_BEARH	0:31	Undefined	
POB0_BEARL	0:31	Undefined	
POB0_BESR0	0:31	0x00000000	
POB0_BESR1	0:31	0x00000000	
POB0_CONFG	0:31	0x00000000	
POB0_LATENCY	0:31	0x00000000	
POB0_REVID	0:31	0x00000000	
PLB Performance Monitor			
PPM0_ISR	0:31	0x00000000	
PPM0_CR	0:31	0x00000000	
PPM0_CCR	0:31	0x00000000	
PPM0_UAR	0:31	0x00000000	
PPM0_LAR	0:31	0x00000000	
PPM0_UAMR	0:31	0x00000000	
PPM0_LAMR	0:31	0x00000000	
PPM0_RIDR	0:31	0x00000000	
PPM0_MCSR0	0:31	0x00000000	
PPM0_MCSR1	0:31	0x00000000	
PPM0_MCSR2	0:31	0x00000000	
PPM0_MCSR3	0:31	0x00000000	
PPM0_SCSR0	0:31	0x00000000	
PPM0_SCSR1	0:31	0x00000000	
PPM0_SCSR2	0:31	0x00000000	
PPM0_SCSR3	0:31	0x00000000	
PPM0_GCSR0	0:31	0x00000000	
PPM0_GCSR1	0:31	0x00000000	
PPM0_GCSR2	0:31	0x00000000	
PPM0_GCSR3	0:31	0x00000000	

Table 0-2. DCR Contents After Reset (continued)

Register	Bits	Reset Value	Comment
PPM0_MCR0	0:31	0x00000000	
PPM0_MCR1	0:31	0x00000000	
PPM0_MCR2	0:31	0x00000000	
PPM0_MCR3	0:31	0x00000000	
PPM0_SCR0	0:31	0x00000000	
PPM0_SCR1	0:31	0x00000000	
PPM0_SCR2	0:31	0x00000000	
PPM0_SCR3	0:31	0x00000000	
PPM0_GCR0	0:31	0x00000000	
PPM0_GCR1	0:31	0x00000000	
PPM0_GCR2	0:31	0x00000000	
PPM0_GCR3	0:31	0x00000000	
PPM0_DCSR0	0:31	0x00000000	
PPM0_DCSR1	0:31	0x00000000	
PPM0_DCMXR0	0:31	0x00000000	
PPM0_DCMXR1	0:31	0x00000000	
PPM0_DCMNR0	0:31	0x00000000	
PPM0_DCMNR1	0:31	0x00000000	
PPM0_DCTVR0	0:31	0x00000000	
PPM0_DCTVR1	0:31	0x00000000	
PPM0_DCOTR0	0:31	0x00000000	
PPM0_DCOTR1	0:31	0x00000000	
DDR_SDRAM Controller			
SDRAM0_B0CR	0:31	0x00000000	
SDRAM0_B1CR	0:31	0x00000000	
SDRAM0_B2CR	0:31	0x00000000	
SDRAM0_B3CR	0:31	0x00000000	
SDRAM0_BEAR	0:31	0x00000000	
SDRAM0_BESR0	0:31	0x00000000	
SDRAM0_BESR1	0:31	0x00000000	
SDRAM0_CFG0	0:31	0x02000000	
SDRAM0_CFG1	0:31	0x00000000	

Table 0-2. DCR Contents After Reset (continued)

Register	Bits	Reset Value	Comment
SDRAM0_CID	0:31	0x320B0000	
SDRAM0_CLKTR	0:31	0x00000000	
SDRAM0_DEVOPT	0:31	0x00000000	
SDRAM0_DLYCAL	0:31	0x20000000	
SDRAM0_ECCESR	0:31	0x00000000	
SDRAM0_MCSTS	0:31	0x20000000	
SDRAM0_MIRQ	0:31	0x00000000	
SDRAM0_PMIT	0:31	0x07C00000	
SDRAM0_RID	0:31	0x00000000	
SDRAM0_RTR	0:31	0x07E00000	
SDRAM0_SLIO	0:31	0x00000000	
SDRAM0_TR0	0:31	0x00894012	
SDRAM0_TR1	0:31	0x40400000	
SDRAM0_UABBA	0:31	0x00000000	
SDRAM0_WDDCTR	0:31	0x00000000	
SRAM			
SRAM0_SB0CR	0:31	0x80000380	
SRAM0_SB1CR	0:31	0x80000380	
SRAM0_SB2CR	0:31	0x80000380	
SRAM0_SB3CR	0:31	0x80000380	
SRAM0_BEAR	0:31	0x00000000	
SRAM0_BESR0	0:31	0x00000000	
SRAM0_BESR1	0:31	0x00000000	
SRAM0_PMEG	0:31	0x01E00000	
SRAM0_DPC	0:31	0x00000000	
Universal Interrupt Controller 0 (UIC0)			
UIC0_CR	0:31	Undefined	
UIC0_ER	0:31	0x00000000	
UIC0_MSR	0:31	Undefined	
UIC0_PR	0:31	Undefined	
UIC0_SR	0:31	Undefined	
UIC0_TR	0:31	Undefined	

Table 0-2. DCR Contents After Reset (continued)

Register	Bits	Reset Value	Comment
UIC0_VCR	0:31	Undefined	
UIC0_VR	0:31	Undefined	
Universal Interrupt Controller 1 (UIC1)			
UIC1_CR	0:31	Undefined	
UIC1_ER	0:31	0x00000000	
UIC1_MSR	0:31	Undefined	
UIC1_PR	0:31	Undefined	
UIC1_SR	0:31	Undefined	
UIC1_TR	0:31	Undefined	
UIC1_VCR	0:31	Undefined	
UIC1_VR	0:31	Undefined	

~~0.2~~

~~0.3 MMIO Register Contents After Reset~~

~~MMIO registers are unaffected by core resets, and are generally identical for chip and system resets.~~

Table 0-3. MMIO Register Contents After Reset

Register	Bits	Reset Value	Comment
Ethernet MAC 0 (EMAC0)			
EMAC0_MR0	0:31	0xC0000000	
EMAC0_MR1	0:31	0x00000000	
EMAC0_TMR0	0:31	0x00000000	
EMAC0_TMR1	0:31	0x380F0000	
EMAC0_RMR	0:31	0x00000000	
EMAC0_ISR	0:31	0x00000000	
EMAC0_ISER	0:31	0x00000000	
EMAC0_IAHR	0:31	0x00000000	
EMAC0_IALR	0:31	0x00000000	
EMAC0_VTPID	0:31	0x00008808	
EMAC0_VTCI	0:31	0x00000000	

Table 0-3. MMIO Register Contents After Reset (continued)

Register	Bits	Reset Value	Comment
EMAC0_PTR	0:31	0x0000FFFF	
EMAC0_IAHT1	0:31	0x00000000	
EMAC0_IAHT2	0:31	0x00000000	
EMAC0_IAHT3	0:31	0x00000000	
EMAC0_IAHT4	0:31	0x00000000	
EMAC0_GAHT1	0:31	0x00000000	
EMAC0_GAHT2	0:31	0x00000000	
EMAC0_GAHT3	0:31	0x00000000	
EMAC0_GAHT4	0:31	0x00000000	
EMAC0_LSAH	0:31	0x00000000	
EMAC0_LSAL	0:31	0x00000000	
EMAC0_IPGVR	0:31	0x00000004	
EMAC0_STACR	0:31	0x00008000	
EMAC0_TRTR	0:31	0x00000000	
EMAC0_RWMR	0:31	0x04001000	
EMAC0_OCTX	0:31	0x00000000	
EMAC0_OCRX	0:31	0x00000000	
Ethernet MAC 1 (EMAC1)			
EMAC1_MR0	0:31	0xC0000000	
EMAC1_MR1	0:31	0x00000000	
EMAC1_TMR0	0:31	0x00000000	
EMAC1_TMR1	0:31	0x380F0000	
EMAC1_RMR	0:31	0x00000000	
EMAC1_ISR	0:31	0x00000000	
EMAC1_ISER	0:31	0x00000000	
EMAC1_IAHR	0:31	0x00000000	
EMAC1_IALR	0:31	0x00000000	
EMAC1_VTPID	0:31	0x00008808	
EMAC1_VTCI	0:31	0x00000000	
EMAC1_PTR	0:31	0x0000FFFF	
EMAC1_IAHT1	0:31	0x00000000	
EMAC1_IAHT2	0:31	0x00000000	

Table 0-3. MMIO Register Contents After Reset (continued)

Register	Bits	Reset Value	Comment
EMAC1_IAHT3	0:31	0x00000000	
EMAC1_IAHT4	0:31	0x00000000	
EMAC1_GAHT1	0:31	0x00000000	
EMAC1_GAHT2	0:31	0x00000000	
EMAC1_GAHT3	0:31	0x00000000	
EMAC1_GAHT4	0:31	0x00000000	
EMAC1_LSAH	0:31	0x00000000	
EMAC1_LSAL	0:31	0x00000000	
EMAC1_IPGVR	0:31	0x00000004	
EMAC1_STACR	0:31	0x00008000	
EMAC1_TRTR	0:31	0x00000000	
EMAC1_RWMR	0:31	0x04001000	
EMAC1_OCTX	0:31	0x00000000	
EMAC1_OCRX	0:31	0x00000000	
General Purpose I/O (GPIO)			
GPIO0_IR	0:31	Undefined	Read-only; follows the GPIO_In input.
GPIO0_ODR	0:31	0x00000000	
GPIO0_OR	0:31	0x00000000	
GPIO0_TCR	0:31	0x00000000	
Inter-Integrated Circuit 0 (IIC0)			
IIC0_CLKDIV	0:7	0x00	
IIC0_CNTL	0:7	0x00	
IIC0_DIRECTCNTL	0:7	0x0F	
IIC0_EXTSTS	0:7	0x60	
IIC0_HMADR	0:7	Undefined	
IIC0_HSADR	0:7	Undefined	
IIC0_INTRMSK	0:7	0x00	
IIC0_LMADR	0:7	Undefined	
IIC0_LSADR	0:7	Undefined	
IIC0_MDBUF	0:7	0x00	
IIC0_MDCNTL	0:7	0x00	
IIC0_SDBUF	0:7	0x00	

Table 0-3. MMIO Register Contents After Reset (continued)

Register	Bits	Reset Value	Comment
IIC0_STS	0:7	0x00	
IIC0_XFRCNT	0:7	0x00	
IIC0_XTCNTLSS	0:7	0x00	
Inter-Integrated Circuit 0 (IIC1)			
IIC1_CLKDIV	0:7	0x00	
IIC1_CNTL	0:7	0x00	
IIC1_DIRECTCNTL	0:7	0x0F	
IIC1_EXTSTS	0:7	0x60	
IIC1_HMADR	0:7	Undefined	
IIC1_HSADR	0:7	Undefined	
IIC1_INTRMSK	0:7	0x00	
IIC1_LMADR	0:7	Undefined	
IIC1_LSADR	0:7	Undefined	
IIC1_MDBUF	0:7	0x00	
IIC1_MDCNTL	0:7	0x00	
IIC1_SDBUF	0:7	0x00	
IIC1_STS	0:7	0x00	
IIC1_XFRCNT	0:7	0x00	
IIC1_XTCNTLSS	0:7	0x00	
OPB Arbiter			
OPBA0_CR	0:7	0x00	
OPBA0_PR	0:7	0x1B	
PCI-X			
PCIX0_ BAR0H	31:0	0x00000000	
PCIX0_ BAR0L	31:0		Determined by I_strap_pim0_prefetch and I_strap_pim0_enable
PCIX0_BAR1	31:0		Determined by I_strap_pim1_enable
PCIX0_ BAR2H	31:0	0x00000000	
PCIX0_ BAR2L	31:0		Determined by I_strap_pim2_prefetch and I_strap_pim2_enable
PCIX0_BIST	7:0	0x00	
PCIX0_BRDGOPT1	31:0	0x00000060	

Table 0-3. MMIO Register Contents After Reset (continued)

Register	Bits	Reset Value	Comment
PCIX0_BRDGOPT2	31:0		Determined by I_strap_host_config_en, I_strap_local_cpu_wait_en, O_strap_pcix_enable, and O_strap_pci_clk_freq
PCIX0_CACHELS	7:0	0x00	
PCIX0_CAP	7:0	0xC0	
PCIX0_CFGADDR	0:31	0x00000000	
PCIX0_CFGDATA	0:31	undefined	
PCIX0_CLS	23:0	0x06800	Determined by I_strap_pci_class
PCIX0_CMD	15:0	0x0000	
PCIX0_DEVID	15:0	0x01EF	Determined by I_strap_pci_device_id
PCIX0_EROMBA	31:0	0x00000000	
PCIX0_ERREN	31:0	0x00000100	
PCIX0_ERRSTS	31:0	0x00000000	
PCIX0_HDTYPE	7:0	0x00	
PCIX0_IM	31:0	0x00000000	
PCIX0_INTLN	7:0	0x00	
PCIX0_INTPN	7:0	0x01	
PCIX0_LATTIM	7:0	0x00	(PCI-CONV) 40h (PCI-X)
PCIX0_MAXLTNCY	7:0	0x00	
PCIX0_MINGNT	7:0	0x00	
PCIX0_MSGIH	31:0	0x00000000	
PCIX0_MSGIL	31:0	0x00000000	
PCIX0_MSGOH	31:0	0x00000000	
PCIX0_MSGOL	31:0	0x00000000	
PCIX0_OMCAPID	7:0	0x05	
PCIX0_OMMA	31:0	0x00000000	
PCIX0_OMMC	15:0	0x0082	
PCIX0_OMMDATA	15:0	0x0000	
PCIX0_OMMEOI	7:0	0x0000	
PCIX0_OMMUA	31:0	0x00000000	
PCIX0_OMNIPTR	7:0	0xD0	
PCIX0_PCIXCAPID	7:0	0x07	
PCIX0_PCIXCID	31:0	032B0000	

Table 0-3. MMIO Register Contents After Reset (continued)

Register	Bits	Reset Value	Comment
PCIX0_PCIXCMD	15:0	0x0030	
PCIX0_PCIXIDR	31:0	0x00000000	
PCIX0_PCIXNIPTR	7:0	0x00	
PCIX0_PCIXRID	31:0		version-specific
PCIX0_PCIXSTS	31:0	0x0583FFF8	
PCIX0_PIM0LAH	31:0	0x00000000	
PCIX0_PIM0LAL	31:0	0x00000000	
PCIX0_PIM0SA	31:0		Determined by I_strap_pim0_size, I_strap_pim0_prefetch, and I_strap_pim0_enable
PCIX0_PIM1LAH	31:0	0x00000000	
PCIX0_PIM1LAL	31:0	0x00000000	
PCIX0_PIM1SA	31:0		FFFF_FF00h or FFFF_FF01h determined by I_strap_pim1_enable
PCIX0_PIM2LAH	31:0	0x00000000	
PCIX0_PIM2LAL	31:0	0x00000000	
PCIX0_PIM2SA	31:0		Determined by value of I_strap_pim2_size, I_strap_pim2_prefetch, and I_strap_pim2_enable
PCIX0_PLBBEARH	31:0		Undefined
PCIX0_PLBBEARL	31:0		Undefined
PCIX0_PLBBESR	31:0		Undefined
PCIX0_PMC	15:0	0x0202	
PCIX0_PMCAPID	7:0	0x01	
PCIX0_PMCSR	15:0	0x0000	
PCIX0_PMCSRBSE	7:0	0x00	
PCIX0_PMDATA	7:0	0x00	
PCIX0_PMNIPTR	7:0	0xDC	
PCIX0_PMSCRR	7:0	0x10	
PCIX0_POM0LAH	31:0		Undefined
PCIX0_POM0LAL	31:0		Undefined
PCIX0_POM0MA	31:0	0xFF000000	
PCIX0_POM0PCIAH	31:0		Undefined
PCIX0_POM0PCIAL	31:0		Undefined
PCIX0_POM1LAH	31:0		Undefined

Table 0-3. MMIO Register Contents After Reset (continued)

Register	Bits	Reset Value	Comment
PCIX0_POM1LAL	31:0		Undefined
PCIX0_POM1MA	31:0	0x00000000	
PCIX0_POM1PCIAH	31:0		Undefined
PCIX0_POM1PCIAL	31:0		Undefined
PCIX0_POM2MA	31:0	0x00000000	
PCIX0_REVID	7:0	0x01	Determined by I_strap_pci_revision_id
PCIX0_SBSYSID	15:0	0x0000	Determined by I_strap_pci_subsys_id
PCIX0_SBSYSVID	15:0	0x0000	Determined by I_strap_pci_subsys_ven_id
PCIX0_STATUS	15:0	0x0230	
PCIX0_VENDID	15:0	0x1014	Determined by I_strap_pci_vendor_id
Serial Port (UART0)			
UART0_DLL	8:15	0x00	
UART0_DLM	0:7	0x00	
UART0_FCR	0:7	0x00	
UART0_IER	0:7	0x00	
UART0_IIR	0:7	0x01	
UART0_LCR	0:7	0x00	
UART0_LSR	0:7	0x60	
UART0_MCR	0:7	0x00	
UART0_MSR	0:7	0x00	$\overline{\text{UART0\_MSR}}_{0:3}$ are driven by $\overline{\text{UART0\_DCD}}$, $\overline{\text{UART0\_RI}}$, $\overline{\text{UART1\_DSR}}$[UART1_CTS], and $\overline{\text{UART1\_RTS}}$[UART1_DTR].
UART0_RBR	0:7	0x00	
UART0_SCR	0:7	Undefined	
UART0_THR	0:7	0x00	
Serial Port (UART1)			
UART1_DLL	8:15	0x00	
UART1_DLM	0:7	0x00	
UART1_FCR	0:7	0x00	
UART1_IER	0:7	0x00	
UART1_IIR	0:7	0x01	
UART1_LCR	0:7	0x00	
UART1_LSR	0:7	0x60	

Table 0-3. MMIO Register Contents After Reset (continued)

Register	Bits	Reset Value	Comment
UART1_MCR	0:7	0x00	
UART1_MSR	0:7	0x00	$\overline{UART0\_MSR}_{0:3}$ are driven by $\overline{UART0\_DCD}$, $\overline{UART0\_RI}$, $\overline{UART1\_DSR}$[UART1_CTS], and $\overline{UART1\_RTS}$[UART1_DTR].
UART1_RBR	0:7	0x00	
UART1_SCR	0:7	Undefined	
UART1_THR	0:7	0x00	
ZMII			
ZMII0_FER	0:31	undefined	
ZMII0_SSR	0:31	undefined	
ZMII0_SMIISR	0:31	undefined	

~~0.4~~

7.5 DCR Contents after Reset

DCR reset values are unaffected by core resets and are generally identical for chip and system resets.

Table 7-2. DCR Contents After Reset

Register	Bits	Reset Value	Comment
Chip Control			
CPC0_SYS0	0:31		Reset according to values received from serial ROM logic
CPC0_SYS1	0:31		Reset according to values received from serial ROM logic
CPC0_CUST0	0:31		Reset according to values received from serial ROM logic
CPC0_CUST1	0:31		Reset according to values received from serial ROM logic
CPC0_STRP0	0:31		Reset according to values received from serial ROM logic
CPC0_STRP1	0:31		Reset according to values received from serial ROM logic
CPC0_STRP2	0:31		Reset according to values received from serial ROM logic
CPC0_STRP3	0:31		Reset according to values received from serial ROM logic
CPC0_GPIO	0:31	0x00002000	
CPC0_PLB	0:31	0x000000F8	
CPC0_CR0	0:31	0x001E01B0	
CPC0_CR1	0:31	0xEAAEA012	
CPC0_MIRQ0	0:31	0x00000000	

Table 7-2. DCR Contents After Reset (continued)

Register	Bits	Reset Value	Comment
CPC0_MIRQ1	0:31	0x00000000	
CPC0_JTAGID	0:31		Refer to *PPC440GP Embedded Processor Data Sheet* for the value of this read-only register.
Clock and Power Management (CPM)			
CPC0_ER	0:31	0x00000000	$CPC0\_ER_{0:16}$ return 1, $CPC0\_ER_{17:31}$ return 0.
CPC0_FR	0:31	0x00000000	
CPC0_SR	0:31	0xFFFFFFFF	
Direct Memory Access (DMA)			
DMA0_CR0	0:31	0x00000000	
DMA0_CR1	0:31	0x00000000	
DMA0_CR2	0:31	0x00000000	
DMA0_CR3	0:31	0x00000000	
DMA0_CT0	0:31	0x00000000	
DMA0_CT1	0:31	0x00000000	
DMA0_CT2	0:31	0x00000000	
DMA0_CT3	0:31	0x00000000	
DMA0_DAH0	0:31	0x00000000	
DMA0_DAH1	0:31	0x00000000	
DMA0_DAH2	0:31	0x00000000	
DMA0_DAH3	0:31	0x00000000	
DMA0_DAL0	0:31	0x00000000	
DMA0_DAL1	0:31	0x00000000	
DMA0_DAL2	0:31	0x00000000	
DMA0_DAL3	0:31	0x00000000	
DMA0_POL	0:31	0x00000000	
DMA0_SAH0	0:31	0x00000000	
DMA0_SAH1	0:31	0x00000000	
DMA0_SAH2	0:31	0x00000000	
DMA0_SAH3	0:31	0x00000000	
DMA0_SAL0	0:31	0x00000000	
DMA0_SAL1	0:31	0x00000000	
DMA0_SAL2	0:31	0x00000000	
DMA0_SAL3	0:31	0x00000000	
DMA0_SGH0	0:31	0x00000000	
DMA0_SGH1	0:31	0x00000000	
DMA0_SGH2	0:31	0x00000000	

Table 7-2. DCR Contents After Reset (continued)

Register	Bits	Reset Value	Comment
DMA0_SGH3	0:31	0x00000000	
DMA0_SGL0	0:31	0x00000000	
DMA0_SGL1	0:31	0x00000000	
DMA0_SGL2	0:31	0x00000000	
DMA0_SGL3	0:31	0x00000000	
DMA0_SGC	0:31	0x00000000	
DMA0_SLP	0:31	0x07C00000	
DMA0_SR	0:31	0x00000000	
External Bus Controller 0 (EBC0)			
EBC0_B0AP	0:31	0x7F8FFE80	Slowest possible bus timings.
EBC0_B0CR	0:31	0xFFE28000	2MB read-only bank.
EBC0_B1AP	0:31	0x00000000	
EBC0_B1CR	0:31	0x00000000	
EBC0_B2AP	0:31	0x00000000	
EBC0_B2CR	0:31	0x00000000	
EBC0_B3AP	0:31	0x00000000	
EBC0_B3CR	0:31	0x00000000	
EBC0_B4AP	0:31	0x00000000	
EBC0_B4CR	0:31	0x00000000	
EBC0_B5AP	0:31	0x00000000	
EBC0_B5CR	0:31	0x00000000	
EBC0_B6AP	0:31	0x00000000	
EBC0_B6CR	0:31	0x00000000	
EBC0_B7AP	0:31	0x00000000	
EBC0_B7CR	0:31	0x00000000	
EBC0_BEAR	0:31	0x00000000	
EBC0_BESR	0:31	0x00000000	
EBC0_CFG	0:31	0x07C00000	
EBC0_CID	0:31	0x00000000	
External Bus Master Interface 0 (EBM0)			
EBM0_BEAR	0:31	0x00000000	
EBM0_BEMR	0:31	0x00000000	
EBM0_BESR	0:31	0x00000000	
EBM0_CID	0:31	undefined	
EBM0_CTL	0:31	0x88400000	
EBM0_FAIR	0:31	0xFFF00000	

Table 7-2. DCR Contents After Reset (continued)

Register	Bits	Reset Value	Comment
EBM0_LCNT	0:31	0x00000000	
EBM0_MISCSTS	0:31	undefined	
EBM0_SLPMD	0:31	0x07C00000	
EBM0_UAM	0:31	0x00000000	
EBM0_UAR	0:31	0x00000000	
Indirect Addressing Registers			
EBM0_CFGADDR	0:31	undefined	
EBM0_CFGDATA	0:31	undefined	
PPM0_CFGADDR	0:31	undefined	
PPM0_CFGDATA	0:31	undefined	
SDRAM0_CFGADDR	0:31	undefined	
SDRAM0_CFGDATA	0:31	undefined	
Memory Access Layer (MAL)			
MAL0_CFG	0:31	0x00004080	
MAL0_ESR	0:31	0x00000000	
MAL0_IER	0:31	0x00000000	
MAL0_RXBADDR	0:31	0x00000000	
MAL0_RCBS0	0:31	Undefined	
MAL0_RCBS1	0:31	Undefined	
MAL0_RXCARR	0:31	0x00000000	
MAL0_RXCASR	0:31	0x00000000	
MAL0_RXCTP0R	0:31	Undefined	
MAL0_RXCTP1R	0:31	Undefined	
MAL0_RXDEIR	0:31	0x00000000	
MAL0_RXEOBISR	0:31	0x00000000	
MAL0_RXTATTRR	0:31	0x00000000	
MAL0_TXBADDR	0:31	0x00000000	
MAL0_TXCARR	0:31	0x00000000	
MAL0_TXCASR	0:31	0x00000000	
MAL0_TXCTP0R	0:31	Undefined	
MAL0_TXCTP1R	0:31	Undefined	
MAL0_TXCTP2R	0:31	Undefined	
MAL0_TXCTP3R	0:31	Undefined	
MAL0_TXDEIR	0:31	0x00000000	
MAL0_TXEOBISR	0:31	0x00000000	
MAL0_TXTATTRR	0:31	0x00000000	

Table 7-2. DCR Contents After Reset (continued)

Register	Bits	Reset Value	Comment
On-Chip Buses			
PLB0_ACR	0:31	0x00000000	
PLB0_BEARH	0:31	Undefined	
PLB0_BEARL	0:31	Undefined	
PLB0_BESR	0:31	0x00000000	
PLB0_REVID	0:31	Undefined	
POB0_BEARH	0:31	Undefined	
POB0_BEARL	0:31	Undefined	
POB0_BESR0	0:31	0x00000000	
POB0_BESR1	0:31	0x00000000	
POB0_CONFG	0:31	0x00000000	
POB0_LATENCY	0:31	0x00000000	
POB0_REVID	0:31	0x00000000	
PLB Performance Monitor			
PPM0_ISR	0:31	0x00000000	
PPM0_CR	0:31	0x00000000	
PPM0_CCR	0:31	0x00000000	
PPM0_UAR	0:31	0x00000000	
PPM0_LAR	0:31	0x00000000	
PPM0_UAMR	0:31	0x00000000	
PPM0_LAMR	0:31	0x00000000	
PPM0_RIDR	0:31	0x00000000	
PPM0_MCSR0	0:31	0x00000000	
PPM0_MCSR1	0:31	0x00000000	
PPM0_MCSR2	0:31	0x00000000	
PPM0_MCSR3	0:31	0x00000000	
PPM0_SCSR0	0:31	0x00000000	
PPM0_SCSR1	0:31	0x00000000	
PPM0_SCSR2	0:31	0x00000000	
PPM0_SCSR3	0:31	0x00000000	
PPM0_GCSR0	0:31	0x00000000	
PPM0_GCSR1	0:31	0x00000000	
PPM0_GCSR2	0:31	0x00000000	
PPM0_GCSR3	0:31	0x00000000	
PPM0_MCR0	0:31	0x00000000	
PPM0_MCR1	0:31	0x00000000	

Table 7-2. DCR Contents After Reset (continued)

Register	Bits	Reset Value	Comment
PPM0_MCR2	0:31	0x00000000	
PPM0_MCR3	0:31	0x00000000	
PPM0_SCR0	0:31	0x00000000	
PPM0_SCR1	0:31	0x00000000	
PPM0_SCR2	0:31	0x00000000	
PPM0_SCR3	0:31	0x00000000	
PPM0_GCR0	0:31	0x00000000	
PPM0_GCR1	0:31	0x00000000	
PPM0_GCR2	0:31	0x00000000	
PPM0_GCR3	0:31	0x00000000	
PPM0_DCSR0	0:31	0x00000000	
PPM0_DCSR1	0:31	0x00000000	
PPM0_DCMXR0	0:31	0x00000000	
PPM0_DCMXR1	0:31	0x00000000	
PPM0_DCMNR0	0:31	0x00000000	
PPM0_DCMNR1	0:31	0x00000000	
PPM0_DCTVR0	0:31	0x00000000	
PPM0_DCTVR1	0:31	0x00000000	
PPM0_DCOTR0	0:31	0x00000000	
PPM0_DCOTR1	0:31	0x00000000	
DDR_SDRAM Controller			
SDRAM0_B0CR	0:31	0x00000000	
SDRAM0_B1CR	0:31	0x00000000	
SDRAM0_B2CR	0:31	0x00000000	
SDRAM0_B3CR	0:31	0x00000000	
SDRAM0_BEAR	0:31	0x00000000	
SDRAM0_BESR0	0:31	0x00000000	
SDRAM0_BESR1	0:31	0x00000000	
SDRAM0_CFG0	0:31	0x02000000	
SDRAM0_CFG1	0:31	0x00000000	
SDRAM0_CID	0:31	0x320B0000	
SDRAM0_CLKTR	0:31	0x00000000	
SDRAM0_DEVOPT	0:31	0x00000000	
SDRAM0_DLYCAL	0:31	0x20000000	
SDRAM0_ECCESR	0:31	0x00000000	
SDRAM0_MCSTS	0:31	0x20000000	

Table 7-2. DCR Contents After Reset (continued)

Register	Bits	Reset Value	Comment
SDRAM0_MIRQ	0:31	0x00000000	
SDRAM0_PMIT	0:31	0x07C00000	
SDRAM0_RID	0:31	0x00000000	
SDRAM0_RTR	0:31	0x07E00000	
SDRAM0_SLIO	0:31	0x00000000	
SDRAM0_TR0	0:31	0x00894012	
SDRAM0_TR1	0:31	0x40400000	
SDRAM0_UABBA	0:31	0x00000000	
SDRAM0_WDDCTR	0:31	0x00000000	
SRAM			
SRAM0_SB0CR	0:31	0x80000380	
SRAM0_SB1CR	0:31	0x80000380	
SRAM0_SB2CR	0:31	0x80000380	
SRAM0_SB3CR	0:31	0x80000380	
SRAM0_BEAR	0:31	0x00000000	
SRAM0_BESR0	0:31	0x00000000	
SRAM0_BESR1	0:31	0x00000000	
SRAM0_PMEG	0:31	0x01E00000	
SRAM0_DPC	0:31	0x00000000	
Universal Interrupt Controller 0 (UIC0)			
UIC0_CR	0:31	Undefined	
UIC0_ER	0:31	0x00000000	
UIC0_MSR	0:31	Undefined	
UIC0_PR	0:31	Undefined	
UIC0_SR	0:31	Undefined	
UIC0_TR	0:31	Undefined	
UIC0_VCR	0:31	Undefined	
UIC0_VR	0:31	Undefined	
Universal Interrupt Controller 1 (UIC1)			
UIC1_CR	0:31	Undefined	
UIC1_ER	0:31	0x00000000	
UIC1_MSR	0:31	Undefined	
UIC1_PR	0:31	Undefined	
UIC1_SR	0:31	Undefined	
UIC1_TR	0:31	Undefined	
UIC1_VCR	0:31	Undefined	

Table 7-2. DCR Contents After Reset (continued)

Register	Bits	Reset Value	Comment
UIC1_VR	0:31	Undefined	

7.6 MMIO Register Contents After Reset

MMIO registers are unaffected by core resets, and are generally identical for chip and system resets.

Table 7-3. MMIO Register Contents After Reset

Register	Bits	Reset Value	Comment
Ethernet MAC 0 (EMAC0)			
EMAC0_MR0	0:31	0xC0000000	
EMAC0_MR1	0:31	0x00000000	
EMAC0_TMR0	0:31	0x00000000	
EMAC0_TMR1	0:31	0x380F0000	
EMAC0_RMR	0:31	0x00000000	
EMAC0_ISR	0:31	0x00000000	
EMAC0_ISER	0:31	0x00000000	
EMAC0_IAHR	0:31	0x00000000	
EMAC0_IALR	0:31	0x00000000	
EMAC0_VTPID	0:31	0x00008808	
EMAC0_VTCI	0:31	0x00000000	
EMAC0_PTR	0:31	0x0000FFFF	
EMAC0_IAHT1	0:31	0x00000000	
EMAC0_IAHT2	0:31	0x00000000	
EMAC0_IAHT3	0:31	0x00000000	
EMAC0_IAHT4	0:31	0x00000000	
EMAC0_GAHT1	0:31	0x00000000	
EMAC0_GAHT2	0:31	0x00000000	
EMAC0_GAHT3	0:31	0x00000000	
EMAC0_GAHT4	0:31	0x00000000	
EMAC0_LSAH	0:31	0x00000000	
EMAC0_LSAL	0:31	0x00000000	
EMAC0_IPGVR	0:31	0x00000004	
EMAC0_STACR	0:31	0x00008000	
EMAC0_TRTR	0:31	0x00000000	
EMAC0_RWMR	0:31	0x04001000	
EMAC0_OCTX	0:31	0x00000000	

Table 7-3. MMIO Register Contents After Reset (continued)

Register	Bits	Reset Value	Comment
EMAC0_OCRX	0:31	0x00000000	
Ethernet MAC 1 (EMAC1)			
EMAC1_MR0	0:31	0xC0000000	
EMAC1_MR1	0:31	0x00000000	
EMAC1_TMR0	0:31	0x00000000	
EMAC1_TMR1	0:31	0x380F0000	
EMAC1_RMR	0:31	0x00000000	
EMAC1_ISR	0:31	0x00000000	
EMAC1_ISER	0:31	0x00000000	
EMAC1_IAHR	0:31	0x00000000	
EMAC1_IALR	0:31	0x00000000	
EMAC1_VTPID	0:31	0x00008808	
EMAC1_VTCI	0:31	0x00000000	
EMAC1_PTR	0:31	0x0000FFFF	
EMAC1_IAHT1	0:31	0x00000000	
EMAC1_IAHT2	0:31	0x00000000	
EMAC1_IAHT3	0:31	0x00000000	
EMAC1_IAHT4	0:31	0x00000000	
EMAC1_GAHT1	0:31	0x00000000	
EMAC1_GAHT2	0:31	0x00000000	
EMAC1_GAHT3	0:31	0x00000000	
EMAC1_GAHT4	0:31	0x00000000	
EMAC1_LSAH	0:31	0x00000000	
EMAC1_LSAL	0:31	0x00000000	
EMAC1_IPGVR	0:31	0x00000004	
EMAC1_STACR	0:31	0x00008000	
EMAC1_TRTR	0:31	0x00000000	
EMAC1_RWMR	0:31	0x04001000	
EMAC1_OCTX	0:31	0x00000000	
EMAC1_OCRX	0:31	0x00000000	

Table 7-3. MMIO Register Contents After Reset (continued)

Register	Bits	Reset Value	Comment
General Purpose I/O (GPIO)			
GPIO0_IR	0:31	Undefined	Read-only; follows the GPIO_In input.
GPIO0_ODR	0:31	0x00000000	
GPIO0_OR	0:31	0x00000000	
GPIO0_TCR	0:31	0x00000000	
Inter-Integrated Circuit 0 (IIC0)			
IIC0_CLKDIV	0:7	0x00	
IIC0_CNTL	0:7	0x00	
IIC0_DIRECTCNTL	0:7	0x0F	
IIC0_EXTSTS	0:7	0x60	
IIC0_HMADR	0:7	Undefined	
IIC0_HSADR	0:7	Undefined	
IIC0_INTRMSK	0:7	0x00	
IIC0_LMADR	0:7	Undefined	
IIC0_LSADR	0:7	Undefined	
IIC0_MDBUF	0:7	0x00	
IIC0_MDCNTL	0:7	0x00	
IIC0_SDBUF	0:7	0x00	
IIC0_STS	0:7	0x00	
IIC0_XFRCNT	0:7	0x00	
IIC0_XTCNTLSS	0:7	0x00	
Inter-Integrated Circuit 0 (IIC1)			
IIC1_CLKDIV	0:7	0x00	
IIC1_CNTL	0:7	0x00	
IIC1_DIRECTCNTL	0:7	0x0F	
IIC1_EXTSTS	0:7	0x60	
IIC1_HMADR	0:7	Undefined	
IIC1_HSADR	0:7	Undefined	
IIC1_INTRMSK	0:7	0x00	
IIC1_LMADR	0:7	Undefined	
IIC1_LSADR	0:7	Undefined	
IIC1_MDBUF	0:7	0x00	
IIC1_MDCNTL	0:7	0x00	
IIC1_SDBUF	0:7	0x00	
IIC1_STS	0:7	0x00	
IIC1_XFRCNT	0:7	0x00	

Table 7-3. MMIO Register Contents After Reset (continued)

Register	Bits	Reset Value	Comment
IIC1_XTCNTLSS	0:7	0x00	
OPB Arbiter			
OPBA0_CR	0:7	0x00	
OPBA0_PR	0:7	0x1B	
PCI-X			
PCIX0_ BAR0H	31:0	0x00000000	
PCIX0_ BAR0L	31:0		Determined by I_strap_pim0_prefetch and I_strap_pim0_enable
PCIX0_BAR1	31:0		Determined by I_strap_pim1_enable
PCIX0_ BAR2H	31:0	0x00000000	
PCIX0_ BAR2L	31:0		Determined by I_strap_pim2_prefetch and I_strap_pim2_enable
PCIX0_BIST	7:0	0x00	
PCIX0_BRDGOPT1	31:0	0x00000060	
PCIX0_BRDGOPT2	31:0		Determined by I_strap_host_config_en, I_strap_local_cpu_wait_en, O_strap_pcix_enable, and O_strap_pci_clk_freq
PCIX0_CACHELS	7:0	0x00	
PCIX0_CAP	7:0	0xC0	
PCIX0_CFGADDR	0:31	0x00000000	
PCIX0_CFGDATA	0:31	undefined	
PCIX0_CLS	23:0	0x06800	Determined by I_strap_pci_class
PCIX0_CMD	15:0	0x0000	
PCIX0_DEVID	15:0	0x01EF	Determined by I_strap_pci_device_id
PCIX0_EROMBA	31:0	0x00000000	
PCIX0_ERREN	31:0	0x00000100	
PCIX0_ERRSTS	31:0	0x00000000	
PCIX0_HDTYPE	7:0	0x00	
PCIX0_IM	31:0	0x00000000	
PCIX0_INTLN	7:0	0x00	
PCIX0_INTPN	7:0	0x01	
PCIX0_LATTIM	7:0	0x00	(PCI-CONV) 40h (PCI-X)
PCIX0_MAXLTNCY	7:0	0x00	
PCIX0_MINGNT	7:0	0x00	
PCIX0_MSGIH	31:0	0x00000000	
PCIX0_MSGIL	31:0	0x00000000	
PCIX0_MSGOH	31:0	0x00000000	
PCIX0_MSGOL	31:0	0x00000000	
PCIX0_OMCAPID	7:0	0x05	

Table 7-3. MMIO Register Contents After Reset (continued)

Register	Bits	Reset Value	Comment
PCIX0_OMMA	31:0	0x00000000	
PCIX0_OMMC	15:0	0x0082	
PCIX0_OMMDATA	15:0	0x0000	
PCIX0_OMMEOI	7:0	0x0000	
PCIX0_OMMUA	31:0	0x00000000	
PCIX0_OMNIPTR	7:0	0xD0	
PCIX0_PCIXCAPID	7:0	0x07	
PCIX0_PCIXCID	31:0	032B0000	
PCIX0_PCIXCMD	15:0	0x0030	
PCIX0_PCIXIDR	31:0	0x00000000	
PCIX0_PCIXNIPTR	7:0	0x00	
PCIX0_PCIXRID	31:0		version-specific
PCIX0_PCIXSTS	31:0	0x0583FFF8	
PCIX0_PIM0LAH	31:0	0x00000000	
PCIX0_PIM0LAL	31:0	0x00000000	
PCIX0_PIM0SA	31:0		Determined by I_strap_pim0_size, I_strap_pim0_prefetch, and I_strap_pim0_enable
PCIX0_PIM1LAH	31:0	0x00000000	
PCIX0_PIM1LAL	31:0	0x00000000	
PCIX0_PIM1SA	31:0		FFFF_FF00h or FFFF_FF01h determined by I_strap_pim1_enable
PCIX0_PIM2LAH	31:0	0x00000000	
PCIX0_PIM2LAL	31:0	0x00000000	
PCIX0_PIM2SA	31:0		Determined by value of I_strap_pim2_size, I_strap_pim2_prefetch, and I_strap_pim2_enable
PCIX0_PLBBEARH	31:0		Undefined
PCIX0_PLBBEARL	31:0		Undefined
PCIX0_PLBBESR	31:0		Undefined
PCIX0_PMC	15:0	0x0202	
PCIX0_PMCAPID	7:0	0x01	
PCIX0_PMCSR	15:0	0x0000	
PCIX0_PMCSRBSE	7:0	0x00	
PCIX0_PMDATA	7:0	0x00	
PCIX0_PMNIPTR	7:0	0xDC	
PCIX0_PMSCRR	7:0	0x10	
PCIX0_POM0LAH	31:0		Undefined
PCIX0_POM0LAL	31:0		Undefined
PCIX0_POM0MA	31:0	0xFF000000	

Table 7-3. MMIO Register Contents After Reset (continued)

Register	Bits	Reset Value	Comment
PCIX0_POM0PCIAH	31:0		Undefined
PCIX0_POM0PCIAL	31:0		Undefined
PCIX0_POM1LAH	31:0		Undefined
PCIX0_POM1LAL	31:0		Undefined
PCIX0_POM1MA	31:0	0x00000000	
PCIX0_POM1PCIAH	31:0		Undefined
PCIX0_POM1PCIAL	31:0		Undefined
PCIX0_POM2MA	31:0	0x00000000	
PCIX0_REVID	7:0	0x01	Determined by I_strap_pci_revision_id
PCIX0_SBSYSID	15:0	0x0000	Determined by I_strap_pci_subsys_id
PCIX0_SBSYSVID	15:0	0x0000	Determined by I_strap_pci_subsys_ven_id
PCIX0_STATUS	15:0	0x0230	
PCIX0_VENDID	15:0	0x1014	Determined by I_strap_pci_vendor_id
Serial Port (UART0)			
UART0_DLL	8:15	0x00	
UART0_DLM	0:7	0x00	
UART0_FCR	0:7	0x00	
UART0_IER	0:7	0x00	
UART0_IIR	0:7	0x01	
UART0_LCR	0:7	0x00	
UART0_LSR	0:7	0x60	
UART0_MCR	0:7	0x00	
UART0_MSR	0:7	0x00	UART0_MSR$_{0:3}$ are driven by $\overline{\text{UART0\_DCD}}$, $\overline{\text{UART0\_RI}}$, $\overline{\text{UART1\_DSR}}$[$\overline{\text{UART1\_CTS}}$], and $\overline{\text{UART1\_RTS}}$[$\overline{\text{UART1\_DTR}}$].
UART0_RBR	0:7	0x00	
UART0_SCR	0:7	Undefined	
UART0_THR	0:7	0x00	
Serial Port (UART1)			
UART1_DLL	8:15	0x00	
UART1_DLM	0:7	0x00	
UART1_FCR	0:7	0x00	
UART1_IER	0:7	0x00	
UART1_IIR	0:7	0x01	
UART1_LCR	0:7	0x00	
UART1_LSR	0:7	0x60	
UART1_MCR	0:7	0x00	

Table 7-3. MMIO Register Contents After Reset (continued)

Register	Bits	Reset Value	Comment
UART1_MSR	0:7	0x00	UART0_MSR$_{0:3}$ are driven by $\overline{\text{UART0\_DCD}}$, $\overline{\text{UART0\_RI}}$, UART1_DSR[UART1_CTS], and UART1_RTS[UART1_DTR].
UART1_RBR	0:7	0x00	
UART1_SCR	0:7	Undefined	
UART1_THR	0:7	0x00	
ZMII			
ZMII0_FER	0:31	undefined	
ZMII0_SSR	0:31	undefined	
ZMII0_SMIISR	0:31	undefined	

7.7 Initialization Software Requirements

After a reset operation occurs, the PPC440GP is initialized to a minimum configuration to enable the fetching and execution of the software initialization code, and to guarantee deterministic behavior of the core during the execution of this code. Initialization software is necessary to complete the configuration of the processor core and the rest of the on-chip and off-chip system.

The system must provide non-volatile memory (or memory initialized by some mechanism other than the PPC440GP) at the real address corresponding to effective address 0xFFFFFFFC, and at the rest of the initial program memory page. The instruction at the initial address must be an unconditional branch backwards to the beginning of the initialization software sequence.

The initialization software functions described in this section perform the configuration tasks required to prepare the PPC440GP to boot an operating system and subsequently execute an application program.

The initialization software must also perform functions associated with hardware resources that are outside the PPC440GP. This section makes reference to some of these functions, but their full scope is described in the chapters describing peripheral operations.

Initialization software should perform the following tasks in order to fully configure the PPC440GP. For more information on the various functions referenced in the initialization sequence, see the corresponding chapters of this document.

1. Branch backwards from effective address 0xFFFFFFFC to the start of the initialization sequence
2. Invalidate the instruction cache (**iccci**)
3. Invalidate the data cache (**dccci**)
4. Synchronize memory accesses (**msync**)

   This step forces any data PLB operations that may have been in progress prior to the reset operation to complete, thereby allowing subsequent data accesses to be initiated and completed properly.
5. Clear DBCR0 register (disable all debug events)

Although the PPC440GP is defined to reset some of the debug event enables during the reset operation (as specified in ~~Table 7-1 on page 7-4~~*Table 7-1* on page 262), this is not required by the architecture and hence the initialization software should not assume this behavior. Software should disable all debug events in order to prevent non-deterministic behavior on the trace interface to the core.

6. Clear DBSR register (initialize all debug event status)

   Although the PPC440GP is defined to reset the DBSR debug event status bits during the reset operation (as specified in ~~Table 7-1 on page 7-4~~*Table 7-1* on page 262), this is not required by the architecture and hence the initialization software should not assume this behavior. Software should clear all such status in order to prevent non-deterministic behavior on the JTAG interface to the core.

7. Initialize CCR0 register
   1. Enable/disable broadcast of instructions to auxiliary processor (save power if no AP attached)
   2. Enable/disable broadcast of trace information (save power if not tracing)
   3. Enable/configure or disable speculative instruction cache line prefetching
   4. Specify behavior for **icbt** and **dcbt**/**dcbtst** instructions
   5. Enable/disable gathering of separate store accesses
   6. Enable/disable hardware support for misaligned data accesses
8. Configure instruction and data cache regions

   These steps must be performed prior to enabling the caches by setting the caching inhibited storage attribute of the corresponding TLB entry to 0.
   1. Clear the instruction and data cache normal victim index registers (INV0–INV3, DNV0–DNV3)
   2. Clear the instruction and data cache transient victim index registers (ITV0–ITV3, DTV0–DTV3)
   3. Set the instruction and data cache victim limit registers (IVLIM and DVLIM) according to the desired size of the normal, locked, and transient regions of each cache
9. Setup TLB entry to cover initial program memory page

   Since the PPC440GP only initializes an architecturally-invisible shadow TLB entry during the reset operation, and since all shadow TLB entries are invalidated upon any context synchronization, special care must be taken during the initialization sequence to prevent any such context synchronizing operations (such as interrupts and the **isync** instruction) until after this step is completed, and an architected TLB entry has been established in the TLB. Particular care should be taken to avoid store operations, since write permission is disabled upon reset, and an attempt to execute any store operation would result in a Data Storage interrupt, thereby invalidating the shadow TLB entry.
   1. Initialize MMUCR
   - Specify TID field to be written to TLB entries
   - Specify TS field to be used for TLB searches
   - Specify store miss allocation behavior
   - Enable/disable transient cache mechanism
   - Enable/disable cache locking exceptions
   2. Write TLB entry for initial program memory page
   - Specify EPN, RPN, ERPN, and SIZE as appropriate for system
   - Set valid bit

- Specify TID = 0 (disable comparison to PID) or else initialize PID register to matching value
- Specify TS = 0 (system address space) or else MSR[IS,DS] must be set to correspond to TS=1
- Specify storage attributes (W, I, M, G, E, U0–U3) as appropriate for system
- Enable supervisor mode fetch, read, and write access (SX, SR, SW)

3. Initialize PID register to match TID field of TLB entry (unless using TID = 0)
4. Setup for subsequent MSR[IS,DS] initialization to correspond to TS field of TLB entry

   Only necessary if TS field of TLB entry being set to 1 (MSR[IS,DS] already reset to 0)

- Write new MSR value into SRR1
- Write address from which to continue execution into SRR0

5. Setup for subsequent change in instruction fetch address

   Only necessary if EPN field of TLB entry changed from the initial value ($EPN_{0:19} \neq$ 0xFFFFF)

- Write initial/new MSR value into SRR1
- Write address from which to continue execution into SRR0

6. Initialize or invalidate all other TLB entries as desired
7. Context synchronize to invalidate shadow TLB contents and cause new TLB contents to take effect

- Use **isync** if not changing MSR contents and not changing the effective address of the rest of the initialization sequence
- Use **rfi** if changing MSR to match new TS field of TLB entry (SRR1 will be copied into MSR, and program execution will resume at value in SRR0)
- Use **rfi** if changing next instruction fetch address to correspond to new EPN field of TLB entry (SRR1 will be copied into MSR, and program execution will resume at value in SRR0)

  Instruction and data caches will now begin to be used, if the corresponding TLB entry has been setup with the caching inhibited storage attribute set to 0. Initialization software can now branch outside of the initial 4KB memory region as controlled by the address and size of the new TLB entry and/or any other TLB entries which have been setup.

10. Initialize interrupt resources

1. Initialize IVPR to specify high-order address of the interrupt handling routines

   Make sure that the corresponding address region is covered by a TLB entry (or entries)

2. Initialize IVOR0–IVOR15 registers (individual interrupt vector addresses)

   Make sure that the corresponding addresses are covered by a TLB entry (or entries)

   Because the low order four bits of IVOR0–IVOR15 are reserved, the values written to those bits are ignored when the registers are written, and are read as zero when the registers are used. Therefore, all interrupt vector offsets are implicitly aligned on quadword boundaries. Software must take care to assure that all interrupt handlers are quadword-aligned.

3. Setup corresponding memory contents with the interrupt handling routines
4. Synchronize any program memory changes as required. (See ~~"Self-Modifying Code" on page 5-11~~*Self-Modifying Code* on page 214 for more information on the instruction sequence necessary to synchronize changes to program memory prior to executing the new instructions.)

11. Configure debug facilities as desired
    1. Write DBCR1 and DBCR2 to specify IAC and DAC event conditions
    2. Clear DBSR to initialize IAC auto-toggle status
    3. Initialize IAC1–IAC4, DAC1–DAC2, DVC1–DVC2 registers to desired values
    4. Write MSR[DWE] to enable Debug Wait mode (if desired)
    5. Write DBCR0 to enable desired debug mode(s) and event(s)
    6. Context synchronize to establish new debug facility context (**isync**)
12. Configure timer facilities as desired
    1. Write DEC to 0 to prevent Decrementer exception after TSR is cleared
    2. Write TBL to 0 to prevent Fixed Interval Timer and Watchdog Timer exceptions after TSR is cleared, and to prevent increment into TBH prior to full initialization
    3.
    4. Clear TSR to clear all timer exception status
    5. Write TCR to configure and enable timers as desired

       Software must take care with respect to the enabling of the Watchdog Timer reset function, as once this function is enabled, it cannot be disabled except by reset itself
    6. Initialize TBH value as desired
    7. Initialize TBL value as desired
    8. Initialize DECAR to desired value (if enabling the auto-reload function)
    9. Initialize DEC to desired value
13. Initialize facilities outside the processor core which are possible sources of asynchronous interrupt requests (including DCRs and/or other memory-mapped resources)

    This must be done prior to enabling asynchronous interrupts in the MSR
14. Initialize the MSR to enable interrupts as desired
    1. Set MSR[CE] to enable/disable Critical Input and Watchdog Timer interrupts
    2. Set MSR[EE] to enable/disable External Input, Decrementer, and Fixed Interval Timer interrupts
    3. Set MSR[DE] to enable/disable Debug interrupts
    4. Set MSR[ME] to enable/disable Machine Check interrupts

       Software should first check the status of the ESR[MCI] field to determine whether any Instruction Machine Check exceptions have occurred after this field was cleared by reset and before Machine Check interrupts were enabled (by this step). Any such exceptions would have set ESR[MCI] to 1, and this status can only be cleared explicitly by software. Once MSR[ME] has been set to 1, ~~and~~ subsequent Machine Check exceptions will result in a Machine Check interrupt.
    5. Context synchronize to establish new MSR context (**isync**)
15. Initialize any other processor core resources as required by the system (GPRs, SPRGs, and so on)
16. Initialize any other facilities outside the processor core as required by the system
17. Initialize system memory as required by the system software

Synchronize any program memory changes as required. (See ~~"Self-Modifying Code" on page 5-11~~*Self-Modifying Code* on page 214 for more information on the instruction sequence necessary to synchronize changes to program memory prior to executing the new instructions)

18. Start the system software

    System software is generally responsible for initializing and/or managing the rest of the MSR fields, including:

    1. MSR[FP] to enable or disable the execution of floating-point instructions
    2. MSR[FE0,FE1] to enable/disable Floating-Point Enabled exception type Program interrupts
    3. MSR[PR] to specify user mode or supervisor mode
    4. MSR[IS,DS] to specify application address space or system address space for instructions and data
    5. MSR[WE] to place the processor into Wait State (halt execution pending an interrupt)

8. IIC Bootstrap Controller

The IIC bootstrap controller provides a flexible alternative to traditional strapping pins as a solution for reset configuration.

Immediately following System Reset, the IIC bootstrap controller, if enabled, sequentially reads 16 bytes from an IIC slave device accessible from the IIC0 interface. The first 8 bytes of data read is configuration data necessary for initializing the PLL settings, clock ratios, boot location and boot width. The second 8 bytes is user definable configuration data. If the bootstrap controller is not enabled, a set of default strap settings are used.

Figure 8-1 illustrates how the IIC bootstrap controller shares the IIC0 interface with the IIC0 controller and connects to the IIC slave device. The serial ROM is the IIC slave device containing the bootstrap settings.

The switch in Figure 8-1 shows two positions, A and B. Immediately after the deassertion of $\overline{\text{SysReset}}$, the switch is in position A giving the IIC bootstrap controller ownership of the IIC0 interface. The switch remains in position A until the IIC bootstrap controller relinquishes control to the IIC0 controller.

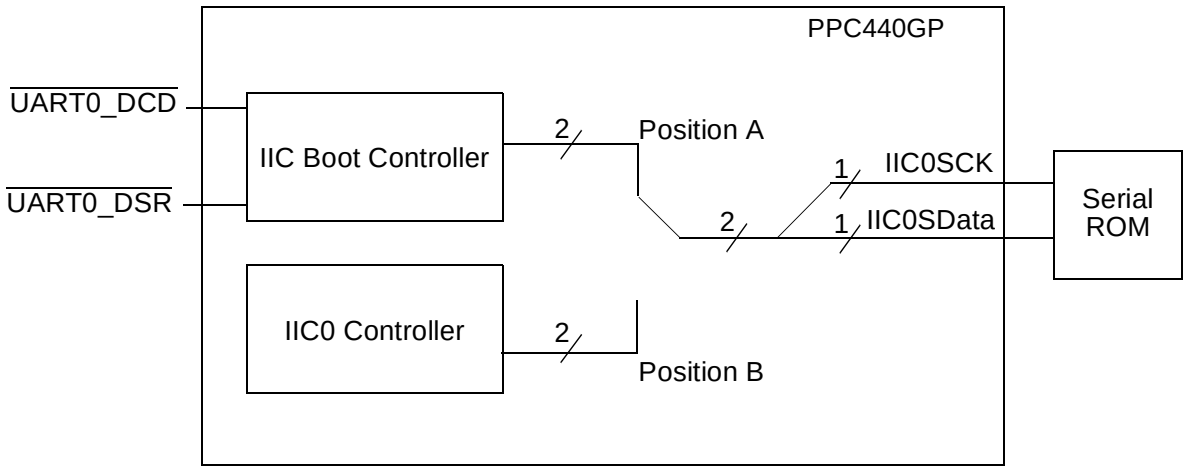


Figure 8-1. IIC Bootstrap Controller Process

8.1 IIC Bootstrap Configuration

The two bootstrap signals, UART0DSR and UART0DCD, shown in Figure 8-1 enable the IIC bootstrap controller and select the IIC address of the serial ROM. These signals are sampled while SysReset is asserted. Table 8-1 describes the configuration of the IIC bootstrap controller as determined by the two bootstrap signals.

UART0DCD, if pulled high during reset, enables the IIC bootstrap controller to read the serial ROM on the IIC0 interface. If UART0DCD is pulled low, the IIC bootstrap controller does not generate any activity on the IIC0 interface and the default strap settings in Table 8-2 are used.

Two possible IIC slave addresses are available for the serial ROM. UART0DSR selects 0xA8 as the address if pulled low during reset and 0xA0 if pulled high during reset.

Table 8-1. IIC Bootstrap Configuration

Bootstrap Signals	IIC Bootstrap Controller Configuration
UART0_DCD	Enables the IIC0 Boot Controller 0 IIC Bootstrap Controller Disabled 1 IIC Bootstrap Controller Enabled
UART0_DSR	Selects serial ROM IIC 7-bit (0:6) Address 0 0xA8 selected 1 0xA0 selected

Table 8-2. Default Strap Settings

Register Bit Field	Value
CPC0_STRP0[TUNE]	0b1000101110
CPC0_STRP0[FBDV]	0b1001
CPC0_STRP0[FWDVA]	0b110
CPC0_STRP0[FWDVB]	0b010
CPC0_STRP0[OPDV]	0b01
CPC0_STRP0[EPDV]	0b00
CPC0_STRP0[EXTSL]	0b0
CPC0_STRP0[RW]	0b00
CPC0_STRP0[RL]	0b0
CPC0_STRP0[ZMIISL]	0b00
CPC0_STRP0[BYPASS]	0b0
CPC0_STRP0[NTO1]	0b0
CPC0_STRP1[TUNE]	0b1000101100
CPC0_STRP1[BYPASS]	0b0
CPC0_STRP1[PAE]	0b0
CPC0_STRP1[PHCE]	0b0
CPC0_STRP1[PISE]	0b0
CPC0_STRP1[PCWE]	0b0

Table 8-2. Default Strap Settings (continued)

Register Bit Field	Value
CPC0_STRP1[PPIM]	0b0000
CPC0_STRP1[PR64E]	0b0
CPC0_STRP1[PXFS]	0b01
CPC0_STRP1[PQXDE]	0b01
CPC0_STRP1 23:31	0b0000000000
CPC0_STRP2 0:31	0x00000000
CPC0_STRP3 0:31	0x00000000

8.1.1 PCI Inbound Map Setting

A PCI master can access internal PCI configuration registers by running type 0 configuration cycles, with the appropriate device select, such that the PLB-PCI bridge's IDSEL is active. A number of strapping options can determine the initial values of the pim0, pim1, and pim2 register bits. There are 4 strapping bits that select 16 different default settings for the registers.

CPC0_SYS1[PPIM] register bits 15:18 control the default setting for various PIM fields as shown in Figure 8-6. Detailed information on PIM0, PIM1 and PIM2 setting is shown in Table 8-3, Table 8-4 and Table 8-5 respectively.

Table 8-3 shows PIM0 settings.

Table 8-3. PCI Inbound Map 0 (PIM 0) Settings

PPIM	Comment	Enable	Prefetch	Size
0000	Off	0	0	FFFFF
0001	4k	1	0	FFFFF
0010	1M	1	0	FFF00
0011	64M	1	0	FC000
0100	4kp	1	1	FFFFF
0101	1Mp	1	1	FFF00
0110	64Mp	1	1	FC000
0111	64k	1	0	FFFF0
1000	1M	1	0	FFF00
1001	64Kp	1	1	FFFF0
1010	1Mp	1	1	FFF00
1011	64k	1	0	FFFF0
1100	1M	1	0	FFF00
1101	1Mp	1	1	FFF00
1110	1M	1	0	FFF00
1111	1M	1	0	FFF00

Table 8-4 shows PIM1 settings.

Table 8-4. PCI Inbound Map 1 (PIM 1) Settings

PPIM	Comment	Size	Prefetch	Enable
0000	Off			0
0001	Off			0
0010	Off			0
0011	Off			0
0100	Off			0
0101	Off			0
0110	Off			0
0111	Off			0
1000	Off			0
1001	Off			0
1010	Off			0
1011	Off			0
1100	Off			0
1101	Off			0
1110	On		0	1
1111	On		0	1

Table 8-5 shows PIM2 settings.

Table 8-5. PCI Inbound Map 2 (PIM 2) Settings

PPIM	Comment	Size	Prefetch	Enable
0000	Off	FFFFF	0	0
0001	Off	FFFFF	0	0
0010	Off	FFFFF	0	0
0011	Off	FFFFF	0	0
0100	Off	FFFFF	0	0
0101	Off	FFFFF	0	0
0110	Off	FFFFF	0	0
0111	16k	FFFFC	0	1
1000	64k	FFFF0	0	1
1001	16k	FFFFC	0	1
1010	64k	FFFF0	0	1
1011	64k	FFFF0	1	1
1100	1Mp	FFF00	1	1
1101	1Mp	FFF00	1	1
1110	Off	FFFFF	0	0
1111	16k	FFFFC	0	1

8.2 IIC Bootstrap Operation

After the deassertion of $\overline{\text{SysReset}}$, the IIC bootstrap controller, if enabled, places the IIC0 bus in a known state through an initialization sequence and accesses the serial ROM as described in Figure 8-2. The initialization sequence assumes that the PPC440GP is the only IIC master on the IIC0 bus. Master arbitration is not supported by the IIC bootstrap controller.

To access the bootstrap settings, the IIC bootstrap controller writes the serial ROM with the base address (0x00) and then reads 16 bytes starting from the base address 0x00 and ending with address 0x0F. If the IIC bootstrap controller is faster than the serial ROM, the serial ROM can hold the IIC0 clock signal low until it prepared for the next transaction. If the serial ROM does not acknowledge its address or the receipt of the base address, the PPC440GP defaults to the settings in Table 8-2 and generates a serial ROM error interrupt, UIC1_SR[SRE]. Once interrupts are enabled, initialization software should check UIC1_SR[SRE] to ensure no errors occurred while accessing the serial ROM.

The data in the serial ROM must be organized as shown in Table 8-6. Once the bootstrap settings are read from the serial ROM, they are stored in the Power-On Configuration Registers, CPC0_STRP0:3. To allow the bootstrap settings to be modified, the read only Power-On Configuration registers have write companion registers. CPC0_SYS0 and CPC_SYS1 are read/write versions of CPC0_STRP0 and CPC0_STRP1. CPC0_CUST0 and CPC_CUST1 are read/write versions of CPC0_STRP2 and CPC0_STRP3. The bootstrap settings stored in CPC0_STRP2 and CPC0_STRP3 (CPC0_CUST0 and CPC_CUST1) are user definable.

Table 8-6. Serial ROM Data Organization

ROM Data Address	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Destination Register CPC0_STRP0								
0x0	TUNE 0:7							
0x1	TUNE 8:9		FBDV 10:13				FBDVA 14:15	
0x2	FWDVA 16	FWDVB 17:19			OPDV 20:21		EPDV 22:23	
0x3	EXTSL 24	RW 25:26		RL 27	ZMIISL 28:29		BYPASS 30	NTO1 31
Destination Register CPC0_STRP1								
0x4	TUNE 0:7							
0x5	TUNE 8:9		BYPASS 10	PAE 11	PHCE 12	PISE 13	PCWE 14	PPIM 15
0x6	PPIM 16:18			PR64E 19	PXFS 20:21		PQXDE 22	Reserved 23
0x7	PDM 24	Reserved 25:31						
Destination Register CPC0_STRP2								
0x8								
0x9								
0xA								
0xB								
Destination Register CPC0_STRP3								
0xC								
0xD								
0xE								
0xF								

Figure 8-2 shows the IIC bootstrap controller flow.

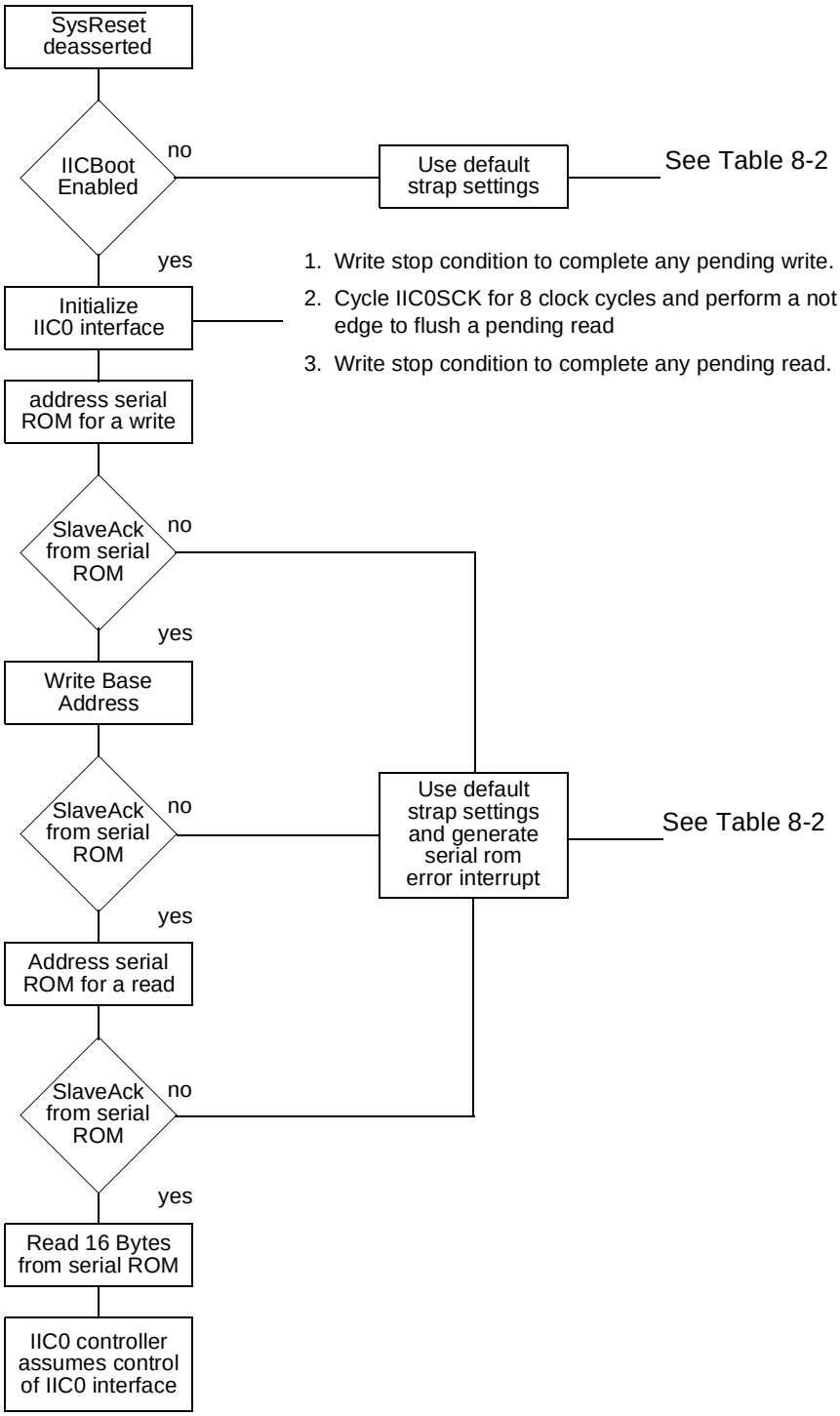


Figure 8-2. IIC Bootstrap Controller Flow

8.3 IIC Bootstrap Initialization

The PPC440GP uses the IIC Bootstrap Controller to preload several system registers from an external serial ROM device after power on reset. The bootstrap controller loads a total of 16 bytes into four r/w registers CPC0_SYS0, CPC0_SYS1, CPC0_CUST0 and CPC0_CUST1 and loads a copy of the same data in four r/o registers CPC0_STRP0, CPC0_STRP1, CPC0_STRP2 and CPC0_STRP3.

The first 8 bytes loaded by the Bootstrap Controller into CPC0_SYS0 and CPC0_SYS1 are used for configuration of the PPC440GP. The remaining 8 bytes in CPC0_CUST0 and CPC0_CUST1 are available for customer application specific purposes.

Table 8-7 lists the system control registers loaded by the Bootstrap Controller.

Table 8-7. IIC Bootstrap Controller Registers

Mnemonic	Register	Address	Access	Page
CPC0_SYS0	System Configuration Register 0	0x0E0	R/W	~~8-9~~291
CPC0_SYS1	System Configuration Register 1	0x0E1	R/W	~~8-13~~295
CPC0_CUST0	Customer Configuration Register 0	0x0E2	R/W	~~8-15~~297
CPC0_CUST1	Customer Configuration Register 1	0x0E3	R/W	~~8-16~~298
CPC0_STRP0	Power-on Configuration Register 0	0x0E4	R	~~8-7~~289
CPC0_STRP1	Power-on Configuration Register 1	0x0E5	R	~~8-11~~293
CPC0_STRP2	Power-on Configuration Register 2	0x0E6	R	~~8-14~~296
CPC0_STRP3	Power-on Configuration Register 3	0x0E7	R	~~8-15~~297

8.3.1 Power-On Configuration Register 0 (CPC0_STRP0)

The CPC0_STRP0 is a 32-bit read-only version of the CPC0_SYS0 register. This register is reserved for system PLL and configuration information. The CPC0_STRP0 is reset according to values read by the IIC Bootstrap controller after System Reset. If the IIC Bootstrap controller is disabled or unable to read the bootstrap information from a serial ROM device, the default values given in Table 8-2 are used.

Note: Writes to this register are ignored unless the SWE bit is set in the CPC0_CR0 register. Also writes to these fields do not affect either SYS or PCI PLL until a CHIP reset occurs.

~~Figure 8-3~~Figure 8-3 describes CPC0_STRP0 bit definitions.

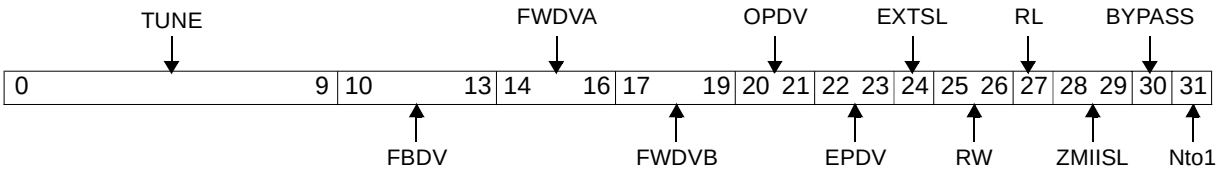


Figure 0-1. Power-On Configuration Register 0 (CPC0_STRP0)

0:9	TUNE	System PLL TUNE bits	See Table 13-5, "CPC0_SYS0[Nto1] Settings," on page 13-5 for tune bit setting information.
10:13	FBDV	PLL Feedback Divisor 0000 PLL Feedback Divisor = 16 0001 PLL Feedback Divisor = 1 0010 PLL Feedback Divisor = 2 0011 PLL Feedback Divisor = 3 0100 PLL Feedback Divisor = 4 0101 PLL Feedback Divisor = 5 0110 PLL Feedback Divisor = 6 0111 PLL Feedback Divisor = 7 1000 PLL Feedback Divisor = 8 1001 PLL Feedback Divisor = 9 1010 PLL Feedback Divisor = 10 1011 PLL Feedback Divisor = 11 1100 PLL Feedback Divisor = 12 1101 PLL Feedback Divisor = 13 1110 PLL Feedback Divisor = 14 1111 PLL Feedback Divisor = 15	
14:16	FWDVA	PLL Forward Divisor A 000 PLL Forward Divisor A = 8 001 PLL Forward Divisor A = 7 010 PLL Forward Divisor A = 6 011 PLL Forward Divisor A = 5 100 PLL Forward Divisor A = 4 101 PLL Forward Divisor A = 3 110 PLL Forward Divisor A = 2 111 PLL Forward Divisor A = 1	
17:19	FWDVB	PLL Forward Divisor B 000 PLL Forward Divisor B = 8 001 PLL Forward Divisor B = 7 010 PLL Forward Divisor B = 6 011 PLL Forward Divisor B = 5 100 PLL Forward Divisor B = 4 101 PLL Forward Divisor B = 3 110 PLL Forward Divisor B = 2	
20:21	OPDV	PLB to OPB Clock Divisor Ratio 00 PLB to OPB Clock Divisor = 1 01 PLB to OPB Clock Divisor = 2 10 PLB to OPB Clock Divisor = 3 11 PLB to OPB Clock Divisor = 4	The OPDV is the divisor ratio between the PLB and OPB clock. For example, if the PLB clock frequency is 133MHz, a divisor ratio of two sets the OPB clock frequency to 66MHz.

22:23	EPDV	EBCO to OPB Clock Divisor Ratio 00 EBCO to OPB Clock Divisor = 1 01 EBCO to OPB Clock Divisor = 2 10 EBCO to OPB Clock Divisor = 3 11 EBCO to OPB Clock Divisor = 4	The EPDV is the divisor ratio between the EBC0 (PerClk) and OPB clock. For example, if the OPB clock frequency is 66MHz, a divisor ratio of two sets the EBC0 clock (PerClk) frequency to 33MHz.
24	EXTSL	EBC0 clock (PerClk) Feedback Path Selection 0 CPU clock selected as PLL feedback path 1 EBC0 clock (PerClk) selected as PLL feedback path	Selecting the CPU clock as the PLL feedback path phase aligns PerClk with the CPU clock. Selecting the PerClk as the PLL feedback path phase aligns PerClk with SysClk.
25:26	RW	ROM Width 00 8-bit ROM 01 16-bit ROM 10 reserved 11 32-bit ROM	
27	RL	ROM Location 0 ROM is connected to EBCO 1 ROM is connected to PCI	If ROM is connected to EBCO, the real address of the reset vector is 0x1 FFFF FFFC. If ROM is connected to PCI, the real address of the reset vector is 0x2 FFFF FFFC.
28:29	ZMIISL	ZMII Selection 00 MII mode selected 01 SMII mode selected 10 RMII 10Mb mode selected 11 RMII 100Mb mode selected	See "EMAC-ZMII Bridge Interfaces" on page 21-3.
30	BYPASS	Bypass System PLL 0 System PLL enabled 1 Bypass system PLL	
31	Nto1	CPU:PLB N to 1 clock ratio 0 CPU:PLB clock ratio is N:P where P is greater than 1 1 CPU:PLB clock ratio is N:1	

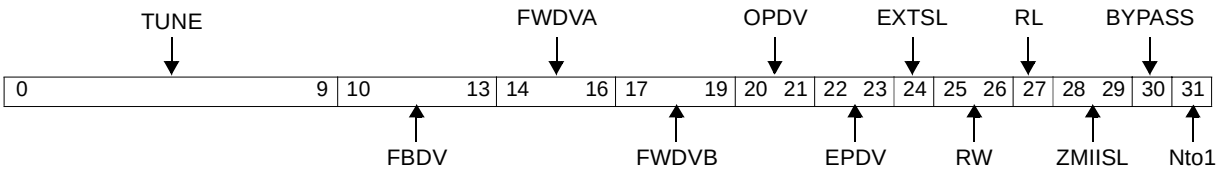


Figure 8-3. Power-On Configuration Register 0 (CPC0_STRP0)

0:9	TUNE	System PLL TUNE bits	See Table 13-5, "CPC0_SYS0[Nto1] Settings," on page -411 for tune bit setting information.
10:13	FBDV	PLL Feedback Divisor 0000 PLL Feedback Divisor = 16 0001 PLL Feedback Divisor = 1 0010 PLL Feedback Divisor = 2 0011 PLL Feedback Divisor = 3 0100 PLL Feedback Divisor = 4 0101 PLL Feedback Divisor = 5 0110 PLL Feedback Divisor = 6 0111 PLL Feedback Divisor = 7 1000 PLL Feedback Divisor = 8 1001 PLL Feedback Divisor = 9 1010 PLL Feedback Divisor = 10 1011 PLL Feedback Divisor = 11 1100 PLL Feedback Divisor = 12 1101 PLL Feedback Divisor = 13 1110 PLL Feedback Divisor = 14 1111 PLL Feedback Divisor = 15	
14:16	FWDVA	PLL Forward Divisor A 000 PLL Forward Divisor A = 8 001 PLL Forward Divisor A = 7 010 PLL Forward Divisor A = 6 011 PLL Forward Divisor A = 5 100 PLL Forward Divisor A = 4 101 PLL Forward Divisor A = 3 110 PLL Forward Divisor A = 2 111 PLL Forward Divisor A = 1	
17:19	FWDVB	PLL Forward Divisor B 000 PLL Forward Divisor B = 8 001 PLL Forward Divisor B = 7 010 PLL Forward Divisor B = 6 011 PLL Forward Divisor B = 5 100 PLL Forward Divisor B = 4 101 PLL Forward Divisor B = 3 110 PLL Forward Divisor B = 2	
20:21	OPDV	PLB to OPB Clock Divisor Ratio 00 PLB to OPB Clock Divisor = 1 01 PLB to OPB Clock Divisor = 2 10 PLB to OPB Clock Divisor = 3 11 PLB to OPB Clock Divisor = 4	The OPDV is the divisor ratio between the PLB and OPB clock. For example, if the PLB clock frequency is 133MHz, a divisor ratio of two sets the OPB clock frequency to 66MHz.
22:23	EPDV	EBCO to OPB Clock Divisor Ratio 00 EBCO to OPB Clock Divisor = 1 01 EBCO to OPB Clock Divisor = 2 10 EBCO to OPB Clock Divisor = 3 11 EBCO to OPB Clock Divisor = 4	The EPDV is the divisor ratio between the EBC0 (PerClk) and OPB clock. For example, if the OPB clock frequency is 66MHz, a divisor ratio of two sets the EBC0 clock (PerClk) frequency to 33MHz.

24	EXTSL	EBC0 clock (PerClk) Feedback Path Selection 0 CPU clock selected as PLL feedback path 1 EBC0 clock (PerClk) selected as PLL feedback path	Selecting the CPU clock as the PLL feedback path phase aligns PerClk with the CPU clock. Selecting the PerClk as the PLL feedback path phase aligns PerClk with SysClk.
25:26	RW	ROM Width 00 8-bit ROM 01 16-bit ROM 10 reserved 11 32-bit ROM	
27	RL	ROM Location 0 ROM is connected to EBCO 1 ROM is connected to PCI	If ROM is connected to EBCO, the real address of the reset vector is 0x1FFFFFFFC. If ROM is connected to PCI, the real address of the reset vector is 0x2FFFFFFFC.
28:29	ZMIISL	ZMII Selection 00 MII mode selected 01 SMII mode selected 10 RMII 10Mb mode selected 11 RMII 100Mb mode selected	See *EMAC-ZMII Bridge Interfaces* on page 721.
30	BYPASS	Bypass System PLL 0 System PLL enabled 1 Bypass system PLL	
31	Nto1	CPU:PLB N to 1 clock ratio 0 CPU:PLB clock ratio is N:P where P is greater than 1 1 CPU:PLB clock ratio is N:1	

8.3.2 System Configuration Register 0 (CPC0_SYS0)

Writes to CPC0_SYS0 are ignored unless CPC0_CR0[SWE] = 1. Writes to any fields in this register do not take effect until a chip reset occurs, however written values can be read back immediately.

Note: Writes to this register are ignored unless the SWE bit is set in the CPC0_CR0 register. Also writes to these fields do not affect either SYS or PCI PLL until a CHIP reset occurs.

Figure 8-4 describes CPC0_SYS0 bit definitions.

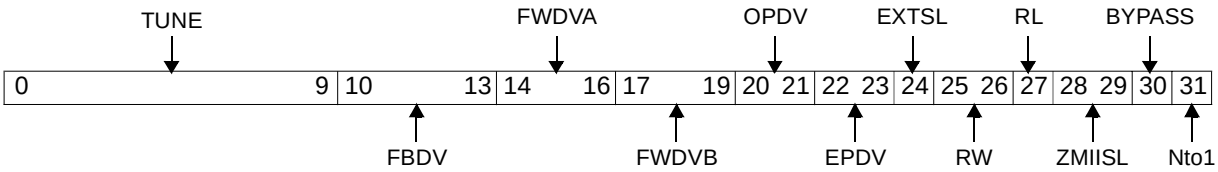


Figure 0-2. System Configuration Register 0 (CPC0_SYS0)

0:9	TUNE	System PLL TUNE bits	See Table 13-5, "CPC0_SYS0[Nto1] Settings," on page 13-5 for tune bit setting information.
10:13	FBDV	PLL Feedback Divisor 0000 PLL Feedback Divisor = 16 0001 PLL Feedback Divisor = 1 0010 PLL Feedback Divisor = 2 0011 PLL Feedback Divisor = 3 0100 PLL Feedback Divisor = 4 0101 PLL Feedback Divisor = 5 0110 PLL Feedback Divisor = 6 0111 PLL Feedback Divisor = 7 1000 PLL Feedback Divisor = 8 1001 PLL Feedback Divisor = 9 1010 PLL Feedback Divisor = 10 1011 PLL Feedback Divisor = 11 1100 PLL Feedback Divisor = 12 1101 PLL Feedback Divisor = 13 1110 PLL Feedback Divisor = 14 1111 PLL Feedback Divisor = 15	
14:16	FWDVA	PLL Forward Divisor A 000 PLL Forward Divisor A = 8 001 PLL Forward Divisor A = 7 010 PLL Forward Divisor A = 6 011 PLL Forward Divisor A = 5 100 PLL Forward Divisor A = 4 101 PLL Forward Divisor A = 3 110 PLL Forward Divisor A = 2 111 PLL Forward Divisor A = 1	
17:19	FWDVB	PLL Forward Divisor B 000 PLL Forward Divisor B = 8 001 PLL Forward Divisor B = 7 010 PLL Forward Divisor B = 6 011 PLL Forward Divisor B = 5 100 PLL Forward Divisor B = 4 101 PLL Forward Divisor B = 3 110 PLL Forward Divisor B = 2 111 PLL Forward Divisor B = 1	

20:21	OPDV	PLB to OPB Clock Divisor Ratio 00 PLB to OPB Clock Divisor = 1 01 PLB to OPB Clock Divisor = 2 10 PLB to OPB Clock Divisor = 3 11 PLB to OPB Clock Divisor = 4	The OPDV is the divisor ratio between the PLB and OPB clock. For example, if the PLB clock frequency is 133MHz, a divisor ratio of two sets the OPB clock frequency to 66MHz.
22:23	EPDV	EBCO to OPB Clock Divisor Ratio 00 EBCO to OPB Clock Divisor = 1 01 EBCO to OPB Clock Divisor = 2 10 EBCO to OPB Clock Divisor = 3 11 EBCO to OPB Clock Divisor = 4	The EPDV is the divisor ratio between the EBC0 (PerClk) and OPB clock. For example, if the OPB clock frequency is 66MHz, a divisor ratio of two sets the EBC0 clock (PerClk) frequency to 33MHz.
24	EXTSL	EBC0 clock (PerClk) Feedback Path Selection 0 CPU clock selected as PLL feedback path 1 EBC0 clock (PerClk) selected as PLL feedback path	Selecting the CPU clock as the PLL feedback path phase aligns PerClk with the CPU clock. Selecting the PerClk as the PLL feedback path phase aligns PerClk with SysClk.
25:26	RW	ROM Width 00 8-bit ROM 01 16-bit ROM 10 reserved 11 32-bit ROM	
27	RL	ROM Location 0 ROM connected to EBCO 1 ROM connected to PCI	If ROM is connected to EBCO, the real address of the reset vector is 0x1FFFFFFFC. If ROM is connected to PCI, the real address of the reset vector is 0x2FFFFFFFC.
28:29	ZMIISL	ZMII Selection 00 MII mode selected 01 SMII mode selected 10 RMII 10Mb mode selected 11 RMII 100Mb mode selected	See "EMAC-ZMII Bridge Interfaces" on page 21-3.
30	BYPASS	Bypass System PLL 0 System PLL enabled 1 Bypass system PLL	
31	Nto1	CPU:PLB N to 1 clock ratio 0 CPU:PLB clock ratio is N:P where P is greater than 1 1 CPU:PLB clock ratio is N:1	

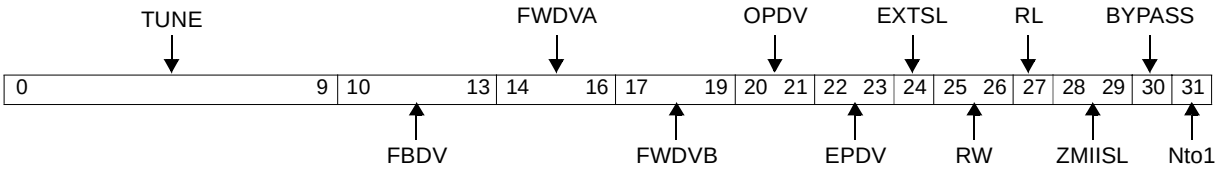


Figure 8-4. System Configuration Register 0 (CPC0_SYS0)

0:9	TUNE	System PLL TUNE bits	See Table 13-5, "CPC0_SYS0[Nto1] Settings," on page -411 for tune bit setting information.
10:13	FBDV	PLL Feedback Divisor 0000 PLL Feedback Divisor = 16 0001 PLL Feedback Divisor = 1 0010 PLL Feedback Divisor = 2 0011 PLL Feedback Divisor = 3 0100 PLL Feedback Divisor = 4 0101 PLL Feedback Divisor = 5 0110 PLL Feedback Divisor = 6 0111 PLL Feedback Divisor = 7 1000 PLL Feedback Divisor = 8 1001 PLL Feedback Divisor = 9 1010 PLL Feedback Divisor = 10 1011 PLL Feedback Divisor = 11 1100 PLL Feedback Divisor = 12 1101 PLL Feedback Divisor = 13 1110 PLL Feedback Divisor = 14 1111 PLL Feedback Divisor = 15	
14:16	FWDVA	PLL Forward Divisor A 000 PLL Forward Divisor A = 8 001 PLL Forward Divisor A = 7 010 PLL Forward Divisor A = 6 011 PLL Forward Divisor A = 5 100 PLL Forward Divisor A = 4 101 PLL Forward Divisor A = 3 110 PLL Forward Divisor A = 2 111 PLL Forward Divisor A = 1	
17:19	FWDVB	PLL Forward Divisor B 000 PLL Forward Divisor B = 8 001 PLL Forward Divisor B = 7 010 PLL Forward Divisor B = 6 011 PLL Forward Divisor B = 5 100 PLL Forward Divisor B = 4 101 PLL Forward Divisor B = 3 110 PLL Forward Divisor B = 2 111 PLL Forward Divisor B = 1	
20:21	OPDV	PLB to OPB Clock Divisor Ratio 00 PLB to OPB Clock Divisor = 1 01 PLB to OPB Clock Divisor = 2 10 PLB to OPB Clock Divisor = 3 11 PLB to OPB Clock Divisor = 4	The OPDV is the divisor ratio between the PLB and OPB clock. For example, if the PLB clock frequency is 133MHz, a divisor ratio of two sets the OPB clock frequency to 66MHz.
22:23	EPDV	EBCO to OPB Clock Divisor Ratio 00 EBCO to OPB Clock Divisor = 1 01 EBCO to OPB Clock Divisor = 2 10 EBCO to OPB Clock Divisor = 3 11 EBCO to OPB Clock Divisor = 4	The EPDV is the divisor ratio between the EBC0 (PerClk) and OPB clock. For example, if the OPB clock frequency is 66MHz, a divisor ratio of two sets the EBC0 clock (PerClk) frequency to 33MHz.

24	EXTSL	EBC0 clock (PerClk) Feedback Path Selection 0 CPU clock selected as PLL feedback path 1 EBC0 clock (PerClk) selected as PLL feedback path	Selecting the CPU clock as the PLL feedback path phase aligns PerClk with the CPU clock. Selecting the PerClk as the PLL feedback path phase aligns PerClk with SysClk.
25:26	RW	ROM Width 00 8-bit ROM 01 16-bit ROM 10 reserved 11 32-bit ROM	
27	RL	ROM Location 0 ROM connected to EBCO 1 ROM connected to PCI	If ROM is connected to EBCO, the real address of the reset vector is 0x1FFFFFFFC. If ROM is connected to PCI, the real address of the reset vector is 0x2FFFFFFFC.
28:29	ZMIISL	ZMII Selection 00 MII mode selected 01 SMII mode selected 10 RMII 10Mb mode selected 11 RMII 100Mb mode selected	See *EMAC-ZMII Bridge Interfaces* on page 721.
30	BYPASS	Bypass System PLL 0 System PLL enabled 1 Bypass system PLL	
31	Nto1	CPU:PLB N to 1 clock ratio 0 CPU:PLB clock ratio is N:P where P is greater than 1 1 CPU:PLB clock ratio is N:1	

8.3.3 Power-On Configuration Register 1 (CPC0_STRP1)

The CPC0_STRP1 is a 32-bit read-only version of the CPC0_SYS1 register. This register is reserved for PCI PLL and PCI configuration information. The CPC0_STRP1 is reset according to values read by the IIC Bootstrap controller after System Reset. If the IIC Bootstrap controller is disabled or unable to read the bootstrap information from a serial ROM device, the default values given inTable 8-2 are used.

Note: Writes to this register are ignored unless the SWE bit is set in the CPC0_CR0 register. Also writes to these fields do not affect either SYS or PCI PLL until a CHIP reset occurs

~~Figure 8-5~~Figure 8-5 describes CPC0_STRP1 bit definitions.

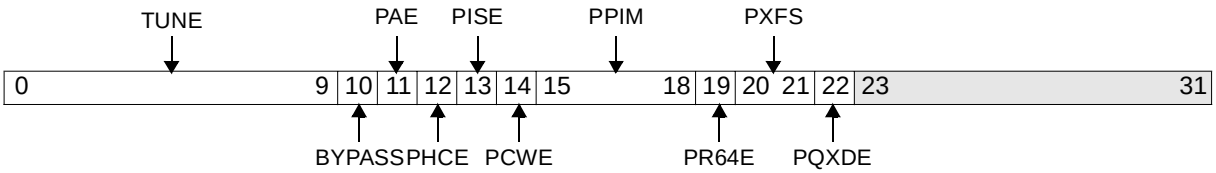


Figure 0-3. Power-On Configuration Register 1 (CPC0_STRP1)

0:9	TUNE	PCI PLL TUNE Bits	See Table 13-5, "CPC0_SYS0[Nto1] Settings," on page 13-5 for tune bit setting information.
10	BYPASS	Bypass PCI PLL 0 PCI PLL enabled 1 Bypass PCI PLL	
11	PAE	PCI arbiter enable	
12	PHCE	PCI host configuration enable	
13	PISE	PCI initial sequence enable	
14	PCWE	PCI local CPU wait enable	
15:18	PPIM	PCI Inbound Map (PIM) Settings 0000 PIM0 off, PIM1 off, PIM2 off 0001 PIM0 4k, PIM1 off, PIM2 off 0010 PIM0 1M, PIM1 off, PIM2 off 0011 PIM0 64M, PIM1 off, PIM2 off 0100 PIM0 4kp, PIM1 off, PIM2 off 0101 PIM0 1Mp, PIM1 off, PIM2 off 0110 PIM0 64Mp, PIM1 off, PIM2 off 0111 PIM0 64k, PIM1 off, PIM2 16k 1000 PIM0 1M, PIM1 off, PIM2 64k 1001 PIM0 64kp, PIM1 off, PIM2 16k 1010 PIM0 1Mp, PIM1 off, PIM2 64k 1011 PIM0 64k, PIM1 off, PIM2 64Kp 1100 PIM0 1M, PIM1 off, PIM2 1Mp 1101 PIM0 1Mp, PIM1 off, PIM2 1Mp 1110 PIM0 1M, PIM1 on, PIM2 off 1111 PIM0 1M, PIM1 on, PIM2 16k	These bits control the default settings for various PIM fields. See "PCI Inbound Map Setting" on page 8-3.
19	PR64E	PCI initialize Req64 Enable	
20:21	PXFS	PCIX Frequency Selection 00 100 - 133MHz 01 66 - 100MHz 10 50 - 66MHz 11 reserved	
22	PQXDE	Quick PCIX capable detect enable	
23:31		Reserved	

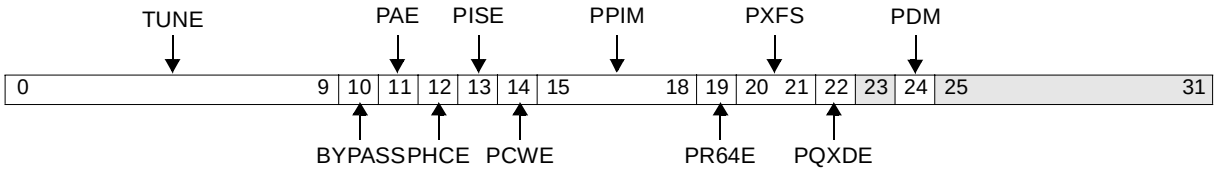


Figure 8-5. Power-On Configuration Register 1 (CPC0_STRP1)

0:9	TUNE	PCI PLL TUNE Bits	See Table 13-5, "CPC0_SYS0[Nto1] Settings," on page -411 for tune bit setting information.
10	BYPASS	Bypass PCI PLL 0 PCI PLL enabled 1 Bypass PCI PLL	
11	PAE	PCI Arbiter Enable 0 PCI arbiter disabled 1 PCI arbiter enabled	
12	PHCE	PCI Host Configuration Enable 0 PCI host configuration disabled 1 PCI host configuration enabled	
13	PISE	PCI initial sequence enable 0 PCI initial sequence disabled 1 PCI initial sequence enabled	
14	PCWE	PCI local CPU wait enable 0 PCI local CPU wait disabled 1 PCI local CPU wait enabled	
15:18	PPIM	PCI Inbound Map (PIM) Settings 0000 PIM0 off, PIM1 off, PIM2 off 0001 PIM0 4k, PIM1 off, PIM2 off 0010 PIM0 1M, PIM1 off, PIM2 off 0011 PIM0 64M, PIM1 off, PIM2 off 0100 PIM0 4kp, PIM1 off, PIM2 off 0101 PIM0 1Mp, PIM1 off, PIM2 off 0110 PIM0 64Mp, PIM1 off, PIM2 off 0111 PIM0 64k, PIM1 off, PIM2 16k 1000 PIM0 1M, PIM1 off, PIM2 64k 1001 PIM0 64kp, PIM1 off, PIM2 16k 1010 PIM0 1Mp, PIM1 off, PIM2 64k 1011 PIM0 64k, PIM1 off, PIM2 64Kp 1100 PIM0 1M, PIM1 off, PIM2 1Mp 1101 PIM0 1Mp, PIM1 off, PIM2 1Mp 1110 PIM0 1M, PIM1 on, PIM2 off 1111 PIM0 1M, PIM1 on, PIM2 16k	These bits control the default settings for various PIM fields. See *PCI Inbound Map Setting* on page 285.
19	PR64E	PCI initialize Req64 Enable 0 PCI initialize Req64 disabled 1 PCI initialize Req64 enabled	
20:21	PXFS	PCIX Frequency Selection 00 100 - 133MHz 01 66 - 100MHz 10 50 - 66MHz 11 reserved	
22	PQXDE	Quick PCIX capable detect enable 0 Quick PCIX capable detect disabled 1 Quick PCIX capable detect enabled	

23		Reserved
24	PDM	PCIX Driver Mode 0 PCIX driver impedence is 20 ohms for multi point mode 1 PCIX driver impedence is 40 ohms for point to point mode
25:31		Reserved

8.3.4 System Configuration Register 1 (CPC0_SYS1)

Writes to CPC0_SYS1 are ignored unless CPC0_CR0[SWE] = 1. Writes to the TUNE and BYPASS fields in this register do not take effect until a chip reset occurs, however written values can be read back immediately. All other fields take effect immediately without requiring a chip reset.

Note: Writes to this register are ignored unless the SWE bit is set in the CPC0_CR0 register. Also writes to these fields do not affect either SYS or PCI PLL until a CHIP reset occurs

Figure 8-6 describes CPC0_SYS1 bit definitions.

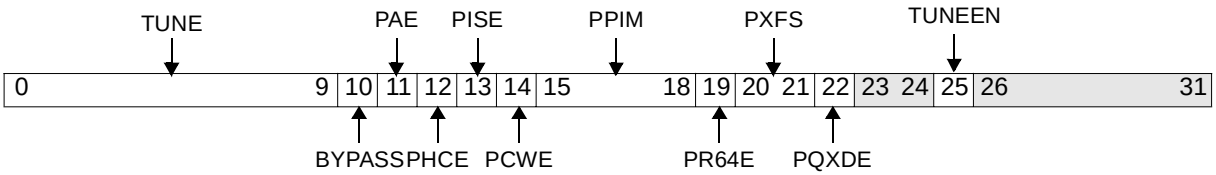


Figure 0-4. System Configuration Register 1 (CPC0_SYS1)

0:9	TUNE	CPC0_SYS1[TUNEEN] 0 CPC0_SYS1[TUNEEN] reads will return the tune bit value which was dynamically determined by the PCI interface at power-on (default). 1 CPC0_SYS1[TUNEEN] reads will return the current value programmed into CPC0_SYS1[TUNE]	CPC0_SYS1[TUNEEN] controls a mux which affects the value read from this field. Also see Table 13-13, "CPC0_SYS1[TUNE] Bit Settings," on page 13-11 for PCI PLL tune bit setting information.
10	BYPASS	Bypass PCI PLL 0 PCI PLL enabled 1 Bypass PCI PLL	
11	PAE	PCI arbiter enable 0 PCI arbiter disabled 1 PCI arbiter enabled	
12	PHCE	PCI host configuration enable 0 PCI host configuration disabled 1 PCI host configuration enabled	

13	PISE	PCI initial sequence enable 0 PCI initial sequence disabled 1 PCI initial sequence enabled	
14	PCWE	PCI local CPU wait enable 0 PCI local CPU wait disabled 1 PCI local CPU wait enabled	
15:18	PPIM	PCI Inbound Map (PIM) Settings 0000 PIM0 off, PIM1 off, PIM2 off 0001 PIM0 4k, PIM1 off, PIM2 off 0010 PIM0 1M, PIM1 off, PIM2 off 0011 PIM0 64M, PIM1 off, PIM2 off 0100 PIM0 4kp, PIM1 off, PIM2 off 0101 PIM0 1Mp, PIM1 off, PIM2 off 0110 PIM0 64Mp, PIM1 off, PIM2 off 0111 PIM0 64k, PIM1 off, PIM2 16k 1000 PIM0 1M, PIM1 off, PIM2 64k 1001 PIM0 64kp, PIM1 off, PIM2 16k 1010 PIM0 1Mp, PIM1 off, PIM2 64k 1011 PIM0 64k, PIM1 off, PIM2 64Kp 1100 PIM0 1M, PIM1 off, PIM2 1Mp 1101 PIM0 1Mp, PIM1 off, PIM2 1Mp 1110 PIM0 1M, PIM1 on, PIM2 off 1111 PIM0 1M, PIM1 on, PIM2 16k	These bits control the default settings for various PIM fields. See "PCI Inbound Map Setting" on page 8-3.
19	PR64E	PCI initialize Req64 Enable	
20:21	PXFS	PCIX Frequency Selection 00 100 - 133MHz frequency selection 01 66 - 100MHz frequency selection 10 50 - 66MHz frequency selection 11 reserved	
22	PQXDE	Quick PCIX capable detect enable 0 Quick PCIX capable detect disabled 1 Quick PCIX capable detect enabled	
23:24		Reserved	
25	TUNEEN	Custom PCI Tune Bit Enable 0 Use dynamically determined tune bits (default) 1 Use tune bits from CPC0_SYS1[TUNE]	Override the PCI PLL tune bit settings which are dynamically determined at power-on with the value programmed in CPC0_SYS1[TUNE]
26:31		Reserved	

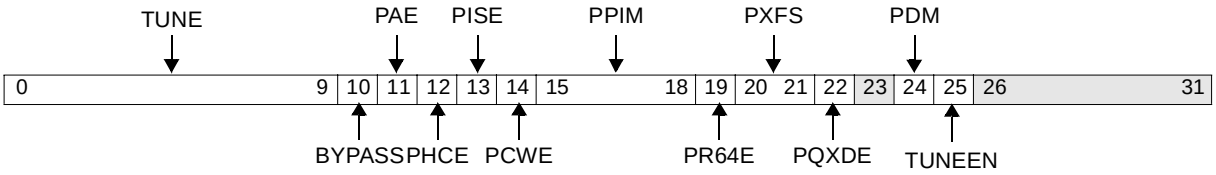


Figure 8-6. System Configuration Register 1 (CPC0_SYS1)

0:9	TUNE	CPC0_SYS1[TUNEEN] 0 CPC0_SYS1[TUNEEN] reads will return the tune bit value which was dynamically determined by the PCI interface at power-on (default). 1 CPC0_SYS1[TUNEEN] reads will return the current value programmed into CPC0_SYS1[TUNE]	CPC0_SYS1[TUNEEN] controls a mux which affects the value read from this field. Also see Table 13-13, "CPC0_SYS1[TUNE] Bit Settings," on page -417 for PCI PLL tune bit setting information.
10	BYPASS	Bypass PCI PLL 0 PCI PLL enabled 1 Bypass PCI PLL	
11	PAE	PCI arbiter enable 0 PCI arbiter disabled 1 PCI arbiter enabled	
12	PHCE	PCI host configuration enable 0 PCI host configuration disabled 1 PCI host configuration enabled	
13	PISE	PCI initial sequence enable 0 PCI initial sequence disabled 1 PCI initial sequence enabled	
14	PCWE	PCI local CPU wait enable 0 PCI local CPU wait disabled 1 PCI local CPU wait enabled	
15:18	PPIM	PCI Inbound Map (PIM) Settings 0000 PIM0 off, PIM1 off, PIM2 off 0001 PIM0 4k, PIM1 off, PIM2 off 0010 PIM0 1M, PIM1 off, PIM2 off 0011 PIM0 64M, PIM1 off, PIM2 off 0100 PIM0 4kp, PIM1 off, PIM2 off 0101 PIM0 1Mp, PIM1 off, PIM2 off 0110 PIM0 64Mp, PIM1 off, PIM2 off 0111 PIM0 64k, PIM1 off, PIM2 16k 1000 PIM0 1M, PIM1 off, PIM2 64k 1001 PIM0 64kp, PIM1 off, PIM2 16k 1010 PIM0 1Mp, PIM1 off, PIM2 64k 1011 PIM0 64k, PIM1 off, PIM2 64Kp 1100 PIM0 1M, PIM1 off, PIM2 1Mp 1101 PIM0 1Mp, PIM1 off, PIM2 1Mp 1110 PIM0 1M, PIM1 on, PIM2 off 1111 PIM0 1M, PIM1 on, PIM2 16k	These bits control the default settings for various PIM fields. See *PCI Inbound Map Setting* on page 285.

19	PR64E	PCI initialize Req64 Enable 0 PCI initialize Req64 disabled 1 PCI initialize Req64 enabled
20:21	PXFS	PCIX Frequency Selection 00 100 - 133MHz frequency selection 01 66 - 100MHz frequency selection 10 50 - 66MHz frequency selection 11 reserved
22	PQXDE	Quick PCIX capable detect enable 0 Quick PCIX capable detect disabled 1 Quick PCIX capable detect enabled
23		Reserved
24	PDM	PCIX driver mode enable 0 PCIX driver impedence is 20 ohms for multi point mode 1 PCIX driver impedence is 40 ohms for point to point mode
25	TUNEEN	Custom PCI Tune Bit Enable 0 Use dynamically determined tune bits (default) 1 Use tune bits from CPC0_SYS1[TUNE] Override the PCI PLL tune bit settings which are dynamically determined at power-on with the value programmed in CPC0_SYS1[TUNE]
26:31		Reserved

8.3.5 Power-On Configuration Register 2 (CPC0_STRP2)

The CPC0_STRP2 is a 32-bit read-only version of the CPC0_CUST0 register. This register is reserved for custom user configuration information and has no predefined bit fields. The CPC0_STRP2 is reset according to values read by the IIC Bootstrap controller after System Reset. If the IIC Bootstrap controller is disabled or unable to read the bootstrap information from a serial ROM device, the default values given in Table 8-2 are used.

Note: Writes to this register are ignored unless the SWE bit is set in the CPC0_CR0 register. Also writes to these fields do not affect either SYS or PCI PLL until a CHIP reset occurs

~~Figure 8-7~~Figure 8-7 describes CPC0_STRP2 register bit definitions.

Figure 0-5. Power-On Configuration Register 2 (CPC0_STRP2)

0:31		Reserved	Reserved for customer use

Figure 8-7. Power-On Configuration Register 2 (CPC0_STRP2)

0:31		Reserved	Reserved for customer use

8.3.6 Customer Configuration Register 0 (CPC0_CUST0)

The CPC0_CUST0 is a 32-bit read/write version of the CPC0_STRP2 register. This register is reserved for custom user configuration information and has no predefined bit fields. The CPC0_CUST0 is reset according to values read by the IIC Bootstrap controller after System Reset. If the IIC Bootstrap controller is disabled or unable to read the bootstrap information from a serial ROM device, the default values given in Table 8-2 are used.

~~Figure 8-8~~Figure 8-8 describes CPC0_CUST0 register bit definitions.

Figure 0-6. Customer Configuration Register 0 (CPC0_CUST0)

0:31		Reserved	Reserved for customer use

Figure 8-8. Customer Configuration Register 0 (CPC0_CUST0)

0:31		Reserved	Reserved for customer use

8.3.7 Power-On Configuration Register 3 (CPC0_STRP3)

The CPC0_STRP3 is a 32-bit read-only version of the CPC0_CUST1 register. This register is reserved for custom user configuration information and has no predefined bit fields. The CPC0_STRP3 is reset according to values read by the IIC Bootstrap controller after System Reset. If the IIC Bootstrap controller is disabled or unable to read the bootstrap information from a serial ROM device, the default values given in Table 8-2 are used.

~~Figure 8-9~~Figure 8-9 describes CPC0_STRP3 register bit definitions.

Figure 0-7. Power-On Configuration Register 3 (CPC0_STRP3)

0:31		Reserved	Reserved for customer use

Figure 8-9. Power-On Configuration Register 3 (CPC0_STRP3)

0:31		Reserved	Reserved for customer use

8.3.8 Customer Configuration Register 0 (CPC0_CUST1)

The CPC0_CUST1 is a 32-bit read/write version of the CPC0_STRP3 register. This register is reserved for custom user configuration information and has no predefined bit fields. The CPC0_CUST1 is reset according to values read by the IIC Bootstrap controller after System Reset. If the IIC Bootstrap controller is disabled or unable to read the bootstrap information from a serial ROM device, the default values given in Table 8-2 are used.

~~Figure 8-10~~Figure 8-10 describes CPC0_CUST1 register bit definitions.

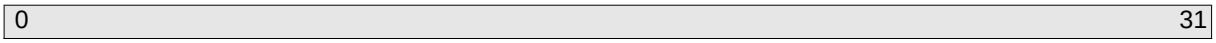


Figure 0-8. Customer Configuration Register 1 (CPC0_CUST1)

0:31		Reserved	Reserved for customer use

Figure 8-10. Customer Configuration Register 1 (CPC0_CUST1)

0:31		Reserved	Reserved for customer use

9. Universal Interrupt Controller

The PPC440GP contains two universal interrupt controllers (UIC0 and UIC1) that provide all necessary control, status, and communication between the various internal and external interrupt sources and the processor core.

9.1 UIC Overview

The UICs support 45 internal interrupts and 13 external interrupts. Status reporting (using the UIC Status Register [UIC*x*_SR]) is provided to ensure that systems software can determine the current and interrupting state of the system and respond appropriately. Software can generate interrupts to simplify software development and for diagnostics.

UIC0 collects interrupts from internal and external sources, including the critical and non-critical interrupt outputs of the secondary interrupt controller, UIC1. The UICs are cascaded as shown in Figure 9-1 below:

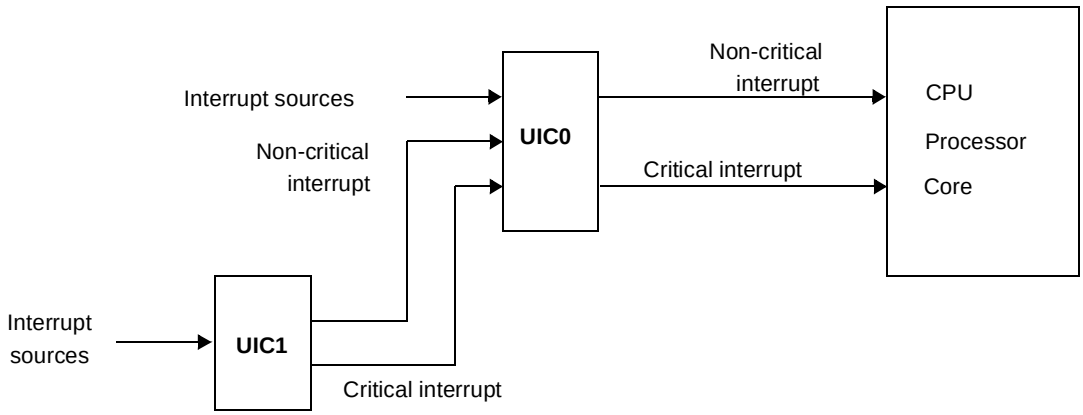


Figure 9-1. Cascaded UIC Organization

The interrupts can be programmed, using the UIC Critical Register (UIC*x*_CR), to generate either a critical or a non-critical interrupt signal to the processor core.

The privileged **mtdcr** and **mfdcr** instructions, which are used by system software, are used to read and write the UIC registers.

An optional critical interrupt vector generator can reduce interrupt handling latency for critical interrupts. Vector calculation is described in detail in ~~"UIC0 Vector Register (UIC0_VR)" on page 9-42~~*UIC0 Vector Register (UIC0_VR)* on page 333.

9.2 UIC Features

- Support for 45 internal and 13 external interrupts
- Support for asynchronous level- or edge-sensitive interrupt types
- Programmable polarity for all interrupt types

- Programmable critical/non-critical interrupt selection for each interrupt bit
- Prioritized critical interrupt vector generation
- A UIC Status Register (UIC*x*_SR) providing the following information:
  - Current state of interrupts
  - Current state of all enabled interrupts (those masked using the UIC Enable Register (UIC*x*_ER))

9.3 UIC Interrupt Assignments

The UIC supports various internal and external interrupt sources as shown in Table 9-1 and Table 9-2.

Table 9-1. UIC0 Interrupt Assignments

Interrupt	Polarity	Sensitivity	Interrupt Source
0	High	Level	UART0
1	High	Level	UART1
2	High	Level	IIC 0
3	High	Level	IIC 1
4	High	Level	PCI Inbound Message
5	High	Level	PCI Command Write Register
6	High	Level	PCI Power Management
7	High	Edge	PCI MSI Level 0
8	High	Edge	PCI MSI Level 1
9	High	Edge	PCI MSI Level 2
10	High	Level	MAL TX EOB
11	High	Level	MAL RX EOB
12	High	Level	DMA Channel 0
13	High	Level	DMA Channel 1
14	High	Level	DMA Channel 2
15	High	Level	DMA Channel 3
16	Reserved		
17			
18	High	Level	GPT Compare Timer 0
19	High	Level	GPT Compare Timer 1
20	High	Level	GPT Compare Timer 2
21	High	Level	GPT Compare Timer 3
22	High	Level	GPT Compare Timer 4
23	Programmable	Programmable	External IRQ 0
24	Programmable	Programmable	External IRQ 1
25	Programmable	Programmable	External IRQ 2
26	Programmable	Programmable	External IRQ 3
27	Programmable	Programmable	External IRQ 4
28	Programmable	Programmable	External IRQ 5
29	Programmable	Programmable	External IRQ 6
30	Programmable	Programmable	UIC1 Non-Critical interrupt
31	Programmable	Programmable	UIC1 Critical Interrupt

Table 9-2. UIC1 Interrupt Assignments

Interrupt	Polarity	Sensitivity	Interrupt Source
0	High	Level	MAL SERR
1	High	Level	MAL TXDE
2	High	Level	MAL RXDE
3	High	Level	DDRSDRAM ECC Uncorrectable Error
4	High	Level	DDRSDRAM ECC Correctable Error
5	High	Level	External Bus Controller
6	High	Level	External Bus Master Interface
7	High	Level	OPB to PLB Bridge
8	High	Edge	PCI MSI Level 3
9	High	Edge	PCI MSI Level 4
10	High	Edge	PCI MSI Level 5
11	High	Edge	PCI MSI Level 6
12	High	Edge	PCI MSI Level 7
13	High	Edge	PCI MSI Level 8
14	High	Edge	PCI MSI Level 9
15	High	Edge	PCI MSI Level 10
16	High	Edge	PCI MSI Level 11
17	High	Level	PLB Performance Monitor
18	Programmable	Programmable	External IRQ 7
19	Programmable	Programmable	External IRQ 8
20	Programmable	Programmable	External IRQ 9
21	Programmable	Programmable	External IRQ 10
22	Programmable	Programmable	External IRQ 11
23	Programmable	Programmable	External IRQ 12
24	High	Level	Serial ROM Error
25	Reserved		
26			
27	High	Level	PCI Asynchronous Error
28	High	Level	EMAC 0
29	High	Level	EMAC 0 Wake-up
30	High	Level	EMAC 1
31	High	Level	EMAC 1 Wake-up

9.4 Interrupt Programmability

The on-chip interrupts and the external IRQs are programmable. However, the polarity and sensitivity of the on-chip interrupts must be programmed as shown in ~~Table 9-1, "UIC0 Interrupt Assignments," on page 9-3~~*Table 9-1 UIC0 Interrupt Assignments* on page 301 and ~~Table 9-2, "UIC1 Interrupt Assignments," on page 9-4~~*Table 9-2 UIC1 Interrupt Assignments* on page 302, using the UIC Polarity Register (UIC*x*_PR) and UIC Trigger Register (UIC*x*_TR), respectively.

9.5 UIC Registers

The UIC includes the Device Control Registers (DCRs) listed in Table 9-3.

The registers are accessed using the **mfdcr** and **mtdcr** instructions.

Table 9-3. UIC Device Control Registers

Mnemonic	Register	Address	Access	Page
UIC0_SR	UIC Status Register 0	0x0C0	Read/Clear	~~9-5~~303
UIC1_SR	UIC Status Register 1	0x0D0	Read/Clear	~~9-8~~306
UIC0_ER	UIC Enable Register 0	0x0C2	R/W	~~9-11~~308
UIC1_ER	UIC Enable Register 1	0x0D2	R/W	~~9-14~~310
UIC0_CR	UIC Critical Register 0	0x0C3	R/W	~~9-16~~313
UIC1_CR	UIC Critical Register 1	0x0D3	R/W	~~9-19~~315
UIC0_PR	UIC Polarity Register 0	0x0C4	R/W	~~9-21~~317
UIC1_PR	UIC Polarity Register 1	0x0D4	R/W	~~9-24~~320
UIC0_TR	UIC Trigger Register 0	0x0C5	R/W	~~9-28~~322
UIC1_TR	UIC Trigger Register 1	0x0D5	R/W	~~9-31~~324
UIC0_MSR	UIC Masked Status Register 0	0x0C6	Read-only	~~9-34~~327
UIC1_MSR	UIC Masked Status Register 1	0x0D6	Read-only	~~9-37~~329
UIC0_VCR	UIC Vector Configuration Register 0	0x0C7	Write-only	~~9-41~~332
UIC1_VCR	UIC Vector Configuration Register 1	0x0D7	Write-only	~~9-42~~332
UIC0_VR	UIC Vector Register 0	0x0C8	Read-only	~~9-42~~333
UIC1_VR	UIC Vector Register 1	0x0D8	Read-only	~~9-44~~335

9.5.1 UIC0 Status Register (UIC0_SR)

To report interrupt status, the UIC0_SR fields capture and hold internal and external interrupts until the fields are intentionally reset. To reset a field, write 1 to the field.

The values of other UIC registers do not affect UIC0_SR fields. ~~Figure 9-2~~Figure 9-2 describes UIC0_SR bit definitions.

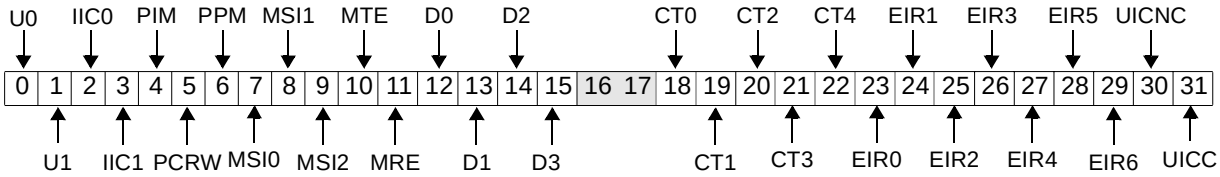


Figure 0-1. UIC0 Status Register (UIC0_SR)

0	U0	UART0 Interrupt Status 0 UART0 interrupt has not occurred. 1 UART0 interrupt occurred.
1	U1	UART1 Interrupt Status 0 UART1 interrupt has not occurred. 1 UART1 interrupt occurred.
2	IIC0	IIC0 Interrupt Status 0 IIC0 interrupt has not occurred. 1 IIC0 interrupt occurred.
3	IIC1	IIC1 Interrupt Status 0 IIC1 interrupt has not occurred. 1 IIC1 interrupt occurred.
4	PIM	PCI Inbound Message Interrupt Status 0 PCI inbound message interrupt has not occurred. 1 PCI inbound message interrupt occurred.
5	PCRW	PCI Command Register Write Interrupt Status 0 PCI command register write interrupt has not occurred. 1 PCI command register write interrupt occurred.
6	PPM	PCI Power Management Interrupt Status 0 PCI power management interrupt has not occurred. 1 PCI power management interrupt occurred.
7	MSI0	PCI MSI Level 0 Interrupt Status 0 PCI MSI Level 0 interrupt has not occurred. 1 PCI MSI Level 0 interrupt occurred.

8	MSI1	PCI MSI Level 1 Interrupt Status 0 PCI MSI Level 1 interrupt has not occurred. 1 PCI MSI Level 1 interrupt occurred.
9	MSI2	PCI MSI Level 2 Interrupt Status 0 PCI MSI Level 2 interrupt has not occurred. 1 PCI MSI Level 2 interrupt occurred.
10	MTE	MAL TX EOB Interrupt Status 0 MAL TX EOB interrupt has not occurred. 1 MAL TX EOB interrupt has occurred.
11	MRE	MAL RX EOB Interrupt Status 0 MAL RX EOB interrupt has not occurred. 1 MAL RX EOB interrupt has occurred.
12	D0	DMA Channel 0 Interrupt Status 0 DMA channel 0 interrupt has not occurred. 1 DMA channel 0 interrupt occurred.
13	D1	DMA Channel 1 Interrupt Status 0 DMA channel 1 interrupt has not occurred. 1 DMA channel 1 interrupt occurred.
14	D2	DMA Channel 2 Interrupt Status 0 DMA channel 2 interrupt has not occurred. 1 DMA channel 2 interrupt occurred.
15	D3	DMA Channel 3 Interrupt Status 0 DMA channel 3 interrupt has not occurred. 1 DMA channel 3 interrupt occurred.
16:17		Reserved
18	CT0	GPT Compare Timer 0 Interrupt Status 0 Compare timer 0 interrupt has not occurred. 1 Compare timer 0 interrupt occurred.
19	CT1	GPT Compare Timer 1 Interrupt Status 0 Compare timer1 interrupt has not occurred. 1 Compare timer 1 interrupt occurred.
20	CT2	GPT Compare Timer 2 Interrupt Status 0 Compare timer 2 interrupt has not occurred. 1 Compare timer 2 interrupt occurred.

21	CT3	GPT Compare Timer 3 Interrupt Status 0 Compare timer 3 interrupt has not occurred. 1 Compare timer 3 interrupt occurred.
22	CT4	GPT Compare Timer 4 Interrupt Status 0 Compare timer 4 interrupt has not occurred. 1 Compare timer 4 interrupt occurred.
23	EIR0	External IRQ 0 Interrupt Status 0 External IRQ 0 interrupt has not occurred. 1 External IRQ 0 interrupt occurred.
24	EIR1	External IRQ 1 Interrupt Status 0 External IRQ 1 interrupt has not occurred. 1 External IRQ 1 interrupt occurred.
25	EIR2	External IRQ 2 Interrupt Status 0 External IRQ 2 interrupt has not occurred. 1 External IRQ 2 interrupt occurred.
26	EIR3	External IRQ 3 Interrupt Status 0 External IRQ 3 interrupt has not occurred. 1 External IRQ 3 interrupt occurred.
27	EIR4	External IRQ 4 Interrupt Status 0 External IRQ 4 interrupt has not occurred. 1 External IRQ 4 interrupt occurred.
28	EIR5	External IRQ 5 Interrupt Status 0 External IRQ 5 interrupt has not occurred. 1 An external IRQ 5 interrupt occurred.
29	EIR6	External IRQ 6 Interrupt Status 0 External IRQ 6 interrupt has not occurred. 1 External IRQ 6 interrupt occurred.
30	UIC1NC	UIC1 Non-Critical Interrupt Status 0 UICNC interrupt has not occurred. 1 UICNC interrupt occurred.
31	UIC1C	UIC1 Critical Interrupt Status 0 UICC interrupt has not occurred. 1 UICC interrupt occurred.

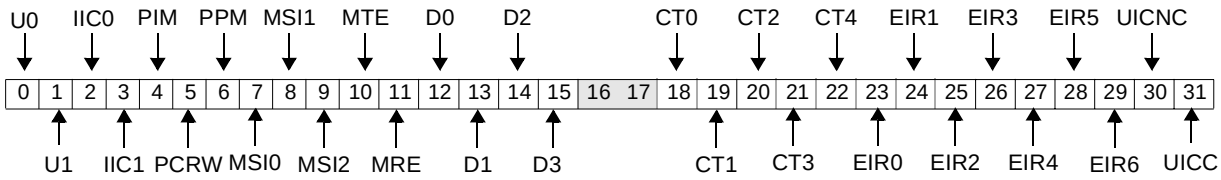


Figure 9-2. UIC0 Status Register (UIC0_SR)

0	U0	UART0 Interrupt Status 0 UART0 interrupt has not occurred. 1 UART0 interrupt occurred.
1	U1	UART1 Interrupt Status 0 UART1 interrupt has not occurred. 1 UART1 interrupt occurred.
2	IIC0	IIC0 Interrupt Status 0 IIC0 interrupt has not occurred. 1 IIC0 interrupt occurred.
3	IIC1	IIC1 Interrupt Status 0 IIC1 interrupt has not occurred. 1 IIC1 interrupt occurred.
4	PIM	PCI Inbound Message Interrupt Status 0 PCI inbound message interrupt has not occurred. 1 PCI inbound message interrupt occurred.
5	PCRW	PCI Command Register Write Interrupt Status 0 PCI command register write interrupt has not occurred. 1 PCI command register write interrupt occurred.
6	PPM	PCI Power Management Interrupt Status 0 PCI power management interrupt has not occurred. 1 PCI power management interrupt occurred.
7	MSI0	PCI MSI Level 0 Interrupt Status 0 PCI MSI Level 0 interrupt has not occurred. 1 PCI MSI Level 0 interrupt occurred.
8	MSI1	PCI MSI Level 1 Interrupt Status 0 PCI MSI Level 1 interrupt has not occurred. 1 PCI MSI Level 1 interrupt occurred.
9	MSI2	PCI MSI Level 2 Interrupt Status 0 PCI MSI Level 2 interrupt has not occurred. 1 PCI MSI Level 2 interrupt occurred.
10	MTE	MAL TX EOB Interrupt Status 0 MAL TX EOB interrupt has not occurred. 1 MAL TX EOB interrupt has occurred.
11	MRE	MAL RX EOB Interrupt Status 0 MAL RX EOB interrupt has not occurred. 1 MAL RX EOB interrupt has occurred.
12	D0	DMA Channel 0 Interrupt Status 0 DMA channel 0 interrupt has not occurred. 1 DMA channel 0 interrupt occurred.

13	D1	DMA Channel 1 Interrupt Status 0 DMA channel 1 interrupt has not occurred. 1 DMA channel 1 interrupt occurred.
14	D2	DMA Channel 2 Interrupt Status 0 DMA channel 2 interrupt has not occurred. 1 DMA channel 2 interrupt occurred.
15	D3	DMA Channel 3 Interrupt Status 0 DMA channel 3 interrupt has not occurred. 1 DMA channel 3 interrupt occurred.
16:17		Reserved
18	CT0	GPT Compare Timer 0 Interrupt Status 0 Compare timer 0 interrupt has not occurred. 1 Compare timer 0 interrupt occurred.
19	CT1	GPT Compare Timer 1 Interrupt Status 0 Compare timer 1 interrupt has not occurred. 1 Compare timer 1 interrupt occurred.
20	CT2	GPT Compare Timer 2 Interrupt Status 0 Compare timer 2 interrupt has not occurred. 1 Compare timer 2 interrupt occurred.
21	CT3	GPT Compare Timer 3 Interrupt Status 0 Compare timer 3 interrupt has not occurred. 1 Compare timer 3 interrupt occurred.
22	CT4	GPT Compare Timer 4 Interrupt Status 0 Compare timer 4 interrupt has not occurred. 1 Compare timer 4 interrupt occurred.
23	EIR0	External IRQ 0 Interrupt Status 0 External IRQ 0 interrupt has not occurred. 1 External IRQ 0 interrupt occurred.
24	EIR1	External IRQ 1 Interrupt Status 0 External IRQ 1 interrupt has not occurred. 1 External IRQ 1 interrupt occurred.
25	EIR2	External IRQ 2 Interrupt Status 0 External IRQ 2 interrupt has not occurred. 1 External IRQ 2 interrupt occurred.
26	EIR3	External IRQ 3 Interrupt Status 0 External IRQ 3 interrupt has not occurred. 1 External IRQ 3 interrupt occurred.
27	EIR4	External IRQ 4 Interrupt Status 0 External IRQ 4 interrupt has not occurred. 1 External IRQ 4 interrupt occurred.
28	EIR5	External IRQ 5 Interrupt Status 0 External IRQ 5 interrupt has not occurred. 1 An external IRQ 5 interrupt occurred.
29	EIR6	External IRQ 6 Interrupt Status 0 External IRQ 6 interrupt has not occurred. 1 External IRQ 6 interrupt occurred.
30	UIC1NC	UIC1 Non-Critical Interrupt Status 0 UICNC interrupt has not occurred. 1 UICNC interrupt occurred.
31	UIC1C	UIC1 Critical Interrupt Status 0 UICC interrupt has not occurred. 1 UICC interrupt occurred.

9.5.2 UIC1 Status Register (UIC1_SR)

To report interrupt status, the UIC1_SR fields capture and hold internal and external interrupts until the fields are intentionally reset. To reset a field, write 1 to the field.

The values of other UIC registers do not affect UIC1_SR fields. ~~Figure 9-3~~Figure 9-3 describes UIC1_SR bit definitions.

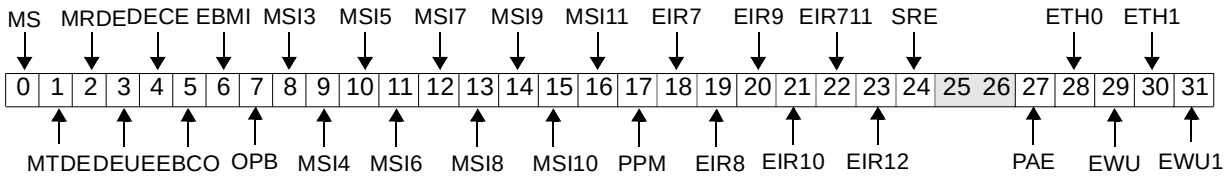


Figure 0-2. UIC1 Status Register (UIC1_SR)

0	MS	MAL SERR Interrupt Status 0 MAL SERR interrupt has not occurred. 1 MAL SERR interrupt occurred.
1	MTDE	MAL TXDE Interrupt Status 0 MAL TXDE interrupt has not occurred. 1 MAL TXDE interrupt occurred.
2	MRDE	MAL RXDE Interrupt Status 0 MAL RXDE interrupt has not occurred. 1 MAL RXDE interrupt occurred.
3	DEUE	DDRSDRAM ECC Uncorrectable Error Interrupt Status 0 DDRSDRAM ECC uncorrectable error interrupt has not occurred. 1 DDRSDRAM ECC uncorrectable error interrupt occurred.
4	DECE	DDRSDRAM ECC Correctable Error Interrupt Status 0 DDRSDRAM ECC correctable error interrupt has not occurred. 1 DDRSDRAM ECC correctable error interrupt occurred.
5	EBCO	EBCO Interrupt Status 0 EBCO interrupt has not occurred. 1 EBCO interrupt occurred.
6	EBMI	EBMI Interrupt Status 0 EBMI interrupt has not occurred. 1 EBMI interrupt occurred.

7	OPB	OPB to PLB Bridge Interrupt Status 0 OPB to PLB bridge interrupt has not occurred. 1 OPB to PLB bridge interrupt occurred.
8	MSI3	PCI MSI Level 3 Interrupt Status 0 PCI MSI level 3 interrupt has not occurred. 1 PCI MSI level 3 interrupt occurred.
9	MSI4	PCI MSI Level 4 Interrupt Status 0 PCI MSI level 4 interrupt has not occurred. 1 PCI MSI level 4 interrupt occurred.
10	MSI5	PCI MSI Level 5 Interrupt Status 0 PCI MSI level 5 interrupt has not occurred. 1 PCI MSI level 5 interrupt occurred.
11	MSI6	PCI MSI Level 6 Interrupt Status 0 PCI MSI level 6 interrupt has not occurred. 1 PCI MSI level 6 interrupt occurred.
12	MSI7	PCI MSI Level 7 Interrupt Status 0 PCI MSI level 7 interrupt has not occurred. 1 PCI MSI level 7 interrupt occurred.
13	MSI8	PCI MSI Level 8 Interrupt Status 0 PCI MSI level 8 interrupt has not occurred. 1 PCI MSI level 8 interrupt occurred.
14	MSI9	PCI MSI Level 9 Interrupt Status 0 PCI MSI level 9 interrupt has not occurred. 1 PCI MSI level 9 interrupt occurred.
15	MSI10	PCI MSI Level 10 Interrupt Status 0 PCI MSI level 10 interrupt has not occurred. 1 PCI MSI level 10 interrupt occurred.
16	MSI11	PCI MSI Level 11 Interrupt Status 0 PCI MSI level 11 interrupt has not occurred. 1 PCI MSI level 11 interrupt occurred.
17	PPM	PPM Interrupt Status 0 PPM interrupt has not occurred. 1 PPM interrupt occurred.
18	EIR7	External IRQ 7 Interrupt Status 0 External IRQ 7 interrupt has not occurred. 1 External IRQ 7 interrupt occurred.
19	EIR8	External IRQ 8 Interrupt Status 0 External IRQ 8 interrupt has not occurred. 1 External IRQ 8 interrupt occurred.
20	EIR9	External IRQ 9 Interrupt Status 0 External IRQ 9 interrupt has not occurred. 1 External IRQ 9 interrupt occurred.

21	EIR10	External IRQ 10 Interrupt Status 0 External IRQ 10 interrupt has not occurred. 1 External IRQ 10 interrupt occurred.
22	EIR11	External IRQ 11 Interrupt Status 0 External IRQ 11 interrupt has not occurred. 1 External IRQ 11 interrupt occurred.
23	EIR12	External IRQ 12 Interrupt Status 0 External IRQ 12 interrupt has not occurred. 1 External IRQ 12 interrupt occurred.
24	SRE	Serial ROM Error Interrupt Status 0 Serial ROM error interrupt has not occurred. 1 Serial ROM error interrupt occurred.
25:26		Reserved
27	PAE	PCI Asynchronous Error Interrupt Status 0 PCI ashynchronous error interrupt has not occurred. 1 PCI asynchronous error interrupt occurred.
28	ETH0	Ethernet 0 Interrupt Status 0 Ethernet 0 interrupt has not occurred. 1 Ethernet 0 interrupt occurred.
29	EWU0	Ethernet 0 Wake-up Interrupt Status 0 Ethernet 0 wake-up interrupt has not occurred. 1 Ethernet 0 wake-up interrupt occurred.
30	ETH1	Ethernet 1 Interrupt Status 0 Ethernet 1 interrupt has not occurred. 1 Ethernet 1 interrupt occurred.
31	EWU1	Ethernet 1 Wake-up Interrupt Status 0 Ethernet 1 wake-up interrupt has not occurred. 1 Ethernet 1 wake-up nterrupt occurred.

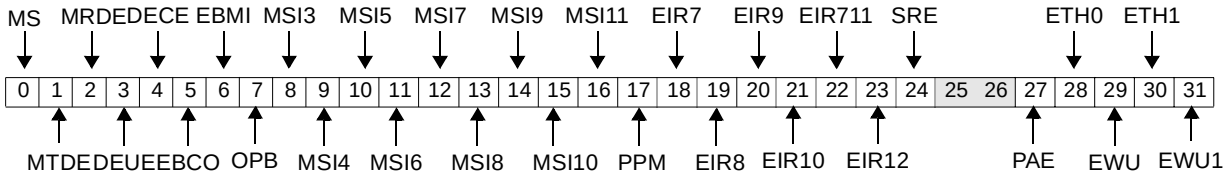


Figure 9-3. UIC1 Status Register (UIC1_SR)

0	MS	MAL SERR Interrupt Status 0 MAL SERR interrupt has not occurred. 1 MAL SERR interrupt occurred.
1	MTDE	MAL TXDE Interrupt Status 0 MAL TXDE interrupt has not occurred. 1 MAL TXDE interrupt occurred.

2	MRDE	MAL RXDE Interrupt Status 0 MAL RXDE interrupt has not occurred. 1 MAL RXDE interrupt occurred.
3	DEUE	DDRSDRAM ECC Uncorrectable Error Interrupt Status 0 DDRSDRAM ECC uncorrectable error interrupt has not occurred. 1 DDRSDRAM ECC uncorrectable error interrupt occurred.
4	DECE	DDRSDRAM ECC Correctable Error Interrupt Status 0 DDRSDRAM ECC correctable error interrupt has not occurred. 1 DDRSDRAM ECC correctable error interrupt occurred.
5	EBCO	EBCO Interrupt Status 0 EBCO interrupt has not occurred. 1 EBCO interrupt occurred.
6	EBMI	EBMI Interrupt Status 0 EBMI interrupt has not occurred. 1 EBMI interrupt occurred.
7	OPB	OPB to PLB Bridge Interrupt Status 0 OPB to PLB bridge interrupt has not occurred. 1 OPB to PLB bridge interrupt occurred.
8	MSI3	PCI MSI Level 3 Interrupt Status 0 PCI MSI level 3 interrupt has not occurred. 1 PCI MSI level 3 interrupt occurred.
9	MSI4	PCI MSI Level 4 Interrupt Status 0 PCI MSI level 4 interrupt has not occurred. 1 PCI MSI level 4 interrupt occurred.
10	MSI5	PCI MSI Level 5 Interrupt Status 0 PCI MSI level 5 interrupt has not occurred. 1 PCI MSI level 5 interrupt occurred.
11	MSI6	PCI MSI Level 6 Interrupt Status 0 PCI MSI level 6 interrupt has not occurred. 1 PCI MSI level 6 interrupt occurred.
12	MSI7	PCI MSI Level 7 Interrupt Status 0 PCI MSI level 7 interrupt has not occurred. 1 PCI MSI level 7 interrupt occurred.
13	MSI8	PCI MSI Level 8 Interrupt Status 0 PCI MSI level 8 interrupt has not occurred. 1 PCI MSI level 8 interrupt occurred.
14	MSI9	PCI MSI Level 9 Interrupt Status 0 PCI MSI level 9 interrupt has not occurred. 1 PCI MSI level 9 interrupt occurred.
15	MSI10	PCI MSI Level 10 Interrupt Status 0 PCI MSI level 10 interrupt has not occurred. 1 PCI MSI level 10 interrupt occurred.
16	MSI11	PCI MSI Level 11 Interrupt Status 0 PCI MSI level 11 interrupt has not occurred. 1 PCI MSI level 11 interrupt occurred.
17	PPM	PPM Interrupt Status 0 PPM interrupt has not occurred. 1 PPM interrupt occurred.

18	EIR7	External IRQ 7 Interrupt Status 0 External IRQ 7 interrupt has not occurred. 1 External IRQ 7 interrupt occurred.
19	EIR8	External IRQ 8 Interrupt Status 0 External IRQ 8 interrupt has not occurred. 1 External IRQ 8 interrupt occurred.
20	EIR9	External IRQ 9 Interrupt Status 0 External IRQ 9 interrupt has not occurred. 1 External IRQ 9 interrupt occurred.
21	EIR10	External IRQ 10 Interrupt Status 0 External IRQ 10 interrupt has not occurred. 1 External IRQ 10 interrupt occurred.
22	EIR11	External IRQ 11 Interrupt Status 0 External IRQ 11 interrupt has not occurred. 1 External IRQ 11 interrupt occurred.
23	EIR12	External IRQ 12 Interrupt Status 0 External IRQ 12 interrupt has not occurred. 1 External IRQ 12 interrupt occurred.
24	SRE	Serial ROM Error Interrupt Status 0 Serial ROM error interrupt has not occurred. 1 Serial ROM error interrupt occurred.
25:26		Reserved
27	PAE	PCI Asynchronous Error Interrupt Status 0 PCI ashynchronous error interrupt has not occurred. 1 PCI asynchronous error interrupt occurred.
28	ETH0	Ethernet 0 Interrupt Status 0 Ethernet 0 interrupt has not occurred. 1 Ethernet 0 interrupt occurred.
29	EWU0	Ethernet 0 Wake-up Interrupt Status 0 Ethernet 0 wake-up interrupt has not occurred. 1 Ethernet 0 wake-up interrupt occurred.
30	ETH1	Ethernet 1 Interrupt Status 0 Ethernet 1 interrupt has not occurred. 1 Ethernet 1 interrupt occurred.
31	EWU1	Ethernet 1 Wake-up Interrupt Status 0 Ethernet 1 wake-up interrupt has not occurred. 1 Ethernet 1 wake-up nterrupt occurred.

9.5.3 UIC0 Enable Register (UIC0_ER)

The fields of the UIC0_ER, which correspond to the fields of the UIC0_SR, enable or disable the reporting of the corresponding fields of the UIC0_SR.

If a UIC0_ER field is set to 1, the corresponding field of the UIC0_SR generates a critical or non-critical interrupt signal to the processor core, if the UIC0_SR field is set to 1. If a UIC0_ER field is set to 0, the corresponding field of the UIC0_SR does not generate a critical or non-critical interrupt signal to the processor core, regardless of the setting of the UIC0_SR field. The critical and non-critical interrupt signals in the processor core are controlled by fields in the Machine State Register (MSR).

The class of generated signals (critical or non-critical) is controlled by the UIC0_CR. ~~Figure 9-4~~*Figure 9-4* describes UIC0_ER bit definitions.

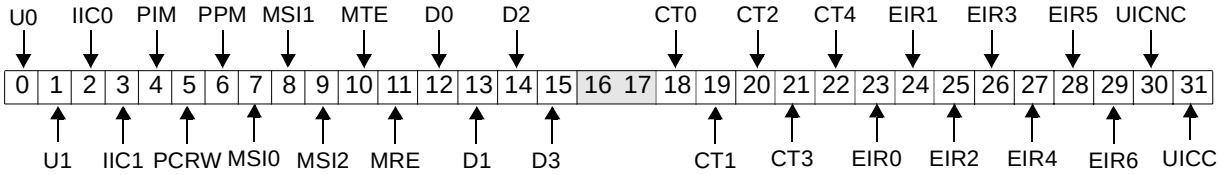


Figure 0-3. UIC0 Enable Register (UIC0_ER)

0	U0	UART0 Interrupt Enable 0 UART0 interrupt is disabled. 1 UART0 interrupt is enabled.
1	U1	UART1 Interrupt Enable 0 UART1 interrupt is disabled. 1 UART1 interrupt is enabled.
2	IIC0	IIC0 Interrupt Enable 0 IIC0 interrupt is disabled. 1 IIC0 interrupt is enabled.
3	IIC1	IIC1 Interrupt Enable 0 IIC1 interrupt is disabled. 1 IIC1 interrupt is enabled.
4	PIM	PCI Inbound Message Interrupt Enable 0 PCI inbound message interrupt is disabled. 1 PCI inbound message interrupt is enabled.
5	PCRW	PCI Command Register Write Interrupt Enable 0 PCI command register write interrupt is disabled. 1 PCI command register write interrupt is enabled.
6	PPM	PCI Power Management Interrupt Enable 0 PCI power management interrupt is disabled. 1 PCI power management interrupt is enabled.
7	MSI0	PCI MSI Level 0 Interrupt Enable 0 PCI MSI Level 0 interrupt is disabled. 1 PCI MSI Level 0 interrupt is enabled.
8	MSI1	PCI MSI Level 1 Interrupt Enable 0 PCI MSI Level 1 Interrupt is disabled. 1 PCI MSI Level 1 interrupt is enabled.
9	MSI2	PCI MSI Level 2 Interrupt Enable 0 PCI MSI Level 2 interrupt is disabled. 1 PCI MSI Level 2 interrupt is enabled.

10	MTE	MAL TX EOB Interrupt Enable 0 MAL TX EOB interrupt is disabled. 1 MAL TX EOB interrupt has is enabled.
11	MRE	MAL RX EOB Interrupt Enable 0 MAL RX EOB interrupt is disabled. 1 MAL RX EOB interrupt has is enabled.
12	D0	DMA Channel 0 Interrupt Enable 0 DMA channel 0 interrupt is disabled. 1 DMA channel 0 interrupt is enabled.
13	D1	DMA Channel 1 Interrupt Enable 0 DMA channel 1 interrupt is disabled. 1 DMA channel 1 interrupt is enabled.
14	D2	DMA Channel 2 Interrupt Enable 0 DMA channel 2 interrupt is disabled. 1 DMA channel 2 interrupt is enabled.
15	D3	DMA Channel 3 Interrupt Enable 0 DMA channel 3 interrupt is disabled. 1 DMA channel 3 interrupt is enabled.
16:17		Reserved
18	CT0	GPT Compare Timer 0 Interrupt Enable 0 Compare timer 0 interrupt is disabled. 1 Compare timer 0 interrupt is enabled.
19	CT1	GPT Compare Timer 1 Interrupt Enable 0 Compare timer1 interrupt is disabled. 1 Compare timer 1 interrupt is enabled.
20	CT2	GPT Compare Timer 2 Interrupt Enable 0 Compare timer 2 interrupt is disabled. 1 Compare timer 2 interrupt is enabled.
21	CT3	GPT Compare Timer 3 Interrupt Enable 0 Compare timer 3 interrupt is disabled. 1 Compare timer 3 interrupt is enabled.
22	CT4	GPT Compare Timer 4 Interrupt Enable 0 Compare timer 4 interrupt is disabled. 1 Compare timer 4 interrupt is enabled.
23	EIR0	External IRQ 0 Interrupt Enable 0 External IRQ 0 interrupt is disabled. 1 External IRQ 0 interrupt is enabled.
24	EIR1	External IRQ 1 Interrupt Enable 0 External IRQ 1 interrupt is disabled. 1 External IRQ 1 interrupt is enabled.
25	EIR2	External IRQ 2 Interrupt Enable 0 External IRQ 2 interrupt is disabled. 1 External IRQ 2 interrupt is enabled.

26	EIR3	External IRQ 3 Interrupt Enable 0 External IRQ 3 interrupt is disabled. 1 External IRQ 3 interrupt is enabled.
27	EIR4	External IRQ 4 Interrupt Enable 0 External IRQ 4 interrupt is disabled. 1 External IRQ 4 interrupt is enabled.
28	EIR5	External IRQ 5 Interrupt Enable 0 External IRQ 5 interrupt is disabled. 1 An external IRQ 5 interrupt is enabled.
29	EIR6	External IRQ 6 Interrupt Enable 0 External IRQ 6 interrupt is disabled. 1 External IRQ 6 interrupt is enabled.
30	UIC1NC	UIC1 Non-Critical Interrupt Enable 0 UICNC interrupt is disabled. 1 UICNC interrupt is enabled.
31	UIC1C	UIC1 Critical Interrupt Enable 0 UICC interrupt is disabled. 1 UICC interrupt is enabled.

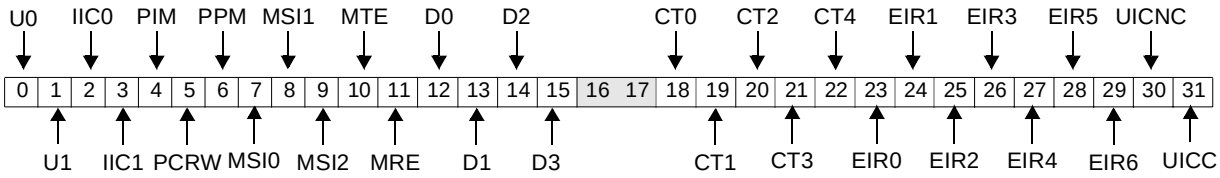


Figure 9-4. UIC0 Enable Register (UIC0_ER)

0	U0	UART0 Interrupt Enable 0 UART0 interrupt is disabled. 1 UART0 interrupt is enabled.
1	U1	UART1 Interrupt Enable 0 UART1 interrupt is disabled. 1 UART1 interrupt is enabled.
2	IIC0	IIC0 Interrupt Enable 0 IIC0 interrupt is disabled. 1 IIC0 interrupt is enabled.
3	IIC1	IIC1 Interrupt Enable 0 IIC1 interrupt is disabled. 1 IIC1 interrupt is enabled.
4	PIM	PCI Inbound Message Interrupt Enable 0 PCI inbound message interrupt is disabled. 1 PCI inbound message interrupt is enabled.
5	PCRW	PCI Command Register Write Interrupt Enable 0 PCI command register write interrupt is disabled. 1 PCI command register write interrupt is enabled.

6	PPM	PCI Power Management Interrupt Enable 0 PCI power management interrupt is disabled. 1 PCI power management interrupt is enabled.
7	MSI0	PCI MSI Level 0 Interrupt Enable 0 PCI MSI Level 0 interrupt is disabled. 1 PCI MSI Level 0 interrupt is enabled.
8	MSI1	PCI MSI Level 1 Interrupt Enable 0 PCI MSI Level 1 Interrupt is disabled. 1 PCI MSI Level 1 interrupt is enabled.
9	MSI2	PCI MSI Level 2 Interrupt Enable 0 PCI MSI Level 2 interrupt is disabled. 1 PCI MSI Level 2 interrupt is enabled.
10	MTE	MAL TX EOB Interrupt Enable 0 MAL TX EOB interrupt is disabled. 1 MAL TX EOB interrupt has is enabled.
11	MRE	MAL RX EOB Interrupt Enable 0 MAL RX EOB interrupt is disabled. 1 MAL RX EOB interrupt has is enabled.
12	D0	DMA Channel 0 Interrupt Enable 0 DMA channel 0 interrupt is disabled. 1 DMA channel 0 interrupt is enabled.
13	D1	DMA Channel 1 Interrupt Enable 0 DMA channel 1 interrupt is disabled. 1 DMA channel 1 interrupt is enabled.
14	D2	DMA Channel 2 Interrupt Enable 0 DMA channel 2 interrupt is disabled. 1 DMA channel 2 interrupt is enabled.
15	D3	DMA Channel 3 Interrupt Enable 0 DMA channel 3 interrupt is disabled. 1 DMA channel 3 interrupt is enabled.
16:17		Reserved
18	CT0	GPT Compare Timer 0 Interrupt Enable 0 Compare timer 0 interrupt is disabled. 1 Compare timer 0 interrupt is enabled.
19	CT1	GPT Compare Timer 1 Interrupt Enable 0 Compare timer 1 interrupt is disabled. 1 Compare timer 1 interrupt is enabled.
20	CT2	GPT Compare Timer 2 Interrupt Enable 0 Compare timer 2 interrupt is disabled. 1 Compare timer 2 interrupt is enabled.
21	CT3	GPT Compare Timer 3 Interrupt Enable 0 Compare timer 3 interrupt is disabled. 1 Compare timer 3 interrupt is enabled.
22	CT4	GPT Compare Timer 4 Interrupt Enable 0 Compare timer 4 interrupt is disabled. 1 Compare timer 4 interrupt is enabled.
23	EIR0	External IRQ 0 Interrupt Enable 0 External IRQ 0 interrupt is disabled. 1 External IRQ 0 interrupt is enabled.
24	EIR1	External IRQ 1 Interrupt Enable 0 External IRQ 1 interrupt is disabled. 1 External IRQ 1 interrupt is enabled.

25	EIR2	External IRQ 2 Interrupt Enable 0 External IRQ 2 interrupt is disabled. 1 External IRQ 2 interrupt is enabled.
26	EIR3	External IRQ 3 Interrupt Enable 0 External IRQ 3 interrupt is disabled. 1 External IRQ 3 interrupt is enabled.
27	EIR4	External IRQ 4 Interrupt Enable 0 External IRQ 4 interrupt is disabled. 1 External IRQ 4 interrupt is enabled.
28	EIR5	External IRQ 5 Interrupt Enable 0 External IRQ 5 interrupt is disabled. 1 An external IRQ 5 interrupt is enabled.
29	EIR6	External IRQ 6 Interrupt Enable 0 External IRQ 6 interrupt is disabled. 1 External IRQ 6 interrupt is enabled.
30	UIC1NC	UIC1 Non-Critical Interrupt Enable 0 UICNC interrupt is disabled. 1 UICNC interrupt is enabled.
31	UIC1C	UIC1 Critical Interrupt Enable 0 UICC interrupt is disabled. 1 UICC interrupt is enabled.

9.5.4 UIC1 Enable Register (UIC1_ER)

The fields of the UIC1_ER, which correspond to the fields of the UIC1_SR, enable or disable the reporting of the corresponding fields of the UIC1_SR.

If a UIC1_ER field is set to 1, the corresponding field of the UIC1_SR generates a critical or non-critical interrupt signal to the processor core, if the UIC1_SR field is set to 1. If a UIC1_ER field is set to 0, the corresponding field of the UIC1_SR does not generate a critical or non-critical interrupt signal to the processor core, regardless of the setting of the UIC1_SR field. The critical and non-critical interrupt signals in the processor core are controlled by fields in the Machine State Register (MSR).

The class of generated signals (critical or non-critical) is controlled by the UIC1_CR. ~~Figure 9-5~~Figure 9-5 described UIC1_ER bit definitions.

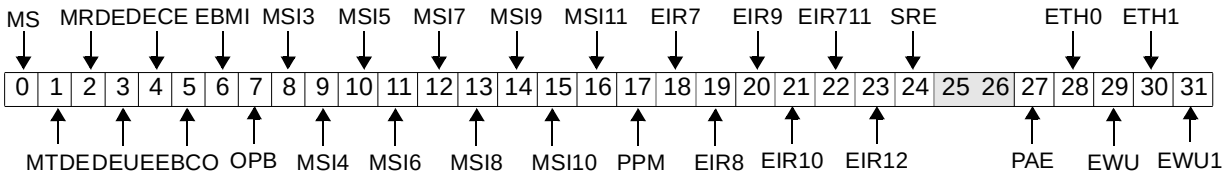


Figure 0-4. UIC1 Enable Register (UIC1_ER)

0	MS	MAL SERR Interrupt Enable 0 MAL SERR interrupt is disabled. 1 MAL SERR interrupt is enabled.
1	MTDE	MAL TXDE Interrupt Enable 0 MAL TXDE interrupt is disabled. 1 MAL TXDE interrupt is enabled.

2	MRDE	MAL RXDE Interrupt Enable 0 MAL RXDE interrupt is disabled. 1 MAL RXDE interrupt is enabled.
3	DEUE	DDRSDRAM ECC Uncorrectable Error Interrupt Enable 0 DDRSDRAM ECC uncorrectable error interrupt is disabled. 1 DDRSDRAM ECC uncorrectable error interrupt is enabled.
4	DECE	DDRSDRAM ECC Correctable Error Interrupt Enable 0 DDRSDRAM ECC correctable error interrupt is disabled. 1 DDRSDRAM ECC correctable error interrupt is enabled.
5	EBCO	EBCO Interrupt Enable 0 EBCO interrupt is disabled. 1 EBCO interrupt is enabled.
6	EBMI	EBMI Interrupt Enable 0 EBMI interrupt is disabled. 1 EBMI interrupt is enabled.
7	OPB	OPB to PLB Bridge Interrupt Enable 0 OPB to PLB bridge interrupt is disabled. 1 OPB to PLB bridge interrupt is enabled.
8	MSI3	PCI MSI Level 3 Interrupt Enable 0 PCI MSI level 3 interrupt is disabled. 1 PCI MSI level 3 interrupt is enabled.
9	MSI4	PCI MSI Level 4 Interrupt Enable 0 PCI MSI level 4 interrupt is disabled. 1 PCI MSI level 4 interrupt is enabled.
10	MSI5	PCI MSI Level 5 Interrupt Enable 0 PCI MSI level 5 interrupt is disabled. 1 PCI MSI level 5 interrupt is enabled.
11	MSI6	PCI MSI Level 6 Interrupt Enable 0 PCI MSI level 6 interrupt is disabled. 1 PCI MSI level 6 interrupt is enabled.
12	MSI7	PCI MSI Level 7 Interrupt Enable 0 PCI MSI level 7 interrupt is disabled. 1 PCI MSI level 7 interrupt is enabled.
13	MSI8	PCI MSI Level 8 Interrupt Enable 0 PCI MSI level 8 interrupt is disabled. 1 PCI MSI level 8 interrupt is enabled.
14	MSI9	PCI MSI Level 9 Interrupt Enable 0 PCI MSI level 9 interrupt is disabled. 1 PCI MSI level 9 interrupt is enabled.

15	MSI10	PCI MSI Level 10 Interrupt Enable 0 PCI MSI level 10 interrupt is disabled. 1 PCI MSI level 10 interrupt is enabled.
16	MSI11	PCI MSI Level 11 Interrupt Enable 0 PCI MSI level 11 interrupt is disabled. 1 PCI MSI level 11 interrupt is enabled.
17	PPM	PPM Interrupt Enable 0 PPM interrupt is disabled. 1 PPM interrupt is enabled.
18	EIR7	External IRQ 7 Interrupt Enable 0 External IRQ 7 interrupt is disabled. 1 External IRQ 7 interrupt is enabled.
19	EIR8	External IRQ 8 Interrupt Enable 0 External IRQ 8 interrupt is disabled. 1 External IRQ 8 interrupt is enabled.
20	EIR9	External IRQ 9 Interrupt Enable 0 External IRQ 9 interrupt is disabled. 1 External IRQ 9 interrupt is enabled.
21	EIR10	External IRQ 10 Interrupt Enable 0 External IRQ 10 interrupt is disabled. 1 External IRQ 10 interrupt is enabled.
22	EIR11	External IRQ 11 Interrupt Enable 0 External IRQ 11 interrupt is disabled. 1 External IRQ 11 interrupt is enabled.
23	EIR12	External IRQ 12 Interrupt Enable 0 External IRQ 12 interrupt is disabled. 1 External IRQ 12 interrupt is enabled.
24	SRE	Serial ROM Error Interrupt Enable 0 Serial ROM error interrupt is disabled. 1 Serial ROM error interrupt is enabled.
25:26		Reserved
27	PAE	PCI Asynchronous Error Interrupt Enable 0 PCI asynchronous error interrupt is disabled. 1 PCI asynchronous error interrupt is enabled.
28	ETH0	Ethernet 0 Interrupt Enable 0 Ethernet 0 interrupt is disabled. 1 Ethernet 0 interrupt is enabled.
29	EWU0	Ethernet 0 Wake-up Interrupt Enable 0 Ethernet 0 wake-up interrupt is disabled. 1 Ethernet 0 wake-up interrupt is enabled.
30	ETH1	Ethernet 1 Interrupt Enable 0 Ethernet 1 interrupt is disabled. 1 Ethernet 1 interrupt is enabled.

31	EWU1	Ethernet 1 Wake-up Interrupt Enable 0 Ethernet 1 wake-up interrupt is disabled. 1 Ethernet 1 wake-up interrupt is enabled.

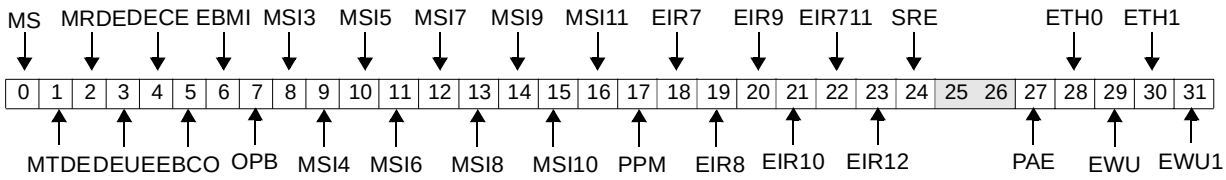


Figure 9-5. UIC1 Enable Register (UIC1_ER)

0	MS	MAL SERR Interrupt Enable 0 MAL SERR interrupt is disabled. 1 MAL SERR interrupt is enabled.
1	MTDE	MAL TXDE Interrupt Enable 0 MAL TXDE interrupt is disabled. 1 MAL TXDE interrupt is enabled.
2	MRDE	MAL RXDE Interrupt Enable 0 MAL RXDE interrupt is disabled. 1 MAL RXDE interrupt is enabled.
3	DEUE	DDRSDRAM ECC Uncorrectable Error Interrupt Enable 0 DDRSDRAM ECC uncorrectable error interrupt is disabled. 1 DDRSDRAM ECC uncorrectable error interrupt is enabled.
4	DECE	DDRSDRAM ECC Correctable Error Interrupt Enable 0 DDRSDRAM ECC correctable error interrupt is disabled. 1 DDRSDRAM ECC correctable error interrupt is enabled.
5	EBCO	EBCO Interrupt Enable 0 EBCO interrupt is disabled. 1 EBCO interrupt is enabled.
6	EBMI	EBMI Interrupt Enable 0 EBMI interrupt is disabled. 1 EBMI interrupt is enabled.
7	OPB	OPB to PLB Bridge Interrupt Enable 0 OPB to PLB bridge interrupt is disabled. 1 OPB to PLB bridge interrupt is enabled.
8	MSI3	PCI MSI Level 3 Interrupt Enable 0 PCI MSI level 3 interrupt is disabled. 1 PCI MSI level 3 interrupt is enabled.
9	MSI4	PCI MSI Level 4 Interrupt Enable 0 PCI MSI level 4 interrupt is disabled. 1 PCI MSI level 4 interrupt is enabled.
10	MSI5	PCI MSI Level 5 Interrupt Enable 0 PCI MSI level 5 interrupt is disabled. 1 PCI MSI level 5 interrupt is enabled.

11	MSI6	PCI MSI Level 6 Interrupt Enable 0 PCI MSI level 6 interrupt is disabled. 1 PCI MSI level 6 interrupt is enabled.
12	MSI7	PCI MSI Level 7 Interrupt Enable 0 PCI MSI level 7 interrupt is disabled. 1 PCI MSI level 7 interrupt is enabled.
13	MSI8	PCI MSI Level 8 Interrupt Enable 0 PCI MSI level 8 interrupt is disabled. 1 PCI MSI level 8 interrupt is enabled.
14	MSI9	PCI MSI Level 9 Interrupt Enable 0 PCI MSI level 9 interrupt is disabled. 1 PCI MSI level 9 interrupt is enabled.
15	MSI10	PCI MSI Level 10 Interrupt Enable 0 PCI MSI level 10 interrupt is disabled. 1 PCI MSI level 10 interrupt is enabled.
16	MSI11	PCI MSI Level 11 Interrupt Enable 0 PCI MSI level 11 interrupt is disabled. 1 PCI MSI level 11 interrupt is enabled.
17	PPM	PPM Interrupt Enable 0 PPM interrupt is disabled. 1 PPM interrupt is enabled.
18	EIR7	External IRQ 7 Interrupt Enable 0 External IRQ 7 interrupt is disabled. 1 External IRQ 7 interrupt is enabled.
19	EIR8	External IRQ 8 Interrupt Enable 0 External IRQ 8 interrupt is disabled. 1 External IRQ 8 interrupt is enabled.
20	EIR9	External IRQ 9 Interrupt Enable 0 External IRQ 9 interrupt is disabled. 1 External IRQ 9 interrupt is enabled.
21	EIR10	External IRQ 10 Interrupt Enable 0 External IRQ 10 interrupt is disabled. 1 External IRQ 10 interrupt is enabled.
22	EIR11	External IRQ 11 Interrupt Enable 0 External IRQ 11 interrupt is disabled. 1 External IRQ 11 interrupt is enabled.
23	EIR12	External IRQ 12 Interrupt Enable 0 External IRQ 12 interrupt is disabled. 1 External IRQ 12 interrupt is enabled.
24	SRE	Serial ROM Error Interrupt Enable 0 Serial ROM error interrupt is disabled. 1 Serial ROM error interrupt is enabled.
25:26		Reserved
27	PAE	PCI Asynchronous Error Interrupt Enable 0 PCI asynchronous error interrupt is disabled. 1 PCI asynchronous error interrupt is enabled.
28	ETH0	Ethernet 0 Interrupt Enable 0 Ethernet 0 interrupt is disabled. 1 Ethernet 0 interrupt is enabled.
29	EWU0	Ethernet 0 Wake-up Interrupt Enable 0 Ethernet 0 wake-up interrupt is disabled. 1 Ethernet 0 wake-up interrupt is enabled.

30	ETH1	Ethernet 1 Interrupt Enable 0 Ethernet 1 interrupt is disabled. 1 Ethernet 1 interrupt is enabled.
31	EWU1	Ethernet 1 Wake-up Interrupt Enable 0 Ethernet 1 wake-up interrupt is disabled. 1 Ethernet 1 wake-up interrupt is enabled.

9.5.5 UIC0 Critical Register (UIC0_CR)

The fields of the UIC0_CR, which correspond to the fields of the UIC0_SR and UIC0_ER, determine whether an interrupt captured in the corresponding fields of the UIC0_SR generates a non-critical or critical interrupt, if the interrupts are enabled in the corresponding fields of the UIC0_ER. The processor handles critical interrupts when MSR[EE] = 1 and non-critical interrupts when MSR[CE]=1.

If a UIC0_CR field is set to 0, an enabled interrupt (captured in the corresponding field of the UIC0_SR and enabled in the corresponding field of the UIC0_ER) generates a non-critical interrupt signal to the processor core. If a UIC0_CR field is a 1, a critical interrupt signal is generated. ~~Figure 9-6~~*Figure 9-6* describes UIC0_CR bit definitions.

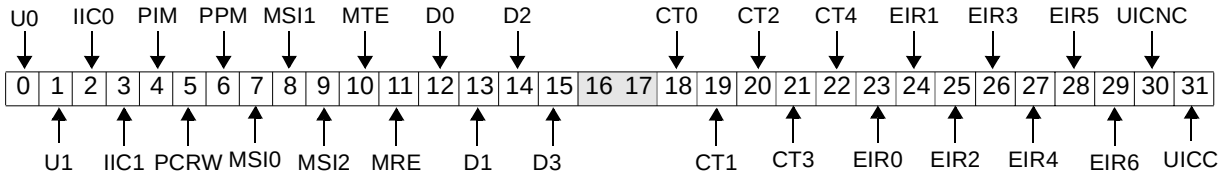


Figure 0-5. UIC0 Critical Register (UIC0_CR)

0	U0	UART0 Interrupt Class 0 UART0 interrupt non-critical. 1 UART0 interrupt is critical.
1	U1	UART1 Interrupt Class 0 UART1 interrupt is non-critical. 1 UART1 interrupt is critical.
2	IIC0	IIC0 Interrupt Class 0 IIC0 interrupt is non-critical. 1 IIC0 interrupt is critical.
3	IIC1	IIC1 Interrupt Class 0 IIC1 interrupt is non-critical. 1 IIC1 interrupt is critical.
4	PIM	PCI Inbound Message Interrupt Class 0 PCI inbound message interrupt is non-critical. 1 PCI inbound message interrupt is critical.

5	PCRW	PCI Command Register Write Interrupt Class 0 PCI command register write interrupt is non-critical. 1 PCI command register write interrupt is critical.
6	PPM	PCI Power Management Interrupt Class 0 PCI power management interrupt is non-critical. 1 PCI power management interrupt is critical.
7	MSI0	PCI MSI Level 0 Interrupt Class 0 PCI MSI level 0 interrupt is non-critical. 1 PCI MSI level 0 interrupt is critical.
8	MSI1	PCI MSI Level 1 Interrupt Class 0 PCI MSI level 1 interrupt is non-critical. 1 PCI MSI level 1 interrupt is critical.
9	MSI2	PCI MSI Level 2 Interrupt Class 0 PCI MSI level 2 interrupt is non-critical. 1 PCI MSI level 2 interrupt is critical.
10	MTE	MAL TX EOB Interrupt Class 0 MAL TX EOB interrupt is non-critical. 1 MAL TX EOB interrupt has is critical.
11	MRE	MAL RX EOB Interrupt Class 0 MAL RX EOB interrupt is non-critical. 1 MAL RX EOB interrupt has is critical.
12	D0	DMA Channel 0 Interrupt Class 0 DMA channel 0 interrupt is non-critical. 1 DMA channel 0 interrupt is critical.
13	D1	DMA Channel 1 Interrupt Class 0 DMA channel 1 interrupt is non-critical. 1 DMA channel 1 interrupt is critical.
14	D2	DMA Channel 2 Interrupt Class 0 DMA channel 2 interrupt is non-critical. 1 DMA channel 2 interrupt is critical.
15	D3	DMA Channel 3 Interrupt Class 0 DMA channel 3 interrupt is non-critical. 1 DMA channel 3 interrupt is critical.
16:17		Reserved
18	CT0	GPT Compare Timer 0 Interrupt Class 0 Compare timer 0 interrupt is non-critical. 1 Compare timer 0 interrupt is critical.
19	CT1	GPT Compare Timer 1 Interrupt Class 0 Compare timer1 interrupt is non-critical. 1 Compare timer 1 interrupt is critical.

20	CT2	GPT Compare Timer 2 Interrupt Class 0 Compare timer 2 interrupt is non-critical. 1 Compare timer 2 interrupt is critical.
21	CT3	GPT Compare Timer 3 Interrupt Class 0 Compare timer 3 interrupt is non-critical. 1 Compare timer 3 interrupt is critical.
22	CT4	GPT Compare Timer 4 Interrupt Class 0 Compare timer 4 interrupt is non-critical. 1 Compare timer 4 interrupt is critical.
23	EIR0	External IRQ 0 Interrupt Class 0 External IRQ 0 interrupt is non-critical. 1 External IRQ 0 interrupt is critical.
24	EIR1	External IRQ 1 Interrupt Class 0 External IRQ 1 interrupt is non-critical. 1 External IRQ 1 interrupt is critical.
25	EIR2	External IRQ 2 Interrupt Class 0 External IRQ 2 interrupt is non-critical. 1 External IRQ 2 interrupt is critical.
26	EIR3	External IRQ 3 Interrupt Class 0 External IRQ 3 interrupt is non-critical. 1 External IRQ 3 interrupt is critical.
27	EIR4	External IRQ 4 Interrupt Class 0 External IRQ 4 interrupt is non-critical. 1 External IRQ 4 interrupt is critical.
28	EIR5	External IRQ 5 Interrupt Class 0 External IRQ 5 interrupt is non-critical. 1 An external IRQ 5 interrupt is critical.
29	EIR6	External IRQ 6 Interrupt Class 0 External IRQ 6 interrupt is non-critical. 1 External IRQ 6 interrupt is critical.
30	UIC1NC	UIC1 Non-Critical Interrupt Class 0 UICNC interrupt is non-critical. 1 UICNC interrupt is critical.
31	UIC1C	UIC1 Critical Interrupt Class 0 UICC interrupt is non-critical. 1 UICC interrupt is critical.

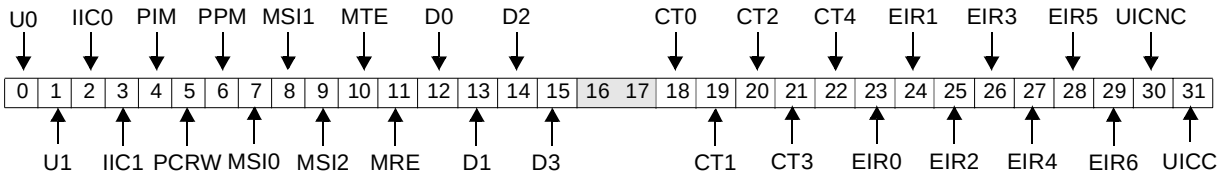


Figure 9-6. UIC0 Critical Register (UIC0_CR)

0	U0	UART0 Interrupt Class 0 UART0 interrupt non-critical. 1 UART0 interrupt is critical.
1	U1	UART1 Interrupt Class 0 UART1 interrupt is non-critical. 1 UART1 interrupt is critical.
2	IIC0	IIC0 Interrupt Class 0 IIC0 interrupt is non-critical. 1 IIC0 interrupt is critical.
3	IIC1	IIC1 Interrupt Class 0 IIC1 interrupt is non-critical. 1 IIC1 interrupt is critical.
4	PIM	PCI Inbound Message Interrupt Class 0 PCI inbound message interrupt is non-critical. 1 PCI inbound message interrupt is critical.
5	PCRW	PCI Command Register Write Interrupt Class 0 PCI command register write interrupt is non-critical. 1 PCI command register write interrupt is critical.
6	PPM	PCI Power Management Interrupt Class 0 PCI power management interrupt is non-critical. 1 PCI power management interrupt is critical.
7	MSI0	PCI MSI Level 0 Interrupt Class 0 PCI MSI level 0 interrupt is non-critical. 1 PCI MSI level 0 interrupt is critical.
8	MSI1	PCI MSI Level 1 Interrupt Class 0 PCI MSI level 1 interrupt is non-critical. 1 PCI MSI level 1 interrupt is critical.
9	MSI2	PCI MSI Level 2 Interrupt Class 0 PCI MSI level 2 interrupt is non-critical. 1 PCI MSI level 2 interrupt is critical.
10	MTE	MAL TX EOB Interrupt Class 0 MAL TX EOB interrupt is non-critical. 1 MAL TX EOB interrupt has is critical.
11	MRE	MAL RX EOB Interrupt Class 0 MAL RX EOB interrupt is non-critical. 1 MAL RX EOB interrupt has is critical.
12	D0	DMA Channel 0 Interrupt Class 0 DMA channel 0 interrupt is non-critical. 1 DMA channel 0 interrupt is critical.

13	D1	DMA Channel 1 Interrupt Class 0 DMA channel 1 interrupt is non-critical. 1 DMA channel 1 interrupt is critical.
14	D2	DMA Channel 2 Interrupt Class 0 DMA channel 2 interrupt is non-critical. 1 DMA channel 2 interrupt is critical.
15	D3	DMA Channel 3 Interrupt Class 0 DMA channel 3 interrupt is non-critical. 1 DMA channel 3 interrupt is critical.
16:17		Reserved
18	CT0	GPT Compare Timer 0 Interrupt Class 0 Compare timer 0 interrupt is non-critical. 1 Compare timer 0 interrupt is critical.
19	CT1	GPT Compare Timer 1 Interrupt Class 0 Compare timer 1 interrupt is non-critical. 1 Compare timer 1 interrupt is critical.
20	CT2	GPT Compare Timer 2 Interrupt Class 0 Compare timer 2 interrupt is non-critical. 1 Compare timer 2 interrupt is critical.
21	CT3	GPT Compare Timer 3 Interrupt Class 0 Compare timer 3 interrupt is non-critical. 1 Compare timer 3 interrupt is critical.
22	CT4	GPT Compare Timer 4 Interrupt Class 0 Compare timer 4 interrupt is non-critical. 1 Compare timer 4 interrupt is critical.
23	EIR0	External IRQ 0 Interrupt Class 0 External IRQ 0 interrupt is non-critical. 1 External IRQ 0 interrupt is critical.
24	EIR1	External IRQ 1 Interrupt Class 0 External IRQ 1 interrupt is non-critical. 1 External IRQ 1 interrupt is critical.
25	EIR2	External IRQ 2 Interrupt Class 0 External IRQ 2 interrupt is non-critical. 1 External IRQ 2 interrupt is critical.
26	EIR3	External IRQ 3 Interrupt Class 0 External IRQ 3 interrupt is non-critical. 1 External IRQ 3 interrupt is critical.
27	EIR4	External IRQ 4 Interrupt Class 0 External IRQ 4 interrupt is non-critical. 1 External IRQ 4 interrupt is critical.
28	EIR5	External IRQ 5 Interrupt Class 0 External IRQ 5 interrupt is non-critical. 1 An external IRQ 5 interrupt is critical.
29	EIR6	External IRQ 6 Interrupt Class 0 External IRQ 6 interrupt is non-critical. 1 External IRQ 6 interrupt is critical.
30	UIC1NC	UIC1 Non-Critical Interrupt Class 0 UICNC interrupt is non-critical. 1 UICNC interrupt is critical.
31	UIC1C	UIC1 Critical Interrupt Class 0 UICC interrupt is non-critical. 1 UICC interrupt is critical.

9.5.6 UIC1 Critical Register (UIC1_CR)

The fields of the UIC1_CR, which correspond to the fields of the UIC1_SR and UIC1_ER, determine whether an interrupt captured in the corresponding fields of the UIC1_SR generates a non-critical or critical interrupt, if the interrupts are enabled in the corresponding fields of the UIC1_ER. The processor handles critical interrupts when MSR[EE] = 1 and non-critical interrupts when MSR[CE]=1.

If a UIC1_CR field is set to 0, an enabled interrupt (captured in the corresponding field of the UIC1_SR and enabled in the corresponding field of the UIC1_ER) generates a non-critical interrupt signal to the processor core. If a UIC1_CR field is a 1, a critical interrupt signal is generated. ~~Figure 9-7~~Figure 9-7 describes UIC1_CR bit definitions.

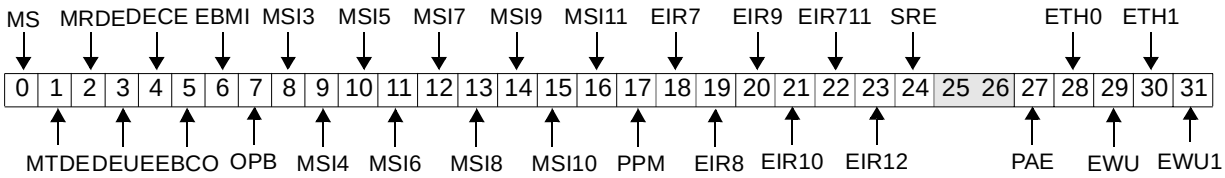


Figure 0-6. UIC1 Critical Register (UIC1_CR)

0	MS	MAL SERR Interrupt Class 0 MAL SERR interrupt is non-critical. 1 MAL SERR interrupt is critical.
1	MTDE	MAL TXDE Interrupt Class 0 MAL TXDE interrupt is non-critical. 1 MAL TXDE interrupt is critical.
2	MRDE	MAL RXDE Interrupt Class 0 MAL RXDE interrupt is non-critical. 1 MAL RXDE interrupt is critical.
3	DEUE	DDRSDRAM ECC Uncorrectable Error Interrupt Class 0 DDRSDRAM ECC uncorrectable error interrupt is non-critical. 1 DDRSDRAM ECC uncorrectable error interrupt is critical.
4	DECE	DDRSDRAM ECC Correctable Error Interrupt Class 0 DDRSDRAM ECC correctable error interrupt is non-critical. 1 DDRSDRAM ECC correctable error interrupt is critical.
5	EBCO	EBCO Interrupt Class 0 EBCO interrupt is non-critical. 1 EBCO interrupt is critical.
6	EBMI	EBMI Interrupt Class 0 EBMI interrupt is non-critical. 1 EBMI interrupt is critical.

7	OPB	OPB to PLB Bridge Interrupt Class 0 OPB to PLB bridge interrupt is non-critical. 1 OPB to PLB bridge interrupt is critical.
8	MSI3	PCI MSI Level 3 Interrupt Class 0 PCI MSI level 3 interrupt is non-critical. 1 PCI MSI level 3 interrupt is critical.
9	MSI4	PCI MSI Level 4 Interrupt Class 0 PCI MSI level 4 interrupt is non-critical. 1 PCI MSI level 4 interrupt is critical.
10	MSI5	PCI MSI Level 5 Interrupt Class 0 PCI MSI level 5 interrupt is non-critical. 1 PCI MSI level 5 interrupt is critical.
11	MSI6	PCI MSI Level 6 Interrupt Class 0 PCI MSI level 6 interrupt is non-critical. 1 PCI MSI level 6 interrupt is critical.
12	MSI7	PCI MSI Level 7 Interrupt Class 0 PCI MSI level 7 interrupt is non-critical. 1 PCI MSI level 7 interrupt is critical.
13	MSI8	PCI MSI Level 8 Interrupt Class 0 PCI MSI level 8 interrupt is non-critical. 1 PCI MSI level 8 interrupt is critical.
14	MSI9	PCI MSI Level 9 Interrupt Class 0 PCI MSI level 9 interrupt is non-critical. 1 PCI MSI level 9 interrupt is critical.
15	MSI10	PCI MSI Level 10 Interrupt Class 0 PCI MSI level 10 interrupt is non-critical. 1 PCI MSI level 10 interrupt is critical.
16	MS11	PCI MSI Level 11 Interrupt Class 0 PCI MSI level 11 interrupt is non-critical. 1 PCI MSI level 11 interrupt is critical.
17	PPM	PPM Interrupt Class 0 PPM interrupt is non-critical. 1 PPM interrupt is critical.
18	EIR7	External IRQ 7 Interrupt Class 0 External IRQ 7 interrupt is non-critical. 1 External IRQ 7 interrupt is critical.
19	EIR8	External IRQ 8 Interrupt Class 0 External IRQ 8 interrupt is non-critical. 1 External IRQ 8 interrupt is critical.
20	EIR9	External IRQ 9 Interrupt Class 0 External IRQ 9 interrupt is non-critical. 1 External IRQ 9 interrupt is critical.

21	EIR10	External IRQ 10 Interrupt Class 0 External IRQ 10 interrupt is non-critical. 1 External IRQ 10 interrupt is critical.
22	EIR11	External IRQ 11 Interrupt Class 0 External IRQ 11 interrupt is non-critical. 1 External IRQ 11 interrupt is critical.
23	EIR12	External IRQ 12 Interrupt Class 0 External IRQ 12 interrupt is non-critical. 1 External IRQ 12 interrupt is critical.
24	SRE	Serial ROM Error Interrupt Class 0 Serial ROM error interrupt is non-critical. 1 Serial ROM error interrupt is critical.
25:26		Reserved
27	PAE	PCI Asynchronous Error Interrupt Class 0 PCI asynchronous error interrupt is non-critical. 1 PCI asynchronous error interrupt is critical.
28	ETH0	Ethernet 0 Interrupt Class 0 Ethernet 0 interrupt is non-critical. 1 Ethernet 0 interrupt is critical.
29	EWU0	Ethernet 0 Wake-up Interrupt Class 0 Ethernet 0 wake-upinterrupt is non-critical. 1 Ethernet 0 wake-up interrupt is critical.
30	ETH1	Ethernet 1 Interrupt Class 0 Ethernet 1 interrupt is non-critical. 1 Ethernet 1 interrupt is critical.
31	EWU1	Ethernet 1 Wake-up Interrupt Class 0 Ethernet 1 interrupt is non-critical. 1 Ethernet 1 interrupt is critical.

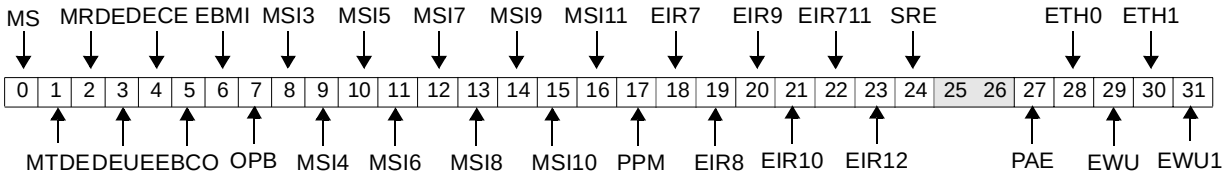


Figure 9-7. UIC1 Critical Register (UIC1_CR)

0	MS	MAL SERR Interrupt Class 0 MAL SERR interrupt is non-critical. 1 MAL SERR interrupt is critical.
1	MTDE	MAL TXDE Interrupt Class 0 MAL TXDE interrupt is non-critical. 1 MAL TXDE interrupt is critical.

2	MRDE	MAL RXDE Interrupt Class 0 MAL RXDE interrupt is non-critical. 1 MAL RXDE interrupt is critical.
3	DEUE	DDRSDRAM ECC Uncorrectable Error Interrupt Class 0 DDRSDRAM ECC uncorrectable error interrupt is non-critical. 1 DDRSDRAM ECC uncorrectable error interrupt is critical.
4	DECE	DDRSDRAM ECC Correctable Error Interrupt Class 0 DDRSDRAM ECC correctable error interrupt is non-critical. 1 DDRSDRAM ECC correctable error interrupt is critical.
5	EBCO	EBCO Interrupt Class 0 EBCO interrupt is non-critical. 1 EBCO interrupt is critical.
6	EBMI	EBMI Interrupt Class 0 EBMI interrupt is non-critical. 1 EBMI interrupt is critical.
7	OPB	OPB to PLB Bridge Interrupt Class 0 OPB to PLB bridge interrupt is non-critical. 1 OPB to PLB bridge interrupt is critical.
8	MSI3	PCI MSI Level 3 Interrupt Class 0 PCI MSI level 3 interrupt is non-critical. 1 PCI MSI level 3 interrupt is critical.
9	MSI4	PCI MSI Level 4 Interrupt Class 0 PCI MSI level 4 interrupt is non-critical. 1 PCI MSI level 4 interrupt is critical.
10	MSI5	PCI MSI Level 5 Interrupt Class 0 PCI MSI level 5 interrupt is non-critical. 1 PCI MSI level 5 interrupt is critical.
11	MSI6	PCI MSI Level 6 Interrupt Class 0 PCI MSI level 6 interrupt is non-critical. 1 PCI MSI level 6 interrupt is critical.
12	MSI7	PCI MSI Level 7 Interrupt Class 0 PCI MSI level 7 interrupt is non-critical. 1 PCI MSI level 7 interrupt is critical.
13	MSI8	PCI MSI Level 8 Interrupt Class 0 PCI MSI level 8 interrupt is non-critical. 1 PCI MSI level 8 interrupt is critical.
14	MSI9	PCI MSI Level 9 Interrupt Class 0 PCI MSI level 9 interrupt is non-critical. 1 PCI MSI level 9 interrupt is critical.
15	MSI10	PCI MSI Level 10 Interrupt Class 0 PCI MSI level 10 interrupt is non-critical. 1 PCI MSI level 10 interrupt is critical.
16	MS11	PCI MSI Level 11 Interrupt Class 0 PCI MSI level 11 interrupt is non-critical. 1 PCI MSI level 11 interrupt is critical.
17	PPM	PPM Interrupt Class 0 PPM interrupt is non-critical. 1 PPM interrupt is critical.

18	EIR7	External IRQ 7 Interrupt Class 0 External IRQ 7 interrupt is non-critical. 1 External IRQ 7 interrupt is critical.
19	EIR8	External IRQ 8 Interrupt Class 0 External IRQ 8 interrupt is non-critical. 1 External IRQ 8 interrupt is critical.
20	EIR9	External IRQ 9 Interrupt Class 0 External IRQ 9 interrupt is non-critical. 1 External IRQ 9 interrupt is critical.
21	EIR10	External IRQ 10 Interrupt Class 0 External IRQ 10 interrupt is non-critical. 1 External IRQ 10 interrupt is critical.
22	EIR11	External IRQ 11 Interrupt Class 0 External IRQ 11 interrupt is non-critical. 1 External IRQ 11 interrupt is critical.
23	EIR12	External IRQ 12 Interrupt Class 0 External IRQ 12 interrupt is non-critical. 1 External IRQ 12 interrupt is critical.
24	SRE	Serial ROM Error Interrupt Class 0 Serial ROM error interrupt is non-critical. 1 Serial ROM error interrupt is critical.
25:26		Reserved
27	PAE	PCI Asynchronous Error Interrupt Class 0 PCI asynchronous error interrupt is non-critical. 1 PCI asynchronous error interrupt is critical.
28	ETH0	Ethernet 0 Interrupt Class 0 Ethernet 0 interrupt is non-critical. 1 Ethernet 0 interrupt is critical.
29	EWU0	Ethernet 0 Wake-up Interrupt Class 0 Ethernet 0 wake-upinterrupt is non-critical. 1 Ethernet 0 wake-up interrupt is critical.
30	ETH1	Ethernet 1 Interrupt Class 0 Ethernet 1 interrupt is non-critical. 1 Ethernet 1 interrupt is critical.
31	EWU1	Ethernet 1 Wake-up Interrupt Class 0 Ethernet 1 interrupt is non-critical. 1 Ethernet 1 interrupt is critical.

9.5.7 UIC0 Polarity Register (UIC0_PR)

The fields of the UIC0_PR, which correspond to the fields of the UIC0_SR, determine whether the corresponding fields in the UIC0_SR have a positive or negative polarity.

For level-sensitive interrupts, a 0 in a UIC0_PR field causes the corresponding interrupt to be negative active. A 1 in a UIC0_PR field causes the corresponding interrupt to be positive active.

For edge-sensitive interrupts, a 0 in a UIC0_PR field causes the corresponding interrupt to be detected on a falling edge (as polarity changes from 1 to 0). A 1 in a UIC0_PR field causes the corresponding interrupt to be detected on a rising edge (as polarity changes from 0 to 1).

Because the on-chip interrupts (those controlled by UIC0_PR) have positive polarity, the associated fields must be set to 1. ~~Figure 9-8~~*Figure 9-8* describes UIC0_PR register bit definitions.

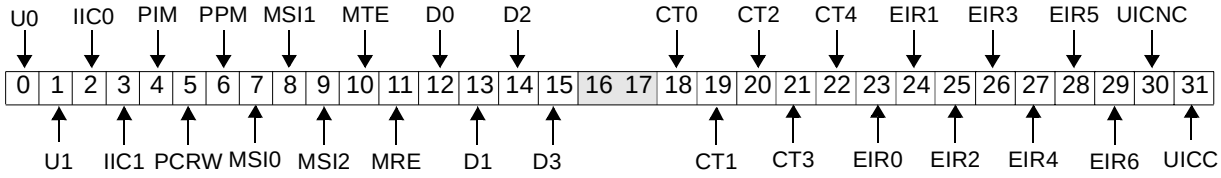


Figure 0-7. UIC0 Polarity Register (UIC0_PR)

0	U0	UART0 Interrupt Polarity 0 UART0 interrupt has negative polarity. 1 UART0 interrupt has positive polarity.
1	U1	UART1 Interrupt Polarity 0 UART1 interrupt has negative polarity. 1 UART1 interrupt has positive polarity.
2	IIC0	IIC0 Interrupt Polarity 0 IIC0 interrupt has negative polarity. 1 IIC0 interrupt has positive polarity.
3	IIC1	IIC1 Interrupt Polarity 0 IIC1 interrupt has negative polarity. 1 IIC1 interrupt has positive polarity.
4	PIM	PCI Inbound Message Interrupt Polarity 0 PCI inbound message interrupt has negative polarity. 1 PCI inbound message interrupt has positive polarity.
5	PCRW	PCI Command Register Write Interrupt Polarity 0 PCI command register write interrupt has negative polarity. 1 PCI command register write interrupt has positive polarity.
6	PPM	PCI Power Management Interrupt Polarity 0 PCI power management interrupt has negative polarity. 1 PCI power management interrupt has positive polarity.
7	MSI0	PCI MSI Level 0 Interrupt Polarity 0 PCI MSI level 0 interrupt has negative polarity. 1 PCI MSI level 0 interrupt has positive polarity.
8	MSI1	PCI MSI Level 1 Interrupt Polarity 0 PCI MSI level 1 interrupt has negative polarity. 1 PCI MSI level 1interrupt has positive polarity.
9	MSI2	PCI MSI Level 2 Interrupt Polarity 0 PCI MSI level 2 interrupt has negative polarity. 1 PCI MSI level 2 interrupt has positive polarity.

10	MTE	MAL TX EOB Interrupt Polarity 0 MAL TX EOB interrupt has negative polarity. 1 MAL TX EOB interrupt has has positive polarity.
11	MRE	MAL RX EOB Interrupt Polarity 0 MAL RX EOB interrupt has negative polarity. 1 MAL RX EOB interrupt has has positive polarity.
12	D0	DMA Channel 0 Interrupt Polarity 0 DMA channel 0 interrupt has negative polarity. 1 DMA channel 0 interrupt has positive polarity.
13	D1	DMA Channel 1 Interrupt Polarity 0 DMA channel 1 interrupt has negative polarity. 1 DMA channel 1 interrupt has positive polarity.
14	D2	DMA Channel 2 Interrupt Polarity 0 DMA channel 2 interrupt has negative polarity. 1 DMA channel 2 interrupt has positive polarity.
15	D3	DMA Channel 3 Interrupt Polarity 0 DMA channel 3 interrupt has negative polarity. 1 DMA channel 3 interrupt has positive polarity.
16:17		Reserved
18	CT0	GPT Compare Timer 0 Interrupt Polarity 0 Compare timer 0 interrupt has negative polarity. 1 Compare timer 0 interrupt has positive polarity.
19	CT1	GPT Compare Timer 1 Interrupt Polarity 0 Compare timer1 interrupt has negative polarity. 1 Compare timer 1 interrupt has positive polarity.
20	CT2	GPT Compare Timer 2 Interrupt Polarity 0 Compare timer 2 interrupt has negative polarity. 1 Compare timer 2 interrupt has positive polarity.
21	CT3	GPT Compare Timer 3 Interrupt Polarity 0 Compare timer 3 interrupt has negative polarity. 1 Compare timer 3 interrupt has positive polarity.
22	CT4	GPT Compare Timer 4 Interrupt Polarity 0 Compare timer 4 interrupt has negative polarity. 1 Compare timer 4 interrupt has positive polarity.

23	EIR0	External IRQ 0 Interrupt Polarity 0 External IRQ 0 interrupt has negative polarity. 1 External IRQ 0 interrupt has positive polarity.
24	EIR1	External IRQ 1 Interrupt Polarity 0 External IRQ 1 interrupt has negative polarity. 1 External IRQ 1 interrupt has positive polarity.
25	EIR2	External IRQ 2 Interrupt Polarity 0 External IRQ 2 interrupt has negative polarity. 1 External IRQ 2 interrupt has positive polarity.
26	EIR3	External IRQ 3 Interrupt Polarity 0 External IRQ 3 interrupt has negative polarity. 1 External IRQ 3 interrupt has positive polarity.
27	EIR4	External IRQ 4 Interrupt Polarity 0 External IRQ 4 interrupt has negative polarity. 1 External IRQ 4 interrupt has positive polarity.
28	EIR5	External IRQ 5 Interrupt Polarity 0 External IRQ 5 interrupt has negative polarity. 1 An external IRQ 5 interrupt has positive polarity.
29	EIR6	External IRQ 6 Interrupt Polarity 0 External IRQ 6 interrupt has negative polarity. 1 External IRQ 6 interrupt has positive polarity.
30	UIC1NC	UIC1 Non-Critical Interrupt Polarity 0 UICNC interrupt has negative polarity. 1 UICNC interrupt has positive polarity.
31	UIC1C	UIC1 Critical Interrupt Polarity 0 UICC interrupt has negative polarity. 1 UICC interrupt has positive polarity.

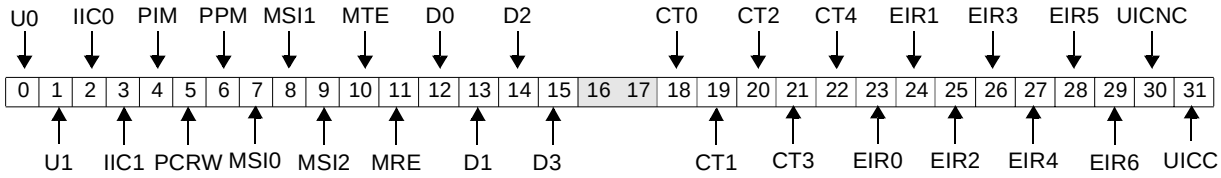


Figure 9-8. UIC0 Polarity Register (UIC0_PR)

0	U0	UART0 Interrupt Polarity 0 UART0 interrupt has negative polarity. 1 UART0 interrupt has positive polarity.
1	U1	UART1 Interrupt Polarity 0 UART1 interrupt has negative polarity. 1 UART1 interrupt has positive polarity.
2	IIC0	IIC0 Interrupt Polarity 0 IIC0 interrupt has negative polarity. 1 IIC0 interrupt has positive polarity.

3	IIC1	IIC1 Interrupt Polarity 0 IIC1 interrupt has negative polarity. 1 IIC1 interrupt has positive polarity.
4	PIM	PCI Inbound Message Interrupt Polarity 0 PCI inbound message interrupt has negative polarity. 1 PCI inbound message interrupt has positive polarity.
5	PCRW	PCI Command Register Write Interrupt Polarity 0 PCI command register write interrupt has negative polarity. 1 PCI command register write interrupt has positive polarity.
6	PPM	PCI Power Management Interrupt Polarity 0 PCI power management interrupt has negative polarity. 1 PCI power management interrupt has positive polarity.
7	MSI0	PCI MSI Level 0 Interrupt Polarity 0 PCI MSI level 0 interrupt has negative polarity. 1 PCI MSI level 0 interrupt has positive polarity.
8	MSI1	PCI MSI Level 1 Interrupt Polarity 0 PCI MSI level 1 interrupt has negative polarity. 1 PCI MSI level 1interrupt has positive polarity.
9	MSI2	PCI MSI Level 2 Interrupt Polarity 0 PCI MSI level 2 interrupt has negative polarity. 1 PCI MSI level 2 interrupt has positive polarity.
10	MTE	MAL TX EOB Interrupt Polarity 0 MAL TX EOB interrupt has negative polarity. 1 MAL TX EOB interrupt has has positive polarity.
11	MRE	MAL RX EOB Interrupt Polarity 0 MAL RX EOB interrupt has negative polarity. 1 MAL RX EOB interrupt has has positive polarity.
12	D0	DMA Channel 0 Interrupt Polarity 0 DMA channel 0 interrupt has negative polarity. 1 DMA channel 0 interrupt has positive polarity.
13	D1	DMA Channel 1 Interrupt Polarity 0 DMA channel 1 interrupt has negative polarity. 1 DMA channel 1 interrupt has positive polarity.
14	D2	DMA Channel 2 Interrupt Polarity 0 DMA channel 2 interrupt has negative polarity. 1 DMA channel 2 interrupt has positive polarity.
15	D3	DMA Channel 3 Interrupt Polarity 0 DMA channel 3 interrupt has negative polarity. 1 DMA channel 3 interrupt has positive polarity.
16:17		Reserved
18	CT0	GPT Compare Timer 0 Interrupt Polarity 0 Compare timer 0 interrupt has negative polarity. 1 Compare timer 0 interrupt has positive polarity.
19	CT1	GPT Compare Timer 1 Interrupt Polarity 0 Compare timer 1 interrupt has negative polarity. 1 Compare timer 1 interrupt has positive polarity.
20	CT2	GPT Compare Timer 2 Interrupt Polarity 0 Compare timer 2 interrupt has negative polarity. 1 Compare timer 2 interrupt has positive polarity.

21	CT3	GPT Compare Timer 3 Interrupt Polarity 0 Compare timer 3 interrupt has negative polarity. 1 Compare timer 3 interrupt has positive polarity.
22	CT4	GPT Compare Timer 4 Interrupt Polarity 0 Compare timer 4 interrupt has negative polarity. 1 Compare timer 4 interrupt has positive polarity.
23	EIR0	External IRQ 0 Interrupt Polarity 0 External IRQ 0 interrupt has negative polarity. 1 External IRQ 0 interrupt has positive polarity.
24	EIR1	External IRQ 1 Interrupt Polarity 0 External IRQ 1 interrupt has negative polarity. 1 External IRQ 1 interrupt has positive polarity.
25	EIR2	External IRQ 2 Interrupt Polarity 0 External IRQ 2 interrupt has negative polarity. 1 External IRQ 2 interrupt has positive polarity.
26	EIR3	External IRQ 3 Interrupt Polarity 0 External IRQ 3 interrupt has negative polarity. 1 External IRQ 3 interrupt has positive polarity.
27	EIR4	External IRQ 4 Interrupt Polarity 0 External IRQ 4 interrupt has negative polarity. 1 External IRQ 4 interrupt has positive polarity.
28	EIR5	External IRQ 5 Interrupt Polarity 0 External IRQ 5 interrupt has negative polarity. 1 An external IRQ 5 interrupt has positive polarity.
29	EIR6	External IRQ 6 Interrupt Polarity 0 External IRQ 6 interrupt has negative polarity. 1 External IRQ 6 interrupt has positive polarity.
30	UIC1NC	UIC1 Non-Critical Interrupt Polarity 0 UICNC interrupt has negative polarity. 1 UICNC interrupt has positive polarity.
31	UIC1C	UIC1 Critical Interrupt Polarity 0 UICC interrupt has negative polarity. 1 UICC interrupt has positive polarity.

9.5.8 UIC1 Polarity Register (UIC1_PR)

The fields of the UIC1_PR, which correspond to the fields of the UIC1_SR, determine whether the corresponding fields in the UIC1_SR have a positive or negative polarity.

For level-sensitive interrupts, a 0 in a UIC1_PR field causes the corresponding interrupt to be negative active. A 1 in a UIC1_PR field causes the corresponding interrupt to be positive active.

For edge-sensitive interrupts, a 0 in a UIC1_PR field causes the corresponding interrupt to be detected on a falling edge (as polarity changes from 1 to 0). A 1 in a UIC1_PR field causes the corresponding interrupt to be detected on a rising edge (as polarity changes from 0 to 1).

Because the on-chip interrupts (those controlled by UIC1_PR) have positive polarity, the associated fields must be set to 1. ~~Figure 9-9~~Figure 9-9 describes UIC1_PR register bit definitions.

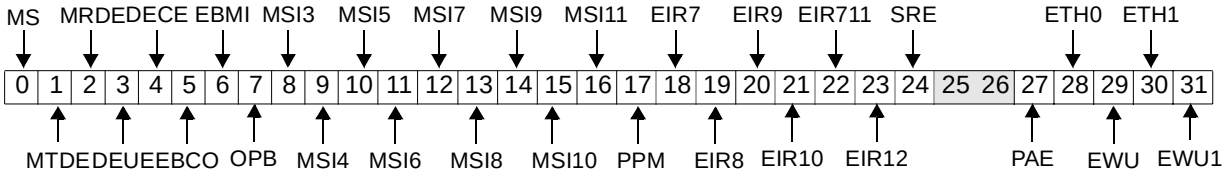


Figure 0-8. UIC1 Polarity Register (UIC1_PR)

0	MS	MAL SERR Interrupt Polarity 0 MAL SERR interrupt has negative polarity. 1 MAL SERR interrupt has positive polarity.
1	MTDE	MAL TXDE Interrupt Polarity 0 MAL TXDE interrupt has negative polarity. 1 MAL TXDE interrupt has positive polarity.
2	MRDE	MAL RXDE Interrupt Polarity 0 MAL RXDE interrupt has negative polarity. 1 MAL RXDE interrupt has positive polarity.
3	DEUE	DDRSDRAM ECC Uncorrectable Error Interrupt Polarity 0 DDRSDRAM ECC uncorrectable error interrupt has negative polarity. 1 DDRSDRAM ECC uncorrectable error interrupt has positive polarity.
4	DECE	DDRSDRAM ECC Correctable Error Interrupt Polarity 0 DDRSDRAM ECC correctable error interrupt has negative polarity. 1 DDRSDRAM ECC correctable error interrupt has positive polarity.
5	EBCO	EBCO Interrupt Polarity 0 EBCO interrupt has negative polarity. 1 EBCO interrupt has positive polarity.
6	EBMI	EBMI Interrupt Polarity 0 EBMI interrupt has negative polarity. 1 EBMI interrupt has positive polarity.
7	OPB	OPB to PLB Bridge Interrupt Polarity 0 OPB to PLB bridge interrupt has negative polarity. 1 OPB to PLB bridge interrupt has positive polarity.

8	MSI3	PCI MSI Level 3 Interrupt Polarity 0 PCI MSI level 3 interrupt has negative polarity. 1 PCI MSI level 3 interrupt has positive polarity.
9	MSI4	PCI MSI Level 4 Interrupt Polarity 0 PCI MSI level 4 interrupt has negative polarity. 1 PCI MSI level 4 interrupt has positive polarity.
10	MSI5	PCI MSI Level 5 Interrupt Polarity 0 PCI MSI level 5 interrupt has negative polarity. 1 PCI MSI level 5 interrupt has positive polarity.
11	MSI6	PCI MSI Level 6 Interrupt Polarity 0 PCI MSI level 6 interrupt has negative polarity. 1 PCI MSI level 6 interrupt has positive polarity.
12	MSI7	PCI MSI Level 7 Interrupt Polarity 0 PCI MSI level 7 interrupt has negative polarity. 1 PCI MSI level 7 interrupt has positive polarity.
13	MSI8	PCI MSI Level 8 Interrupt Polarity 0 PCI MSI level 8 interrupt has negative polarity. 1 PCI MSI level 8 interrupt has positive polarity.
14	MSI9	PCI MSI Level 9 Interrupt Polarity 0 PCI MSI level 9 interrupt has negative polarity. 1 PCI MSI level 9 interrupt has positive polarity.
15	MSI10	PCI MSI Level 10 Interrupt Polarity 0 PCI MSI level 10 interrupt has negative polarity. 1 PCI MSI level 10 interrupt has positive polarity.
16	MSI11	PCI MSI Level 11 Interrupt Polarity 0 PCI MSI level 11 interrupt has negative polarity. 1 PCI MSI level 11 interrupt has positive polarity.
17	PPM	PPM Interrupt Polarity 0 PPM interrupt has negative polarity. 1 PPM interrupt has positive polarity.

18	EIR7	External IRQ 7 Interrupt Polarity 0 External IRQ 7 interrupt has negative polarity. 1 External IRQ 7 interrupt has positive polarity.
19	EIR8	External IRQ 8 Interrupt Polarity 0 External IRQ 8 interrupt has negative polarity. 1 External IRQ 8 interrupt has positive polarity.
20	EIR9	External IRQ 9 Interrupt Polarity 0 External IRQ 9 interrupt has negative polarity. 1 External IRQ 9 interrupt has positive polarity.
21	EIR10	External IRQ 10 Interrupt Polarity 0 External IRQ 10 interrupt has negative polarity. 1 External IRQ 10 interrupt has positive polarity.
22	EIR11	External IRQ 11 Interrupt Polarity 0 External IRQ 11 interrupt has negative polarity. 1 External IRQ 11 interrupt has positive polarity.
23	EIR12	External IRQ 12 Interrupt Polarity 0 External IRQ 12 interrupt has negative polarity. 1 External IRQ 12 interrupt has positive polarity.
24	SRE	Serial ROM Error Interrupt Polarity 0 Serial ROM error interrupt has negative polarity. 1 Serial ROM error interrupt has positive polarity.
25:26		Reserved
27	PAE	PCI Asynchronous Error Interrupt Polarity 0 PCI asynchronous error interrupt has negative polarity. 1 PCI asynchronous error interrupt has positive polarity.
28	ETH0	Ethernet 0 Interrupt Polarity 0 Ethernet 0 interrupt has negative polarity. 1 Ethernet 0 interrupt has positive polarity.

29	EWU0	Ethernet 0 Wake-up Interrupt Polarity 0 Ethernet 0 wake-up interrupt has negative polarity. 1 Ethernet 0 wake-up interrupt has positive polarity.
30	ETH1	Ethernet 1 Interrupt Polarity 0 Ethernet 1 interrupt has negative polarity. 1 Ethernet 1 interrupt has positive polarity.
31	EWU1	Ethernet 1 Wake-up Interrupt Polarity 0 Ethernet 1 interrupt has negative polarity. 1 Ethernet 1 interrupt has positive polarity.

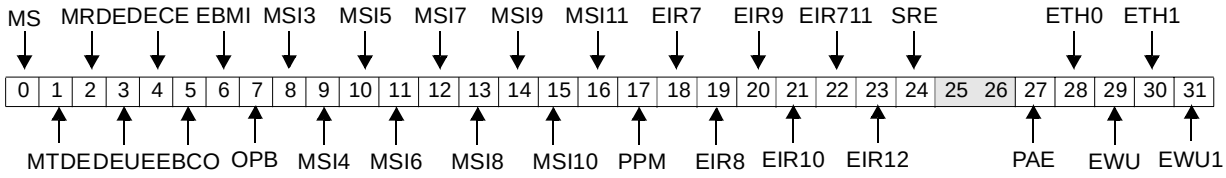


Figure 9-9. UIC1 Polarity Register (UIC1_PR)

0	MS	MAL SERR Interrupt Polarity 0 MAL SERR interrupt has negative polarity. 1 MAL SERR interrupt has positive polarity.
1	MTDE	MAL TXDE Interrupt Polarity 0 MAL TXDE interrupt has negative polarity. 1 MAL TXDE interrupt has positive polarity.
2	MRDE	MAL RXDE Interrupt Polarity 0 MAL RXDE interrupt has negative polarity. 1 MAL RXDE interrupt has positive polarity.
3	DEUE	DDRSDRAM ECC Uncorrectable Error Interrupt Polarity 0 DDRSDRAM ECC uncorrectable error interrupt has negative polarity. 1 DDRSDRAM ECC uncorrectable error interrupt has positive polarity.
4	DECE	DDRSDRAM ECC Correctable Error Interrupt Polarity 0 DDRSDRAM ECC correctable error interrupt has negative polarity. 1 DDRSDRAM ECC correctable error interrupt has positive polarity.
5	EBCO	EBCO Interrupt Polarity 0 EBCO interrupt has negative polarity. 1 EBCO interrupt has positive polarity.
6	EBMI	EBMI Interrupt Polarity 0 EBMI interrupt has negative polarity. 1 EBMI interrupt has positive polarity.

7	OPB	OPB to PLB Bridge Interrupt Polarity 0 OPB to PLB bridge interrupt has negative polarity. 1 OPB to PLB bridge interrupt has positive polarity.
8	MSI3	PCI MSI Level 3 Interrupt Polarity 0 PCI MSI level 3 interrupt has negative polarity. 1 PCI MSI level 3 interrupt has positive polarity.
9	MSI4	PCI MSI Level 4 Interrupt Polarity 0 PCI MSI level 4 interrupt has negative polarity. 1 PCI MSI level 4 interrupt has positive polarity.
10	MSI5	PCI MSI Level 5 Interrupt Polarity 0 PCI MSI level 5 interrupt has negative polarity. 1 PCI MSI level 5 interrupt has positive polarity.
11	MSI6	PCI MSI Level 6 Interrupt Polarity 0 PCI MSI level 6 interrupt has negative polarity. 1 PCI MSI level 6 interrupt has positive polarity.
12	MSI7	PCI MSI Level 7 Interrupt Polarity 0 PCI MSI level 7 interrupt has negative polarity. 1 PCI MSI level 7 interrupt has positive polarity.
13	MSI8	PCI MSI Level 8 Interrupt Polarity 0 PCI MSI level 8 interrupt has negative polarity. 1 PCI MSI level 8 interrupt has positive polarity.
14	MSI9	PCI MSI Level 9 Interrupt Polarity 0 PCI MSI level 9 interrupt has negative polarity. 1 PCI MSI level 9 interrupt has positive polarity.
15	MSI10	PCI MSI Level 10 Interrupt Polarity 0 PCI MSI level 10 interrupt has negative polarity. 1 PCI MSI level 10 interrupt has positive polarity.
16	MSI11	PCI MSI Level 11 Interrupt Polarity 0 PCI MSI level 11 interrupt has negative polarity. 1 PCI MSI level 11 interrupt has positive polarity.
17	PPM	PPM Interrupt Polarity 0 PPM interrupt has negative polarity. 1 PPM interrupt has positive polarity.
18	EIR7	External IRQ 7 Interrupt Polarity 0 External IRQ 7 interrupt has negative polarity. 1 External IRQ 7 interrupt has positive polarity.
19	EIR8	External IRQ 8 Interrupt Polarity 0 External IRQ 8 interrupt has negative polarity. 1 External IRQ 8 interrupt has positive polarity.
20	EIR9	External IRQ 9 Interrupt Polarity 0 External IRQ 9 interrupt has negative polarity. 1 External IRQ 9 interrupt has positive polarity.
21	EIR10	External IRQ 10 Interrupt Polarity 0 External IRQ 10 interrupt has negative polarity. 1 External IRQ 10 interrupt has positive polarity.
22	EIR11	External IRQ 11 Interrupt Polarity 0 External IRQ 11 interrupt has negative polarity. 1 External IRQ 11 interrupt has positive polarity.
23	EIR12	External IRQ 12 Interrupt Polarity 0 External IRQ 12 interrupt has negative polarity. 1 External IRQ 12 interrupt has positive polarity.

24	SRE	Serial ROM Error Interrupt Polarity 0 Serial ROM error interrupt has negative polarity. 1 Serial ROM error interrupt has positive polarity.
25:26		Reserved
27	PAE	PCI Asynchronous Error Interrupt Polarity 0 PCI asynchronous error interrupt has negative polarity. 1 PCI asynchronous error interrupt has positive polarity.
28	ETH0	Ethernet 0 Interrupt Polarity 0 Ethernet 0 interrupt has negative polarity. 1 Ethernet 0 interrupt has positive polarity.
29	EWU0	Ethernet 0 Wake-up Interrupt Polarity 0 Ethernet 0 wake-up interrupt has negative polarity. 1 Ethernet 0 wake-up interrupt has positive polarity.
30	ETH1	Ethernet 1 Interrupt Polarity 0 Ethernet 1 interrupt has negative polarity. 1 Ethernet 1 interrupt has positive polarity.
31	EWU1	Ethernet 1 Wake-up Interrupt Polarity 0 Ethernet 1 interrupt has negative polarity. 1 Ethernet 1 interrupt has positive polarity.

9.5.9 UIC0 Trigger Register (UIC0_TR)

The fields of the UIC0_TR, which correspond to the fields of the UIC0_SR, determine whether corresponding fields in the UIC0_SR are edge-sensitive or level-sensitive.

Edge-sensitive interrupts are triggered depending on whether the associated interrupt signal is rising or falling (changing from 0 to 1 or 1 to 0, respectively). Whether a rising or falling edge causes the trigger is controlled by bits in the UIC0_PR.

Level-sensitive interrupts are triggered depending on whether the associated interrupt signal is high (1) or low (0).

If a UIC0_TR field is 0, the associated interrupt is level-sensitive. If the UIC0_TR field is 1, the interrupt is edge-sensitive.~~Figure 9-10~~Figure 9-10 describes UIC0_TR bit definitions.

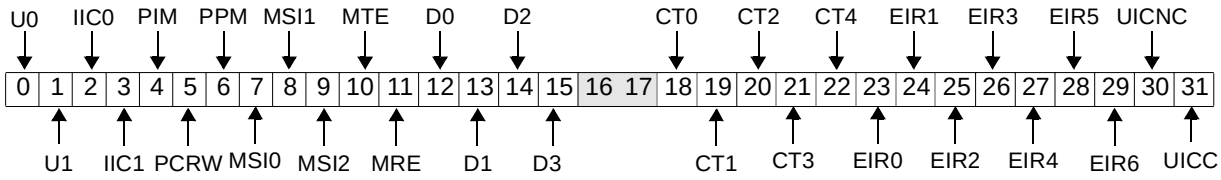


Figure 0-9. UIC0 Trigger Register (UIC0_TR)

0	U0	UART0 Interrupt Trigger 0 UART0 interrupt is level-sensitive. 1 UART0 interrupt is edge-sensitive.

1	U1	UART1 Interrupt Trigger 0 UART1 interrupt is level-sensitive. 1 UART1 interrupt is edge-sensitive.
2	IIC0	IIC0 Interrupt Trigger 0 IIC0 interrupt is level-sensitive. 1 IIC0 interrupt is edge-sensitive.
3	IIC1	IIC1 Interrupt Trigger 0 IIC1 interrupt is level-sensitive. 1 IIC1 interrupt is edge-sensitive.
4	PIM	PCI Inbound Message Interrupt Trigger 0 PCI inbound message interrupt is level-sensitive. 1 PCI inbound message interrupt is edge-sensitive.
5	PCRW	PCI Command Register Write Interrupt Trigger 0 PCI command register write interrupt is level-sensitive. 1 PCI command register write interrupt is edge-sensitive.
6	PPM	PCI Power Management Interrupt Trigger 0 PCI power management interrupt is level-sensitive. 1 PCI power management interrupt is edge-sensitive.
7	MSI0	PCI MSI Level 0 Interrupt Trigger 0 PCI MSI level 0 interrupt is level-sensitive. 1 PCI MSI level 0 interrupt is edge-sensitive.
8	MSI1	PCI MSI Level 1 Interrupt Trigger 0 PCI MSI level 1 interrupt is level-sensitive. 1 PCI MSI level 1 interrupt is edge-sensitive.
9	MSI2	PCI MSI Level 2 Interrupt Trigger 0 PCI MSI level 2 interrupt is level-sensitive. 1 PCI MSI level 2 interrupt is edge-sensitive.
10	MTE	MAL TX EOB Interrupt Trigger 0 MAL TX EOB interrupt is level-sensitive. 1 MAL TX EOB interrupt has is edge-sensitive.

11	MRE	MAL RX EOB Interrupt Trigger 0 MAL RX EOB interrupt is level-sensitive. 1 MAL RX EOB interrupt has is edge-sensitive.
12	D0	DMA Channel 0 Interrupt Trigger 0 DMA channel 0 interrupt is level-sensitive. 1 DMA channel 0 interrupt is edge-sensitive.
13	D1	DMA Channel 1 Interrupt Trigger 0 DMA channel 1 interrupt is level-sensitive. 1 DMA channel 1 interrupt is edge-sensitive.
14	D2	DMA Channel 2 Interrupt Trigger 0 DMA channel 2 interrupt is level-sensitive. 1 DMA channel 2 interrupt is edge-sensitive.
15	D3	DMA Channel 3 Interrupt Trigger 0 DMA channel 3 interrupt is level-sensitive. 1 DMA channel 3 interrupt is edge-sensitive.
16:17		Reserved
18	CT0	GPT Compare Timer 0 Interrupt Trigger 0 Compare timer 0 interrupt is level-sensitive. 1 Compare timer 0 interrupt is edge-sensitive.
19	CT1	GPT Compare Timer 1 Interrupt Trigger 0 Compare timer1 interrupt is level-sensitive. 1 Compare timer 1 interrupt is edge-sensitive.
20	CT2	GPT Compare Timer 2 Interrupt Trigger 0 Compare timer 2 interrupt is level-sensitive. 1 Compare timer 2 interrupt is edge-sensitive.
21	CT3	GPT Compare Timer 3 Interrupt Trigger 0 Compare timer 3 interrupt is level-sensitive. 1 Compare timer 3 interrupt is edge-sensitive.

22	CT4	GPT Compare Timer 4 Interrupt Trigger 0 Compare timer 4 interrupt is level-sensitive. 1 Compare timer 4 interrupt is edge-sensitive.
23	EIR0	External IRQ 0 Interrupt Trigger 0 External IRQ 0 interrupt is level-sensitive. 1 External IRQ 0 interrupt is edge-sensitive.
24	EIR1	External IRQ 1 Interrupt Trigger 0 External IRQ 1 interrupt is level-sensitive. 1 External IRQ 1 interrupt is edge-sensitive.
25	EIR2	External IRQ 2 Interrupt Trigger 0 External IRQ 2 interrupt is level-sensitive. 1 External IRQ 2 interrupt is edge-sensitive.
26	EIR3	External IRQ 3 Interrupt Trigger 0 External IRQ 3 interrupt is level-sensitive. 1 External IRQ 3 interrupt is edge-sensitive.
27	EIR4	External IRQ 4 Interrupt Trigger 0 External IRQ 4 interrupt is level-sensitive. 1 External IRQ 4 interrupt is edge-sensitive.
28	EIR5	External IRQ 5 Interrupt Trigger 0 External IRQ 5 interrupt is level-sensitive. 1 An external IRQ 5 interrupt is edge-sensitive.
29	EIR6	External IRQ 6 Interrupt Trigger 0 External IRQ 6 interrupt is level-sensitive. 1 External IRQ 6 interrupt is edge-sensitive.
30	UIC1NC	UIC1 Non-Critical Interrupt Trigger 0 UICNC interrupt is level-sensitive. 1 UICNC interrupt is edge-sensitive.
31	UIC1C	UIC1 Critical Interrupt Trigger 0 UICC interrupt is level-sensitive. 1 UICC interrupt is edge-sensitive.

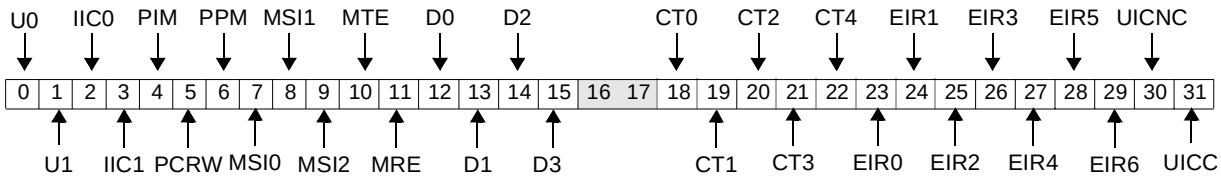


Figure 9-10. UIC0 Trigger Register (UIC0_TR)

0	U0	UART0 Interrupt Trigger 0 UART0 interrupt is level-sensitive. 1 UART0 interrupt is edge-sensitive.
1	U1	UART1 Interrupt Trigger 0 UART1 interrupt is level-sensitive. 1 UART1 interrupt is edge-sensitive.
2	IIC0	IIC0 Interrupt Trigger 0 IIC0 interrupt is level-sensitive. 1 IIC0 interrupt is edge-sensitive.
3	IIC1	IIC1 Interrupt Trigger 0 IIC1 interrupt is level-sensitive. 1 IIC1 interrupt is edge-sensitive.
4	PIM	PCI Inbound Message Interrupt Trigger 0 PCI inbound message interrupt is level-sensitive. 1 PCI inbound message interrupt is edge-sensitive.
5	PCRW	PCI Command Register Write Interrupt Trigger 0 PCI command register write interrupt is level-sensitive. 1 PCI command register write interrupt is edge-sensitive.
6	PPM	PCI Power Management Interrupt Trigger 0 PCI power management interrupt is level-sensitive. 1 PCI power management interrupt is edge-sensitive.
7	MSI0	PCI MSI Level 0 Interrupt Trigger 0 PCI MSI level 0 interrupt is level-sensitive. 1 PCI MSI level 0 interrupt is edge-sensitive.
8	MSI1	PCI MSI Level 1 Interrupt Trigger 0 PCI MSI level 1 interrupt is level-sensitive. 1 PCI MSI level 1 interrupt is edge-sensitive.
9	MSI2	PCI MSI Level 2 Interrupt Trigger 0 PCI MSI level 2 interrupt is level-sensitive. 1 PCI MSI level 2 interrupt is edge-sensitive.
10	MTE	MAL TX EOB Interrupt Trigger 0 MAL TX EOB interrupt is level-sensitive. 1 MAL TX EOB interrupt has is edge-sensitive.
11	MRE	MAL RX EOB Interrupt Trigger 0 MAL RX EOB interrupt is level-sensitive. 1 MAL RX EOB interrupt has is edge-sensitive.

12	D0	DMA Channel 0 Interrupt Trigger 0 DMA channel 0 interrupt is level-sensitive. 1 DMA channel 0 interrupt is edge-sensitive.
13	D1	DMA Channel 1 Interrupt Trigger 0 DMA channel 1 interrupt is level-sensitive. 1 DMA channel 1 interrupt is edge-sensitive.
14	D2	DMA Channel 2 Interrupt Trigger 0 DMA channel 2 interrupt is level-sensitive. 1 DMA channel 2 interrupt is edge-sensitive.
15	D3	DMA Channel 3 Interrupt Trigger 0 DMA channel 3 interrupt is level-sensitive. 1 DMA channel 3 interrupt is edge-sensitive.
16:17		Reserved
18	CT0	GPT Compare Timer 0 Interrupt Trigger 0 Compare timer 0 interrupt is level-sensitive. 1 Compare timer 0 interrupt is edge-sensitive.
19	CT1	GPT Compare Timer 1 Interrupt Trigger 0 Compare timer 1 interrupt is level-sensitive. 1 Compare timer 1 interrupt is edge-sensitive.
20	CT2	GPT Compare Timer 2 Interrupt Trigger 0 Compare timer 2 interrupt is level-sensitive. 1 Compare timer 2 interrupt is edge-sensitive.
21	CT3	GPT Compare Timer 3 Interrupt Trigger 0 Compare timer 3 interrupt is level-sensitive. 1 Compare timer 3 interrupt is edge-sensitive.
22	CT4	GPT Compare Timer 4 Interrupt Trigger 0 Compare timer 4 interrupt is level-sensitive. 1 Compare timer 4 interrupt is edge-sensitive.
23	EIR0	External IRQ 0 Interrupt Trigger 0 External IRQ 0 interrupt is level-sensitive. 1 External IRQ 0 interrupt is edge-sensitive.
24	EIR1	External IRQ 1 Interrupt Trigger 0 External IRQ 1 interrupt is level-sensitive. 1 External IRQ 1 interrupt is edge-sensitive.
25	EIR2	External IRQ 2 Interrupt Trigger 0 External IRQ 2 interrupt is level-sensitive. 1 External IRQ 2 interrupt is edge-sensitive.
26	EIR3	External IRQ 3 Interrupt Trigger 0 External IRQ 3 interrupt is level-sensitive. 1 External IRQ 3 interrupt is edge-sensitive.
27	EIR4	External IRQ 4 Interrupt Trigger 0 External IRQ 4 interrupt is level-sensitive. 1 External IRQ 4 interrupt is edge-sensitive.
28	EIR5	External IRQ 5 Interrupt Trigger 0 External IRQ 5 interrupt is level-sensitive. 1 An external IRQ 5 interrupt is edge-sensitive.
29	EIR6	External IRQ 6 Interrupt Trigger 0 External IRQ 6 interrupt is level-sensitive. 1 External IRQ 6 interrupt is edge-sensitive.
30	UIC1NC	UIC1 Non-Critical Interrupt Trigger 0 UICNC interrupt is level-sensitive. 1 UICNC interrupt is edge-sensitive.

31	UIC1C	UIC1 Critical Interrupt Trigger 0 UICC interrupt is level-sensitive. 1 UICC interrupt is edge-sensitive.

9.5.10 UIC1 Trigger Register (UIC1_TR)

The fields of the UIC1_TR, which correspond to the fields of the UIC1_SR, determine whether corresponding fields in the UIC1_SR are edge-sensitive or level-sensitive.

Edge-sensitive interrupts are triggered depending on whether the associated interrupt signal is rising or falling (changing from 0 to 1 or 1 to 0, respectively). Whether a rising or falling edge causes the trigger is controlled by bits in the UIC1_PR.

Level-sensitive interrupts are triggered depending on whether the associated interrupt signal is high (1) or low (0).

If a UIC1_TR field is 0, the associated interrupt is level-sensitive. If the UIC1_TR field is 1, the interrupt is edge-sensitive. ~~Figure 9-11~~Figure 9-11 describes UIC1_TR bit definitions.

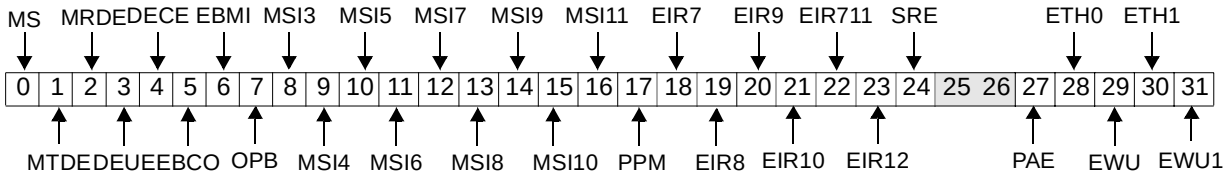


Figure 0-10. UIC1 Trigger Register (UIC1_TR)

0	MS	MAL SERR Interrupt Trigger 0 MAL SERR interrupt is level-sensitive. 1 MAL SERR interrupt is edge-sensitive.
1	MTDE	MAL TXDE Interrupt Trigger 0 MAL TXDE interrupt is level-sensitive. 1 MAL TXDE interrupt is edge-sensitive.
2	MRDE	MAL RXDE Interrupt Trigger 0 MAL RXDE interrupt is level-sensitive. 1 MAL RXDE interrupt is edge-sensitive.
3	DEUE	DDRSDRAM ECC Uncorrectable Error Interrupt Trigger 0 DDRSDRAM ECC uncorrectable error interrupt is level-sensitive. 1 DDRSDRAM ECC uncorrectable error interrupt is edge-sensitive.
4	DECE	DDRSDRAM ECC Correctable Error Interrupt Trigger 0 DDRSDRAM ECC correctable error interrupt is level-sensitive. 1 DDRSDRAM ECC correctable error interrupt is edge-sensitive.

5	EBCO	EBCO Interrupt Trigger 0 EBCO interrupt is level-sensitive. 1 EBCO interrupt is edge-sensitive.
6	EBMI	EBMI Interrupt Trigger 0 EBMI interrupt is level-sensitive. 1 EBMI interrupt is edge-sensitive.
7	OPB	OPB to PLB Bridge Interrupt Trigger 0 OPB to PLB bridge interrupt is level-sensitive. 1 OPB to PLB bridge interrupt is edge-sensitive.
8	MSI3	PCI MSI Level 3 Interrupt Trigger 0 PCI MSI level 3 interrupt is level-sensitive. 1 PCI MSI level 3 interrupt is edge-sensitive.
9	MSI4	PCI MSI Level 4 Interrupt Trigger 0 PCI MSI level 4 interrupt is level-sensitive. 1 PCI MSI level 4 interrupt is edge-sensitive.
10	MSI5	PCI MSI Level 5 Interrupt Trigger 0 PCI MSI level 5 interrupt is level-sensitive. 1 PCI MSI level 5 interrupt is edge-sensitive.
11	MSI6	PCI MSI Level 6 Interrupt Trigger 0 PCI MSI level 6 interrupt is level-sensitive. 1 PCI MSI level 6 interrupt is edge-sensitive.
12	MSI7	PCI MSI Level 7 Interrupt Trigger 0 PCI MSI level 7 interrupt is level-sensitive. 1 PCI MSI level 7 interrupt is edge-sensitive.
13	MSI8	PCI MSI Level 8 Interrupt Trigger 0 PCI MSI level 8 interrupt is level-sensitive. 1 PCI MSI level 8 interrupt is edge-sensitive.
14	MSI9	PCI MSI Level 9 Interrupt Trigger 0 PCI MSI level 9 interrupt is level-sensitive. 1 PCI MSI level 9 interrupt is edge-sensitive.
15	MSI10	PCI MSI Level 10 Interrupt Trigger 0 PCI MSI level 10 interrupt is level-sensitive. 1 PCI MSI level 10 interrupt is edge-sensitive.
16	MSI11	PCI MSI Level 11 Interrupt Trigger 0 PCI MSI level 11 interrupt is level-sensitive. 1 PCI MSI level 11 interrupt is edge-sensitive.
17	PPM	PPM Interrupt Trigger 0 PPM interrupt is level-sensitive. 1 PPM interrupt is edge-sensitive.
18	EIR7	External IRQ 7 Interrupt Trigger 0 External IRQ 7 interrupt is level-sensitive. 1 External IRQ 7 interrupt is edge-sensitive.

19	EIR8	External IRQ 8 Interrupt Trigger 0 External IRQ 8 interrupt is level-sensitive. 1 External IRQ 8 interrupt is edge-sensitive.
20	EIR9	External IRQ 9 Interrupt Trigger 0 External IRQ 9 interrupt is level-sensitive. 1 External IRQ 9 interrupt is edge-sensitive.
21	EIR10	External IRQ 10 Interrupt Trigger 0 External IRQ 10 interrupt is level-sensitive. 1 External IRQ 10 interrupt is edge-sensitive.
22	EIR11	External IRQ 11 Interrupt Trigger 0 External IRQ 11 interrupt is level-sensitive. 1 External IRQ 11 interrupt is edge-sensitive.
23	EIR12	External IRQ 12 Interrupt Trigger 0 External IRQ 12 interrupt is level-sensitive. 1 External IRQ 12 interrupt is edge-sensitive.
24	SRE	Serial ROM Error Interrupt Trigger 0 Serial ROM error interrupt is level-sensitive. 1 Serial ROM error interrupt is edge-sensitive.
25:26		Reserved
27	PAE	PCI Asynchronous Error Interrupt Trigger 0 PCI asynchronous error interrupt is level-sensitive. 1 PCI asynchronous error interrupt is edge-sensitive.
28	ETH0	Ethernet 0 Interrupt Trigger 0 Ethernet 0 interrupt is level-sensitive. 1 Ethernet 0 interrupt is edge-sensitive.
29	EWU0	Ethernet 0 Wake-up Interrupt Trigger 0 Ethernet 0 wake-up interrupt is level-sensitive. 1 Ethernet 0 wake-up interrupt is edge-sensitive.
30	ETH1	Ethernet 1 Interrupt Trigger 0 Ethernet 1 interrupt is level-sensitive. 1 Ethernet 1 interrupt is edge-sensitive.
31	EWU1	Ethernet 1 Wake-up Interrupt Trigger 0 Ethernet 1 interrupt is level-sensitive. 1 Ethernet 1 interrupt is edge-sensitive.

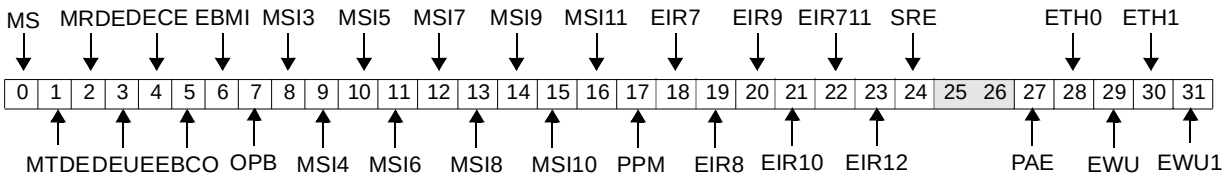


Figure 9-11. UIC1 Trigger Register (UIC1_TR)

0	MS	MAL SERR Interrupt Trigger 0 MAL SERR interrupt is level-sensitive. 1 MAL SERR interrupt is edge-sensitive.
1	MTDE	MAL TXDE Interrupt Trigger 0 MAL TXDE interrupt is level-sensitive. 1 MAL TXDE interrupt is edge-sensitive.
2	MRDE	MAL RXDE Interrupt Trigger 0 MAL RXDE interrupt is level-sensitive. 1 MAL RXDE interrupt is edge-sensitive.
3	DEUE	DDRSDRAM ECC Uncorrectable Error Interrupt Trig-ger 0 DDRSDRAM ECC uncorrectable error interrupt is level-sensitive. 1 DDRSDRAM ECC uncorrectable error interrupt is edge-sensitive.
4	DECE	DDRSDRAM ECC Correctable Error Interrupt Trigger 0 DDRSDRAM ECC correctable error interrupt is level-sensitive. 1 DDRSDRAM ECC correctable error interrupt is edge-sensitive.
5	EBCO	EBCO Interrupt Trigger 0 EBCO interrupt is level-sensitive. 1 EBCO interrupt is edge-sensitive.
6	EBMI	EBMI Interrupt Trigger 0 EBMI interrupt is level-sensitive. 1 EBMI interrupt is edge-sensitive.
7	OPB	OPB to PLB Bridge Interrupt Trigger 0 OPB to PLB bridge interrupt is level-sensitive. 1 OPB to PLB bridge interrupt is edge-sensitive.
8	MSI3	PCI MSI Level 3 Interrupt Trigger 0 PCI MSI level 3 interrupt is level-sensitive. 1 PCI MSI level 3 interrupt is edge-sensitive.
9	MSI4	PCI MSI Level 4 Interrupt Trigger 0 PCI MSI level 4 interrupt is level-sensitive. 1 PCI MSI level 4 interrupt is edge-sensitive.
10	MSI5	PCI MSI Level 5 Interrupt Trigger 0 PCI MSI level 5 interrupt is level-sensitive. 1 PCI MSI level 5 interrupt is edge-sensitive.
11	MSI6	PCI MSI Level 6 Interrupt Trigger 0 PCI MSI level 6 interrupt is level-sensitive. 1 PCI MSI level 6 interrupt is edge-sensitive.
12	MSI7	PCI MSI Level 7 Interrupt Trigger 0 PCI MSI level 7 interrupt is level-sensitive. 1 PCI MSI level 7 interrupt is edge-sensitive.

13	MSI8	PCI MSI Level 8 Interrupt Trigger 0 PCI MSI level 8 interrupt is level-sensitive. 1 PCI MSI level 8 interrupt is edge-sensitive.
14	MSI9	PCI MSI Level 9 Interrupt Trigger 0 PCI MSI level 9 interrupt is level-sensitive. 1 PCI MSI level 9 interrupt is edge-sensitive.
15	MSI10	PCI MSI Level 10 Interrupt Trigger 0 PCI MSI level 10 interrupt is level-sensitive. 1 PCI MSI level 10 interrupt is edge-sensitive.
16	MSI11	PCI MSI Level 11 Interrupt Trigger 0 PCI MSI level 11 interrupt is level-sensitive. 1 PCI MSI level 11 interrupt is edge-sensitive.
17	PPM	PPM Interrupt Trigger 0 PPM interrupt is level-sensitive. 1 PPM interrupt is edge-sensitive.
18	EIR7	External IRQ 7 Interrupt Trigger 0 External IRQ 7 interrupt is level-sensitive. 1 External IRQ 7 interrupt is edge-sensitive.
19	EIR8	External IRQ 8 Interrupt Trigger 0 External IRQ 8 interrupt is level-sensitive. 1 External IRQ 8 interrupt is edge-sensitive.
20	EIR9	External IRQ 9 Interrupt Trigger 0 External IRQ 9 interrupt is level-sensitive. 1 External IRQ 9 interrupt is edge-sensitive.
21	EIR10	External IRQ 10 Interrupt Trigger 0 External IRQ 10 interrupt is level-sensitive. 1 External IRQ 10 interrupt is edge-sensitive.
22	EIR11	External IRQ 11 Interrupt Trigger 0 External IRQ 11 interrupt is level-sensitive. 1 External IRQ 11 interrupt is edge-sensitive.
23	EIR12	External IRQ 12 Interrupt Trigger 0 External IRQ 12 interrupt is level-sensitive. 1 External IRQ 12 interrupt is edge-sensitive.
24	SRE	Serial ROM Error Interrupt Trigger 0 Serial ROM error interrupt is level-sensitive. 1 Serial ROM error interrupt is edge-sensitive.
25:26		Reserved
27	PAE	PCI Asynchronous Error Interrupt Trigger 0 PCI asynchronous error interrupt is level-sensitive. 1 PCI asynchronous error interrupt is edge-sensitive.
28	ETH0	Ethernet 0 Interrupt Trigger 0 Ethernet 0 interrupt is level-sensitive. 1 Ethernet 0 interrupt is edge-sensitive.
29	EWU0	Ethernet 0 Wake-up Interrupt Trigger 0 Ethernet 0 wake-up interrupt is level-sensitive. 1 Ethernet 0 wake-up interrupt is edge-sensitive.
30	ETH1	Ethernet 1 Interrupt Trigger 0 Ethernet 1 interrupt is level-sensitive. 1 Ethernet 1 interrupt is edge-sensitive.
31	EWU1	Ethernet 1 Wake-up Interrupt Trigger 0 Ethernet 1 interrupt is level-sensitive. 1 Ethernet 1 interrupt is edge-sensitive.

9.5.11 UIC0 Masked Status Register (UIC0_MSR)

This read-only register contains the result of masking the UIC0_SR with the UIC0_ER. Reading this register, instead of the actual UIC0_SR, eliminates the need for software to read and apply the enable mask to the contents of the UIC0_SR to determine which enabled interrupt fields are active.

If an interrupt is configured as level-sensitive, and a clear is attempted on the UIC0_SR, the UIC0_SR field is not cleared if the incoming interrupt signal is at the asserted polarity. The interrupt signal must be reset before the UIC0_SR can be successfully cleared. ~~Figure 9-12~~Figure 9-12 describes UIC0_MSR bit definitions.

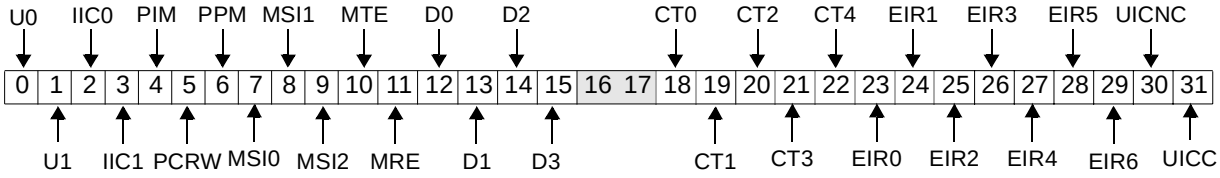


Figure 0-11. UIC0 Masked Status Register (UIC0_MSR)

0	U0	UART0 Masked Interrupt Status 0 UART0masked interrupt has not occurred. 1 UART0 masked interrupt occurred.
1	U1	UART1 Masked Interrupt Status 0 UART1 masked interrupt has not occurred. 1 UART1 masked interrupt occurred.
2	IIC0	IIC0 Masked Interrupt Status 0 IIC0 masked interrupt has not occurred. 1 IIC0 masked interrupt occurred.
3	IIC1	IIC1 Masked Interrupt Status 0 IIC1 masked interrupt has not occurred. 1 IIC1 masked interrupt occurred.
4	PIM	PCI Inbound Message Masked Interrupt Status 0 PCI inbound message masked interrupt has not occurred. 1 PCI inbound message masked interrupt occurred.
5	PCRW	PCI Command Register Write Masked Interrupt Status 0 PCI command register write masked interrupt has not occurred. 1 PCI command register write masked interrupt occurred.

6	PPM	PCI Power Management Masked Interrupt Status 0 PCI power management masked interrupt has not occurred. 1 PCI power management masked interrupt occurred.
7	MSI0	PCI MSI Level 0 Masked Interrupt Status 0 PCI MSI level 0 masked interrupt has not occurred. 1 PCI MSI level 0 masked interrupt occurred.
8	MSI1	PCI MSI Level 1 Masked Interrupt Status 0 PCI MSI level 1 masked interrupt has not occurred. 1 PCI MSI level 1 masked interrupt occurred.
9	MSI2	PCI MSI Level 2 Masked Interrupt Status 0 PCI MSI level 2 masked interrupt has not occurred. 1 PCI MSI level 2 masked interrupt occurred.
10	MTE	MAL TX EOB Masked Interrupt Status 0 MAL TX EOB interrupt has not occurred. 1 MAL TX EOB interrupt has occurred.
11	MRE	MAL RX EOB Masked Interrupt Status 0 MAL RX EOB interrupt has not occurred. 1 MAL RX EOB interrupt has occurred.
12	D0	DMA Channel 0 Masked Interrupt Status 0 DMA channel 0 interrupt has not occurred. 1 DMA channel 0 interrupt occurred.
13	D1	DMA Channel 1 Masked Interrupt Status 0 DMA channel 1 interrupt has not occurred. 1 DMA channel 1 interrupt occurred.
14	D2	DMA Channel 2 Masked Interrupt Status 0 DMA channel 2 interrupt has not occurred. 1 DMA channel 2 interrupt occurred.
15	D3	DMA Channel 3 Masked Interrupt Status 0 DMA channel 3 interrupt has not occurred. 1 DMA channel 3 interrupt occurred.
16:17		Reserved

18	CT0	GPT Compare Timer 0 Masked Interrupt Status 0 Compare timer 0 interrupt has not occurred. 1 Compare timer 0 interrupt occurred.
19	CT1	GPT Compare Timer 1 Masked Interrupt Status 0 Compare timer1 interrupt has not occurred. 1 Compare timer 1 interrupt occurred.
20	CT2	GPT Compare Timer 2 Masked Interrupt Status 0 Compare timer 2 interrupt has not occurred. 1 Compare timer 2 interrupt occurred.
21	CT3	GPT Compare Timer 3 Masked Interrupt Status 0 Compare timer 3 interrupt has not occurred. 1 Compare timer 3 interrupt occurred.
22	CT4	GPT Compare Timer 4 Masked Interrupt Status 0 Compare timer 4 interrupt has not occurred. 1 Compare timer 4 interrupt occurred.
23	EIR0	External IRQ 0 Masked Interrupt Status 0 External IRQ 0 interrupt has not occurred. 1 External IRQ 0 interrupt occurred.
24	EIR1	External IRQ 1 Masked Interrupt Status 0 External IRQ 1 interrupt has not occurred. 1 External IRQ 1 interrupt occurred.
25	EIR2	External IRQ 2 Masked Interrupt Status 0 External IRQ 2 interrupt has not occurred. 1 External IRQ 2 interrupt occurred.
26	EIR3	External IRQ 3 Masked Interrupt Status 0 External IRQ 3 interrupt has not occurred. 1 External IRQ 3 interrupt occurred.
27	EIR4	External IRQ 4 Masked Interrupt Status 0 External IRQ 4 interrupt has not occurred. 1 External IRQ 4 interrupt occurred.

28	EIR5	External IRQ 5 Masked Interrupt Status 0 External IRQ 5 interrupt has not occurred. 1 An external IRQ 5 interrupt occurred.
29	EIR6	External IRQ 6 Masked Interrupt Status 0 External IRQ 6 interrupt has not occurred. 1 External IRQ 6 interrupt occurred.
30	UIC1NC	UIC1 Non-Critical Masked Interrupt Status 0 UICNC interrupt has not occurred. 1 UICNC interrupt occurred.
31	UIC1C	UIC1 Critical Masked Interrupt Status 0 UICC interrupt has not occurred. 1 UICC interrupt occurred.

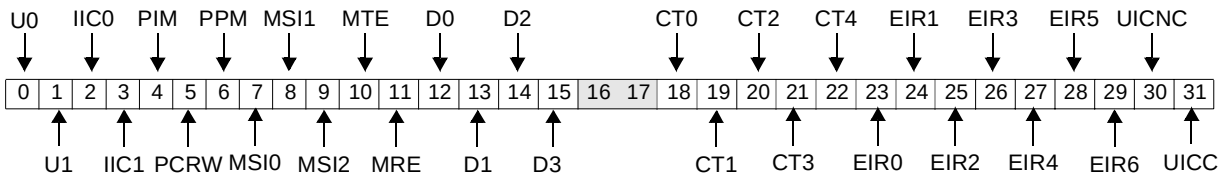


Figure 9-12. UIC0 Masked Status Register (UIC0_MSR)

0	U0	UART0 Masked Interrupt Status 0 UART0masked interrupt has not occurred. 1 UART0 masked interrupt occurred.
1	U1	UART1 Masked Interrupt Status 0 UART1 masked interrupt has not occurred. 1 UART1 masked interrupt occurred.
2	IIC0	IIC0 Masked Interrupt Status 0 IIC0 masked interrupt has not occurred. 1 IIC0 masked interrupt occurred.
3	IIC1	IIC1 Masked Interrupt Status 0 IIC1 masked interrupt has not occurred. 1 IIC1 masked interrupt occurred.
4	PIM	PCI Inbound Message Masked Interrupt Status 0 PCI inbound message masked interrupt has not occurred. 1 PCI inbound message masked interrupt occurred.
5	PCRW	PCI Command Register Write Masked Interrupt Status 0 PCI command register write masked interrupt has not occurred. 1 PCI command register write masked interrupt occurred.

6	PPM	PCI Power Management Masked Interrupt Status 0 PCI power management masked interrupt has not occurred. 1 PCI power management masked interrupt occurred.
7	MSI0	PCI MSI Level 0 Masked Interrupt Status 0 PCI MSI level 0 masked interrupt has not occurred. 1 PCI MSI level 0 masked interrupt occurred.
8	MSI1	PCI MSI Level 1 Masked Interrupt Status 0 PCI MSI level 1 masked interrupt has not occurred. 1 PCI MSI level 1 masked interrupt occurred.
9	MSI2	PCI MSI Level 2 Masked Interrupt Status 0 PCI MSI level 2 masked interrupt has not occurred. 1 PCI MSI level 2 masked interrupt occurred.
10	MTE	MAL TX EOB Masked Interrupt Status 0 MAL TX EOB interrupt has not occurred. 1 MAL TX EOB interrupt has occurred.
11	MRE	MAL RX EOB Masked Interrupt Status 0 MAL RX EOB interrupt has not occurred. 1 MAL RX EOB interrupt has occurred.
12	D0	DMA Channel 0 Masked Interrupt Status 0 DMA channel 0 interrupt has not occurred. 1 DMA channel 0 interrupt occurred.
13	D1	DMA Channel 1 Masked Interrupt Status 0 DMA channel 1 interrupt has not occurred. 1 DMA channel 1 interrupt occurred.
14	D2	DMA Channel 2 Masked Interrupt Status 0 DMA channel 2 interrupt has not occurred. 1 DMA channel 2 interrupt occurred.
15	D3	DMA Channel 3 Masked Interrupt Status 0 DMA channel 3 interrupt has not occurred. 1 DMA channel 3 interrupt occurred.
16:17		Reserved
18	CT0	GPT Compare Timer 0 Masked Interrupt Status 0 Compare timer 0 interrupt has not occurred. 1 Compare timer 0 interrupt occurred.
19	CT1	GPT Compare Timer 1 Masked Interrupt Status 0 Compare timer 1 interrupt has not occurred. 1 Compare timer 1 interrupt occurred.
20	CT2	GPT Compare Timer 2 Masked Interrupt Status 0 Compare timer 2 interrupt has not occurred. 1 Compare timer 2 interrupt occurred.
21	CT3	GPT Compare Timer 3 Masked Interrupt Status 0 Compare timer 3 interrupt has not occurred. 1 Compare timer 3 interrupt occurred.
22	CT4	GPT Compare Timer 4 Masked Interrupt Status 0 Compare timer 4 interrupt has not occurred. 1 Compare timer 4 interrupt occurred.

23	EIR0	External IRQ 0 Masked Interrupt Status 0 External IRQ 0 interrupt has not occurred. 1 External IRQ 0 interrupt occurred.
24	EIR1	External IRQ 1 Masked Interrupt Status 0 External IRQ 1 interrupt has not occurred. 1 External IRQ 1 interrupt occurred.
25	EIR2	External IRQ 2 Masked Interrupt Status 0 External IRQ 2 interrupt has not occurred. 1 External IRQ 2 interrupt occurred.
26	EIR3	External IRQ 3 Masked Interrupt Status 0 External IRQ 3 interrupt has not occurred. 1 External IRQ 3 interrupt occurred.
27	EIR4	External IRQ 4 Masked Interrupt Status 0 External IRQ 4 interrupt has not occurred. 1 External IRQ 4 interrupt occurred.
28	EIR5	External IRQ 5 Masked Interrupt Status 0 External IRQ 5 interrupt has not occurred. 1 An external IRQ 5 interrupt occurred.
29	EIR6	External IRQ 6 Masked Interrupt Status 0 External IRQ 6 interrupt has not occurred. 1 External IRQ 6 interrupt occurred.
30	UIC1NC	UIC1 Non-Critical Masked Interrupt Status 0 UICNC interrupt has not occurred. 1 UICNC interrupt occurred.
31	UIC1C	UIC1 Critical Masked Interrupt Status 0 UICC interrupt has not occurred. 1 UICC interrupt occurred.

9.5.12 UIC1 Masked Status Register (UIC1_MSR)

This read-only register contains the result of masking the UIC1_SR with the UIC1_ER. Reading this register, instead of the actual UIC1_SR, eliminates the need for software to read and apply the enable mask to the contents of the UIC1_SR to determine which enabled interrupt fields are active.

If an interrupt is configured as level-sensitive, and a clear is attempted on the UIC1_SR, the UIC1_SR field is not cleared if the incoming interrupt signal is at the asserted polarity. The interrupt signal must be reset before the UIC1_SR can be successfully cleared. ~~Figure 9-13~~Figure 9-13 describes UIC1_MSR bit definitions.

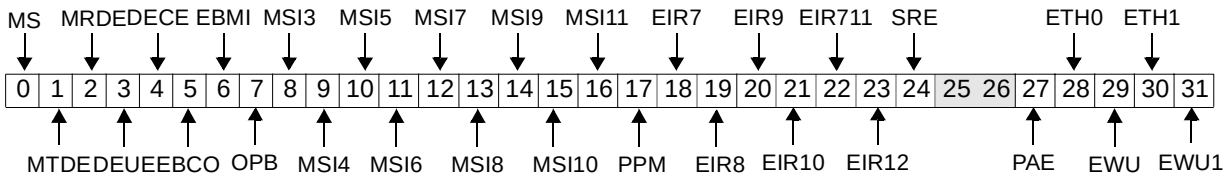


Figure 0-12. UIC1 Masked Status Register (UIC1_MSR)

0	MS	MAL SERR Masked Interrupt Status 0 MAL SERR masked interrupt has not occurred. 1 MAL SERR masked interrupt occurred.

1	MTDE	MAL TXDE Masked Interrupt Status 0 MAL TXDE masked interrupt has not occurred. 1 MAL TXDE masked interrupt occurred.
2	MRDE	MAL RXDE Masked Interrupt Status 0 MAL RXDE masked interrupt has not occurred. 1 MAL RXDE masked interrupt occurred.
3	DEUE	DDRSDRAM ECC Uncorrectable Error Masked Interrupt Status 0 DDRSDRAM ECC uncorrectable error masked interrupt has not occurred. 1 DDRSDRAM ECC uncorrectable error masked interrupt occurred.
4	DECE	DDRSDRAM ECC Correctable Error Masked Interrupt Status 0 DDRSDRAM ECC correctable error masked interrupt has not occurred. 1 DDRSDRAM ECC correctable error masked interrupt occurred.
5	EBCO	EBCO Masked Interrupt Status 0 EBCO interrupt has not occurred. 1 EBCO interrupt occurred.
6	EBMI	EBMI Masked Interrupt Status 0 EBMI masked interrupt has not occurred. 1 EBMI masked interrupt occurred.
7	OPB	OPB to PLB Bridge Masked Interrupt Status 0 OPB to PLB bridge masked interrupt has not occurred. 1 OPB to PLB bridge masked interrupt occurred.
8	MSI3	PCI MSI Level 3 Masked Interrupt Status 0 PCI MSI level 3 masked interrupt has not occurred. 1 PCI MSI level 3 masked interrupt occurred.
9	MSI4	PCI MSI Level 4 Masked Interrupt Status 0 PCI MSI level 4 masked interrupt has not occurred. 1 PCI MSI level 4 masked interrupt occurred.
10	MSI5	PCI MSI Level 5 Masked Interrupt Status 0 PCI MSI level 5 masked interrupt has not occurred. 1 PCI MSI level 5 masked interupt occurred.

11	MSI6	PCI MSI Level 6 Masked Interrupt Status 0 PCI MSI level 6 masked interrupt has not occurred. 1 PCI MSI level 6 masked interrupt occurred.
12	MSI7	PCI MSI Level 7 Masked Interrupt Status 0 PCI MSI level 7 masked interrupt has not occurred. 1 PCI MSI level 7 masked interrupt input occurred.
13	MSI8	PCI MSI Level 8 Masked Interrupt Status 0 PCI MSI level 8 masked interrupt has not occurred. 1 PCI MSI level 8 masked interrupt occurred.
14	MSI9	PCI MSI Level 9 Masked Interrupt Status 0 PCI MSI level 9 masked interrupt has not occurred. 1 PCI MSI level 9 masked interrupt occurred.
15	MSI10	PCI MSI Level 10 Masked Interrupt Status 0 PCI MSI level 10 masked interrupt has not occurred. 1 PCI MSI level 10 masked interrupt occurred.
16	MSI11	PCI MSI Level 11 Masked Interrupt Status 0 PCI MSI level 11 masked interrupt has not occurred. 1 PCI MSI level 11 masked interrupt occurred.
17	PPM	PPM Masked Interrupt Status 0 PPM masked interrupt has not occurred. 1 PPM masked interrupt occurred.
18	EIR7	External IRQ 7 Masked Interrupt Status 0 External IRQ 7 interrupt has not occurred. 1 External IRQ 7 interrupt occurred.
19	EIR8	External IRQ 8 Masked Interrupt Status 0 External IRQ 8 interrupt has not occurred. 1 External IRQ 8 interrupt occurred.
20	EIR9	External IRQ 9 Masked Interrupt Status 0 External IRQ 9 interrupt has not occurred. 1 External IRQ 9 interrupt occurred.

21	EIR10	External IRQ 10 Masked Interrupt Status 0 External IRQ 10 interrupt has not occurred. 1 External IRQ 10 interrupt occurred.
22	EIR11	External IRQ 11 Masked Interrupt Status 0 External IRQ 11 interrupt has not occurred. 1 External IRQ 11 interrupt occurred.
23	EIR12	External IRQ 12 Masked Interrupt Status 0 External IRQ 12 interrupt has not occurred. 1 External IRQ 12 interrupt occurred.
24	SRE	Serial ROM Error Interrupt Status 0 Serial ROM error interrupt has not occurred. 1 Serial ROM error interrupt occurred.
25:26		Reserved
27	PAE	PCI Asynchronous Error Masked Interrupt Status 0 PCI asynchronous error interrupt has not occurred. 1 PCI asynchronous error interrupt occurred.
28	ETH0	Ethernet 0 Masked Interrupt Status 0 Ethernet 0 interrupt has not occurred. 1 Ethernet 0 interrupt occurred.
29	EWU0	Ethernet 0 Wake-up Masked Interrupt Status 0 Ethernet 0 wake-up interrupt has not occurred. 1 Ethernet 0 wake-up interrupt occurred.
30	ETH1	Ethernet 1 Masked Interrupt Status 0 Ethernet 1 interrupt has not occurred. 1 Ethernet 1 interrupt occurred.
31	EWU1	Ethernet 1 Wake-up Masked Interrupt Status 0 Ethernet 1 wake-up interrupt has not occurred. 1 Ethernet 1wake-up interrupt occurred.

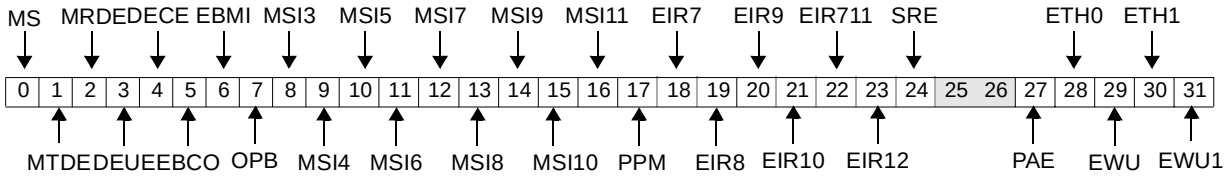


Figure 9-13. UIC1 Masked Status Register (UIC1_MSR)

0	MS	MAL SERR Masked Interrupt Status 0 MAL SERR masked interrupt has not occurred. 1 MAL SERR masked interrupt occurred.
1	MTDE	MAL TXDE Masked Interrupt Status 0 MAL TXDE masked interrupt has not occurred. 1 MAL TXDE masked interrupt occurred.
2	MRDE	MAL RXDE Masked Interrupt Status 0 MAL RXDE masked interrupt has not occurred. 1 MAL RXDE masked interrupt occurred.
3	DEUE	DDRSDRAM ECC Uncorrectable Error Masked Interrupt Status 0 DDRSDRAM ECC uncorrectable error masked interrupt has not occurred. 1 DDRSDRAM ECC uncorrectable error masked interrupt occurred.
4	DECE	DDRSDRAM ECC Correctable Error Masked Interrupt Status 0 DDRSDRAM ECC correctable error masked interrupt has not occurred. 1 DDRSDRAM ECC correctable error masked interrupt occurred.
5	EBCO	EBCO Masked Interrupt Status 0 EBCO interrupt has not occurred. 1 EBCO interrupt occurred.
6	EBMI	EBMI Masked Interrupt Status 0 EBMI masked interrupt has not occurred. 1 EBMI masked interrupt occurred.
7	OPB	OPB to PLB Bridge Masked Interrupt Status 0 OPB to PLB bridge masked interrupt has not occurred. 1 OPB to PLB bridge masked interrupt occurred.
8	MSI3	PCI MSI Level 3 Masked Interrupt Status 0 PCI MSI level 3 masked interrupt has not occurred. 1 PCI MSI level 3 masked interrupt occurred.
9	MSI4	PCI MSI Level 4 Masked Interrupt Status 0 PCI MSI level 4 masked interrupt has not occurred. 1 PCI MSI level 4 masked interrupt occurred.
10	MSI5	PCI MSI Level 5 Masked Interrupt Status 0 PCI MSI level 5 masked interrupt has not occurred. 1 PCI MSI level 5 masked interupt occurred.

11	MSI6	PCI MSI Level 6 Masked Interrupt Status 0 PCI MSI level 6 masked interrupt has not occurred. 1 PCI MSI level 6 masked interrupt occurred.
12	MSI7	PCI MSI Level 7 Masked Interrupt Status 0 PCI MSI level 7 masked interrupt has not occurred. 1 PCI MSI level 7 masked interrupt input occurred.
13	MSI8	PCI MSI Level 8 Masked Interrupt Status 0 PCI MSI level 8 masked interrupt has not occurred. 1 PCI MSI level 8 masked interrupt occurred.
14	MSI9	PCI MSI Level 9 Masked Interrupt Status 0 PCI MSI level 9 masked interrupt has not occurred. 1 PCI MSI level 9 masked interrupt occurred.
15	MSI10	PCI MSI Level 10 Masked Interrupt Status 0 PCI MSI level 10 masked interrupt has not occurred. 1 PCI MSI level 10 masked interrupt occurred.
16	MSI11	PCI MSI Level 11 Masked Interrupt Status 0 PCI MSI level 11 masked interrupt has not occurred. 1 PCI MSI level 11 masked interrupt occurred.
17	PPM	PPM Masked Interrupt Status 0 PPM masked interrupt has not occurred. 1 PPM masked interrupt occurred.
18	EIR7	External IRQ 7 Masked Interrupt Status 0 External IRQ 7 interrupt has not occurred. 1 External IRQ 7 interrupt occurred.
19	EIR8	External IRQ 8 Masked Interrupt Status 0 External IRQ 8 interrupt has not occurred. 1 External IRQ 8 interrupt occurred.
20	EIR9	External IRQ 9 Masked Interrupt Status 0 External IRQ 9 interrupt has not occurred. 1 External IRQ 9 interrupt occurred.
21	EIR10	External IRQ 10 Masked Interrupt Status 0 External IRQ 10 interrupt has not occurred. 1 External IRQ 10 interrupt occurred.
22	EIR11	External IRQ 11 Masked Interrupt Status 0 External IRQ 11 interrupt has not occurred. 1 External IRQ 11 interrupt occurred.
23	EIR12	External IRQ 12 Masked Interrupt Status 0 External IRQ 12 interrupt has not occurred. 1 External IRQ 12 interrupt occurred.
24	SRE	Serial ROM Error Interrupt Status 0 Serial ROM error interrupt has not occurred. 1 Serial ROM error interrupt occurred.
25:26		Reserved
27	PAE	PCI Asynchronous Error Masked Interrupt Status 0 PCI asynchronous error interrupt has not occurred. 1 PCI asynchronous error interrupt occurred.

28	ETH0	Ethernet 0 Masked Interrupt Status 0 Ethernet 0 interrupt has not occurred. 1 Ethernet 0 interrupt occurred.
29	EWU0	Ethernet 0 Wake-up Masked Interrupt Status 0 Ethernet 0 wake-up interrupt has not occurred. 1 Ethernet 0 wake-up interrupt occurred.
30	ETH1	Ethernet 1 Masked Interrupt Status 0 Ethernet 1 interrupt has not occurred. 1 Ethernet 1 interrupt occurred.
31	EWU1	Ethernet 1 Wake-up Masked Interrupt Status 0 Ethernet 1 wake-up interrupt has not occurred. 1 Ethernet 1wake-up interrupt occurred.

9.5.13 UIC0 Vector Configuration Register (UIC0_VCR)

The write-only UIC0_VCR enables software control of interrupt vector generation for critical interrupts. UIC0_VCR contains an address, used as an interrupt vector base address, and specifies interrupt ordering priority. Vector generation is not performed for non-critical interrupts.

UIC0_VCR[VBA] can contain either the base address for an interrupt handler vector table or the base address for the interrupt handler associated with each interrupt. The actual interrupt vector (the address of the interrupt handler that services the interrupt) is generated in the UIC0_VR, using UIC0_VCR[VBA]. Vector generation is described in ~~"UIC0 Vector Register (UIC0_VR)" on page 9-42~~*UIC0 Vector Register (UIC0_VR)* on page 333. Because the two lowest-order bits of an interrupt handler address are assumed to be 00 to ensure word alignment, 30 bits are sufficient to form the base address.

A general interrupt handler uses the vector to access a table of interrupt vectors. Each interrupt vector table entry contains the address of an interrupt handler for a specific interrupt. Alternatively, UIC0_VCR[VBA] can directly address the interrupt handlers for specific interrupts, which in memory are separated by an offset calculated in UIC0_VR.

UIC0_VCR[PRO] controls whether the interrupt associated with UIC0_SR[0] or UIC0_SR[31] has the highest priority. If UIC0_VCR[PRO] = 0, the interrupt associated with UIC0_SR[31] has the highest priority; if UIC0_VCR[PRO] = 1, the interrupt associated with UIC0_SR[0] has the highest priority. The bit closest to the highest priority field that is programmed in the UIC0_CR as a interrupt has the second highest priority. Priority decreases across the UIC0_SR to the end opposite the highest priority field. ~~Figure 9-14~~Figure 9-14 describes UIC0_VCR bit definitions.

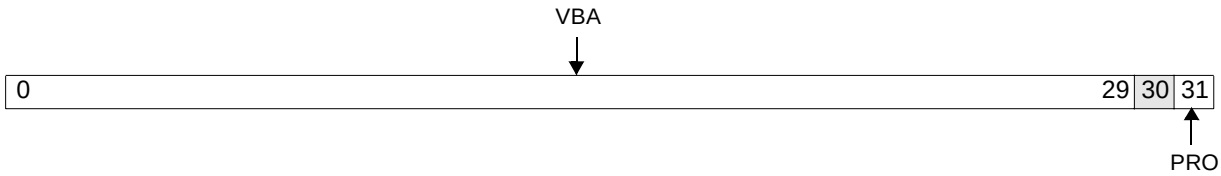


Figure 0-13. UIC0 Vector Configuration Register (UIC0_VCR)

0:29	VBA	Vector Base Address
30		Reserved

31	PRO	Priority Ordering 0 UIC0_SR[31] is the highest priority interrupt. 1 UIC0_SR[0] is the highest priority interrupt. Note: Vector generation is not performed for non-critical interrupts.

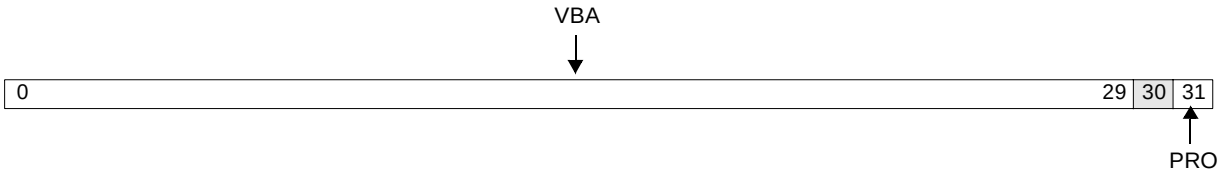


Figure 9-14. UIC0 Vector Configuration Register (UIC0_VCR)

0:29	VBA	Vector Base Address
30		Reserved
31	PRO	Priority Ordering 0 UIC0_SR[31] is the highest priority interrupt. 1 UIC0_SR[0] is the highest priority interrupt. **Note:** Vector generation is not performed for non-critical interrupts.

9.5.14 UIC1 Vector Configuration Register (UIC1_VCR)

The write-only UIC1_VCR enables software control of interrupt vector generation for critical interrupts. UIC1_VCR contains an address, used as an interrupt vector base address, and specifies interrupt ordering priority. Vector generation is not performed for non-critical interrupts.

UIC1_VCR[VBA] can contain either the base address for an interrupt handler vector table or the base address for the interrupt handler associated with each interrupt. The actual interrupt vector (the address of the interrupt handler that services the interrupt) is generated in the UIC1_VR, using UIC1_VCR[VBA]. Vector generation is described in ~~"UIC1 Vector Register (UIC1_VR)" on page 9-44~~*UIC1 Vector Register (UIC1_VR)* on page 335. Because the two lowest-order bits of an interrupt handler address are assumed to be 00 to ensure word alignment, 30 bits are sufficient to form the base address.

A general interrupt handler uses the vector to access a table of interrupt vectors. Each interrupt vector table entry contains the address of an interrupt handler for a specific interrupt. Alternatively, UIC1_VCR[VBA] can directly address the interrupt handlers for specific interrupts, which in memory are separated by an offset calculated in UIC1_VR.

UIC1_VCR[PRO] controls whether the interrupt associated with UIC1_SR[0] or UIC1_SR[31] has the highest priority. If UIC1_VCR[PRO] = 0, the interrupt associated with UIC1_SR[0] has the highest priority; if UIC1_VCR[PRO] = 1, the interrupt associated with UIC1_SR[31] has the highest priority. The bit closest to the highest priority field that is programmed in the UIC*x*_CR as a interrupt has the second highest priority. Priority decreases across the UIC1_SR to the end opposite the highest priority field. ~~Figure 9-15~~Figure 9-15 describes UIC1_VCR bit definitions.

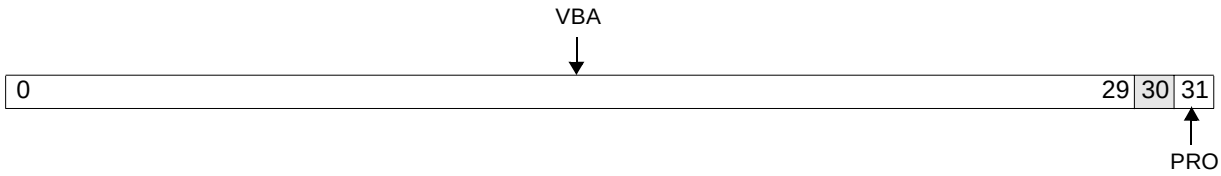


Figure 0-14. UIC1 Vector Configuration Register (UIC1_VCR)

0:29	VBA	Vector Base Address
30		Reserved
31	PRO	Priority Ordering 0 UIC1_SR[31] is the highest priority interrupt. 1 UIC1_SR[0] is the highest priority interrupt. Note: Vector generation is not performed for non-critical interrupts.

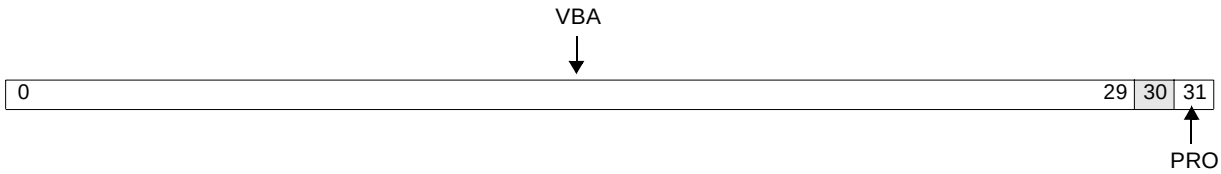


Figure 9-15. UIC1 Vector Configuration Register (UIC1_VCR)

0:29	VBA	Vector Base Address
30		Reserved
31	PRO	Priority Ordering 0 UIC1_SR[31] is the highest priority interrupt. 1 UIC1_SR[0] is the highest priority interrupt. **Note:** Vector generation is not performed for non-critical interrupts.

9.5.15 UIC0 Vector Register (UIC0_VR)

The read-only UIC0_VR contains an interrupt vector that can reduce interrupt handling latency for critical interrupts. Vector generation logic adds an offset to UIC0_VCR[VBA], and the sum is returned in the UIC0_VR. Vectors are not computed for non-critical interrupts.

The interrupt vector is based on the field position of the current highest priority, enabled, active, critical interrupt relative to the highest priority interrupt in the UIC0_SR. The generated vectors can be programmed to point directly to the interrupt handlers.

Programming Note: Regardless of the programming of UIC0_VCR and UIC0_VR registers, the processor always vectors to IVPR and IVOR0 when a critical interrupt occurs.

The interrupt vector offset is based on the bit position of the current highest priority, enabled, active, critical interrupt relative to the highest priority interrupt in the UIC0_SR. The offset has a fixed value of 512 per bit. The main critical interrupt handler can interpret the vector returned by UIC0_VR as the address of the interrupt handler for that interrupt, assuming the routine is 512 bytes or smaller. Alternatively, the main critical interrupt handler can interpret the vector as a look-up table entry for the address of the interrupt handler for that interrupt. Figure 9-16 describes UIC0_VR bit definitions.

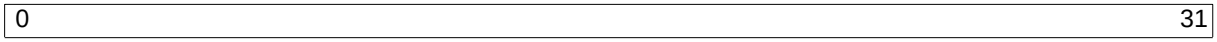


Figure 0-15. UIC Vector Register (UIC*x*_VR)

0:31		Interrupt Vector

Figure 9-16. UIC Vector Register (UIC0_VR)

0:31		Interrupt Vector

The following example illustrates the generation of a UIC0_VR vector for external interrupt request IRQ2.

For the example, assume that UIC0_VCR[PRO] = 1, so that UIC0_SR[EIR6S] (UIC0_SR$_{31}$)has the highest interrupt priority, and that UIC0_SR[EIR2S] (UIC0_SR$_{27}$) is the current highest priority, enabled, active, critical interrupt. To generate the vector for the interrupt associated with UIC0_SR[EIR2S], internal logic multiplies the difference between the highest priority interrupt bit and the active enabled priority interrupt bit by 512. The interrupt vector offset is therefore $(31 - 27) \times 512 = 4 \times 512$. This offset is added to the base address in UIC0_VCR[VBA], and the UIC0_VR returns UIC0_VCR[VBA] + (4×512).

9.5.15.1 Using the Value in UIC0_VR as a Vector Address or Entry Table Lookup

If an interrupt handler is 512 bytes or smaller, system software can interpret the value returned in the UIC0_VR as an address. In this case, when the interrupt is received, the UIC0_VR is read and software simply jumps to the address represented by the UIC0_VR value. Alternatively, the routine can be at a different address, and system software can treat the value of the UIC0_VR as a pointer, storing the interrupt handler address in the UIC0_VR during system initialization. In this case, when the interrupt is handled, software must read the UIC0_VR, read the entry at the UIC0_VR value, and jump to the entry. Hardware has no knowledge of the method is used, which is determined by system software.

9.5.15.2 Vector Generation Scenarios

For the following sequence, assume that the interrupts are enabled and critical (vectors are not generated for disabled or non-critical interrupts). The sequence illustrates several scenarios for vector generation.

1. An intermediate priority interrupt goes active; its vector is stored in UIC0_VR.
2. A low priority interrupt goes active; UIC0_VR is unchanged.
3. Software reads the vector; UIC0_VR is unchanged.
4. Software resets the intermediate priority interrupt; UIC0_VR contains the vector for the low priority interrupt.
5. A high priority interrupt goes active; UIC0_VR contains the vector for the high priority interrupt.
6. Software resets the high priority interrupt; UIC0_VR contains the vector for the low priority interrupt.
7. Software resets the UIC0_ER field for the low priority interrupt, disabling it; UIC0_VR contains 0x00000000.
8. UIC0_CR is reprogrammed to make the low priority interrupt non-critical and UIC0_ER is reprogrammed to re-enable the low priority interrupt; UIC0_VR continues to contain 0x00000000.

9.5.16 UIC1 Vector Register (UIC1_VR)

The read-only UIC1_VR contains an interrupt vector that can reduce interrupt handling latency for critical interrupts. Vector generation logic adds an offset to UIC1_VCR[VBA], and the sum is returned in the UIC1_VR. Vectors are not computed for non-critical interrupts.

The interrupt vector is based on the field position of the current highest priority, enabled, active, critical interrupt relative to the highest priority interrupt in the UIC1_SR. The generated vectors can be programmed to point directly to the interrupt handlers.

> **Programming Note:** Regardless of the programming of UIC1_VCR and UIC1_VR registers, the processor always vectors to IVPR and IVOR0 when a critical interrupt occurs.

The interrupt vector offset is based on the bit position of the current highest priority, enabled, active, critical interrupt relative to the highest priority interrupt in the UIC1_SR. The offset has a fixed value of 512 per bit. The main critical interrupt handler can interpret the vector returned by UIC1_VR as the address of the interrupt handler for that interrupt, assuming the routine is 512 bytes or smaller. Alternatively, the main critical interrupt handler can interpret the vector as a look-up table entry for the address of the interrupt handler for that interrupt. ~~Figure 9-17~~Figure 9-17 describes UIC1_VR bit definitions.

Figure 0-16. UIC1 Vector Register (UIC1_VR)

0:31		Interrupt Vector

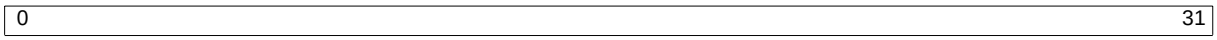


Figure 9-17. UIC1 Vector Register (UIC1_VR)

0:31		Interrupt Vector

The following example illustrates the generation of a UIC1_VR vector for external interrupt request IRQ2.

For the example, assume that UIC1_VCR[PRO] = 1, so that UIC1_SR[EIR6S] (UIC1_SR_{31})has the highest interrupt priority, and that UIC1_SR[EIR2S] (UIC1_SR_{27}) is the current highest priority, enabled, active, critical interrupt. To generate the vector for the interrupt associated with UIC1_SR[EIR2S], internal logic multiplies the difference between the highest priority interrupt bit and the active enabled priority interrupt bit by 512. The interrupt vector offset is therefore $(31 - 27) \times 512 = 4 \times 512$. This offset is added to the base address in UIC1_VCR[VBA], and the UIC1_VR returns UIC1_VCR[VBA] + (4×512).

9.5.16.1 Using the Value in UIC1_VR as a Vector Address or Entry Table Lookup

If an interrupt handler is 512 bytes or smaller, system software can interpret the value returned in the UIC1_VR as an address. In this case, when the interrupt is received, the UIC1_VR is read and software simply jumps to the address represented by the UIC1_VR value. Alternatively, the routine can be at a different address, and system software can treat the value of the UIC1_VR as a pointer, storing the interrupt handler address in the UIC1_VR during system initialization. In this case, when the interrupt is handled, software must read the UIC1_VR, read the entry at the UIC1_VR value, and jump to the entry. Hardware has no knowledge of the method is used, which is determined by system software.

9.5.16.2 Vector Generation Scenarios

For the following sequence, assume that the interrupts are enabled and critical (vectors are not generated for disabled or non-critical interrupts). The sequence illustrates several scenarios for vector generation.

1. An intermediate priority interrupt goes active; its vector is stored in UIC1_VR.
2. A low priority interrupt goes active; UIC1_VR is unchanged.
3. Software reads the vector; UIC1_VR is unchanged.
4. Software resets the intermediate priority interrupt; UIC1_VR contains the vector for the low priority interrupt.
5. A high priority interrupt goes active; UIC1_VR contains the vector for the high priority interrupt.
6. Software resets the high priority interrupt; UIC1_VR contains the vector for the low priority interrupt.
7. Software resets the UIC1_ER field for the low priority interrupt, disabling it; UIC1_VR contains 0x00000000.
8. UIC1_CR is reprogrammed to make the low priority interrupt non-critical and UIC1_ER is reprogrammed to re-enable the low priority interrupt; UIC1_VR continues to contain 0x00000000.

10. Interrupts and Exceptions

This chapter begins by defining the terminology and classification of interrupts and exceptions in ~~"Overview"~~*Overview* and ~~"Interrupt Classes"~~*Interrupt Classes*.

~~"Interrupt Processing" on page 10-4~~*Interrupt Processing* on page 340 explains in general how interrupts are processed, including the requirements for partial execution of instructions.

Several registers support interrupt handling and control. ~~"Interrupt Processing Registers" on page 10-7~~*Interrupt Processing Registers* on page 343 describes these registers.

~~Table 10-2, "Interrupt and Exception Types," on page 10-16~~*Table 10-2 Interrupt and Exception Types* on page 351 lists the interrupts and exceptions handled by the PPC440GP, in the order of Interrupt Vector Offset Register (IVOR) usage. Detailed descriptions of each interrupt type follow, in the same order.

Finally, ~~"Interrupt Ordering and Masking" on page 10-39~~*Interrupt Ordering and Masking* on page 373 and ~~"Exception Priorities" on page 10-42~~*Exception Priorities* on page 376 define the priority order for the processing of simultaneous interrupts and exceptions.

10.1 Overview

An *interrupt* is the action in which the processor saves its old context (Machine State Register (MSR) and next instruction address) and begins execution at a pre-determined interrupt-handler address, with a modified MSR. *Exceptions* are the events that may cause the processor to take an interrupt, if the corresponding interrupt type is enabled.

Exceptions may be generated by the execution of instructions, or by signals from devices external to the PPC440GP, the internal timer facilities, debug events, or error conditions.

10.2 Interrupt Classes

All interrupts, except for Machine Check, can be categorized according to two independent characteristics of the interrupt:

- Asynchronous or synchronous
- Critical or non-critical

10.2.1 Asynchronous Interrupts

Asynchronous interrupts are caused by events that are independent of instruction execution. For asynchronous interrupts, the address reported to the interrupt handling routine is the address of the instruction that would have executed next, had the asynchronous interrupt not occurred.

10.2.2 Synchronous Interrupts

Synchronous interrupts are those that are caused directly by the execution (or attempted execution) of instructions, and are further divided into two classes, *precise* and *imprecise*.

Synchronous, precise interrupts are those that *precisely* indicate the address of the instruction causing the exception that generated the interrupt; or, for certain synchronous, precise interrupt types, the address of the immediately following instruction.

Synchronous, imprecise interrupts are those that may indicate the address of the instruction which caused the exception that generated the interrupt, or the address of some instruction after the one which caused the exception.

10.2.2.1 Synchronous, Precise Interrupts

When the execution or attempted execution of an instruction causes a synchronous, precise interrupt, the following conditions exist when the associated interrupt handler begins execution:

- SRR0 (see ~~"Save/Restore Register 0 (SRR0)" on page 10-9~~*Save/Restore Register 0 (SRR0)* on page 345) or CSRR0 (see ~~"Critical Save/Restore Register 0 (CSRR0)" on page 10-10~~*Critical Save/Restore Register 0 (CSRR0)* on page 346) addresses either the instruction which caused the exception that generated the interrupt, or the instruction immediately following this instruction. Which instruction is addressed can be determined from a combination of the interrupt type and the setting of certain fields of the ESR (see ~~"Exception Syndrome Register (ESR)" on page 10-13~~*Exception Syndrome Register (ESR)* on page 349).
- The interrupt is generated such that all instructions preceding the instruction which caused the exception appear to have completed with respect to the executing processor. However, some storage accesses associated with these preceding instructions may not have been performed with respect to other processors and mechanisms.
- The instruction which caused the exception may appear not to have begun execution (except for having caused the exception), may have been partially executed, or may have completed, depending on the interrupt type (see ~~"Partially Executed Instructions" on page 10-6~~*Partially Executed Instructions* on page 341).
- Architecturally, no instruction beyond the one which caused the exception has executed.

10.2.2.2 Synchronous, Imprecise Interrupts

When the execution or attempted execution of an instruction causes a synchronous, imprecise interrupt, the following conditions exist when the associated interrupt handler begins execution:

- SRR0 or CSRR0 addresses either the instruction which caused the exception that generated the interrupt, or some instruction following this instruction.
- The interrupt is generated such that all instructions preceding the instruction addressed by SRR0 or CSRR0 appear to have completed with respect to the executing processor.
- If the imprecise interrupt is forced by the context synchronizing mechanism, due to an instruction that causes another exception that generates an interrupt (for example, Alignment, Data Storage), then SRR0 addresses the interrupt-forcing instruction, and the interrupt-forcing instruction may have been partially executed (see ~~"Partially Executed Instructions" on page 10-6~~*Partially Executed Instructions* on page 341).
- If the imprecise interrupt is forced by the execution synchronizing mechanism, due to executing an execution synchronizing instruction other than **msync** or **isync**, then SRR0 or CSRR0 addresses the interrupt-forcing instruction, and the interrupt-forcing instruction appears not to have begun execution (except for its forcing the imprecise interrupt). If the imprecise interrupt is forced by an **msync** or **isync** instruc-

tion, then SRR0 or CSRR0 may address either the **msync** or **isync** instruction, or the following instruction.

- If the imprecise interrupt is not forced by either the context synchronizing mechanism or the execution synchronizing mechanism, then the instruction addressed by SRR0 or CSRR0 may have been partially executed (see ~~"Partially Executed Instructions" on page 10-6~~*Partially Executed Instructions* on page 341).
- No instruction following the instruction addressed by SRR0 or CSRR0 has executed.

The only synchronous, imprecise interrupts in the PPC440GP are the "special cases" of "delayed" interrupts, which can result when certain kinds of exceptions occur while the corresponding interrupt type is disabled. The first of these is the Floating-Point Enabled exception type Program interrupt. For this type of interrupt to occur, a floating-point unit must be attached to the auxiliary processor interface of the PPC440GP, and the Floating-point Enabled Exception Summary bit of the Floating-Point Status and Control Register (FPSCR[FEX]) must be set while Floating-point Enabled exception type Program interrupts are disabled due to MSR[FE0,FE1] both being 0. If and when such interrupts are subsequently enabled, by setting one or the other (or both) of MSR[FE0,FE1] to 1 while FPSCR[FEX] is still 1, then a synchronous, imprecise form of Floating-Point Enabled exception type Program interrupt will occur, and SRR0 will be set to the address of the instruction which would have executed next (that is, the instruction after the one which updated MSR[FE0,FE1]). If the MSR was updated by an **rfi** or **rfci** instruction, then SRR0 will be set to the address to which the **rfi** or **rfci** was returning, and not to the instruction address which is sequentially after the **rfi** or **rfci**.

The second type of delayed interrupt which may be handled as a synchronous, imprecise interrupt is the Debug interrupt. Similar to the Floating-Point Enabled exception type Program interrupt, the Debug interrupt can be temporarily disabled by an MSR bit, MSR[DE]. Accordingly, certain kinds of Debug exceptions may occur and be recorded in the DBSR while MSR[DE] is 0, and later lead to a delayed Debug interrupt if MSR[DE] is set to 1 while a Debug exception is still set in the DBSR. If and when this occurs, the interrupt will either be synchronous and imprecise, or it will be asynchronous, depending on the type of Debug exception causing the interrupt. In either case, CSRR0 is set to the address of the instruction which would have executed next (that is, the instruction after the one which set MSR[DE] to 1). If MSR[DE] is set to 1 by **rfi** or **rfci**, then CSRR0 is set to the address to which the **rfi** or **rfci** was returning, and not to the address of the instruction which was sequentially after the **rfi** or **rfci**.

Besides these special cases of Program and Debug interrupts, all other synchronous interrupts are handled precisely by the PPC440GP.

See ~~"Program Interrupt" on page 10-27~~*Program Interrupt* on page 362 and ~~"Debug Interrupt" on page 10-35~~*Debug Interrupt* on page 369 for a more detailed description of these interrupt types, including both the precise and imprecise cases.

10.2.3 Critical and Non-Critical Interrupts

Interrupts can also be classified as critical or noncritical interrupts. Certain interrupt types demand immediate attention, even if other interrupt types are currently being processed and have not yet had the opportunity to save the state of the machine (that is, return address and captured state of the MSR). To enable taking a critical interrupt immediately after a non-critical interrupt has occurred (that is, before the state of the machine has been saved), two sets of Save/Restore Register pairs are provided. Critical interrupts use the Save/Restore Register pair CSRR0/CSRR1. Non-Critical interrupts use Save/Restore Register pair SRR0/SRR1.

10.2.4 Machine Check Interrupts

Machine Check interrupts are a special case. They are typically caused by some kind of hardware or storage subsystem failure, or by an attempt to access an invalid address. A Machine Check may be caused indirectly by the execution of an instruction, but not be recognized and/or reported until long after the processor has executed past the instruction that caused the Machine Check. As such, Machine Check interrupts cannot properly be classified as either synchronous or asynchronous, nor as precise or imprecise. They do belong to the critical class of interrupts however.

Architecturally, the following general rules apply for Machine Check interrupts:

1. No instruction after the one whose address is reported to the Machine Check interrupt handler in CSRR0 has begun execution.
2. The instruction whose address is reported to the Machine Check interrupt handler in CSRR0, and all prior instructions, may or may not have completed successfully. All those instructions that are ever going to complete appear to have done so already, and have done so within the context existing prior to the Machine Check interrupt. No further interrupt (other than possible additional Machine Check interrupts) will occur as a result of those instructions.

With the PPC440GP, Machine Check interrupts can be caused by Machine Check exceptions on a memory access either for an instruction fetch or for a data access. In the case of an Instruction Machine Check exception, the PPC440GP will handle the interrupt as a synchronous, precise interrupt, assuming Machine Check interrupts are enabled (MSR[ME] = 1). That is, if a Machine Check exception is detected during an instruction fetch, the exception will not be *reported* to the interrupt mechanism unless and until execution is attempted for the instruction address at which the Machine Check exception occurred. If, for example, the direction of the instruction stream is changed (perhaps due to a branch instruction), such that the instruction at the address associated with the Machine Check exception will not be executed, then the exception will not be reported and no interrupt will occur. If and when an Instruction Machine Check exception is reported, and if Machine Check interrupts are enabled at the time of the reporting of the exception, then the interrupt will be synchronous and precise and CSRR0 will be set to the instruction address which led to the exception. If Machine Check interrupts are *not* enabled at the time of the reporting of an Instruction Machine Check exception, then a Machine Check interrupt will *not* be generated (*ever*, even if and when MSR[ME] is subsequently set to 1), although the ESR[MCI] field will be set to 1 to indicate that the exception has occurred and that the instruction associated with the exception has been executed.

Data Machine Check exceptions, on the other hand, are handled in an "asynchronous" fashion. That is, instruction execution may proceed past the instruction which prompted the data access which led to the Data Machine Check exception, such that when the exception is reported and a Machine Check interrupt occurs, the address reported in CSRR0 will not be related to the instruction which led (directly or indirectly) to the Data Machine Check exception. If MSR[ME] is 0 at the time of a Data Machine Check exception, a Machine Check interrupt *will* subsequently occur if and when MSR[ME] is set to 1.

See ~~"Machine Check Interrupt" on page 10-19~~*Machine Check Interrupt* on page 354 for more detailed information on Machine Check interrupts.

10.3 Interrupt Processing

Associated with each kind of interrupt is an *interrupt vector*, that is, the address of the initial instruction that is executed when the corresponding interrupt occurs.

Interrupt processing consists of saving a small part of the processor state in certain registers, identifying the cause of the interrupt in another register, and continuing execution at the corresponding interrupt vector location. When an exception exists and the corresponding interrupt type is enabled, the following actions are performed, in order:

1. SRR0 (for non-critical class interrupts) or CSRR0 (for critical class interrupts) is loaded with an instruction address that depends on the type of interrupt; see the specific interrupt description for details.
2. The ESR is loaded with information specific to the exception type. Note that many interrupt types can only be caused by a single type of exception, and thus do not need nor use an ESR setting to indicate the cause of the interrupt. Machine Check interrupts load the MCSR
3. SRR1 (for non-critical class interrupts) or CSRR1 (for critical class interrupts) is loaded with a copy of the contents of the MSR.
4. The MSR is updated as described below. The new values take effect beginning with the first instruction following the interrupt.
   - MSR[WE,EE,PR,FP,FE0,DWE,FE1,IS,DS] are set to 0 by all interrupts.
   - MSR[CE,DE] are set to 0 by all critical class interrupts and left unchanged by all non-critical class interrupts.
   - MSR[ME] is set to 0 by Machine Check interrupts and left unchanged by all other interrupts.

   See "Machine State Register (MSR)" on page 10-7 *Machine State Register (MSR)* on page 343 for more detail on the definition of the MSR.
5. Instruction fetching and execution resumes, using the new MSR value, at the interrupt vector address, which is specific to the interrupt type, and is determined as follows:

   $$IVPR_{0:15} \,||\, IVOR n_{16:27} \,||\, 0b0000$$

   where *n* specifies the IVOR register to be used for a particular interrupt type (see "Interrupt Vector Offset Registers (IVOR0–IVOR15)" on page 10-11 *Interrupt Vector Offset Registers (IVOR0–IVOR15)* on page 347).

At the end of a non-critical interrupt handling routine, execution of an **rfi** causes the MSR to be restored from the contents of SRR1 and instruction execution to resume at the address contained in SRR0. Likewise, execution of an **rfci** performs the same function at the end of a critical interrupt handling routine, using CSRR0 instead of SRR0 and CSRR1 instead of SRR1.

Programming Note: In general, at process switch, due to possible process interlocks and possible data availability requirements, the operating system needs to consider executing the following instructions.

- **stwcx.**, to clear the reservation if one is outstanding, to ensure that a **lwarx** in the "old" process is not paired with a **stwcx.** in the "new" process. See the instruction descriptions for **lwarx** and **stwcx.** in Chapter 28, "Instruction Set" *Instruction Set* on page 893 for more information on storage reservations.
- **msync**, to ensure that all storage operations of an interrupted process are complete with respect to other processors before that process begins executing on another processor.
- **isync**, **rfi** or **rfci**, to ensure that the instructions in the "new" process execute in the "new" context.

10.3.1 Partially Executed Instructions

In general, the architecture permits load and store instructions to be partially executed, interrupted, and then to be restarted from the beginning upon return from the interrupt. In order to guarantee that a particular load or store instruction will complete without being interrupted and restarted, software must mark the storage being referred to as Guarded, and must use an elementary (not a string or multiple) load or store that is aligned on an operand-sized boundary.

In order to guarantee that load and store instructions can, in general, be restarted and completed correctly without software intervention, the following rules apply when an instruction is partially executed and then interrupted:

- For an elementary load, no part of the target register (GPR(RT), FPR(FRT), or auxiliary processor register) will have been altered.
- For the "update" forms of load and store instructions, the update register, GPR(RA), will not have been altered.

On the other hand, the following effects are permissible when certain instructions are partially executed and then restarted:

- For any store instruction, some of the bytes at the addressed storage location may have been accessed and/or updated (if write access to that page in which bytes were altered is permitted by the access control mechanism). In addition, for the **stwcx.** instruction, if the address is not aligned on a word boundary, then the value in CR[CR0] is undefined, as is whether or not the reservation (if one existed) has been cleared.
- For any load, some of the bytes at the addressed storage location may have been accessed (if read access to that page in which bytes were accessed is permitted by the access control mechanism). In addition, for the **lwarx** instruction, if the address is not aligned on a word boundary, it is undefined whether or not a reservation has been set.
- For load multiple and load string instructions, some of the registers in the range to be loaded may have been altered. Including the addressing registers (GPR(RA), and possibly GPR(RB)) in the range to be loaded is an invalid form of these instructions (and a programming error), and thus the rules for partial execution do not protect against overwriting of these registers. Such possible overwriting of the addressing registers makes these invalid forms of load multiple and load strings inherently non-restartable.

In no case will access control be violated.

As previously stated, the only load or store instructions that are guaranteed to not be interrupted after being partially executed are elementary, aligned, guarded loads and stores. All others may be interrupted after being partially executed. The following list identifies the specific instruction types for which interruption after partial execution may occur, as well as the specific interrupt types that could cause the interruption:

1. Any load or store (except elementary, aligned, guarded):

   Critical Input

   Machine Check

   External Input

   Program (Imprecise Mode Floating-Point Enabled)

   > Note that this type of interrupt can lead to partial execution of a load or store instruction under the architectural definition only; the PPC440GP handles the imprecise modes of the Floating-Point Enabled exceptions precisely, and hence this type of interrupt will not lead to partial execution.

Decrementer

Fixed-Interval Timer

Watchdog Timer

Debug (Unconditional Debug Event)

2. Unaligned elementary load or store, or any load or store multiple or string:

   All of the above listed under item 1, plus the following:

   Alignment

   Data Storage (if the access crosses a memory page boundary)

   Debug (Data Address Compare, Data Value Compare)

10.4 Interrupt Processing Registers

The interrupt processing registers include the Save/Restore Registers (SRR0–SRR1), Critical Save/Restore Registers (CSRR0–CSRR1), Data Exception Address Register (DEAR), Interrupt Vector Offset Registers (IVOR0–IVOR15), Interrupt Vector Prefix Register (IVPR), and Exception Syndrome Register (ESR). Also described in this section is the Machine State Register (MSR), which belongs to the category of processor control registers.

10.4.1 Machine State Register (MSR)

The MSR is a register of its own unique type that controls important chip functions, such as the enabling or disabling of various interrupt types.

The MSR can be written from a GPR using the **mtmsr** instruction. The contents of the MSR can be read into a GPR using the **mfmsr** instruction. The MSR[EE] bit can be set or cleared atomically using the **wrtee** or **wrteei** instructions. The MSR contents are also automatically saved, altered, and restored by the interrupt-handling mechanism.

Figure 10-1 shows the MSR bit definitions and describes the function of each bit.

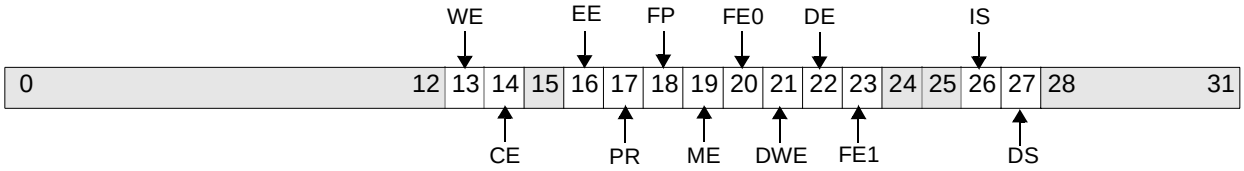


Figure 0-1. Machine State Register (MSR)

0:12		Reserved	
13	WE	Wait State Enable 0 The processor is not in the wait state. 1 The processor is in the wait state.	If MSR[WE] = 1, the processor remains in the wait state until an interrupt is taken, a reset occurs, or an external debug tool clears WE.

14	CE	Critical Interrupt Enable 0 Critical Input and Watchdog Timer interrupts are disabled. 1 Critical Input and Watchdog Timer interrupts are enabled.
15		Reserved
16	EE	External Interrupt Enable 0 External Input, Decrementer, and Fixed Interval Timer interrupts are disabled. 1 External Input, Decrementer, and Fixed Interval Timer interrupts are enabled.
17	PR	Problem State 0 Supervisor state (privileged instructions can be executed) 1 Problem state (privileged instructions can not be executed)
18	FP	Floating Point Available 0 The processor cannot execute floating-point instructions 1 The processor can execute floating-point instructions
19	ME	Machine Check Enable 0 Machine Check interrupts are disabled 1 Machine Check interrupts are enabled.
20	FE0	Floating-point exception mode 0 0 If MSR[FE1] = 0, ignore exceptions mode; if MSR[FE1] = 1, imprecise nonrecoverable mode 1 If MSR[FE1] = 0, imprecise recoverable mode; if MSR[FE1] = 1, precise mode
21	DWE	Debug Wait Enable 0 Disable debug wait mode. 1 Enable debug wait mode.
22	DE	Debug interrupt Enable 0 Debug interrupts are disabled. 1 Debug interrupts are enabled.
23	FE1	Floating-point exception mode 1 0 If MSR[FE0] = 0, ignore exceptions mode; if MSR[FE0] = 1, imprecise recoverable mode 1 If MSR[FE0] = 0, imprecise non-recoverable mode; if MSR[FE0] = 1, precise mode
24:25		Reserved

26	IS	Instruction Address Space 0 All instruction storage accesses are directed to address space 0 (TS = 0 in the relevant TLB entry). 1 All instruction storage accesses are directed to address space 1 (TS = 1 in the relevant TLB entry).
27	DS	Data Address Space 0 All data storage accesses are directed to address space 0 (TS = 0 in the relevant TLB entry). 1 All data storage accesses are directed to address space 1 (TS = 1 in the relevant TLB entry).
28:31		Reserved

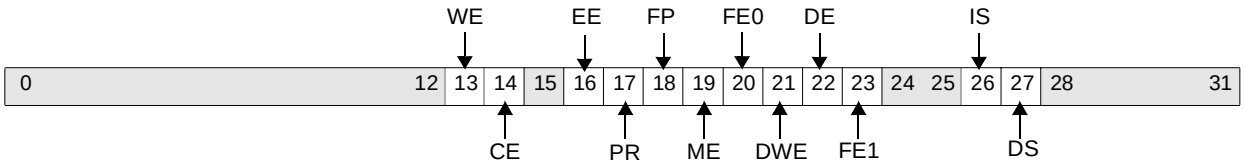


Figure 10-1. Machine State Register (MSR)

0:12		Reserved	
13	WE	Wait State Enable 0 The processor is not in the wait state. 1 The processor is in the wait state.	If MSR[WE] = 1, the processor remains in the wait state until an interrupt is taken, a reset occurs, or an external debug tool clears WE.
14	CE	Critical Interrupt Enable 0 Critical Input and Watchdog Timer interrupts are disabled. 1 Critical Input and Watchdog Timer interrupts are enabled.	
15		Reserved	
16	EE	External Interrupt Enable 0 External Input, Decrementer, and Fixed Interval Timer interrupts are disabled. 1 External Input, Decrementer, and Fixed Interval Timer interrupts are enabled.	
17	PR	Problem State 0 Supervisor state (privileged instructions can be executed) 1 Problem state (privileged instructions can not be executed)	
18	FP	Floating Point Available 0 The processor cannot execute floating-point instructions 1 The processor can execute floating-point instructions	

19	ME	Machine Check Enable 0 Machine Check interrupts are disabled 1 Machine Check interrupts are enabled.
20	FE0	Floating-point exception mode 0 0 If MSR[FE1] = 0, ignore exceptions mode; if MSR[FE1] = 1, imprecise nonrecoverable mode 1 If MSR[FE1] = 0, imprecise recoverable mode; if MSR[FE1] = 1, precise mode
21	DWE	Debug Wait Enable 0 Disable debug wait mode. 1 Enable debug wait mode.
22	DE	Debug interrupt Enable 0 Debug interrupts are disabled. 1 Debug interrupts are enabled.
23	FE1	Floating-point exception mode 1 0 If MSR[FE0] = 0, ignore exceptions mode; if MSR[FE0] = 1, imprecise recoverable mode 1 If MSR[FE0] = 0, imprecise non-recoverable mode; if MSR[FE0] = 1, precise mode
24:25		Reserved
26	IS	Instruction Address Space 0 All instruction storage accesses are directed to address space 0 (TS = 0 in the relevant TLB entry). 1 All instruction storage accesses are directed to address space 1 (TS = 1 in the relevant TLB entry).
27	DS	Data Address Space 0 All data storage accesses are directed to address space 0 (TS = 0 in the relevant TLB entry). 1 All data storage accesses are directed to address space 1 (TS = 1 in the relevant TLB entry).
28:31		Reserved

10.4.2 Save/Restore Register 0 (SRR0)

SRR0 is an SPR that is used to save machine state on non-critical interrupts, and to restore machine state when an **rfi** is executed. When a non-critical interrupt occurs, SRR0 is set to an address associated with the process which was executing at the time. When **rfi** is executed, instruction execution returns to the address in SRR0.

In general, SRR0 contains the address of the instruction that caused the non-critical interrupt, or the address of the instruction to return to after a non-critical interrupt is serviced. See the individual descriptions under ~~"Interrupt Definitions" on page 10-16~~*Interrupt Definitions* on page 351 for an explanation of the precise address recorded in SRR0 for each non-critical interrupt type.

SRR0 can be written from a GPR using **mtspr**, and can be read into a GPR using **mfspr**.

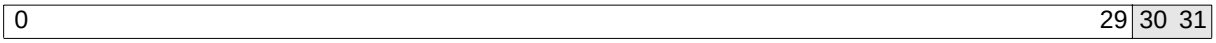


Figure 0-2. Save/Restore Register 0 (SRR0)

0:29		Return address for non-critical interrupts
30:31		Reserved

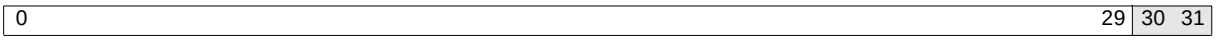


Figure 10-2. Save/Restore Register 0 (SRR0)

0:29		Return address for non-critical interrupts
30:31		Reserved

10.4.3 Save/Restore Register 1 (SRR1)

SRR1 is an SPR that is used to save machine state on non-critical interrupts, and to restore machine state when an **rfi** is executed. When a non-critical interrupt is taken, the contents of the MSR (prior to the MSR being cleared by the interrupt) are placed into SRR1. When **rfi** is executed, the MSR is restored with the contents of SRR1.

Bits of SRR1 that correspond to reserved bits in the MSR are also reserved.

Programming Note: An MSR bit that is reserved may be altered by **rfi**, consistent with the value being restored from SRR1.

SRR1 can be written from a GPR using **mtspr**, and can be read into a GPR using **mfspr**.

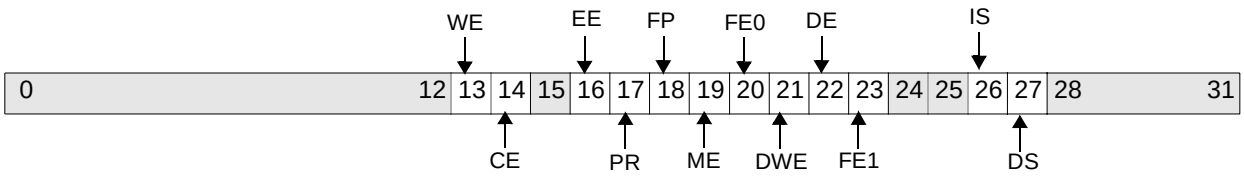


Figure 0-3. Save/Restore Register 1 (SRR1)

0:31		Copy of the MSR at the time of a non-critical interrupt.

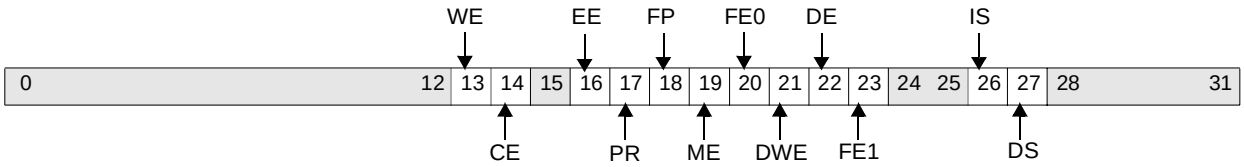


Figure 10-3. Save/Restore Register 1 (SRR1)

0:31		Copy of the MSR at the time of a non-critical interrupt.

10.4.4 Critical Save/Restore Register 0 (CSRR0)

CSRR0 is an SPR that is used to save machine state on critical interrupts, and to restore machine state when an **rfci** is executed. When a critical interrupt occurs, CSRR0 is set to an address associated with the process which was executing at the time. When **rfci** is executed, instruction execution returns to the address in CSRR0.

In general, CSRR0 contains the address of the instruction that caused the critical interrupt, or the address of the instruction to return to after a critical interrupt is serviced. See the individual descriptions under ~~"Interrupt Definitions" on page 10-16~~*Interrupt Definitions* on page 351 for an explanation of the precise address recorded in CSRR0 for each critical interrupt type.

CSRR0 can be written from a GPR using **mtspr**, and can be read into a GPR using **mfspr**.

Figure 0-4. Critical Save/Restore Register 0 (CSRR0)

0:29		Return address for critical interrupts
30:31		Reserved

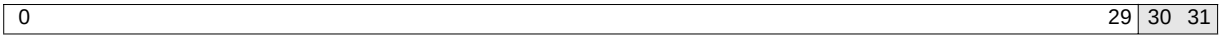


Figure 10-4. Critical Save/Restore Register 0 (CSRR0)

0:29		Return address for critical interrupts
30:31		Reserved

10.4.5 Critical Save/Restore Register 1 (CSRR1)

CSRR1 is an SPR that is used to save machine state on critical interrupts, and to restore machine state when an **rfci** is executed. When a critical interrupt is taken, the contents of the MSR (prior to the MSR being cleared by the interrupt) are placed into CSRR1. When **rfci** is executed, the MSR is restored with the contents of CSRR1.

Bits of CSRR1 that correspond to reserved bits in the MSR are also reserved.

Programming Note: An MSR bit that is reserved may be altered by **rfci**, consistent with the value being restored from CSRR1.

CSRR1 can be written from a GPR using **mtspr**, and can be read into a GPR using **mfspr**.

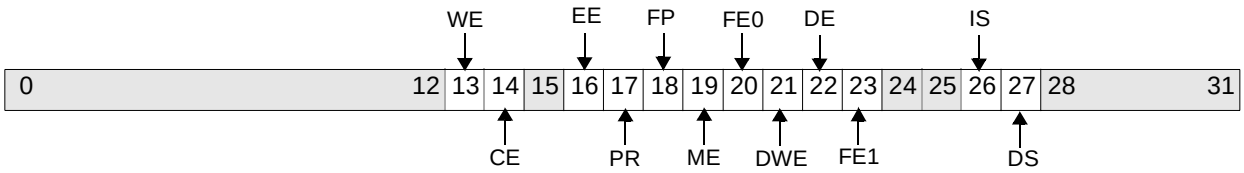


Figure 0-5. Critical Save/Restore Register 1 (CSRR1)

0:31	Copy of the MSR when a critical interrupt is taken

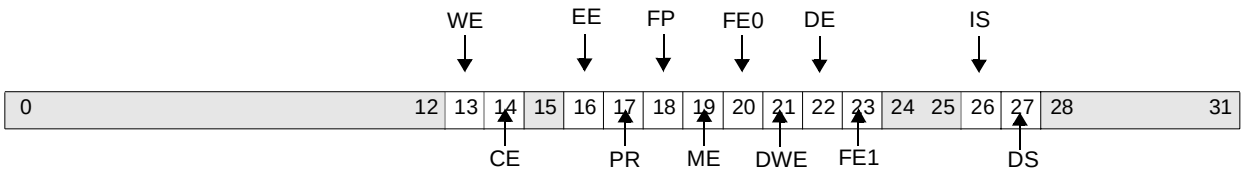


Figure 10-5. Critical Save/Restore Register 1 (CSRR1)

0:31	Copy of the MSR when a critical interrupt is taken

10.4.6 Data Exception Address Register (DEAR)

The DEAR contains the address that was referenced by a load, store, or cache management instruction that caused an Alignment, Data TLB Miss, or Data Storage exception.

The DEAR can be written from a GPR using **mtspr**, and can be read into a GPR using **mfspr**.

Figure 0-6. Data Exception Address Register (DEAR)

0:31		Address of data exception for Data Storage, Alignment, and Data TLB Error interrupts

Figure 10-6. Data Exception Address Register (DEAR)

0:31		Address of data exception for Data Storage, Alignment, and Data TLB Error interrupts

10.4.7 Interrupt Vector Offset Registers (IVOR0–IVOR15)

An IVOR specifies the quadword (16 byte)-aligned interrupt vector offset from the base address provided by the IVPR (see ~~"Interrupt Vector Prefix Register (IVPR)" on page 10-12~~*Interrupt Vector Prefix Register (IVPR)* on page 348) for its respective interrupt type. IVOR0–IVOR15 are provided for the defined interrupt types. The interrupt vector effective address is formed as follows:

$$IVPR_{0:15} \,||\, IVOR n_{16:27} \,||\, 0b0000$$

where *n* specifies the IVOR register to be used for the particular interrupt type.

Any IVOR can be written from a GPR using **mtspr**, and can be read into a GPR using **mfspr**.

~~Figure 10-7~~Figure 10-7 shows the IVOR field definitions, while Table 10-1 identifies the specfic IVOR register associated with each interrupt type.

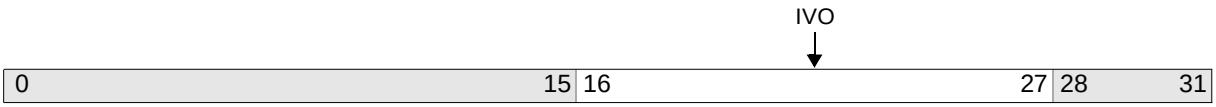


Figure 0-7. Interrupt Vector Offset Registers (IVOR0–IVOR15)

0:15		Reserved
16:27	IVO	Interrupt Vector Offset
28:31		Reserved

Table 0-1. Interrupt Types Associated with each IVOR

IVOR	Interrupt Type
IVOR0	Critical Input
IVOR1	Machine Check
IVOR2	Data Storage
IVOR3	Instruction Storage
IVOR4	External Input
IVOR5	Alignment
IVOR6	Program
IVOR7	Floating Point Unavailable
IVOR8	System Call
IVOR9	Auxiliary Processor Unavailable
IVOR10	Decrementer
IVOR11	Fixed Interval Timer
IVOR12	Watchdog Timer
IVOR13	Data TLB Error
IVOR14	Instruction TLB Error
IVOR15	Debug

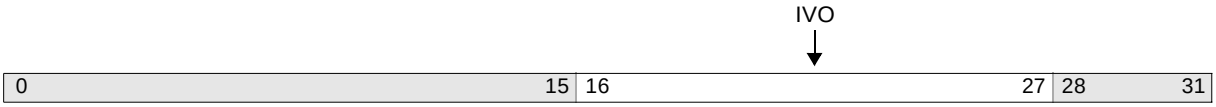


Figure 10-7. Interrupt Vector Offset Registers (IVOR0–IVOR15)

0:15		Reserved

16:27	IVO	Interrupt Vector Offset
28:31		Reserved

Table 10-1. Interrupt Types Associated with each IVOR

IVOR	Interrupt Type
IVOR0	Critical Input
IVOR1	Machine Check
IVOR2	Data Storage
IVOR3	Instruction Storage
IVOR4	External Input
IVOR5	Alignment
IVOR6	Program
IVOR7	Floating Point Unavailable
IVOR8	System Call
IVOR9	Auxiliary Processor Unavailable
IVOR10	Decrementer
IVOR11	Fixed Interval Timer
IVOR12	Watchdog Timer
IVOR13	Data TLB Error
IVOR14	Instruction TLB Error
IVOR15	Debug

10.4.8 Interrupt Vector Prefix Register (IVPR)

The IVPR provides the high-order 16 bits of the effective address of the interrupt vectors, for all interrupt types. The interrupt vector effective address is formed as follows:

$$IVPR_{0:15} \,||\, IVOR\mathit{n}_{16:27} \,||\, 0b0000$$

where *n* specifies the IVOR register to be used for the particular interrupt type.

The IVPR can be written from a GPR using **mtspr**, and can be read into a GPR using **mfspr**.

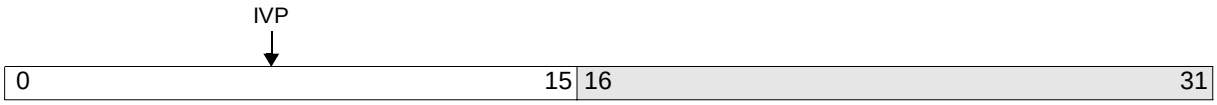


Figure 0-8. Interrupt Vector Prefix Register (IVPR)

0:15	IVP	Interrupt Vector Prefix
16:31		Reserved

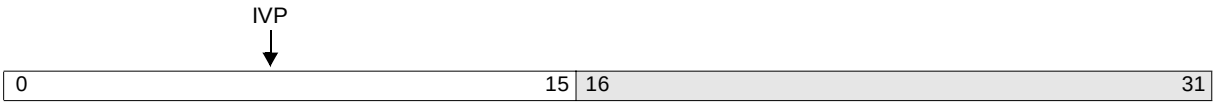


Figure 10-8. Interrupt Vector Prefix Register (IVPR)

0:15	IVP	Interrupt Vector Prefix
16:31		Reserved

10.4.9 Exception Syndrome Register (ESR)

The ESR provides a *syndrome* to differentiate between the different kinds of exceptions that can generate the same interrupt type. Upon the generation of one of these types of interrupt, the bit or bits corresponding to the specific exception that generated the interrupt is set, and all other ESR bits are cleared. Other interrupt types do not affect the contents of the ESR. ~~Figure 10-9 on page 10-13~~*Figure 10-9* on page 349 provides a summary of the fields of the ESR along with their definitions. See the individual interrupt descriptions under ~~"Interrupt Definitions" on page 10-16~~*Interrupt Definitions* on page 351 for an explanation of the ESR settings for each interrupt type, as well as a more detailed explanation of the function of certain ESR fields.

The ESR can be written from a GPR using **mtspr**, and can be read into a GPR using **mfspr**.

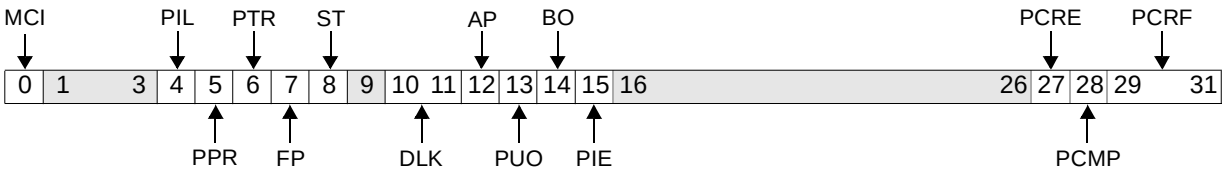


Figure 0-9. Exception Syndrome Register (ESR)

0	MCI	Machine Check—Instruction Fetch Exception 0 Instruction Machine Check exception did not occur. 1 Instruction Machine Check exception occurred.	This is an implementation-dependent field of the ESR and is not part of the PowerPC Book-E Architecture.
1:3		Reserved	
4	PIL	Program Interrupt—Illegal Instruction Exception 0 Illegal Instruction exception did not occur. 1 Illegal Instruction exception occurred.	

5	PPR	Program Interrupt—Privileged Instruction Exception 0 Privileged Instruction exception did not occur. 1 Privileged Instruction exception occurred.
6	PTR	Program Interrupt—Trap Exception 0 Trap exception did not occur. 1 Trap exception occurred.
7	FP	Floating Point Operation 0 Exception was not caused by a floating point instruction. 1 Exception was caused by a floating point instruction.
8	ST	Store Operation 0 Exception was not caused by a store-type storage access or cache management instruction. 1 Exception was caused by a store-type storage access or cache management instruction.
9		Reserved
10:11	DLK	Data Storage Interrupt—Locking Exception 00 Locking exception did not occur. 01 Locking exception was caused by **dcbf**. 10 Locking exception was caused by **icbi**. 11 Reserved
12	AP	AP Operation 0 Exception was not caused by an auxiliary processor instruction. 1 Exception was caused by an auxiliary processor instruction.
13	PUO	Program Interrupt—Unimplemented Operation Exception 0 Unimplemented Operation exception did not occur. 1 Unimplemented Operation exception occurred.
14	BO	Byte Ordering Exception 0 Byte Ordering exception did not occur. 1 Byte Ordering exception occurred.

15	PIE	Program Interrupt—Imprecise Exception 0 Exception occurred precisely; SRR0 contains the address of the instruction that caused the exception. 1 Exception occurred imprecisely; SRR0 contains the address of an instruction after the one which caused the exception.	This field is only set for a Floating-Point Enabled exception type Program interrupt, and then only when the interrupt occurs imprecisely due to MSR[FE0,FE1] being set to a non-zero value when an attached floating-point unit is already signaling the Floating-Point Enabled exception (that is, FPSCR[FEX] is already 1).
16:26		Reserved	
27	PCRE	Program Interrupt—Condition Register Enable 0 Instruction which caused the exception is not a floating-point CR-updating instruction. 1 Instruction which caused the exception is a floating-point CR-updating instruction.	This is an implementation-dependent field of the ESR and is not part of the PowerPC Book-E Architecture. This field is only defined for a Floating-Point Enabled exception type Program interrupt, and then only when ESR[PIE] is 0.
28	PCMP	Program Interrupt—Compare 0 Instruction which caused the exception is not a floating-point compare type instruction 1 Instruction which caused the exception is a floating-point compare type instruction.	This is an implementation-dependent field of the ESR and is not part of the PowerPC Book-E Architecture. This field is only defined for a Floating-Point Enabled exception type Program interrupt, and then only when ESR[PIE] is 0.
29:31	PCRF	Program Interrupt—Condition Register Field If ESR[PCRE]=1, this field indicates which CR field was to be updated by the floating-point instruction which caused the exception.	This is an implementation-dependent field of the ESR and is not part of the PowerPC Book-E Architecture. This field is only defined for a Floating-Point Enabled exception type Program interrupt, and then only when ESR[PIE] is 0.

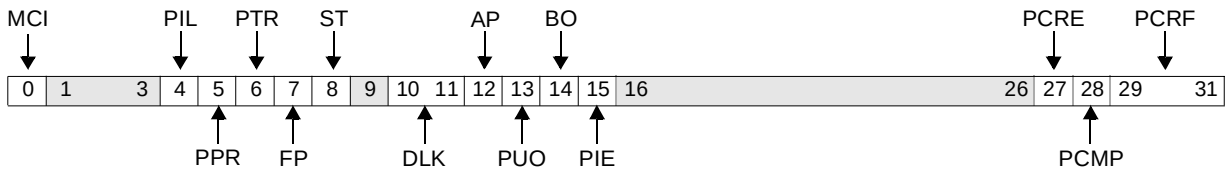


Figure 10-9. Exception Syndrome Register (ESR)

0	MCI	Machine Check—Instruction Fetch Exception 0 Instruction Machine Check exception did not occur. 1 Instruction Machine Check exception occurred.	This is an implementation-dependent field of the ESR and is not part of the PowerPC Book-E Architecture.
1:3		Reserved	

4	PIL	Program Interrupt—Illegal Instruction Exception 0 Illegal Instruction exception did not occur. 1 Illegal Instruction exception occurred.
5	PPR	Program Interrupt—Privileged Instruction Exception 0 Privileged Instruction exception did not occur. 1 Privileged Instruction exception occurred.
6	PTR	Program Interrupt—Trap Exception 0 Trap exception did not occur. 1 Trap exception occurred.
7	FP	Floating Point Operation 0 Exception was not caused by a floating point instruction. 1 Exception was caused by a floating point instruction.
8	ST	Store Operation 0 Exception was not caused by a store-type storage access or cache management instruction. 1 Exception was caused by a store-type storage access or cache management instruction.
9		Reserved
10:11	DLK	Data Storage Interrupt—Locking Exception 00 Locking exception did not occur. 01 Locking exception was caused by **dcbf**. 10 Locking exception was caused by **icbi**. 11 Reserved
12	AP	AP Operation 0 Exception was not caused by an auxiliary processor instruction. 1 Exception was caused by an auxiliary processor instruction.
13	PUO	Program Interrupt—Unimplemented Operation Exception 0 Unimplemented Operation exception did not occur. 1 Unimplemented Operation exception occurred.
14	BO	Byte Ordering Exception 0 Byte Ordering exception did not occur. 1 Byte Ordering exception occurred.

15	PIE	Program Interrupt—Imprecise Exception 0 Exception occurred precisely; SRR0 contains the address of the instruction that caused the exception. 1 Exception occurred imprecisely; SRR0 contains the address of an instruction after the one which caused the exception.	This field is only set for a Floating-Point Enabled exception type Program interrupt, and then only when the interrupt occurs imprecisely due to MSR[FE0,FE1] being set to a non-zero value when an attached floating-point unit is already signaling the Floating-Point Enabled exception (that is, FPSCR[FEX] is already 1).
16:26		Reserved	
27	PCRE	Program Interrupt—Condition Register Enable 0 Instruction which caused the exception is not a floating-point CR-updating instruction. 1 Instruction which caused the exception is a floating-point CR-updating instruction.	This is an implementation-dependent field of the ESR and is not part of the PowerPC Book-E Architecture. This field is only defined for a Floating-Point Enabled exception type Program interrupt, and then only when ESR[PIE] is 0.
28	PCMP	Program Interrupt—Compare 0 Instruction which caused the exception is not a floating-point compare type instruction 1 Instruction which caused the exception is a floating-point compare type instruction.	This is an implementation-dependent field of the ESR and is not part of the PowerPC Book-E Architecture. This field is only defined for a Floating-Point Enabled exception type Program interrupt, and then only when ESR[PIE] is 0.
29:31	PCRF	Program Interrupt—Condition Register Field If ESR[PCRE]=1, this field indicates which CR field was to be updated by the floating-point instruction which caused the exception.	This is an implementation-dependent field of the ESR and is not part of the PowerPC Book-E Architecture. This field is only defined for a Floating-Point Enabled exception type Program interrupt, and then only when ESR[PIE] is 0.

10.5 Interrupt Definitions

Table 10-2 provides a summary of each interrupt type, in the order corresponding to their associated IVOR register. The table also summarizes the various exception types that may cause that interrupt type; the classification of the interrupt; which ESR bit(s) can be set, if any; and which mask bit(s) can mask the interrupt type, if any.

Detailed descrptions of each of the interrupt types follow the table..

Table 0-2. Interrupt and Exception Types

IVOR	Interrupt Type	Exception Type	Asynchronous	Synchronous, Precise	Synchronous, Imprecise	Critical	ESR (See Note 4)	MSR Mask Bit(s)	DBCR0/TCR Mask Bit	Notes
IVOR0	Critical Input	Critical Input	x			x		CE		1
IVOR1	Machine Check	Instruction Machine Check				x	MCI	ME		2
		Data Machine Check				x		ME		2
IVOR2	Data Storage	Read Access Control		x			[FP,AP]			
		Write Access Control		x			ST,[FP,AP]			
		Cache Locking		x			$\{DLK_0,DLK_1\}$			
		Byte Ordering		x			BO,[ST],[FP,AP]			5
IVOR3	Instruction Storage	Execute Access Control		x						
		Byte Ordering		x			BO			6
IVOR4	External Input	External Input	x					EE		1
IVOR5	Alignment	Alignment		x			[ST],[FP,AP]			
IVOR6	Program	Illegal Instruction		x			PIL			
		Privileged Instruction		x			PPR,[AP]			
		Trap		x			PTR			
		FP Enabled		x	x		FP,[PIE],[PCRE] {PCMP,PCRF}	FE0 FE1		8
		AP Enabled		x			AP			8
		Unimplemented Op		x			PUO,[FP,AP]			7
IVOR7	FP Unavailable	FP Unavailable		x						8
IVOR8	System Call	System Call		x						
IVOR9	AP Unavailable	AP Unavailable		x						8
IVOR10	Decrementer	Decrementer	x					EE	DIE	
IVOR11	Fixed Interval Timer	Fixed Interval Timer	x					EE	FIE	
IVOR12	Watchdog Timer	Watchdog Timer	x			x		CE	WIE	
IVOR13	Data TLB Error	Data TLB Miss		x			[ST],[FP,AP]			
IVOR14	Instruction TLB Error	Instruction TLB Miss		x						

Table 0-2. Interrupt and Exception Types

IVOR	Interrupt Type	Exception Type	Asynchronous	Synchronous, Precise	Synchronous, Imprecise	Critical	ESR (See Note 4)	MSR Mask Bit(s)	DBCR0/TCR Mask Bit	Notes
IVOR15	Debug	Trap		x	x	x		DE	IDM	3
		Instruction Address Compare		x	x	x		DE	IDM	3
		Data Address Compare	x	x	x	x		DE	IDM	3
		Data Value Compare	x	x	x	x		DE	IDM	3
		Instruction Complete		x	x	x		DE	IDM	3
		Branch Taken		x		x		DE	IDM	3
		Return		x	x	x		DE	IDM	3
		Interrupt	x			x		DE	IDM	
		Unconditional	x			x		DE	IDM	

Table 0-2. Interrupt and Exception Types

IVOR	Interrupt Type	Exception Type	Asynchronous	Synchronous, Precise	Synchronous, Imprecise	Critical	ESR (See Note 4)	MSR Mask Bit(s)	DBCR0/TCR Mask Bit	Notes

Table Notes

1. Although it is not specified as part of Book E, it is common for system implementations to provide, as part of the interrupt controller, independent mask and status bits for the various sources of Critical Input and External Input interrupts.
2. Machine Check interrupts are not classified as asynchronous nor synchronous. See "Machine Check Interrupts" on page 10-3.
3. Debug exceptions have special rules regarding their interrupt classification (synchronous or asynchronous, and precise or imprecise), depending on the particular debug mode being used and other conditions (see "Debug Interrupt" on page 10-35).
4. In general, when an interrupt causes a particular ESR bit or bits to be set as indicated in the table, it also causes all other ESR bits to be cleared. Special rules apply to the ESR[MCI] field; see "Machine Check Interrupt" on page 10-19. If no ESR setting is indicated for any of the exception types within a given interrupt type, then the ESR is unchanged for that interrupt type.

   The syntax for the ESR setting indication is as follows:

   [xxx] means ESR[xxx] *may* be set

   [xxx,yyy,zzz] means any *one* (or none) of ESR[xxx] or ESR[yyy] or ESR[zzz] *may* be set, but never more than one

   {xxx,yyy,zzz} means that any combination of ESR[xxx], ESR[yyy], and ESR[zzz] *may* be set, including all or none

   xxx means ESR[xxx] *will* be set
5. Byte Ordering exception type Data Storage interrupts can only occur when the PPC440GP is connected to a floating-point unit or auxiliary processor, and then only when executing FP or AP load or store instructions. See "Data Storage Interrupt" on page 10-21 for more detailed information on these kinds of exceptions.
6. Byte Ordering exception type Instruction Storage interrupts are defined by the PowerPC Book-E architecture, but cannot occur within the PPC440GP. The core is capable of executing instructions from both big endian and little endian code pages.
7. Unimplemented Operation exception type Program interrupts can only occur when the PPC440GP is connected to a floating-point unit or auxiliary processor, and then only when executing instruction opcodes which are recognized by the floating-point unit or auxiliary processor but are not implemented within the hardware.
8. Floating-Point Unavailable and Auxiliary Processor Unavailable interrupts, as well as Floating-Point Enabled and Auxiliary Processor Enabled exception type Program interrupts, can only occur when the PPC440GP is connected to a floating-point unit or auxiliary processor, and then only when executing instruction opcodes which are recognized by the floating-point unit or auxiliary processor, respectively.

Table 10-2. Interrupt and Exception Types

IVOR	Interrupt Type	Exception Type	Asynchronous	Synchronous, Precise	Synchronous, Imprecise	Critical	ESR (See Note 4)	MSR Mask Bit(s)	DBCR0/TCR Mask Bit	Notes
IVOR0	Critical Input	Critical Input	x			x		CE		1
IVOR1	Machine Check	Instruction Machine Check				x	MCI	ME		2
		Data Machine Check				x		ME		2
IVOR2	Data Storage	Read Access Control		x			[FP,AP]			
		Write Access Control		x			ST,[FP,AP]			
		Cache Locking		x			{DLK_0,DLK_1}			
		Byte Ordering		x			BO,[ST],[FP,AP]			5
IVOR3	Instruction Storage	Execute Access Control		x						
		Byte Ordering		x			BO			6
IVOR4	External Input	External Input	x					EE		1
IVOR5	Alignment	Alignment		x			[ST],[FP,AP]			
IVOR6	Program	Illegal Instruction		x			PIL			
		Privileged Instruction		x			PPR,[AP]			
		Trap		x			PTR			
		FP Enabled		x	x		FP,[PIE],[PCRE] {PCMP,PCRF}	FE0 FE1		8
		AP Enabled		x			AP			8
		Unimplemented Op		x			PUO,[FP,AP]			7
IVOR7	FP Unavailable	FP Unavailable		x						8
IVOR8	System Call	System Call		x						
IVOR9	AP Unavailable	AP Unavailable		x						8
IVOR10	Decrementer	Decrementer	x					EE	DIE	
IVOR11	Fixed Interval Timer	Fixed Interval Timer	x					EE	FIE	
IVOR12	Watchdog Timer	Watchdog Timer	x			x		CE	WIE	
IVOR13	Data TLB Error	Data TLB Miss		x			[ST],[FP,AP]			
IVOR14	Instruction TLB Error	Instruction TLB Miss		x						

Table 10-2. Interrupt and Exception Types

IVOR	Interrupt Type	Exception Type	Asynchronous	Synchronous, Precise	Synchronous, Imprecise	Critical	ESR (See Note 4)	MSR Mask Bit(s)	DBCR0/TCR Mask Bit	Notes
IVOR15	Debug	Trap		x	x	x		DE	IDM	3
		Instruction Address Compare		x	x	x		DE	IDM	3
		Data Address Compare	x	x	x	x		DE	IDM	3
		Data Value Compare	x	x	x	x		DE	IDM	3
		Instruction Complete		x	x	x		DE	IDM	3
		Branch Taken		x		x		DE	IDM	3
		Return		x	x	x		DE	IDM	3
		Interrupt	x			x		DE	IDM	
		Unconditional	x			x		DE	IDM	

Table 10-2. Interrupt and Exception Types

IVOR	Interrupt Type	Exception Type	Asynchronous	Synchronous, Precise	Synchronous, Imprecise	Critical	ESR (See Note 4)	MSR Mask Bit(s)	DBCR0/TCR Mask Bit	Notes

Table Notes

1. Although it is not specified as part of Book E, it is common for system implementations to provide, as part of the interrupt controller, independent mask and status bits for the various sources of Critical Input and External Input interrupts.
2. Machine Check interrupts are not classified as asynchronous nor synchronous. See *Machine Check Interrupts* on page 339, and *Machine Check Interrupt* on page 354.
3. Debug exceptions have special rules regarding their interrupt classification (synchronous or asynchronous, and precise or imprecise), depending on the particular debug mode being used and other conditions (see *Debug Interrupt* on page 369).
4. In general, when an interrupt causes a particular ESR bit or bits to be set as indicated in the table, it also causes all other ESR bits to be cleared. Special rules apply to the ESR[MCI] field; see *Machine Check Interrupt* on page 354. If no ESR setting is indicated for any of the exception types within a given interrupt type, then the ESR is unchanged for that interrupt type.

   The syntax for the ESR setting indication is as follows:

   [xxx] means ESR[xxx] *may* be set

   [xxx,yyy,zzz] means any *one* (or none) of ESR[xxx] or ESR[yyy] or ESR[zzz] *may* be set, but never more than one

   {xxx,yyy,zzz} means that any combination of ESR[xxx], ESR[yyy], and ESR[zzz] *may* be set, including all or none

   xxx means ESR[xxx] *will* be set
5. Byte Ordering exception type Data Storage interrupts can only occur when the PPC440GP is connected to a floating-point unit or auxiliary processor, and then only when executing FP or AP load or store instructions. See *Data Storage Interrupt* on page 356 for more detailed information on these kinds of exceptions.
6. Byte Ordering exception type Instruction Storage interrupts are defined by the PowerPC Book-E architecture, but cannot occur within the PPC440GP. The core is capable of executing instructions from both big endian and little endian code pages.
7. Unimplemented Operation exception type Program interrupts can only occur when the PPC440GP is connected to a floating-point unit or auxiliary processor, and then only when executing instruction opcodes which are recognized by the floating-point unit or auxiliary processor but are not implemented within the hardware.
8. Floating-Point Unavailable and Auxiliary Processor Unavailable interrupts, as well as Floating-Point Enabled and Auxiliary Processor Enabled exception type Program interrupts, can only occur when the PPC440GP is connected to a floating-point unit or auxiliary processor, and then only when executing instruction opcodes which are recognized by the floating-point unit or auxiliary processor, respectively.

10.5.1 Critical Input Interrupt

A Critical Input interrupt occurs when no higher priority exception exists, a Critical Input exception is presented to the interrupt mechanism, and MSR[CE] = 1. A Critical Input exception is caused by the activation of an asynchronous input to the PPC440GP. Although the only mask for this interrupt type within the core is the MSR[CE] bit, system implementations typically provide an alternative means for independently masking the interrupt requests from the various devices which collectively may activate the processor core Critical Input interrupt request input.

Note: MSR[CE] also enables the Watchdog Timer interrupt.

When a Critical Input interrupt occurs, the interrupt processing registers are updated as indicated below (all registers not listed are unchanged), and instruction execution resumes at address IVPR[IVP] || IVOR0[IVO] || 0b0000.

Critical Save/Restore Register 0 (CSRR0)

Set to the effective address of the next instruction to be executed.

Critical Save/Restore Register 1 (CSRR1)

Set to the contents of the MSR at the time of the interrupt.

Machine State Register (MSR)

ME Unchanged.

All other MSR bits set to 0.

Programming Note: Software is responsible for taking any action(s) that are required by the implementation in order to clear any Critical Input exception status (such that the Critical Input interrupt request input signal is deasserted) before reenabling MSR[CE], in order to avoid another, redundant Critical Input interrupt.

10.5.2 Machine Check Interrupt

A Machine Check interrupt occurs when no higher priority exception exists, a Machine Check exception is presented to the interrupt mechanism, and MSR[ME] = 1. The processor core includes two types of Machine Check ~~exception~~exceptions. They are:

Instruction Machine Check exception

An Instruction Machine Check exception is caused when a timeout or read error is signaled on the instruction read PLB interface, during an instruction fetch operation. Such an exception is not presented to the interrupt handling mechanism, however, unless and until such time as the execution is attempted of an instruction at an address associated with the instruction fetch for which the Instruction Machine Check exception was asserted.

If MSR[ME] is 1 when the Instruction Machine Check exception is presented to the interrupt mechanism, then execution of the instruction associated with the exception will be suppressed, a Machine Check interrupt will occur, and the interrupt processing registers will be updated as described on Page 355. If MSR[ME] is 0, however, then the instruction associated with the exception will be processed as though the exception did not exist and a Machine Check interrupt will *not* occur (*ever*, even if and when MSR[ME] is subsequently set to 1), although the ESR will still be updated as described on Page 355.

Data Machine Check exception

A Data Machine Check exception is caused when one of the following occurs:

- a timeout, read error, or read interrupt request is signaled on the data read PLB interface, during a data read operation
- a timeout, write error, or write interrupt request is signaled on the data write PLB interface, during a data write operation
- a read interrupt request is signaled on the instruction read PLB interface. Note that this asynchonous mechanism for signaling an interrupt on the instruction PLB interface is handled as a Data Machine Check exception, and not as an Instruction Machine Check exception.

When a Data Machine Check exception occurs, it is immediately presented to the interrupt handling mechanism. A Machine Check interrupt will occur immediately if MSR[ME] is 1, and the interrupt processing registers will be updated as described below. If MSR[ME] is 0, however, then the exception will be "recorded" in a manner which is not visible to the programming model. If and when MSR[ME] is subsequently set to 1, a "delayed" Machine Check interrupt will then occur, clearing the recorded Data Machine Check exception.

When a Machine Check interrupt occurs, the interrupt processing registers are updated as indicated below (all registers not listed are unchanged), and instruction execution resumes at address IVPR[IVP] || IVOR1[IVO] || 0b0000.

•

When a Machine Check interrupt occurs, the interrupt processing registers are updated as indicated below (all registers not listed are unchanged), and instruction execution resumes at address IVPR[IVP] || IVOR1[IVO] || 0b0000.

Critical Save/Restore Register 0 (CSRR0)

For an Instruction Machine Check exception, set to the effective address of the instruction presenting the exception. For a Data Machine Check exception, set to the effective address of the next instruction to be executed.

Critical Save/Restore Register 1 (CSRR1)

Set to the contents of the MSR at the time of the interrupt.

Machine State Register (MSR)

All MSR bits set to 0.

Exception Syndrome Register (ESR)

MCI Set to 1 for an Instruction Machine Check exception; otherwise left unchanged.

All other defined ESR bits are set to 0 for an Instruction Machine Check exception; otherwise they are left unchanged.

Programming Note: If an ~~Instruction Machine~~ InstructionMachine Check exception is associated with an instruction, and execution of that instruction is attempted while MSR[ME] is 0, then no Machine Check interrupt will occur, but ESR[MCI] will still be set to 1 when the instruction actually

executes. Once set, ESR[MCI] cannot be cleared except by software, using the **mtspr** instruction. When processing a Machine Check interrupt handler, software should query ESR[MCI] to determine the type of Machine Check exception, and then clear ESR[MCI]. Then, prior to re-enabling Machine Check interrupts by setting MSR[ME] to 1, software should query the status of ESR[MCI] again to determine whether any additional Instruction Machine Check exceptions have occurrred while MSR[ME] was disabled.

~~See "Machine Check Interrupts" on page 10-3 for more information on the handling of Machine Check interrupts within the PPC440GP.~~

Programming Note: If an Instruction Machine Check exception occurs, any data associated with the exception will *not* be placed into the instruction cache. On the other hand, if a Data Machine Check exception occurs during a cacheable read operation, the data associated with the exception will be placed into the data cache, and could subsequently be loaded into a register. Similarly, if a Data Machine Check exception occurs during a caching inhibited read operation, the data associated with the exception will be read into a register.

Programming Note: Since a **dcbz** instruction establishes a real address in the data cache without actually reading the block of data from memory, it is possible for a delayed Data Machine Check exception to occur if and when a line established by a **dcbz** instruction is cast-out of the data cache and written to memory, if the address of the cache line is not valid within the system implementation.

10.5.3 Data Storage Interrupt

A Data Storage interrupt *may* occur when no higher priority exception exists and a Data Storage exception is presented to the interrupt mechanism. The PPC440GP includes four types of Data Storage exception. They are:

Read Access Control exception

A Read Access Control exception is caused by one of the following:

- While in user mode (MSR[PR] = 1), a load, **icbi**, **icbt**, **dcbst**, **dcbf**, **dcbt**, or **dcbtst** instruction attempts to access a location in storage that is not enabled for read access in user mode (that is, the TLB entry associated with the memory page being accessed has UR=0).
- While in supervisor mode (MSR[PR] = 0), a load, **icbi**, **icbt**, **dcbst**, **dcbf**, **dcbt**, or **dcbtst** instruction attempts to access a location in storage that is not enabled for read access in supervisor mode (that is, the TLB entry associated with the memory page being accessed has SR=0).

Programming Note: The instruction cache management instructions **icbi** and **icbt** are treated as "loads" from the addressed byte with respect to address translation and protection. These instruction cache management instructions use MSR[DS] rather than MSR[IS] to determine translation for their target effective address. Similarly, they use the read access control field (UR or SR) rather than the execute access control field (UX or SX) of the TLB entry to determine whether a Data Storage exception should occur. Instruction Storage exceptions and Instruction TLB Miss exceptions are

associated with the *fetching* of instructions not with the *execution* of instructions. Data Storage exceptions and Data TLB Miss exceptions are associated with the *execution* of instruction cache management instructions, as well as with the execution of load, store, and data cache management instructions.

Write Access Control exception

A Write Access Control exception is caused by one of the following:

- While in user mode (MSR[PR] = 1), a store, **dcbz**, or **dcbi** instruction attempts to access a location in storage that is not enabled for write access in user mode (that is, the TLB entry associated with the memory page being accessed has UW=0).
- While in supervisor mode (MSR[PR] = 0), a store, **dcbz**, or **dcbi** instruction attempts to access a location in storage that is not enabled for write access in supervisor mode (that is, the TLB entry associated with the memory page being accessed has SW=0).

Byte Ordering exception

A Byte Ordering exception will occur when a floating-point unit or auxiliary processor is attached to the PPC440GP, and a floating-point or auxiliary processor load or store instruction attempts to access a memory page with a byte order which is not supported by the attached processor. Whether or not a given load or store instruction type is supported for a given byte order is dependent on the implementation of the floating-point or auxiliary processor. All integer load and store instructions supported by the PPC440GP are supported for both big endian and little endian memory pages.

Cache Locking exception

A Cache Locking exception is caused by one of the following:

- While in user mode (MSR[PR] = 1) with MMUCR[IULXE]=1, execution of an **icbi** instruction is attempted. The exception occurs whether or not the cache line targeted by the **icbi** instruction is actually locked in the instruction cache.
- While in user mode (MSR[PR] = 1) with MMUCR[DULXE]=1, execution of a **dcbf** instruction is attempted. The exception occurs whether or not the cache line targeted by the **dcbf** instruction is actually locked in the data cache.

See ~~Chapter 5, "Instruction and Data Caches"~~*Instruction and Data Caches* on page 205 and ~~"Memory Management Unit Control Register (MMUCR)" on page 6-16~~*Memory Management Unit Control Register (MMUCR)* on page 247 for more information on cache locking and Cache Locking exceptions, respectively.

If a **stwcx.** instruction causes a Write Access Control exception, but the processor does not have the reservation from a **lwarx** instruction, then a Data Storage interrupt does not occur and the instruction completes, updating CR[CR0] to indicate the failure of the store due to the lost reservation.

If a Data Storage exception occurs on any of the following instructions, then the instruction is treated as a no-op, and a Data Storage interrupt does not occur.

- **lswx** or **stswx** with a length of zero (although the target register of **lswx** will still be undefined, as it is whether or not a Data Storage exception occurs)
- **icbt**
- **dcbt**

- **dcbtst**

For all other instructions, if a Data Storage exception occurs, then execution of the instruction causing the exception is suppressed, a Data Storage interrupt is generated, the interrupt processing registers are updated as indicated below (all registers not listed are unchanged), and instruction execution resumes at address IVPR[IVP] || IVOR2[IVO] || 0b0000.

Save/Restore Register 0 (SRR0)

Set to the effective address of the instruction causing the Data Storage interrupt.

Save/Restore Register 1 (SRR1)

Set to the contents of the MSR at the time of the interrupt.

Machine State Register (MSR)

CE, ME, DE Unchanged.

All other MSR bits set to 0.

Data Exception Address Register (DEAR)

If the instruction causing the Data Storage exception does so with respect to the memory page targeted by the initial effective address calculated by the instruction, then the DEAR is set to this calculated effective address. On the other hand, if the Data Storage exception only occurs due to the instruction causing the exception crossing a memory page boundary, in that the exception is with respect to the attributes of the page accessed after crossing the boundary, then the DEAR is set to the address of the first byte within that page.

For example, consider a misaligned load word instruction that targets effective address 0x00000FFF, and that the page containing that address is a 4KB page. The load word will thus cross the page boundary, and access the next page starting at address 0x00001000. If a Read Access Control exception exists within the first page (because the Read Access Control field for that page is 0), the DEAR will be set to 0x00000FFF. On the other hand, if the Read Access Control field of the first page is 1, but the same field is 0 for the next page, then the Read Access Control exception exists only for the second page and the DEAR will be set to 0x00001000. Furthermore, the load word instruction in this latter scenario will have been partially executed (see ~~"Partially Executed Instructions" on page 10-6~~*Partially Executed Instructions* on page 341).

Exception Syndrome Register (ESR)

FP Set to 1 if the instruction causing the interrupt is a floating-point load or store; otherwise set to 0.

ST Set to 1 if the instruction causing the interrupt is a store, **dcbz**, or **dcbi** instruction; otherwise set to 0.

$DLK_{0:1}$ Set to 0b10 if an **icbi** instruction caused a Cache Locking exception; set to 0b01 if a **dcbf** instruction caused a Cache Locking exception; otherwise set to 0b00. Note that a Read Access Control exception may occur in combination with a Cache Locking exception, in which case software would need to examine the TLB entry associated with the address reported in the DEAR to determine whether both exceptions had occurred, or just a Cache Locking exception.

AP Set to 1 if the instruction causing the interrupt is an auxiliary processor load or store; otherwise set to 0.

BO Set to 1 if the instruction caused a Byte Ordering exception; otherwise set to 0. Note that a Read or Write Access Control exception may occur in combination with a Byte Ordering exception, in which case software would need to examine the TLB entry associated with the address reported in the DEAR to determine whether both exceptions had occurred, or just a Byte Ordering exception.

MCI Unchanged.

All other defined ESR bits are set to 0.

10.5.4 Instruction Storage Interrupt

An Instruction Storage interrupt occurs when no higher priority exception exists and an Instruction Storage exception is presented to the interrupt mechanism. Note that although an Instruction Storage exception may occur during an attempt to *fetch* an instruction, such an exception is not actually presented to the interrupt mechanism until an attempt is made to *execute* that instruction. The PPC440GP includes one type of Instruction Storage exception. That is:

Execute Access Control exception

An Execute Access Control exception is caused by one of the following:

- While in user mode (MSRPR=1), an instruction fetch attempts to access a location in storage that is not enabled for execute access in user mode (that is, the TLB entry associated with the memory page being accessed has UX=0).
- While in supervisor mode (MSRPR=0), an instruction fetch attempts to access a location in storage that is not enabled for execute access in supervisor mode (that is, the TLB entry associated with the memory page being accessed has SX=0).

Architecture Note: The PowerPC Book-E architecture defines an additional Instruction Storage exception -- the Byte Ordering exception. This exception is defined to assist implementations that cannot support dynamically switching byte ordering between consecutive instruction fetches and/or cannot support a given byte order at all. The PPC440GP however supports instruction fetching from both big endian and little endian memory pages, so this exception cannot occur.

When an Instruction Storage interrupt occurs, the processor suppresses the execution of the instruction causing the Instruction Storage exception, the interrupt processing registers are updated as indicated below (all registers not listed are unchanged), and instruction execution resumes at address IVPR[IVP] || IVOR3[IVO] || 0b0000.

Save/Restore Register 0 (SRR0)

Set to the effective address of the instruction causing the Instruction Storage interrupt.

Save/Restore Register 1 (SRR1)

Set to the contents of the MSR at the time of the interrupt.

Machine State Register (MSR)

CE, ME, DE Unchanged.

All other MSR bits set to 0.

Exception Syndrome Register (ESR)

BO Set to 0.

MCI Unchanged.

All other defined ESR bits are set to 0.

10.5.5 External Input Interrupt

An External Input interrupt occurs when no higher priority exception exists, an External Input exception is presented to the interrupt mechanism, and MSR[EE] = 1. An External Input exception is caused by the activation of an asynchronous input to the PPC440GP. Although the only mask for this interrupt type within the core is the MSR[EE] bit, system implementations typically provide an alternative means for independently masking the interrupt requests from the various devices which collectively may activate the core's External Input interrupt request input.

Note: MSR[EE] also enables the External Input and Fixed-Interval Timer interrupts.

When an External Input interrupt occurs, the interrupt processing registers are updated as indicated below (all registers not listed are unchanged), and instruction execution resumes at address IVPR[IVP] || IVOR4[IVO] || 0b0000.

Save/Restore Register 0 (SRR0)

Set to the effective address of the next instruction to be executed.

Save/Restore Register 1 (SRR1)

Set to the contents of the MSR at the time of the interrupt.

Machine State Register (MSR)

CE, ME, DE Unchanged.

All other MSR bits set to 0.

Programming Note: Software is responsible for taking any action(s) that are required by the implementation in order to clear any External Input exception status (such that the External Input interrupt request input signal is deasserted) before reenabling MSR[EE], in order to avoid another, redundant External Input interrupt.

10.5.6 Alignment Interrupt

An Alignment interrupt occurs when no higher priority exception exists and an Alignment exception is presented to the interrupt mechanism. An Alignment exception occurs if execution of any of the following is attempted:

- An integer load or store instruction that references a data storage operand that is not aligned on an operand-sized boundary, when CCR0[FLSTA] is 1. Load and store multiple instructions are considered to ref-

erence word operands, and hence word-alignment is required for the target address of these instructions when CCR0[FLSTA] is 1. Load and store string instructions are considered to reference byte operands, and hence they cannot cause an Alignment exception due to CCR0[FLSTA] being 1, regardless of the target address alignment.

- A floating-point or auxiliary processor load or store instruction that references a data storage operand that crosses a quadword (16 byte) boundary.
- A floating-point or auxiliary processor load or store instruction that references a data storage operand that is not aligned on an operand-sized boundary, when the attached processing unit indicates to the PPC440GP that the instruction requires operand-alignment.
- A floating-point or auxiliary processor load or store instruction that references a data storage operand that is not aligned on a word boundary, when the attached processing unit indicates to the PPC440GP that the instruction requires word-alignment.
- A **dcbz** instruction that targets a memory page that is either write-through required or caching inhibited.

If a **stwcx.** instruction causes an Alignment exception, and the processor does not have the reservation from a **lwarx** instruction, then an Alignment interrupt still occurs.

Programming Note: The architecture does not support the use of an unaligned effective address by the **lwarx** and **stwcx.** instructions. If an Alignment interrupt occurs due to the attempted execution of one of these instructions, the Alignment interrupt handler must not attempt to emulate the instruction, but instead should treat the instruction as a programming error.

When an Alignment interrupt occurs, the processor suppresses the execution of the instruction causing the Alignment exception, the interrupt processing registers are updated as indicated below (all registers not listed are unchanged), and instruction execution resumes at address IVPR[IVP] || IVOR5[IVO] || 0b0000.

Save/Restore Register 0 (SRR0)

Set to the effective address of the instruction causing the Alignment interrupt.

Save/Restore Register 1 (SRR1)

Set to the contents of the MSR at the time of the interrupt.

Machine State Register (MSR)

CE, ME, DE Unchanged.

All other MSR bits set to 0.

Data Exception Address Register (DEAR)

Set to the effective address of the target data operand as calculated by the instruction causing the Alignment exception. Note that for **dcbz**, this effective address is not necessarily the address of the first byte of the targeted cache block, but could be the address of any byte within the block (it will be the address calculated by the **dcbz** instruction).

Exception Syndrome Register (ESR)

FP Set to 1 if the instruction causing the interrupt is a floating-point load or store; otherwise set to 0.

ST Set to 1 if the instruction causing the interrupt is a store, **dcbz**, or **dcbi** instruction;

otherwise set to 0.

AP Set to 1 if the instruction causing the interrupt is an auxiliary processor load or store; otherwise set to 0.

All other defined ESR bits are set to 0.

10.5.7 Program Interrupt

A Program interrupt occurs when no higher priority exception exists, a Program exception is presented to the interrupt mechanism, and -- for the Floating-Point Enabled form of Program exception only -- MSR[FE0,FE1] is non-zero. The PPC440GP includes six types of Program exception. They are:

Illegal Instruction exception

An Illegal Instruction exception occurs when execution is attempted of any of the following kinds of instructions:

- a reserved-illegal instruction
- when MSR[PR] = 1 (user mode), an **mtspr** or **mfspr** that specifies an SPRN value with $SPRN_5 = 0$ (user-mode accessible) that represents an unimplemented Special Purpose Register. For **mtspr**, this includes any SPR number other than the XER, LR, CTR, or USPRG0. For **mfspr**, this includes any SPR number other than the ones listed for **mtspr**, plus SPRG4-7, TBH, and TBL.
- a defined instruction which is not implemented within the PPC440GP, and which is not a floating-point instruction. This includes all instructions that are defined for 64-bit implementations only, as well as **tlbiva** and **mfapidi** (see the PowerPC Book-E specification)
- a defined floating-point instruction that is not recognized by an attached floating-point unit (or when no such floating-point unit is attached)
- an allocated instruction that is not implemented within the PPC440GP and which is not recognized by an attached auxiliary processor (or when no such auxiliary processor is attached)

See ~~"Instruction Classes" on page 4-35~~*Instruction Classes* on page 175 for more information on the PPC440GP's support for defined and allocated instructions.

Privileged Instruction exception

A Privileged Instruction exception occurs when MSR[PR] = 1 and execution is attempted of any of the following kinds of instructions:

- a privileged instruction
- an **mtspr** or **mfspr** instruction that specifies an SPRN value with $SPRN_5 = 1$ (a Privileged Instruction exception occurs regardless of whether or not the SPR referenced by the SPRN value is defined)

Trap exception

A Trap exception occurs when any of the conditions specified in a **tw** or **twi** instruction are met. However, if Trap debug events are enabled (DBCR0[TRAP]=1), internal debug mode is enabled (DBCR0[IDM]=1), and Debug interrupts are enabled (MSR[DE]=1), then a Trap exception will cause a Debug interrupt to occur, rather than a Program interrupt.

See ~~Chapter 15, "Debug Facilities"~~*Debug Facilities* on page 429 for more information on Trap debug events.

Unimplemented Operation exception

An Unimplemented Operation exception occurs when execution is attempted of any of the following kinds of instructions:

- a defined floating-point instruction that is recognized but not supported by an attached floating-point unit, when floating-point instruction processing is enabled (MSR[FP]=1).
- an allocated instruction that is not implemented within the PPC440GP, and is recognized but not supported by an attached auxiliary processor, when auxiliary processor instruction processing is enabled. The enabling of auxiliary processor instruction processing is implementation-dependent.

Floating-Point Enabled exception

A Floating-Point Enabled exception occurs when the execution or attempted execution of a defined floating-point instruction causes FPSCR[FEX] to be set to 1, in an attached floating-point unit. FPSCR[FEX] is the Floating-Point Status and Control Register Floating-Point Enabed Exception Summary bit (see the user's manual for the floating-point unit implementation for more details).

If MSR[FE0,FE1] is non-zero when the Floating-Point Enabled exception is presented to the interrupt mechanism, then a Program interrupt will occur, and the interrupt processing registers will be updated as described below. If MSR[FE0,FE1] are both 0, however, then a Program interrupt will *not* occur and the instruction associated with the exception will execute according to the definition of the floating-point unit (see the user's manual for the floating-point unit implementation). If and when MSR[FE0,FE1] are subsequently set to a non-zero value, and the Floating-Point Enabled exception is still being presented to the interrupt mechanism (that is, FPSCR[FEX] is still set), then a "delayed" Program interrupt will occur, updating the interrupt processing registers as described below.

See ~~"Synchronous, Imprecise Interrupts" on page 10-2~~*Synchronous, Imprecise Interrupts* on page 338 for more information on this special form of "delayed" Floating-Point Enabled exception.

Auxiliary Processor Enabled exception

An Auxiliary Processor Enabled exception may occur due to the execution or attempted execution of an allocated instruction that is not implemented within the PPC440GP, but is recognized and supported by an attached auxiliary processor. The cause of such an exception is implementation-dependent. See the user's manual for the auxiliary processor implementation for more details.

When a Program interrupt occurs, the processor suppresses the execution of the instruction causing the Program exception (for all cases except the "delayed" form of Floating-Point Enabled exception described above), the interrupt processing registers are updated as indicated below (all registers not listed are unchanged), and instruction execution resumes at address IVPR[IVP] || IVOR6[IVO] || 0b0000.

Save/Restore Register 0 (SRR0)

Set to the effective address of the instruction causing the Program interrupt, for all cases except the "delayed" form of Floating-Point Enabled exception described above.

For the special case of the delayed Floating-Point Enabled exception, where the exception was already being presented to the interrupt mechanism at the time MSR[FE0,FE1] was changed from 0 to a non-zero value, SRR0 is set to the address of the instruction that would have executed after the MSR-changing instruction. If the instruction which set MSR[FE0,FE1] was **rfi** or **rfci**, then CSRR0 is set to the address to which the **rfi** or **rfci** was returning, and not to the address of the instruction which was sequentially after the **rfi** or **rfci**.

Save/Restore Register 1 (SRR1)

Set to the contents of the MSR at the time of the interrupt.

Machine State Register (MSR)

CE, ME, DE Unchanged.

All other MSR bits set to 0.

Exception Syndrome Register (ESR)

PIL Set to 1 for an Illegal Instruction exception; otherwise set to 0

PPR Set to 1 for a Privileged Instruction exception; otherwise set to 0

PTR Set to 1 for a Trap exception; otherwise set to 0

PUO Set to 1 for an Unimplemented Operation exception; otherwise set to 0

FP Set to 1 if the instruction causing the interrupt is a floating-point instruction; otherwise set to 0.

AP Set to 1 if the instruction causing the interrupt is an auxiliary processor instruction; otherwise set to 0.

PIE Set to 1 if a "delayed" form of Floating-point Enabled exception type Program interrupt; otherwise set to 0. The setting of ESR[PIE] to 1 indicates to the Program interrupt handler that the interrupt was imprecise due to being caused by the changing of MSR[FE0,FE1] and not directly by the execution of the floating-point instruction which caused the exception by setting FPSCR[FEX]. Thus the Program interrupt handler can recognize that SRR0 contains the address of the instruction after the MSR-changing instruction, and not the address of the instruction that caused the Floating-Point Enabled exception.

PCRE Set to 1 if a Floating-Point Enabled exception and the floating-point instruction which caused the exception was a CR-updating instruction. Note that ESR[PCRE] is undefined if ESR[PIE] is 1.

PCMP Set to 1 if a Floating-Point Enabled exception and the instruction which caused the exception was a floating-point compare instruction. Note that ESR[PCMP] is undefined if ESR[PIE] is 1.

PCRF Set to the number of the CR field (0 - 7) which was to be updated, if a Floating-Point Enabled exception and the floating-point instruction which caused the exception was a CR-updating instruction. Note that ESR[PCRF] is undefined if ESR[PIE] is 1.

All other defined ESR bits are set to 0.

Programming Note: The ESR[PCRE,PCMP,PCRF] fields are provided to assist the Program interrupt handler with the emulation of part of the function of the various floating-point CR-updating instructions, when any of these instructions cause a precise ("non-delayed") Floating-Point Enabled exception type Program interrupt. The PowerPC Book-E floating-point architecture defines that when such exceptions occur, the CR is to be updated even though the rest of the instruction execution may be suppressed. The PPC440GP, however, does not support such CR updates when the instruction which is supposed to cause the update is being suppressed due to the occurrence of a synchronous, precise interrupt. Instead, the

PPC440GP records in the ESR[PCRE,PCMP,PCRF] fields information about the instruction causing the interrupt, to assist the Program interrupt handler software in performing the appropriate CR update manually.

10.5.8 Floating-Point Unavailable Interrupt

A Floating-Point Unavailable interrupt occurs when no higher priority exception exists, an attempt is made to execute a floating-point instruction which is recognized by an attached floating-point unit, and MSR[FP]=0.

When a Floating-Point Unavailable interrupt occurs, the processor suppresses the execution of the instruction causing the Floating-Point Unavailable exception, the interrupt processing registers are updated as indicated below (all registers not listed are unchanged), and instruction execution resumes at address IVPR[IVP] || IVOR7[IVO] || 0b0000.

Save/Restore Register 0 (SRR0)

Set to the effective address of the next instruction causing the Floating-Point Unavailable interrupt.

Save/Restore Register 1 (SRR1)

Set to the contents of the MSR at the time of the interrupt.

Machine State Register (MSR)

CE, ME, DE Unchanged.

All other MSR bits set to 0.

10.5.9 System Call Interrupt

A System Call interrupt occurs when no higher priority exception exists and a system call (**sc**) instruction is executed.

When a System Call interrupt occurs, the interrupt processing registers are updated as indicated below (all registers not listed are unchanged), and instruction execution resumes at address IVPR[IVP] || IVOR8[IVO] || 0b0000.

Save/Restore Register 0 (SRR0)

Set to the effective address of the instruction after the system call instruction.

Save/Restore Register 1 (SRR1)

Set to the contents of the MSR at the time of the interrupt.

Machine State Register (MSR)

CE, ME, DE Unchanged.

All other MSR bits set to 0.

10.5.10 Auxiliary Processor Unavailable Interrupt

An Auxiliary Processor Unavailable interrupt occurs when no higher priority exception exists, an attempt is made to execute an auxiliary processor instruction which is not implemented within the PPC440GP but which is recognized by an attached auxiliary processor, and auxiliary processor instruction processing is not enabled. The enabling of auxiliary processor instruction processing is implementation-dependent. See the user's manual for the attached auxiliary processor.

When an Auxiliary Processor Unavailable interrupt occurs, the processor suppresses the execution of the instruction causing the Auxiliary Processor Unavailable exception, the interrupt processing registers are updated as indicated below (all registers not listed are unchanged), and instruction execution resumes at address IVPR[IVP] || IVOR9[IVO] || 0b0000.

Save/Restore Register 0 (SRR0)

Set to the effective address of the next instruction causing the Auxiliary Processor Unavailable interrupt.

Save/Restore Register 1 (SRR1)

Set to the contents of the MSR at the time of the interrupt.

Machine State Register (MSR)

CE, ME, DE Unchanged.

All other MSR bits set to 0.

10.5.11 Decrementer Interrupt

A Decrementer interrupt occurs when no higher priority exception exists, a Decrementer exception exists (TSR[DIS] = 1), and the interrupt is enabled (TCR[DIE] = 1 and MSR[EE] = 1). See ~~Chapter 11, "Timer Facilities"~~ *Timer Facilities* on page 383 for more information on Decrementer exceptions.

Note: MSR[EE] also enables the External Input and Fixed-Interval Timer interrupts.

When a Decrementer interrupt occurs, the interrupt processing registers are updated as indicated below (all registers not listed are unchanged), and instruction execution resumes at address IVPR[IVP] || IVOR10[IVO] || 0b0000.

Save/Restore Register 0 (SRR0)

Set to the effective address of the next instruction to be executed.

Save/Restore Register 1 (SRR1)

Set to the contents of the MSR at the time of the interrupt.

Machine State Register (MSR)

CE, ME, DE Unchanged.

All other MSR bits set to 0.

Programming Note: Software is responsible for clearing the Decrementer exception status by writing to TSR[DIS], prior to reenabling MSR[EE], in order to avoid another, redundant Decrementer interrupt.

10.5.12 Fixed-Interval Timer Interrupt

A Fixed-Interval Timer interrupt occurs when no higher priority exception exists, a Fixed-Interval Timer exception exists (TSR[FIS] = 1), and the interrupt is enabled (TCR[FIE] = 1 and MSR[EE]=1). See ~~Chapter 11, "Timer Facilities"~~ *Timer Facilities* on page 383 for more information on Fixed Interval Timer exceptions.

Note: MSR[EE] also enables the External Input and Decrementer interrupts.

When a Fixed interval Timer interrupt occurs, the interrupt processing registers are updated as indicated below (all registers not listed are unchanged), and instruction execution resumes at address IVPR[IVP] || IVOR11[IVO] || 0b0000.

Save/Restore Register 0 (SRR0)

Set to the effective address of the next instruction to be executed.

Save/Restore Register 1 (SRR1)

Set to the contents of the MSR at the time of the interrupt.

Machine State Register (MSR)

CE, ME, DE Unchanged.

All other MSR bits set to 0.

Programming Note: Software is responsible for clearing the Fixed Interval Timer exception status by writing to TSR[FIS], prior to reenabling MSR[EE], in order to avoid another, redundant Fixed Interval Timer interrupt.

10.5.13 Watchdog Timer Interrupt

A Watchdog Timer interrupt occurs when no higher priority exception exists, a Watchdog Timer exception exists (TSR[WIS] = 1), and the interrupt is enabled (TCR[WIE] = 1 and MSR[CE] = 1). See ~~Chapter 11, "Timer Facilities"~~ *Timer Facilities* on page 383 for more information on Watchdog Timer exceptions.

Note: MSR[CE] also enables the Critical Input interrupt.

When a Watchdog Timer interrupt occurs, the interrupt processing registers are updated as indicated below (all registers not listed are unchanged), and instruction execution resumes at address IVPR[IVP] || IVOR12[IVO] || 0b0000.

Critical Save/Restore Register 0 (CSRR0)

Set to the effective address of the next instruction to be executed.

Critical Save/Restore Register 1 (CSRR1)

Set to the contents of the MSR at the time of the interrupt.

Machine State Register (MSR)

ME Unchanged.

All other MSR bits set to 0.

Programming Note: Software is responsible for clearing the Watchdog Timer exception status by writing to TSR[WIS], prior to reenabling MSR[CE], in order to avoid another, redundant Watchdog Timer interrupt.

10.5.14 Data TLB Error Interrupt

A Data TLB Error interrupt *may* occur when no higher priority exception exists and a Data TLB Miss exception is presented to the interrupt mechanism. A Data TLB Miss exception occurs when a load, store, **icbi**, **icbt**, **dcbst**, **dcbf**, **dcbz**, **dcbi**, **dcbt**, or **dcbtst** instruction attempts to access a virtual address for which a valid TLB entry does not exist. See ~~Chapter 6, "Memory Management"~~ *Memory Management* on page 233 for more information on the TLB.

Programming Note: The instruction cache management instructions **icbi** and **icbt** are treated as "loads" from the addressed byte with respect to address translation and protection, and therefore use MSR[DS] rather than MSR[IS] as part of the calculated virtual address when searching the TLB to determine translation for their target storage address. Instruction TLB Miss exceptions are associated with the *fetching* of instructions not with the *execution* of instructions. Data TLB Miss exceptions are associated with the *execution* of instruction cache management instructions, as well as with the execution of load, store, and data cache management instructions.

If a **stwcx.** instruction causes a Data TLB Miss exception, and the processor does not have the reservation from a **lwarx** instruction, then a Data TLB Error interrupt still occurs.

If a Data TLB Miss exception occurs on any of the following instructions, then the instruction is treated as a no-op, and a Data TLB Error interrupt does not occur.

- **lswx** or **stswx** with a length of zero (although the target register of **lswx** will be undefined)
- **icbt**
- **dcbt**
- **dcbtst**

For all other instructions, if a Data TLB Miss exception occurs, then execution of the instruction causing the exception is suppressed, a Data TLB Error interrupt is generated, the interrupt processing registers are updated as indicated below (all registers not listed are unchanged), and instruction execution resumes at address IVPR[IVP] || IVOR13[IVO] || 0b0000.

Save/Restore Register 0 (SRR0)

Set to the effective address of the instruction causing the Data TLB Error interrupt.

Save/Restore Register 1 (SRR1)

Set to the contents of the MSR at the time of the interrupt.

Machine State Register (MSR)

CE, ME, DE Unchanged.

All other MSR bits set to 0.

Data Exception Address Register (DEAR)

If the instruction causing the Data TLB Miss exception does so with respect to the memory page targeted by the initial effective address calculated by the instruction, then the DEAR is set to this calculated effective address. On the other hand, if the Data TLB Miss exception only occurs due to the instruction causing the exception crossing a memory page boundary, in that the missing TLB entry is for the page accessed after crossing the boundary, then the DEAR is set to the address of the first byte within that page.

As an example, consider a misaligned load word instruction that targets effective address 0x00000FFF, and that the page containing that address is a 4KB page. The load word will thus cross the page boundary, and attempt to access the next page starting at address 0x00001000. If a valid TLB entry does not exist for the first page, then the DEAR will be set to 0x00000FFF. On the other hand, if a valid TLB entry *does* exist for the first page, but not for the second, then the DEAR will be set to 0x00001000. Furthermore, the load word instruction in this latter scenario will have been partially executed (see ~~"Partially Executed Instructions" on page 10-6~~*Partially Executed Instructions* on page 341).

Exception Syndrome Register (ESR)

FP Set to 1 if the instruction causing the interrupt is a floating-point load or store; otherwise set to 0.

ST Set to 1 if the instruction causing the interrupt is a store, **dcbz**, or **dcbi** instruction; otherwise set to 0.

AP Set to 1 if the instruction causing the interrupt is an auxiliary processor load or store; otherwise set to 0.

MCI Unchanged.

All other defined ESR bits are set to 0.

10.5.15 Instruction TLB Error Interrupt

An Instruction TLB Error interrupt occurs when no higher priority exception exists and an Instruction TLB Miss exception is presented to the interrupt mechanism. Note that although an Instruction TLB Miss exception may occur during an attempt to *fetch* an instruction, such an exception is not actually presented to the interrupt mechanism until an attempt is made to *execute* that instruction. An Instruction TLB Miss exception occurs when an instruction fetch attempts to access a virtual address for which a valid TLB entry does not exist. See ~~Chapter 6, "Memory Management"~~*Memory Management* on page 233 for more information on the TLB.

When an Instruction TLB Error interrupt occurs, the processor suppresses the execution of the instruction causing the Instruction TLB Miss exception, the interrupt processing registers are updated as indicated below (all registers not listed are unchanged), and instruction execution resumes at address IVPR[IVP] || IVOR14[IVO] || 0b0000.

Save/Restore Register 0 (SRR0)

Set to the effective address of the instruction causing the Instruction TLB Error interrupt.

Save/Restore Register 1 (SRR1)

Set to the contents of the MSR at the time of the interrupt.

Machine State Register (MSR)

CE, ME, DE Unchanged.

All other MSR bits set to 0.

10.5.16 Debug Interrupt

A Debug interrupt occurs when no higher priority exception exists, a Debug exception exists in the Debug Status Register (DBSR), the processor is in internal debug mode (DBCR0[IDM]=1), and Debug interrupts are enabled (MSR[DE] = 1). A Debug exception occurs when a debug event causes a corresponding bit in the DBSR to be set.

There are several types of Debug exception, as follows:

Instruction Address Compare (IAC) exception

An IAC Debug exception occurs when execution is attempted of an instruction whose address matches the IAC conditions specified by the various debug facility registers. This exception can occur regardless of debug mode, and regardless of the value of MSR[DE].

Data Address Compare (DAC) exception

A DAC Debug exception occurs when the DVC mechanism is not enabled, and execution is attempted of a load, store, **icbi**, **icbt**, **dcbst**, **dcbf**, **dcbz**, **dcbi**, **dcbt**, or **dcbtst** instruction whose target storage operand address matches the DAC conditions specified by the various debug facility registers. This exception can occur regardless of debug mode, and regardless of the value of MSR[DE].

Programming Note: The instruction cache management instructions **icbi** and **icbt** are treated as "loads" from the addressed byte with respect to Debug exceptions. IAC Debug exceptions are associated with the *fetching* of instructions not with the *execution* of instructions. DAC Debug exceptions are associated with the *execution* of instruction cache management instructions, as well as with the execution of load, store, and data cache management instructions.

Data Value Compare (DVC) exception

A DVC Debug exception occurs when execution is attempted of a load, store, or **dcbz** instruction whose target storage operand address matches the DAC and DVC conditions specified by the various debug facility registers. This exception can occur regardless of debug mode, and regardless of the value of MSR[DE].

Branch Taken (BRT) exception

A BRT Debug exception occurs when BRT debug events are enabled (DBCR0[BRT]=1) and execution is attempted of a branch instruction for which the branch conditions are met. This exception cannot occur in internal debug mode when MSR[DE]=0, unless external debug mode or debug wait mode is also enabled.

Trap (TRAP) exception

A TRAP Debug exception occurs when TRAP debug events are enabled (DBCR0[TRAP]=1) and execution is attempted of a **tw** or **twi** instruction that matches any of the specified trap conditions. This exception can occur regardless of debug mode, and regardless of the value of MSR[DE].

Return (RET) exception

A RET Debug exception occurs when RET debug events are enabled (DBCR0[RET]=1) and execution is attempted of an **rfi** or **rfci** instruction. For **rfi**, the RET Debug exception can occur regardless of debug mode and regardless of the value of MSR[DE]. For **rfci**, the RET Debug exception cannot occur in internal debug mode when MSR[DE]=0, unless external debug mode or debug wait mode is also enabled.

Instruction Complete (ICMP) exception

An ICMP Debug exception occurs when ICMP debug events are enabled (DBCR0[ICMP]=1) and execution of any instruction is completed. This exception cannot occur in internal debug mode when MSR[DE]=0, unless external debug mode or debug wait mode is also enabled.

Interrupt (IRPT) exception

An IRPT Debug exception occurs when IRPT debug events are enabled (DBCR0[IRPT]=1) and an interrupt occurs. For non-critical class interrupt types, the IRPT Debug exception can occur regardless of debug mode and regardless of the value of MSR[DE]. For critical class interrupt types, the IRPT Debug exception cannot occur in internal debug mode (regardless of the value of MSR[DE]), unless external debug mode or debug wait mode is also enabled.

Unconditional Debug Event (UDE) exception

A UDE Debug exception occurs when an Unconditional Debug Event is signaled over the JTAG interface to the PPC440GP. This exception can occur regardless of debug mode, and regardless of the value of MSR[DE].

There are four debug modes supported by the PPC440GP. They are: internal debug mode, external debug mode, debug wait mode, and trace mode. Debug exceptions and interrupts are affected by the debug mode(s) which are enabled at the time of the Debug exception. Debug interrupts occur only when internal debug mode is enabled, although it is possible for external debug mode and/or debug wait mode to be enabled as well. The remainder of this section assumes that internal debug mode is enabled and that external debug mode and debug wait mode are not enabled, at the time of a Debug exception.

See ~~Chapter 15, "Debug Facilities"~~ *Debug Facilities* on page 429 for more information on the different debug modes and the behavior of each of the Debug exception types when operating in each of the modes.

Programming Note: It is a programming error for software to enable internal debug mode (by setting DBCR0[IDM] to 1) while Debug exceptions are already present in the DBSR. Software must first clear all DBSR Debug exception status (that is, all fields except IDE, MRR, IAC12ATS, and IAC34ATS) before setting DBCR0[IDM] to 1.

If a **stwcx.** instruction causes a DAC or DVC Debug exception, but the processor does not have the reservation from a **lwarx** instruction, then the Debug exception is not recorded in the DBSR and a Debug interrupt does not occur. Instead, the instruction completes and updates CR[CR0] to indicate the failure of the store due to the lost reservation.

If a DAC exception occurs on an **lswx** or **stswx** with a length of zero, then the instruction is treated as a no-op, the Debug exception is not recorded in the DBSR, and a Debug interrupt does not occur.

If a DAC exception occurs on an **icbt**, **dcbt**, or **dcbtst** instruction which is being no-op'ed due to some other reason (either the referenced cache block is in a caching inhibited memory page, or a Data Storage or Data TLB Miss exception occurs), then the Debug exception is not recorded in the DBSR and a Debug interrupt does not occur. On the other hand, if the **icbt**, **dcbt**, or **dcbtst** instruction is not being no-op'ed for one of these other reasons, the DAC Debug exception does occur and is handled in the same fashion as other DAC Debug exceptions (see below).

For all other cases, when a Debug exception occurs, it is immediately presented to the interrupt handling mechanism. A Debug interrupt will occur immediately if MSR[DE] is 1, and the interrupt processing registers will be updated as described below. If MSR[DE] is 0, however, then the exception condition remains set in the DBSR. If and when MSR[DE] is subsequently set to 1, and the exception condition is still present in the DBSR, a "delayed" Debug interrupt will then occur either as a synchronous, imprecise interrupt, or as an asynchronous interrupt, depending on the type of Debug exception.

When a Debug interrupt occurs, the interrupt processing registers are updated as indicated below (all registers not listed are unchanged) and instruction execution resumes at address IVPR[IVP] || IVOR15[IVO] || 0b0000.

Critical Save/Restore Register 0 (CSRR0)

For Debug exceptions that occur while Debug interrupts are enabled (MSR[DE] = 1), CSRR0 is set as follows:

- For IAC, BRT, TRAP, and RET Debug exceptions, set to the address of the instruction causing the Debug interrupt. Execution of the instruction causing the Debug exception is suppressed, and the interrupt is synchronous and precise.
- For DAC and DVC Debug exceptions, if DBCR2[DAC12A] is 0, set to the address of the instruction causing the Debug interrupt. Execution of the instruction causing the Debug exception is suppressed, and the interrupt is synchronous and precise.

  If DBCR2[DAC12A] is 1, however, then DAC and DVC Debug exceptions are handled asynchronously, and CSRR0 is set to the address of the instruction that would have executed next had the Debug interrupt not occurred. This could either be the address of the instruction causing the DAC or DVC Debug exception, or the address of a subsequent instruction.
- For ICMP Debug exceptions, set to the address of the next instruction to be executed (the instruction after the one whose completion caused the ICMP Debug exception). The interrupt is synchronous and precise.

  Since the ICMP Debug exception does not suppress the execution of the instruction causing the exception, but rather allows it to complete before causing the interrupt, the behavior of the interrupt is different in the special case where the instruction causing the ICMP Debug exception is itself setting MSR[DE] to 0. In this case, the interrupt will be delayed and will occur if and when MSR[DE] is again set to 1, assuming DBSR[ICMP] is still set. If the Debug interrupt occurs in this fashion, it will be synchronous and imprecise, and CSRR0 will be set to the address of the instruction after the one which set MSR[DE] to 1 (not the one which originally caused the ICMP Debug exception and in so doing set MSR[DE] to 0). If the instruction which set MSR[DE] to 1 was **rfi** or **rfci**, then CSRR0 is set to the address to which the **rfi** or **rfci** was returning, and not to the address of the instruction which was sequentially after the **rfi** or **rfci**.

- For IRPT Debug exceptions, set to the address of the first instruction in the interrupt handler associated with the interrupt type that caused the IRPT Debug exception. The interrupt is asynchronous.
- For UDE Debug exceptions, set to the address of the instruction that would have executed next if the Debug interrupt had not occurred. The interrupt is asynchronous.

For all Debug exceptions that occur while Debug interrupts are disabled (MSR[DE] = 0), the Debug interrupt will be delayed and will occur if and when MSR[DE] is again set to 1, assuming the Debug exception status is still set in the DBSR. If the Debug interrupt occurs in this fashion, CSRR0 is set to the address of the instruction after the one which set MSR[DE]. If the instruction which set MSR[DE] was **rfi** or **rfci**, then CSRR0 is set to the address to which the **rfi** or **rfci** was returning, and not to the address of the instruction which was sequentially after the **rfi** or **rfci**. The interrupt is either synchronous and imprecise, or asynchronous, depending on the type of Debug exception, as follows:

- For IAC and RET Debug exceptions, the interrupt is synchronous and imprecise.
- For BRT Debug exceptions, this scenario cannot occur. BRT Debug exceptions are not recognized when MSR[DE]=0 if operating in internal debug mode.
- For TRAP Debug exceptions, the Debug interrupt is synchronous and imprecise. However, under these conditions (TRAP Debug exception occurring while MSR[DE] is 0), the attempted execution of the trap instruction for which one or more of the trap conditions is met will itself lead to a Trap exception type Program interrupt. The corresponding Debug interrupt which will occur later if and when Debug interrupts are enabled will be *in addition* to the Program interrupt.
- For DAC and DVC Debug exceptions, if DBCR2[DAC12A] is 0, then the interrupt is synchronous and imprecise. If DBCR2[DAC12A] is 1, then the interrupt is asynchronous.
- For ICMP Debug exceptions, this scenario cannot occur in this fashion. ICMP Debug exceptions are not recognized when MSR[DE]=0 if operating in internal debug mode. However, a similar scenario can occur when MSR[DE] is 1 at the time of the ICMP Debug exception, but the instruction whose completion is causing the exception is itself setting MSR[DE] to 0. This scenario is described above in the subsection on the ICMP Debug exception for which MSR[DE] is 1 at the time of the exception. In that scenario, the interrupt is synchronous and imprecise.
- For IRPT and UDE Debug exceptions, the interrupt is asynchronous.

Critical Save/Restore Register 1 (CSRR1)

Set to the contents of the MSR at the time of the interrupt.

Machine State Register (MSR)

ME Unchanged.

All other MSR bits set to 0.

10.6 Interrupt Ordering and Masking

It is possible for multiple exceptions to exist simultaneously, each of which could cause the generation of an interrupt. Furthermore, the PowerPC Book-E architecture does not provide for the generation of more than one interrupt of the same class (critical or non-critical) at a time. Therefore, the architecture defines that interrupts are ordered with respect to each other, and provides a masking mechanism for certain persistent interrupt types.

When an interrupt type is masked (disabled), and an event causes an exception that would normally generate an interrupt of that type, the exception *persists* as a *status* bit in a register (which register depends upon the exception type). However, no interrupt is generated. Later, if the interrupt type is enabled (unmasked), and the exception status has not been cleared by software, the interrupt due to the original exception event will then finally be generated.

All asynchronous interrupt types can be masked. In addition, certain synchronous interrupt types can be masked. The two synchronous interrupt types which can be masked are the Floating-Point Enabled exception type Program interrupt (masked by MSR[FE0,FE1), and the IAC, DAC, DVC, RET, and ICMP exception type Debug interrupts (masked by MSR[DE]).

Architecture Note: When an otherwise synchronous, *precise* interrupt type is "delayed" in this fashion via masking, and the interrupt type is later enabled, the interrupt that is then generated due to the exception event that occurred while the interrupt type was disabled is then considered a synchronous, *imprecise* class of interrupt.

In order to prevent a subsequent interrupt from causing the state information (saved in SRR0/SRR1 or CSRR0/CSRR1) from a previous interrupt to be overwritten and lost, the PPC440GP performs certain functions. As a first step, upon any non-critical class interrupt, the processor automatically disables any further asynchronous, non-critical class interrupts (External Input, Decrementer, and Fixed Interval Timer) by clearing MSR[EE]. Likewise, upon any critical class interrupt, hardware automatically disables any further asynchronous interrupts of either class (critical and non-critical) by clearing MSR[CE] and MSR[DE], in addition to MSR[EE]. The additional interrupt types that are disabled by the clearing of MSR[CE,DE] are the Critical Input, Watchdog Timer, and Debug interrupts.

This first step of clearing MSR[EE] (and MSR[CE,DE] for critical class interrupts) prevents any subsequent asynchronous interrupts from overwriting the relevant save/restore registers (SRR0/SRR1 or CSRR0/CSRR1), prior to software being able to save their contents. The processor also automatically clears, on any interrupt, MSR[WE,PR,FP,FE0,FE1,IS,DS]. The clearing of these bits assists in the avoidance of subsequent interrupts of certain other types. However, *guaranteeing* that these interrupt types do not occur and thus do not overwrite the save/restore registers also requires the cooperation of system software. Specifically, system software must avoid the execution of instructions that could cause (or enable) a subsequent interrupt, if the contents of the save/restore registers have not yet been saved.

10.6.1 Interrupt Ordering Software Requirements

The following list identifies the actions that system software must *avoid*, prior to having saved the save/restore registers' contents:

- Reenabling of MSR[EE] (or MSR[CE,DE] in critical class interrupt handlers)

  This prevents any asynchronous interrupts, as well as (in the case of MSR[DE]) any Debug interrupts (which include both synchronous and asynchronous types).

- Branching (or sequential execution) to addresses not mapped by the TLB, or mapped without execute access permission

  This prevents Instruction Storage and Instruction TLB Error interrupts.

- Load, store, or cache management instructions to addresses not mapped by the TLB or not having the necessary access permission (read or write)

  This prevents Data Storage and Data TLB Error interrupts.

- Execution of system call (**sc**) or trap (**tw**, **twi**) instructions

  This prevents System Call and Trap exception type Program interrupts.

- Execution of any floating-point instructions

  This prevents Floating-Point Unavailable interrupts. Note that this interrupt would occur upon the execution of any floating-point instruction, due to the automatic clearing of MSR[FP]. However, even if software were to re-enable MSR[FP], floating-point instructions must still be avoided in order to prevent Program interrupts due to the possibility of Floating-Point Enabled and/or Unimplemented Operation exceptions.

- Reenabling of MSR[PR]

  This prevents Privileged Instruction exception type Program interrupts. Alternatively, software could re-enable MSR[PR], but avoid the execution of any privileged instructions.

- Execution of any Auxiliary Processor instructions that are not implemented in the PPC440GP

  This prevents Auxiliary Processor Unavailable interrupts, as well as Auxiliary Processor Enabled and Unimplemented Operation exception type Program interrupts. Note that the auxiliary processor instructions that are implemented within the PPC440GP do not cause any of these types of exceptions, and can therefore be executed prior to software having saved the save/restore registers' contents.

- Execution of any illegal instructions, or any defined instructions not implemented within the PPC440GP (64-bit instructions, **tlbiva**, **mfapidi**)

  This prevents Illegal Instruction exception type Program interrupts.

- Execution of any instruction that could cause an Alignment interrupt

  This prevents Alignment interrupts. See ~~"Alignment Interrupt" on page 10-25~~*Alignment Interrupt* on page 360 for a complete list of instructions that may cause Alignment interrupts.

Machine Check interrupts are a special case. Machine Checks are critical class interrupts, but *normal* critical class interrupts (Critical Input, Watchdog Timer and Debug) do not automatically disable Machine Check interrupts. Instead, Machine Check interrupts are disabled by clearing MSR[ME], which is only cleared by a Machine Check interrupt itself. Thus there is always the risk that a Machine Check interrupt could occur within a *normal*, critical interrupt handler, prior to the contents of CSRR0 and CSRR1 having been saved. In such a case, the interrupt may not be recoverable.

It is not necessary for hardware or software to avoid critical class interrupts from within non-critical class interrupt handlers (and hence the processor does not automatically clear MSR[CE,ME,DE] upon a non-critical interrupt), since the two classes of interrupts use different pairs of save/restore registers to save the instruction address and MSR. The converse, however, is not true. That is, hardware and software must cooperate in the avoidance of both critical *and* non-critical class interrupts from within critical class interrupt handlers, even though the two classes of interrupts use different save/restore register pairs. This is because the critical class interrupt may have occurred from within a non-critical class interrupt handler, prior to the non-critical class interrupt handler having saved SRR0 and SRR1. Therefore, within the critical class interrupt handler, both pairs of save/restore registers may contain data that is necessary to the system software.

10.6.2 Interrupt Order

The following is a prioritized listing of the various enabled interrupt types for which exceptions might exist simultaneously:

1. Synchronous (non-debug) interrupts:

1. Data Storage
2. Instruction Storage
3. Alignment
4. Program
5. Floating-Point Unavailable
6. System Call
7. Auxiliary Processor Unavailable
8. Data TLB Error
9. Instruction TLB Error

   Only one of the above types of synchronous interrupts may have an existing exception generating it at any given time. This is guaranteed by the exception priority mechanism (see ~~"Exception Priorities" on page 10-42~~*Exception Priorities* on page 376) and the requirements of the sequential execution model defined by the PowerPC Book-E architecture.

2. Machine Check
3. Debug
4. Critical Input
5. Watchdog Timer
6. External Input
7. Fixed-Interval Timer
8. Decrementer

Even though, as indicated above, the non-critical, synchronous exception types listed under item 1 are generated with higher priority than the critical interrupt types listed in items 2-5, the fact is that these non-critical interrupts will immediately be followed by the highest priority existing critical interrupt type, without executing any instructions at the non-critical interrupt handler. This is because the non-critical interrupt types do not automatically clear MSR[ME,DE,CE] and hence do not automatically disable the critical interrupt types. In all other cases, a particular interrupt type from the above list will automatically disable any subsequent interrupts of the same type, as well as all other interrupt types that are listed below it in the priority order.

10.7 Exception Priorities

PowerPC Book-E requires all synchronous (precise and imprecise) interrupts to be reported in program order, as implied by the sequential execution model. The one exception to this rule is the case of multiple synchronous imprecise interrupts. Upon a synchronizing event, all previously executed instructions are required to report any synchronous imprecise interrupt-generating exceptions, and the interrupt(s) will then be generated according to the general interrupt ordering rules outlined in ~~"Interrupt Order" on page 10-41~~*Interrupt Order* on page 375. For example, if a **mtmsr** instruction causes MSR[FE0,FE1,DE] to all be set, it is possible that a previous Floating-Point Enabled exception and a previous Debug exception both are still being presented (in the FPSCR and DBSR, respectively). In such a scenario, a Floating-Point Enabled exception type Program interrupt will occur first, followed immediately by a Debug interrupt.

For any single instruction attempting to cause multiple exceptions for which the corresponding synchronous interrupt types are enabled, this section defines the priority order by which the instruction will be permitted to cause a *single* enabled exception, thus generating a particular synchronous interrupt. Note that it is this exception priority mechanism, along with the requirement that synchronous interrupts be generated in program order, that guarantees that at any given time there exists for consideration only one of the synchronous interrupt types listed in item 1 of ~~"Interrupt Order" on page 10-41~~*Interrupt Order* on page 375. The exception priority mechanism also prevents certain debug exceptions from existing in combination with certain other synchronous interrupt-generating exceptions.

This section does not define the permitted setting of multiple exceptions for which the corresponding interrupt types are disabled. The generation of exceptions for which the corresponding interrupt types are disabled will have no effect on the generation of other exceptions for which the corresponding interrupt types are enabled. Conversely, if a particular exception for which the corresponding interrupt type is enabled is shown in the following sections to be of a higher priority than another exception, the occurrence of that enabled higher priority exception will prevent the setting of the other exception, independent of whether that other exception's corresponding interrupt type is enabled or disabled.

Except as specifically noted below, only one of the exception types listed for a given instruction type will be permitted to be generated at any given time, assuming the corresponding interrupt type is enabled. The priority of the exception types are listed in the following sections ranging from highest to lowest, within each instruction type.

10.7.1 Exception Priorities for Integer Load, Store, and Cache Management Instructions

The following list identifies the priority order of the exception types that may occur within the PPC440GP as the result of the attempted execution of any integer load, store, or cache management instruction. Included in this category is the former opcode for the **icbt** instruction, which is an allocated opcode still supported by the PPC440GP.

1. Debug (IAC exception)
2. Instruction TLB Error (Instruction TLB Miss exception)
3. Instruction Storage (Execute Access Control exception)
4. Program (Illegal Instruction exception)

   Only applies to the defined 64-bit load, store, and cache management instructions, which are not recognized by the PPC440GP.
5. Program (Privileged Instruction)

   Only applies to the **dcbi** instruction, and only occurs if MSR[PR]=1.
6. Data TLB Error (Data TLB Miss exception)
7. Data Storage (all exception types except Byte Ordering exception)
8. Alignment (Alignment exception)
9. Debug (DAC or DVC exception)
10. Debug (ICMP exception)

10.7.2 Exception Priorities for Floating-Point Load and Store Instructions

The following list identifies the priority order of the exception types that may occur within the PPC440GP as the result of the attempted execution of any floating-point load or store instruction.

1. Debug (IAC exception)
2. Instruction TLB Error (Instruction TLB Miss exception)
3. Instruction Storage (Execute Access Control exception)
4. Program (Illegal Instruction exception)

   This exception will occur if no floating-point unit is attached to the PPC440GP, or if the particular floating-point load or store instruction is not recognized by the attached floating-point unit.
5. Floating-Point Unavailable (Floating-Point Unavailable exception)

   This exception will occur if an attached floating-point unit recognizes the instruction, but floating-point instruction processing is disabled (MSR[FP]=0).
6. Program (Unimplemented Operation exception)

   This exception will occur if an attached floating-point unit recognizes but does not support the instruction, and floating-point instruction processing is enabled (MSR[FP]=1).
7. Data TLB Error (Data TLB Miss exception)
8. Data Storage (all exception types except Cache Locking exception)
9. Alignment (Alignment exception)
10. Debug (DAC or DVC exception)
11. Debug (ICMP exception)

10.7.3 Exception Priorities for Allocated Load and Store Instructions

The following list identifies the priority order of the exception types that may occur within the PPC440GP as the result of the attempted execution of any allocated load or store instruction.

1. Debug (IAC exception)
2. Instruction TLB Error (Instruction TLB Miss exception)
3. Instruction Storage (Execute Access Control exception)
4. Program (Illegal Instruction exception)

   This exception will occur if no auxiliary processor unit is attached to the PPC440GP, or if the particular allocated load or store instruction is not recognized by the attached auxiliary processor.
5. Program (Privileged Instruction exception)

   This exception will occur if an attached auxiliary processor unit recognizes the instruction and indicates that the instruction is privileged, but MSR[PR]=1.
6. Auxiliary Processor Unavailable (Auxiliary Processor Unavailable exception)

   This exception will occur if an attached auxiliary processor recognizes the instruction, but indicates that auxiliary processor instruction processing is disabled (whether or not auxiliary processor instruction processing is enabled is implementation-dependent).

7. Program (Unimplemented Operation exception)

   This exception will occur if an attached auxiliary processor recognizes but does not support the instruction, and also indicates that auxiliary processor instruction processing is enabled (whether or not auxiliary processor instruction processing is enabled is implementation-dependent).
8. Data TLB Error (Data TLB Miss exception)
9. Data Storage (all exception types except Cache Locking exception)
10. Alignment (Alignment exception)
11. Debug (DAC or DVC exception)
12. Debug (ICMP exception)

10.7.4 Exception Priorities for Floating-Point Instructions (Other)

The following list identifies the priority order of the exception types that may occur within the PPC440GP as the result of the attempted execution of any floating-point instruction other than a load or store.

1. Debug (IAC exception)
2. Instruction TLB Error (Instruction TLB Miss exception)
3. Instruction Storage (Execute Access Control exception)
4. Program (Illegal Instruction exception)

   This exception will occur if no floating-point unit is attached to the PPC440GP, or if the particular floating-point instruction is not recognized by the attached floating-point unit.
5. Floating-Point Unavailable (Floating-Point Unavailable exception)

   This exception will occur if an attached floating-point unit recognizes the instruction, but floating-point instruction processing is disabled (MSR[FP]=0).
6. Program (Unimplemented Operation exception)

   This exception will occur if an attached floating-point unit recognizes but does not support the instruction, and floating-point instruction processing is enabled (MSR[FP]=1).
7. Program (Floating-Point Enabled exception)

   This exception will occur if an attached floating-point unit recognizes and supports the instruction, floating-point instruction processing is enabled (MSR[FP]=1), and the instruction sets FPSCR[FEX] to 1.
8. Debug (ICMP exception)

10.7.5 Exception Priorities for Allocated Instructions (Other)

The following list identifies the priority order of the exception types that may occur within the PPC440GP as the result of the attempted execution of any allocated instruction other than a load or store, and which is not one of the allocated instructions implemented within the PPC440GP.

1. Debug (IAC exception)
2. Instruction TLB Error (Instruction TLB Miss exception)
3. Instruction Storage (Execute Access Control exception)

4. Program (Illegal Instruction exception)

   This exception will occur if no auxiliary processor unit is attached to the PPC440GP, or if the particular allocated instruction is not recognized by the attached auxiliary processor and is not one of the allocated instructions implemented within the PPC440GP.

5. Program (Privileged Instruction exception)

   This exception will occur if an attached auxiliary processor unit recognizes the instruction and indicates that the instruction is privileged, but MSR[PR]=1.

6. Auxiliary Processor Unavailable (Auxiliary Processor Unavailable exception)

   This exception will occur if an attached auxiliary processor recognizes the instruction, but indicates that auxiliary processor instruction processing is disabled (whether or not auxiliary processor instruction processing is enabled is implementation-dependent).

7. Program (Unimplemented Operation exception)

   This exception will occur if an attached auxiliary processor recognizes but does not support the instruction, and also indicates that auxiliary processor instruction processing is enabled (whether or not auxiliary processor instruction processing is enabled is implementation-dependent).

8. Program (Auxiliary Processor Enabled exception)

   This exception will occur if an attached auxiliary processor recognizes and supports the instruction, indicates that auxiliary processor instruction processing is enabled, and the instruction execution results in an Auxiliary Processor Enabled exception. Whether or not auxiliary processor instruction processing is enabled is implementation-dependent, as is whether or not a given auxiliary processor instruction results in an Auxiliary Processor Enabled exception.

9. Debug (ICMP exception)

10.7.6 Exception Priorities for Privileged Instructions

The following list identifies the priority order of the exception types that may occur within the PPC440GP as the result of the attempted execution of any privileged instruction other than **dcbi**, **rfi**, **rfci**, or any allocated instruction not implemented within the PPC440GP (all of which are covered elsewhere). This list *does* cover, however, the **dccci**, **dcread**, **iccci**, and **icread** instructions, which are privileged, allocated instructions that *are* implemented within the PPC440GP. This list also covers the defined 64-bit privileged instructions, the **tlbiva** instruction, and the **mfapidi** instruction, all of which are not implemented by the PPC440GP.

1. Debug (IAC exception)
2. Instruction TLB Error (Instruction TLB Miss exception)
3. Instruction Storage (Execute Access Control exception)
4. Program (Illegal Instruction exception)

   Only applies to the defined 64-bit privileged instructions, the **tlbiva** instruction, and the **mfapidi** instruction.

5. Program (Privileged Instruction exception)

   Does not apply to the defined 64-bit privileged instructions, the **tlbiva** instruction, nor the **mfapidi** instruction.

6. Debug (ICMP exception)

Does not apply to the defined 64-bit privileged instructions, the **tlbiva** instruction, nor the **mfapidi** instruction.

10.7.7 Exception Priorities for Trap Instructions

The following list identifies the priority order of the exception types that may occur within the PPC440GP as the result of the attempted execution of a trap (**tw**, **twi**) instruction.

1. Debug (IAC exception)
2. Instruction TLB Error (Instruction TLB Miss exception)
3. Instruction Storage (Execute Access Control exception)
4. Debug (TRAP exception)
5. Program (Trap exception)
6. Debug (ICMP exception)

10.7.8 Exception Priorities for System Call Instruction

The following list identifies the priority order of the exception types that may occur within the PPC440GP as the result of the attempted execution of a system call (**sc**) instruction.

1. Debug (IAC exception)
2. Instruction TLB Error (Instruction TLB Miss exception)
3. Instruction Storage (Execute Access Control exception)
4. System Call (System Call exception)
5. Debug (ICMP exception)

Since the System Call exception does not suppress the execution of the **sc** instruction, but rather the exception occurs once the instruction has completed, it is possible for an **sc** instruction to cause both a System Call exception and an ICMP Debug exception at the same time. In such a case, the associated interrupts will occur in the order indicated in ~~"Interrupt Order" on page 10-41~~*Interrupt Order* on page 375.

10.7.9 Exception Priorities for Branch Instructions

The following list identifies the priority order of the exception types that may occur within the PPC440GP as the result of the attempted execution of a branch instruction.

1. Debug (IAC exception)
2. Instruction TLB Error (Instruction TLB Miss exception)
3. Instruction Storage (Execute Access Control exception)
4. Debug (BRT exception)
5. Debug (ICMP exception)

10.7.10 Exception Priorities for Return From Interrupt Instructions

The following list identifies the priority order of the exception types that may occur within the PPC440GP as the result of the attempted execution of an **rfi** or **rfci** instruction.

1. Debug (IAC exception)
2. Instruction TLB Error (Instruction TLB Miss exception)
3. Instruction Storage (Execute Access Control exception)
4. Debug (RET exception)
5. Debug (ICMP exception)

10.7.11 Exception Priorities for Preserved Instructions

The following list identifies the priority order of the exception types that may occur within the PPC440GP as the result of the attempted execution of a preserved instruction.

1. Debug (IAC exception)
2. Instruction TLB Error (Instruction TLB Miss exception)
3. Instruction Storage (Execute Access Control exception)
4. Program (Illegal Instruction exception)

   Applies to all preserved instructions except the **mftb** instruction, which is the only preserved class instruction implemented within the PPC440GP.
5. Debug (ICMP exception)

   Only applies to the **mftb** instruction, which is the only preserved class instruction implemented within the PPC440GP.

10.7.12 Exception Priorities for Reserved Instructions

The following list identifies the priority order of the exception types that may occur within the PPC440GP as the result of the attempted execution of a reserved instruction.

1. Debug (IAC exception)
2. Instruction TLB Error (Instruction TLB Miss exception)
3. Instruction Storage (Execute Access Control exception)
4. Program (Illegal Instruction exception)

   Applies to all reserved instruction opcodes except the reserved-nop instruction opcodes.
5. Debug (ICMP exception)

   Only applies to the reserved-nop instruction opcodes.

10.7.13 Exception Priorities for All Other Instructions

The following list identifies the priority order of the exception types that may occur within the PPC440GP as the result of the attempted execution of all other instructions (that is, those not covered by one of the sections 10.7.1 through 10.7.12). This includes both defined instructions and allocated instructions implemented within the PPC440GP.

1. Debug (IAC exception)
2. Instruction TLB Error (Instruction TLB Miss exception)

3. Instruction Storage (Execute Access Control exception)
4. Program (Illegal Instruction exception)

   Applies only to the defined 64-bit instructions, as these are not implemented within the PPC440GP.
5. Debug (ICMP exception)

   Does not apply to the defined 64-bit instructions, as these are not implemented by the PPC440GP.

11. Timer Facilities

The PPC440GP provides four timer facilities: a time base, a Decrementer (DEC), a Fixed Interval Timer (FIT), and a Watchdog Timer. These facilities, which share the same source clock frequency, can support:

- Time-of-day functions
- General software timing functions
- Peripherals requiring periodic service
- General system maintenance
- System error recoverability

Figure 11-1 shows the relationship between these facilities and the clock source.

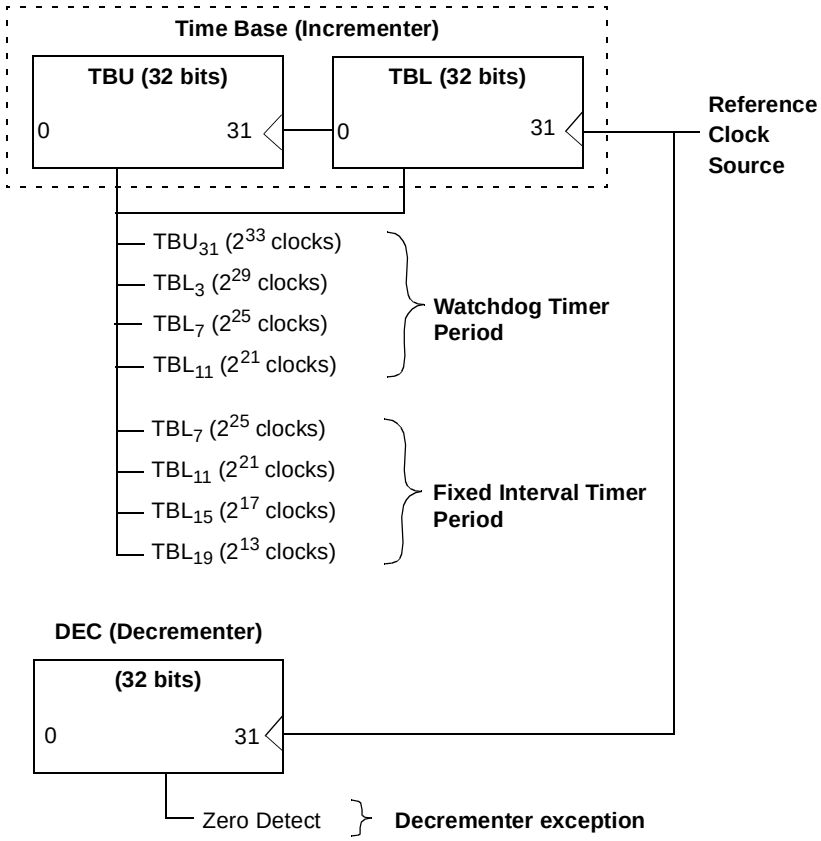


Figure 11-1. Relationship of Timer Facilities to the Time Base

11.1 Time Base

The time base is a 64-bit register which increments once during each period of the source clock, and provides a time reference. Access to the time base is via two Special Purpose Registers (SPRs). The Time Base Upper (TBU) SPR contains the high-order 32 bits of the time base, while the Time Base Lower (TBL) SPR contains the low-order 32 bits.

Software access to TBU and TBL is non-privileged for read but privileged for write, and hence different SPR numbers are used for reading than for writing. TBU and TBL are written using **mtspr** and read using **mfspr**.

The period of the 64-bit time base is approximately 1462 years for a 400 MHz clock source. The time base value itself does not generate any exceptions, even when it wraps. For most applications, the time base is set once at system reset and only read thereafter. Note that Fixed Interval Timer and Watchdog Timer exceptions (discussed below) are caused by 0→1 transitions of selected bits from the time base. Transitions of these bits caused by software alteration of the time base have the same effect as transitions caused by normal incrementing of the time base.

~~Figure 11-2~~Figure 11-2 illustrates the TBL.

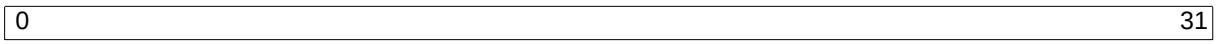


Figure 0-1. Time Base Lower (TBL)

0:31		Time Base Lower	Low-order 32 bits of time base.

Figure 11-2. Time Base Lower (TBL)

0:31		Time Base Lower	Low-order 32 bits of time base.

~~Figure 11-3~~Figure 11-3 illustrates the TBU.

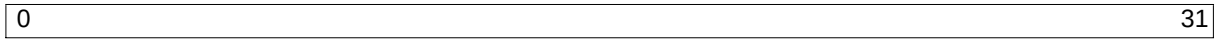


Figure 0-2. Time Base Upper (TBU)

0:31		Time Base Upper	High-order 32 bits of time base.

Figure 11-3. Time Base Upper (TBU)

0:31		Time Base Upper	High-order 32 bits of time base.

11.1.1 Reading the Time Base

The following code provides an example of reading the time base.

```
loop:
        mfspr Rx,TBU            # read TBU into GPR Rx
        mfspr Ry,TBL            # read TBL into GPR Ry
        mfspr Rz,TBU            # read TBU again, this time into GPR Rz
        cmpw Rz, Rx             # see if old = new
        bne   loop              # loop/reread if rollover occurred
```

The comparison and loop ensure that a consistent pair of values is obtained.

11.1.2 Writing the Time Base

The following code provides an example of writing the time base.

```
lwz     Rx, upper               # load 64-bit time base value into GPRs Rx and Ry
lwz     Ry, lower
li      Rz, 0                   # set GPR Rz to 0
mtspr   TBL,Rz                  # force TBL to 0 (thereby preventing wrap into TBU)
mtspr   TBU,Rx                  # set TBU to initial value
mtspr   TBL,Ry                  # set TBL to initial value
```

11.2 Decrementer (DEC)

The DEC is a 32-bit privileged SPR that decrements at the same rate that the time base increments. The DEC is read and written using **mfspr** and **mtspr**, respectively. When a non-zero value is written to the DEC, it begins to decrement with the next time base clock. A Decrementer exception is signalled when a decrement occurs on a DEC count of 1, and the Decrementer Interrupt Status field of the Timer Status Register (TSR[DIS]; see Page 390) is set. A Decrementer interrupt will occur if it is enabled by both the Decrementer Interrupt Enable field of the Timer Control Register (TCR[DIE]; see Page 389) and by the External Interrupt Enable field of the Machine State Register (MSR[EE]; see ~~"Machine State Register (MSR)" on page 10-7~~*Machine State Register (MSR)* on page 343). ~~"Interrupts and Exceptions" on page 10-1~~*Interrupts and Exceptions* on page 337 provides more information on the handling of Decrementer interrupts.

The Decrementer interrupt handler software should clear TSR[DIS] before re-enabling MSR[EE], in order to avoid another Decrementer interrupt due to the same exception (unless TCR[DIE] is cleared instead).

The behavior of the DEC itself upon a decrement from a DEC value of 1 depends on which of two modes it is operating in -- normal, or auto-reload. The mode is controlled by the Auto-Reload Enable (ARE) field of the TCR. When operating in normal mode (TCR[ARE]=0), the DEC simply decrements to the value 0 and then stops decrementing until it is re-initialized by software.

When operating in auto-reload mode (TCR[ARE]=1), however, instead of decrementing to the value 0, the DEC is reloaded with the value in the Decrementer Auto-Reload (DECAR) register (see ~~Figure 11-5 on page 11-4~~*Figure 11-5* on page 386), and continues to decrement with the next time base clock (assuming the DECAR value was non-zero). The DECAR register is a 32-bit privileged, write-only SPR, and is written using **mtspr**.

The auto-reload feature of the DEC is disabled upon reset, and must be enabled by software.

~~Figure 11-4~~Figure 11-4 illustrates the DEC.

Figure 0-3. Decrementer (DEC)

0:31		Decrement value

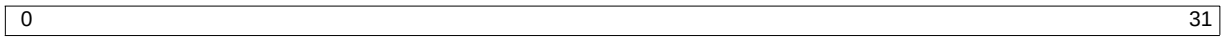


Figure 11-4. Decrementer (DEC)

0:31		Decrement value

~~Figure 11-5~~Figure 11-5 illustrates the DECAR.

Figure 0-4. Decrementer Auto-Reload (DECAR)

0:31		Decrementer auto-reload value	Copied to DEC at next time base clock when DEC = 1 and auto-reload is enabled (TCR[ARE] = 1).

Figure 11-5. Decrementer Auto-Reload (DECAR)

0:31		Decrementer auto-reload value	Copied to DEC at next time base clock when DEC = 1 and auto-reload is enabled (TCR[ARE] = 1).

Using **mtspr** to force the DEC to 0 does *not* cause a Decrementer exception and thus does not cause TSR[DIS] to be set. However, if a time base clock causes a decrement from a DEC value of 1 to occur simultaneously with the writing of the DEC by a **mtspr** instruction, then the Decrementer exception *will* occur, TSR[DIS] will be set, and the DEC will be written with the value from the **mtspr**.

In order for software to quiesce the activity of the DEC and eliminate all DEC exceptions, the following procedure should be followed:

1. Write 0 to TCR[DIE]. This prevents a Decrementer exception from causing a Decrementer interrupt.
2. Write 0 to TCR[ARE]. This disables the DEC auto-reload feature.
3. Write 0 to the DEC to halt decrementing. Although this action does not itself cause a Decrementer exception, it is possible that a decrement from a DEC value of 1 has occurred since the last time that TSR[DIS] was cleared.
4. Write 1 to TSR[DIS] (DEC Interrupt Status bit). This clears the Decrementer exception by setting TSR[DIS] to 0. Because the DEC is no longer decrementing (due to having been written with 0 in step 3), no further Decrementer exceptions are possible.

11.3 Fixed Interval Timer (FIT)

The FIT provides a mechanism for causing periodic exceptions with a regular period. The FIT would typically be used by system software to invoke a periodic system maintenance function, executed by the Fixed Interval Timer interrupt handler.

A Fixed Interval Timer exception occurs on a 0→1 transition of a selected bit from the time base. Note that a Fixed Interval Timer exception will also occur if the selected time base bit transitions from 0→1 due to a **mtspr** instruction that writes 1 to that time base bit when its previous value was 0.

The Fixed Interval Timer Period (FP) field of the TCR selects one of four bits from the time base, as shown in ~~Table 11-1~~*Table 11-1*.

Table 11-1. Fixed Interval Timer Period Selection

TCR[FP]	Time Base Bit	Period (Time Base Clocks)	Period (400 Mhz Clock)
0b00	TBL_{19}	2^{13} clocks	20.48 µs
0b01	TBL_{15}	2^{17} clocks	327.68 µs
0b10	TBL_{11}	2^{21} clocks	5.2 ms

Table 11-1. Fixed Interval Timer Period Selection (continued)

TCR[FP]	Time Base Bit	Period (Time Base Clocks)	Period (400 Mhz Clock)
0b11	TBL_7	2^{25} clocks	83.9 ms

When a Fixed Interval Timer exception occurs, the exception status is recorded by setting the Fixed interval Timer Interrupt Status (FIS) field of the TSR to 1. A Fixed Interval Timer interrupt will occur if it is enabled by both the Fixed Interval Timer Interrupt Enable (FIE) field of the TCR and by MSR[EE]. ~~"Fixed-Interval Timer Interrupt" on page 10-32~~*Fixed-Interval Timer Interrupt* on page 366 provides more information on the handling of Fixed Interval Timer interrupts.

The Fixed Interval Timer interrupt handler software should clear TSR[FIS] before re-enabling MSR[EE], in order to avoid another Fixed Interval Timer interrupt due to the same exception (unless TCR[FIE] is cleared instead).

11.4 Watchdog Timer

The Watchdog Timer provides a mechanism for system error recovery in the event that the program running on the PPC440GP has stalled and cannot be interrupted by the normal interrupt mechanism. The Watchdog Timer can be configured to cause a critical-class Watchdog Timer interrupt upon the expiration of a single period of the Watchdog Timer. It can also be configured to invoke a processor-initiated reset upon the expiration of a second period of the Watchdog Timer.

A Watchdog Timer exception occurs on a 0→1 transition of a selected bit from the time base. Note that a Watchdog Timer exception will also occur if the selected time base bit transitions from 0→1 due to a **mtspr** instruction that writes 1 to that time base bit when its previous value was 0.

The Watchdog Timer Period (WP) field of the TCR selects one of four bits from the time base, as shown in ~~Table 11-2~~*Table 11-2*.

Table 11-2. Watchdog Timer Period Selection

TCR[WP]	Time Base Bit	Period (Time Base Clocks)	Period (400 MHz Clock)
0b00	TBL_{11}	2^{21} clocks	5.2 ms
0b01	TBL_7	2^{25} clocks	83.9 ms
0b10	TBL_3	2^{29} clocks	1.34 s
0b11	TBU_{31}	2^{33} clocks	21.47 s

The action taken upon a Watchdog Timer exception depends upon the status of the Enable Next Watchdog (ENW) and Watchdog Timer Interrupt Status (WIS) fields of the TSR at the time of the exception. When TSR[ENW] = 0, the next Watchdog Timer exception is "disabled", and the only action to be taken upon the exception is to set TSR[ENW] to 1. By clearing TSR[ENW], software can guarantee that the time until the next *enabled* Watchdog Timer exception will be *at least* one full Watchdog Timer period (and a maximum of *two* full Watchdog Timer periods).

When TSR[ENW] = 1, the next Watchdog Timer exception is enabled, and the action to be taken upon the exception depends on the value of TSR[WIS] at the time of the exception. If TSR[WIS] = 0, then the action is to set TSR[WIS] to 1, at which time a Watchdog Timer interrupt will occur if enabled by both the Watchdog Timer Interrupt Enable (WIE) field of the TCR and by the Critical Interrupt Enable (CE) field of the MSR. The

Watchdog Timer interrupt handler software should clear TSR[WIS] before re-enabling MSR[CE], in order to avoid another Watchdog Timer interrupt due to the same exception (unless TCR[WIE] is cleared instead). ~~"Watchdog Timer Interrupt" on page 10-32~~*Watchdog Timer Interrupt* on page 367 provides more information on the handling of Watchdog Timer interrupts.

If TSR[WIS] is already 1 at the time of the next Watchdog Timer exception, then the action to take depends on the value of the Watchdog Reset Control (TRC) field of the TCR. If TCR[WRC] is non-zero, then the value of the TCR[WRC] field will be copied into TSR[WRS], TCR[WRC] will be cleared, and a core reset will occur (see ~~"Processor Core State After Reset" on page 7-3~~*Processor Core State After Reset* on page 261 for more information on core behavior when reset).

Note that once software has set TCR[WRC] to a non-zero value, it cannot be reset by software; this feature prevents errant software from disabling the Watchdog Timer reset capability.

~~Table 11-3~~*Table 11-3* summarizes the action to be taken upon a Watchdog Timer exception according to the values of TSR[ENW] and TSR[WIS].

Table 11-3. Watchdog Timer Exception Behavior

TSR[ENW]	TSR[WIS]	Action upon Watchdog Timer exception
0	0	Set TSR[ENW] to 1
0	1	Set TSR[ENW] to 1
1	0	Set TSR[WIS] to 1. If Watchdog Timer interrupts are enabled (TCR[WIE]=1 and MSR[CE]=1), then interrupt.
1	1	Cause Watchdog Timer reset action specified by TCR[WRC]. Reset will copy pre-reset TCR[WRC] into TSR[WRS], then clear TCR[WRC].

A typical system usage of the Watchdog Timer function is to enable the Watchdog Timer interrupt and the Watchdog Timer reset function in the TCR (and MSR), and to start out with both TSR[ENW] and TSR[WIS] clear. Then, a recurring software loop of reliable duration (or perhaps the interrupt handler for a periodic interrupt such as the Fixed Interval Timer interrup) performs a periodic check of system integrity. Upon successful completion of the system check, software clears TSR[ENW], thereby ensuring that a minimum of one full Watchdog Timer period and a maximum of two full Watchdog Timer periods must expire before an enabled Watchdog Timer exception will occur.

If for some reason the recurring software loop is not successfully completed (and TSR[ENW] thus not cleared) during this period of time, then an enabled Watchdog Timer exeption will occur. This will set TSR[WIS] and a Watchdog Timer interrupt will occur (if enabled by both TCR[WIE] and MSR[CE]). The occurrence of a Watchdog Timer interrupt in this kind of system is interpreted as a "system error", insofar as the system was for some reason unable to complete the periodic system integrity check in time to avoid the enabled Watchdog Timer exception. The action taken by the Watchdog Timer interrupt handler is of course system-dependent, but typically the software will attempt to determine the nature of the problem and correct it if possible. If and when the system attempts to resume operation, the software would typically clear both TSR[WIS] and TSR[ENW], thus providing a minimum of another full Watchdog Timer period for a new system integrity check to occur.

Finally, if for some reason the Watchdog Timer interrupt is disabled, and/or the Watchdog Timer interrupt handler is unsuccessful in clearing TSR[WIS] and TSR[ENW] prior to another Watchdog Timer exception, then the next exception will cause a processor reset operation to occur, according to the value of TCR[WRC].

Figure 11-6*Figure 11-6* illustrates the sequence of Watchdog Timer events which occurs according to this

typical system usage.

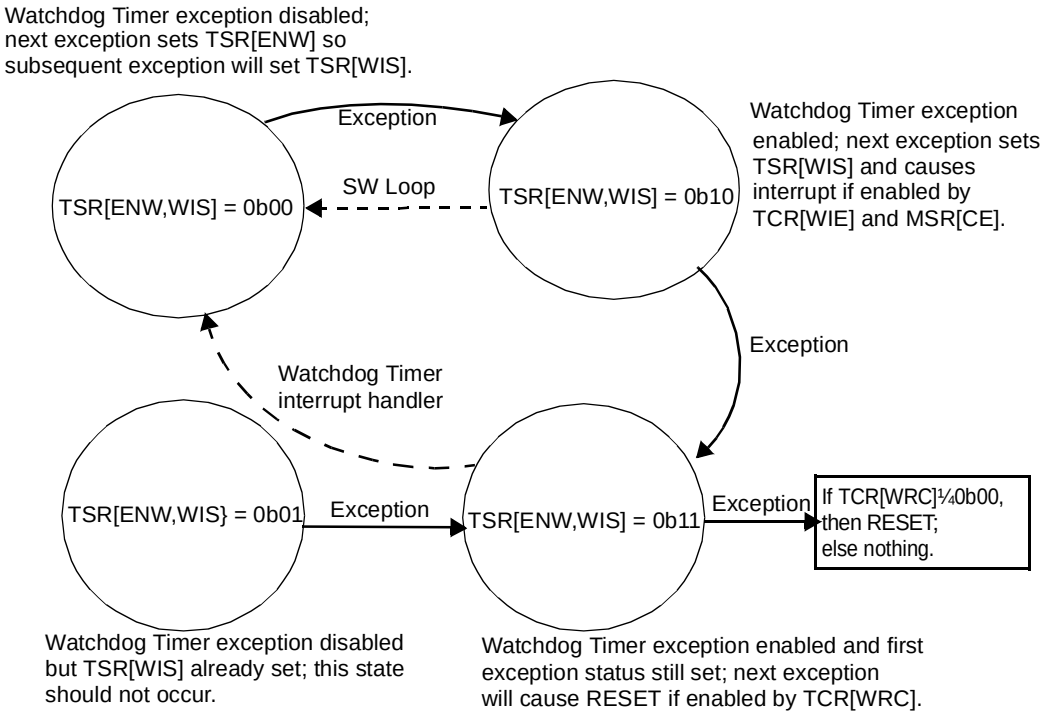


Figure 0-5. Watchdog State Machine

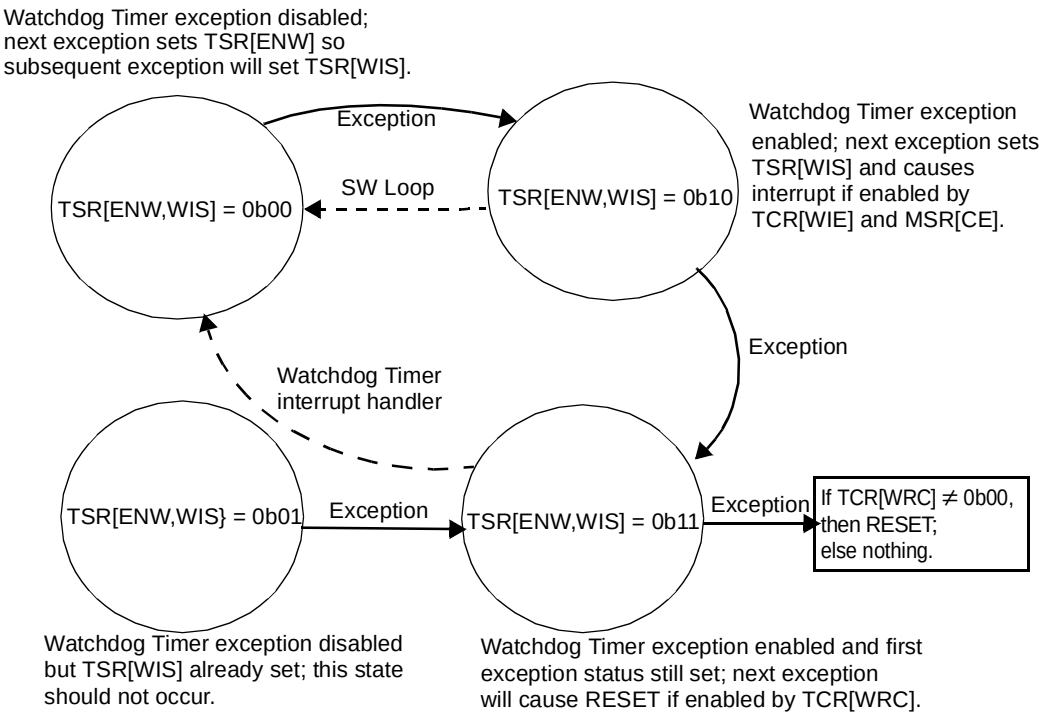


Figure 11-6. Watchdog State Machine

11.5 Timer Control Register (TCR)

The TCR is a privileged SPR that controls DEC, FIT, and Watchdog Timer operation. The TCR is read into a GPR using **mfspr**, and is written from a GPR using **mtspr**.

The Watchdog Timer Reset Control (WRC) field of the TCR is cleared to 0 by processor reset (see ~~Chapter 7, "Reset and Initialization"~~ *Reset and Initialization* on page 259). Each bit of this 2-bit field is set only by software and is cleared only by hardware. For each bit of the field, once software has written it to 1, that bit remains 1 until processor reset occurs. This is to prevent errant code from disabling the Watchdog Timer reset function.

The Auto-Reload Enable (ARE) field of the TCR is also cleared to zero by processor reset. This disables the auto-reload feature of the DEC.

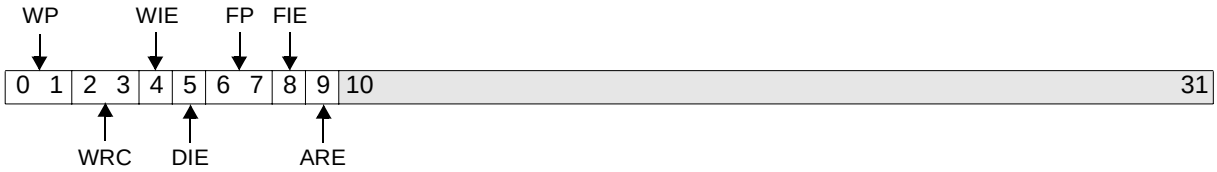


Figure 0-6. Timer Control Register (TCR)

0:1	WP	Watchdog Timer Period 00 2^{21} time base clocks 01 2^{25} time base clocks 10 2^{29} time base clocks 11 2^{33} time base clocks	
2:3	WRC	Watchdog Timer Reset Control 00 No Watchdog Timer reset will occur. 01 Core reset 10 Chip reset 11 System reset	TCR[WRC] resets to 0b00. Type of reset to cause upon Watchdog Timer exception with TSR[ENW,WIS]=0b11. This field can be set by software, but cannot be cleared by software, except by a software-induced reset.
4	WIE	Watchdog Timer Interrupt Enable 0 Disable Watchdog Timer interrupt. 1 Enable Watchdog Timer interrupt.	
5	DIE	Decrementer Interrupt Enable 0 Disable Decrementer interrupt. 1 Enable Decrementer interrupt.	
6:7	FP	Fixed Interval Timer (FIT) Period 00 2^{13} time base clocks 01 2^{17} time base clocks 10 2^{21} time base clocks 11 2^{25} time base clocks	
8	FIE	FIT Interrupt Enable 0 Disable Fixed Interval Timer interrupt. 1 Enable Fixed Interval Timer interrupt.	
9	ARE	Auto-Reload Enable 0 Disable auto reload. 1 Enable auto reload.	TCR[ARE] resets to 0b0.
10:31		Reserved	

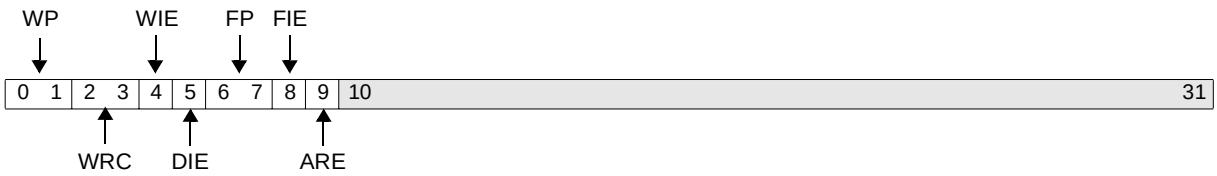


Figure 11-7. Timer Control Register (TCR)

0:1	WP	Watchdog Timer Period 00 2^{21} time base clocks 01 2^{25} time base clocks 10 2^{29} time base clocks 11 2^{33} time base clocks	
2:3	WRC	Watchdog Timer Reset Control 00 No Watchdog Timer reset will occur. 01 Core reset 10 Chip reset 11 System reset	TCR[WRC] resets to 0b00. Type of reset to cause upon Watchdog Timer exception with TSR[ENW,WIS]=0b11. This field can be set by software, but cannot be cleared by software, except by a software-induced reset.
4	WIE	Watchdog Timer Interrupt Enable 0 Disable Watchdog Timer interrupt. 1 Enable Watchdog Timer interrupt.	
5	DIE	Decrementer Interrupt Enable 0 Disable Decrementer interrupt. 1 Enable Decrementer interrupt.	
6:7	FP	Fixed Interval Timer (FIT) Period 00 2^{13} time base clocks 01 2^{17} time base clocks 10 2^{21} time base clocks 11 2^{25} time base clocks	
8	FIE	FIT Interrupt Enable 0 Disable Fixed Interval Timer interrupt. 1 Enable Fixed Interval Timer interrupt.	
9	ARE	Auto-Reload Enable 0 Disable auto reload. 1 Enable auto reload.	TCR[ARE] resets to 0b0.
10:31		Reserved	

11.6 Timer Status Register (TSR)

The TSR is a privileged SPR that records the status of DEC, FIT, and Watchdog Timer events. The fields of the TSR are generally set to 1 only by hardware and cleared to 0 only by software. Hardware cannot clear any fields in the TSR, nor can software set any fields. Software can read the TSR into a GPR using **mfspr**. Clearing the TSR is performed using **mtspr** by placing a 1 in the GPR source register in all bit positions which are to be cleared in the TSR, and a 0 in all other bit positions. The data written from the GPR to the TSR is not direct data, but a mask. A 1 clears the bit and a 0 leaves the corresponding TSR bit unchanged.

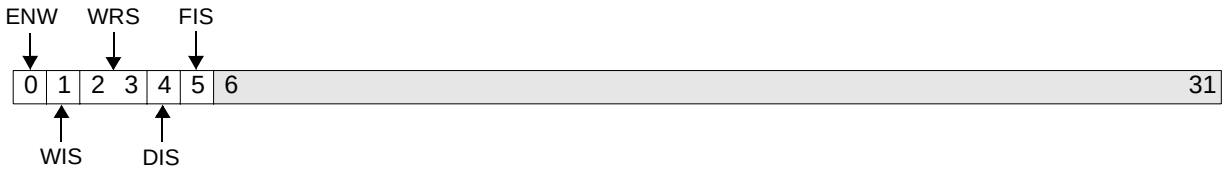


Figure 0-7. Timer Status Register (TSR)

0	ENW	Enable Next Watchdog Timer Exception 0 Action on next Watchdog Timer exception is to set TSR[ENW] = 1. 1 Action on next Watchdog Timer exception is governed by TSR[WIS].
1	WIS	Watchdog Timer Interrupt Status 0 Watchdog Timer exception has not occurred. 1 Watchdog Timer exception has occurred.
2:3	WRS	Watchdog Timer Reset Status 00 No Watchdog Timer reset has occurred. 01 Core reset was forced by Watchdog Timer. 10 Chip reset was forced by Watchdog Timer. 11 System reset was forced by Watchdog Timer.
4	DIS	Decrementer Interrupt Status 0 Decrementer exception has not occurred. 1 Decrementer exception has occurred.
5	FIS	Fixed Interval Timer (FIT) Interrupt Status 0 Fixed Interval Timer exception has not occurred. 1 Fixed Interval Timer exception has occurred.
6:31		Reserved

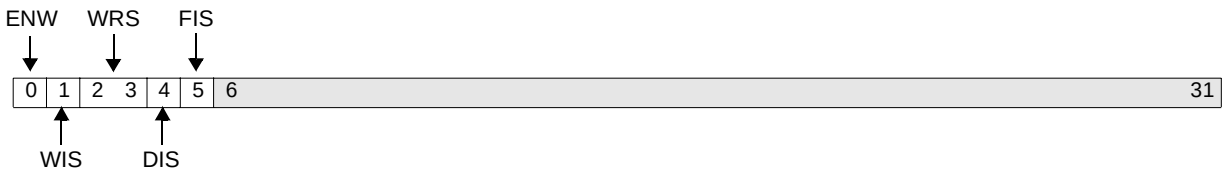


Figure 11-8. Timer Status Register (TSR)

0	ENW	Enable Next Watchdog Timer Exception 0 Action on next Watchdog Timer exception is to set TSR[ENW] = 1. 1 Action on next Watchdog Timer exception is governed by TSR[WIS].
1	WIS	Watchdog Timer Interrupt Status 0 Watchdog Timer exception has not occurred. 1 Watchdog Timer exception has occurred.

2:3	WRS	Watchdog Timer Reset Status 00 No Watchdog Timer reset has occurred. 01 Core reset was forced by Watchdog Timer. 10 Chip reset was forced by Watchdog Timer. 11 System reset was forced by Watchdog Timer.
4	DIS	Decrementer Interrupt Status 0 Decrementer exception has not occurred. 1 Decrementer exception has occurred.
5	FIS	Fixed Interval Timer (FIT) Interrupt Status 0 Fixed Interval Timer exception has not occurred. 1 Fixed Interval Timer exception has occurred.
6:31		Reserved

11.7 Freezing the Timer Facilities

The debug mechanism provides a means for temporarily "freezing" the timers upon a debug exception. Specifically, the time base and Decrementer can be prevented from incrementing and decrementing, respectively, whenever a debug exception is recorded in the Debug Status Register (DBSR). This allows a debugger to simulate the appearance of "real time", even though the application has been temporarily "halted" to service the debug event.

See ~~Chapter 15, "Debug Facilities"~~*Debug Facilities* on page 429 for more information on freezing the timers.

12. General Purpose Timers

The General Purpose Timer (GPT) is an on-chip peripheral bus (OPB) soft core that provides a separate time base counter and additional system timers beyond those defined in the PowerPC® architecture. Five compare timers are implemented in PPC440GP. The following sections include a list of major features, an overview, supported configurability, I/O signal descriptions, and register descriptions of the General Purpose Timer core.

12.1 Overview

The GPT core is an OPB-compliant soft core that provides additional system timers. A time base is provided by the 32-bit Time Base Counter Register (GPT0_TBC) that continually increments by one unless written or reset. From its maximum value, it rolls to 0. ~~A separate clock pin (TBCClk) is provided for the Time Base Counter but must be driven by the system clock (SysClk).~~ GPT_chipReset resets the entire GPT core, including the GPT0_TBC. TBCreset resets just the GPT0_TBC.

Each of the five compare timers is 32 bits wide and provides a reference value that is compared to the GPT0_TBC (see ~~Figure 12-1 on page 12-3~~*Figure 12-1* on page 395 for the compare timers Logic/Block Diagram). Each compare timer is an OPB register. For each compare timer, a corresponding mask register (also an OPB register) allows the masking of individual bits during the compare. If a compare is made, an interrupt status bit may be set (if not masked), and an interrupt may be generated (if enabled). In addition, GPT_CompOut may be activated (if enabled separately).

12.1.1 Features

- Direct control of all functions from registers programmed via OPB bus master accesses
- 32 bit time base

  A SysClk time base counter clock input for all I/O signals)
- OPB slave interface for access to all control, timer and status registers

  Registers provide direct control of all GPT functions
- Five compare timers with unique outputs (GPT_CompOut)

  Separately programmable output levels
- Five interrupt outputs (GPT_uicIntrpt), one per each compare timer
- Two reset inputs, one for the entire core, one just for the time base
- OPB address decode configurable to use a configurable base address or core select input

12.2 Programmability

The GPT core is fully programmable through the OPB interface. Programmability features include:

- Programmable time base register (sets the Time Base Counter)
- Maskable time-base comparison support for each compare timer

- Programmable compare timer values
- Enable/disable control of all compare interrupts
- Mask control of interrupt status bits
- Programmable level for all compare timers
- Programmable secondary output signals for the compare timers

12.3 Mode of Operation

12.3.1 Time Base Counter

The Time Base Counter (TBC) is both an OPB register and an unsigned counter, and provides the reference time for all compare timers. It increments by one with each clock period and is 32 bits wide. When the Time Base Counter is at its maximum value (all bits set to 1) it will roll back to zero upon the next clock.

The TBC clock source is provided by the TBCClk core input. It must be driven by the same clock source which drives SysClk. In other words, the TBCClk and SysClk pins must be tied together at the chip level.

The TBC is synchronously reset to zero upon a full chip reset (GPT_chipReset) or when TBCreset is active. It may be read and written via software through the OPB interface. When written the new value is stored with the next rising edge of TBCClk.

12.3.2 .Compare Timers

Each of the five compare timers continually compares its 32-bit programmed value with the TBC value on a bit-by-bit basis (see Figure 12-1 for the Compare Timer Logic/Block Diagram). For each compare timer, selected bits may be individually masked from the comparator logic by programming the corresponding Mask register (GPT0_MASKn). This forces a valid comparison for these selected bits.

If the corresponding GPT Interrupt Mask (GPT0_IM) Register bit is ~~disabled~~ low (not masked), and when all bits either compare or are masked, then the GPT Interrupt Status (GPT0_IS) Register bit is set, indicating a valid comparison. Further, if the corresponding GPT Interrupt Enable (GPT0_IE) Register bit is set (enabled), then GPT_uicIntrpt[1], the core output interrupt signal, is generated. Typically, this signal goes to an Interrupt Controller core.

A valid comparison will also activate the GPT_CompOut signal if the corresponding GPT Output Enable (GPT0_OE) Register bit is set. The active level for each GPT_CompOut signal is individually programmed through the GPT Output Level (GPT0_OL) Register. Each GPT_CompOut signal is also registered to avoid glitches on the output.

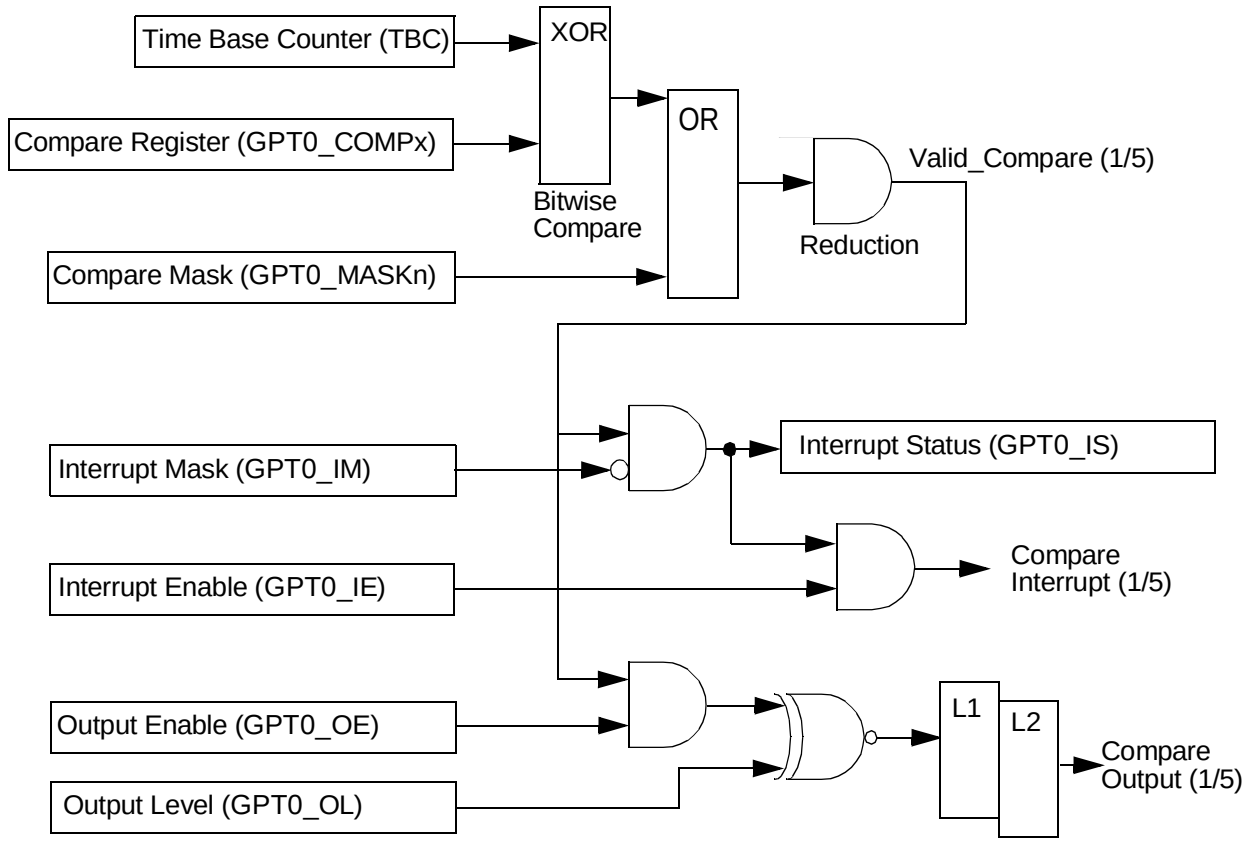


Figure 12-1. Compare Timers Logic/Block Diagram

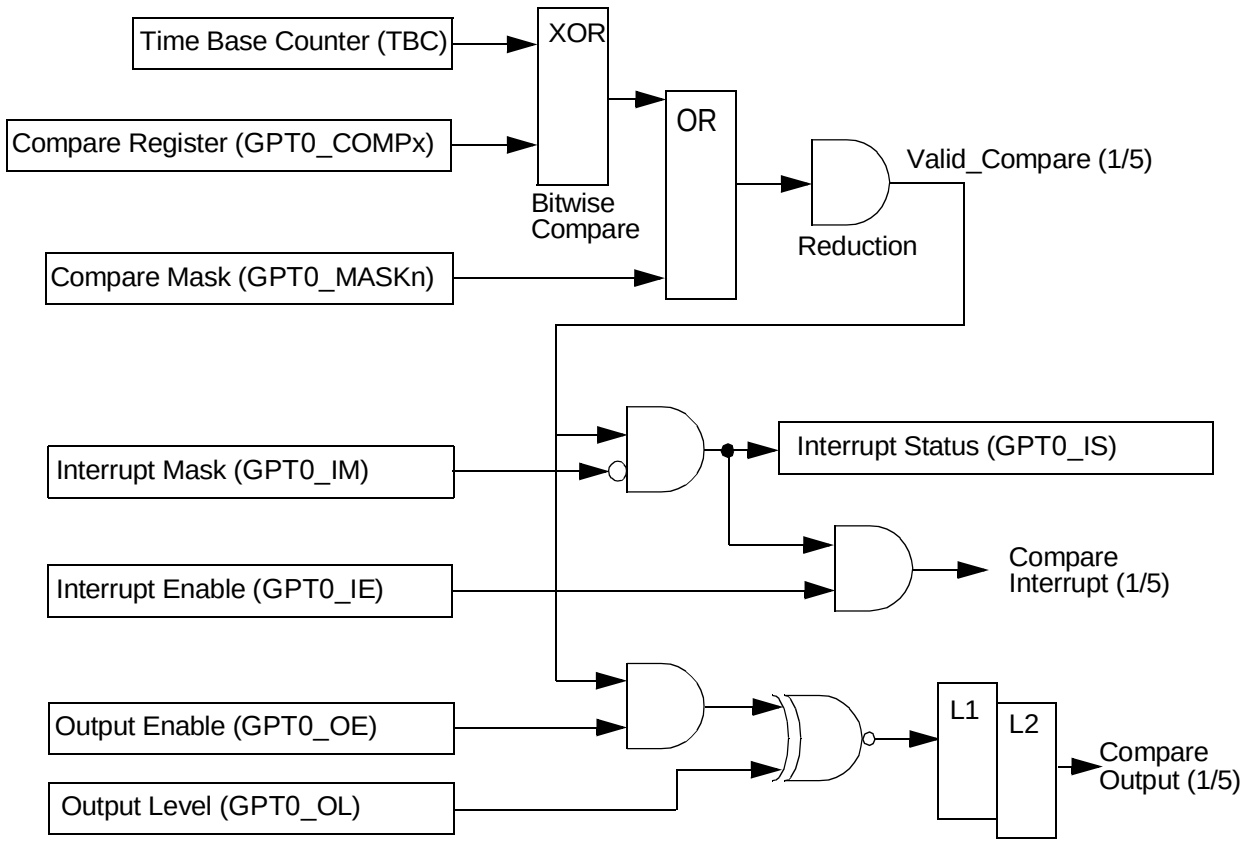


Figure 0-1. Compare Timers Logic/Block Diagram

12.3.3 Compare Timers Interrupt

The following steps must happen in order for a compare timer interrupt to occur.

1. The corresponding Compare Timer Register (GPT0_COMPn) must be programmed to the desired compare value.
2. The corresponding Time Base Counter Mask Register (GPT0_MASKn) must be programmed with the desired mask bit pattern.
3. The corresponding Interrupt Mask bit must be reset (0) in the GPT Interrupt Mask (GPT0_IM) Register.
4. The corresponding Interrupt Enable bit must be set (1) in the GPT Interrupt Enable (GPT0_IE) Register.
5. The Time Base Counter must increment to the same value programmed in the corresponding Compare Timer (GPT0_COMPn) Register, excluding Compare Mask (GPT0_MASKn) Register bits.

12.3.4 Compare Timers Output

The following steps must happen in order for a compare timer output to occur.

1. The corresponding Compare Timer (GPT0_COMPn) Register must be programmed to the desired compare value.

2. The corresponding Compare Mask Register (GPT0_MASKn) must be programmed with the desired mask bit pattern.
3. The corresponding bit must be set (1) in the GPT Output Enable (GPT0_OE) Register.
4. The corresponding bit must be programmed in the GPT Output Level (GPT0_OL) Register to determine the signal level of the GPT_CompOut.
5. The Time Base Counter (TBC) must increment to the same value programmed in the corresponding Compare Timer (GPT0_COMPn) Register, excluding Compare Mask (GPT0_MASKn) Register bits.

12.4 Interrupt Generation

GPT_uicIntrpt is implemented as five separate interrupt lines, one for each of the five compare timers. Since all OPB registers and the TBC run off SysClk, all interrupts are also synchronized to SysClk.

12.5 GPT Registers

The GPT includes the device control registers (DCRs) listed in ~~Table 12-1~~*Table 12-1*. The registers are accessed using the **mfdcr** and **mtdcr** instructions.

Table 12-1. GPT Register Addresses, Names, and Access Modes

Mnemonic	Name	Address	Access	Page
GPT0_TBC	Time Base Counter	0x1 40000A00	R/W	~~12-6~~398
GPT0_OE	GPT Output Enable	0x1 40000A10	R/W	~~12-7~~399
GPT0_OL	GPT Output Level	0x1 40000A14	R/W	~~12-8~~400
GPT0_IM	GPT Interrupt Mask	0x1 40000A18	R/W	~~12-9~~401
GPT0_ISS	GPT Interrupt Status (Set bits if write 1)	0x1 40000A1C	R/W	~~12-10~~402
GPT0_ISC	GPT Interrupt Status (Clear bits if write 1)	~~020~~0x1 40000A20	R/W	~~12-10~~402
GPT0_IE	GPT Interrupt Enable	0x1 40000A24	R/W	~~12-11~~403
GPT0_COMP0	Compare Timer 0	0x 40000A80	R/W	~~12-12~~403
GPT0_COMP1	Compare Timer 1	0x 40000A84	R/W	~~12-12~~403
GPT0_COMP2	Compare Timer 2	0x 40000A88	R/W	~~12-12~~403
GPT0_COMP3	Compare Timer 3	0x 40000A8C	R/W	~~12-12~~403
GPT0_COMP4	Compare Timer 4	0x 40000A90	R/W	~~12-12~~403
GPT0_MASK0	Compare Mask (Compare Timer 0)	0x 40000AC0	R/W	~~12-13~~405
GPT0_MASK1	Compare Mask (Compare Timer 1)	0x 40000AC4	R/W	~~12-13~~405
GPT0_MASK2	Compare Mask (Compare Timer 2)	0x 40000AC8	R/W	~~12-13~~405
GPT0_MASK3	Compare Mask (Compare Timer 3)	0x 40000ACC	R/W	~~12-13~~405
GPT0_MASK4	Compare Mask (Compare Timer 4)	0x 40000AD0	R/W	~~12-13~~405

All GPT registers are memory mapped and accessed via load/store instructions at the address of the register. All registers are accessed on 32-bit boundaries relative to the configurable base address.

12.5.1 GPT Register Reset Values

All GPT registers are reset to zero (0) except for the GPT Interrupt Mask Register (GPT0_IM) and the GPT Output Level Register (GPT0_OL), which are both reset to one (1) for all implemented bits. Bits not implemented will still read as 0 when read via the OPB interface.

The GPT can be reset by three different means. The first is a scan flush reset, forced at the chip level by forcing both LSSD_AClk and LSSD_BClk to one (1) and LSSD_ScanIn to zero (0). Because all latches in the GPT core are connected in a scan chain, this causes all latches to flush to a certain value (see ~~Table 12-2~~*Table 12-2*).

The second way to reset the GPT is with the GPT_chipReset signal. To save on power consumption, only those registers required to bring the GPT to a stable state are reset with GPT_chipReset. Other registers should be programmed first before being used. ~~Table 12-2~~*Table 12-2* shows which registers are reset with GPT_chipReset and which are not.

Table 12-2. GPT Register Reset Values

Mnemonic	Scan Flush Reset Value	Width	Bus I/F	Reset with GPT_chipReset?	GPT_chipResetReset Value	Page
GPT0_TBC	32'h 00000000	32	OPB	yes	32'h 00000000	~~12-6~~398
GPT0_OE	32'h 00000000	7	OPB	yes	32'h 00000000	~~12-7~~399
GPT0_OL	32'h 00000000	7	OPB	no	NA	~~12-8~~400
GPT0_IM	32'h 00000000	3+7	OPB	yes	32'h 00000000	~~12-9~~401
GPTx_ISS	32'h 00000000	3+7	OPB	no	NA	~~12-10~~402
GPT0_ISC	32'h 00000000	3+7	OPB	no	NA	~~12-10~~402
GPT0_IE	32'h 00000000	3+7	OPB	no	NA	~~12-11~~403
GPT0_COMP0	32'h 00000000	32	OPB	no	NA	~~12-12~~403
GPT0_COMP1	32'h 00000000	32	OPB	no	N/A	~~12-12~~403
GPT0_COMP2	32'h 00000000	32	OPB	no	N/A	~~12-12~~403
GPT0_COMP3	32'h 00000000	32	OPB	no	N/A	~~12-12~~403
GPT0_COMP4	32'h 00000000	32	OPB	no	N/A	~~12-12~~403
GPT0_MASK0	32'h 00000000	32	OPB	no	NA	~~12-13~~405
GPT0_MASK1	32'h 00000000	32	OPB	no	N/A	~~12-13~~405
GPT0_MASK2	32'h 00000000	32	OPB	no	N/A	~~12-13~~405
GPT0_MASK3	32'h 00000000	32	OPB	no	N/A	~~12-13~~405
GPT0_MASK4	32'h 00000000	32	OPB	no	N/A	~~12-13~~405

The third way to reset the GPT core is with the TBCReset signal. It resets only the GPT0_TBC register. Both GPT_chipReset and TBCReset reset the GPT0_TBC register to zero.

12.5.2 Detailed Register Descriptions

The following sections provide a complete bit description of the GPT registers. All registers are accessed from the OPB. The GPT Interrupt Status Register (GPT0_ISS and GPTx_ISC) bits are either set or cleared when written, depending upon which one of two addresses are used. All other registers are both read and write accessible in the normal manner.

12.5.2.1 GPT Time Base Counter Register (GPT0_TBC)

The 32-bit time base is used by the compare timers as a reference for determining event occurrences and for software to use as a general timer.

~~Figure 12-2~~*Figure 12-2* describes GPT0_TBC register bits.

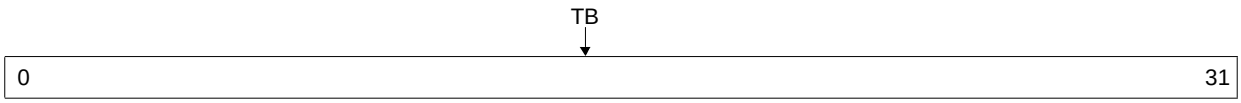


Figure 0-2. Time Base Counter Register (GPT0_TBC)

0:31	TB	Time Base

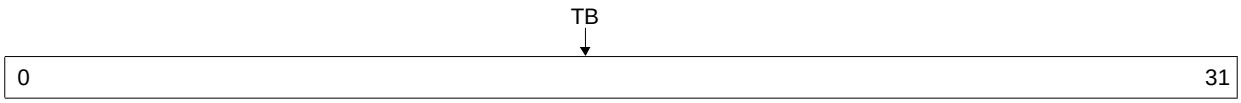


Figure 12-2. Time Base Counter Register (GPT0_TBC)

0:31	TB	Time Base

12.5.2.2 GPT Output Enable Register (GPT0_OE)

Each bit location of the GPT0_OE corresponds to that particular compare timer when the comparison match is satisfied; therefore, if the value of Bit 3 of the GPT0_OE is a 1, then the compare timer #3 level, as specified in the GPT0_OL, is seen when the comparison match is satisfied. All implemented bits reset to zero. ~~Figure 12-3~~*Figure 12-3* describes GPT0_OE register bits.

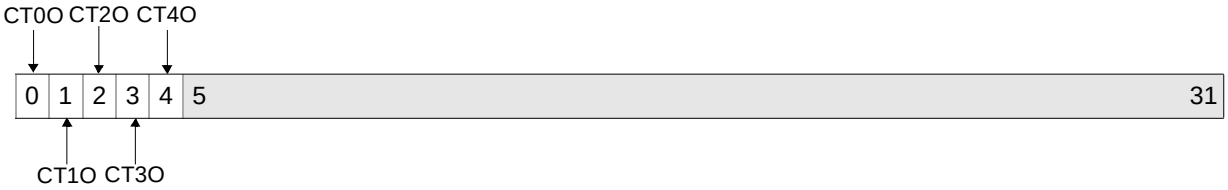


Figure 0-3. GPT Output Enable Register (GPT0_OE)

0	CT0O	Compare Timer 0 Output 0 Compare timer 0 output disabled 1 Compare timer 0 output enabled
1	CT1O	Compare Timer 1 Output 0 Compare timer 1 output disabled 1 Compare timer 1 output enabled
2	CT2O	Compare Timer 2 Output 0 Compare timer 2 output disabled 1 Compare timer 2 output enabled
3	CT3O	Compare Timer 3 Output 0 Compare timer 3 output disabled 1 Compare timer 3 output enabled
4	CT4O	Compare Timer 4 Output 0 Compare timer 4 output disabled 1 Compare timer 4 output enabled
5:31		Reserved

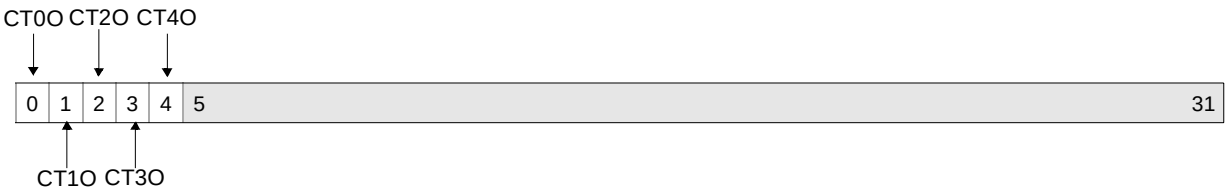


Figure 12-3. GPT Output Enable Register (GPT0_OE)

0	CT0O	Compare Timer 0 Output 0 Compare timer 0 output disabled 1 Compare timer 0 output enabled
1	CT1O	Compare Timer 1 Output 0 Compare timer 1 output disabled 1 Compare timer 1 output enabled

2	CT2O	Compare Timer 2 Output 0 Compare timer 2 output disabled 1 Compare timer 2 output enabled
3	CT3O	Compare Timer 3 Output 0 Compare timer 3 output disabled 1 Compare timer 3 output enabled
4	CT4O	Compare Timer 4 Output 0 Compare timer 4 output disabled 1 Compare timer 4 output enabled
5:31		Reserved

12.5.2.3 GPT Output Level Register (GPT0_OL)

Each bit location of the GPT0_OL corresponds to the level that will be seen for that particular compare timer when the comparison match is satisfied (and the respective compare output is enabled); therefore, the value of Bit 3 of the Compare Timers Output Register is what will be seen on the output of compare timer #3 when the comparison match is satisfied. All implemented bits reset to zero. ~~Figure 12-4~~*Figure 12-4* describes GPT0_OL register bits.

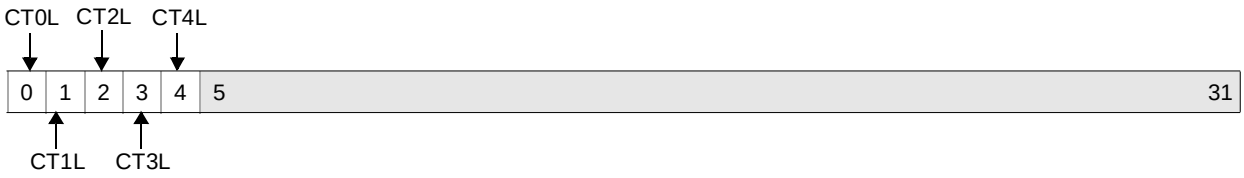


Figure 0-4. GPT Output Level Register (GPT0_OL)

0	CT0L	Compare Timer 0 Output Level 0 Compare timer 0 output level disabled 1 Compare timer 0 output level enabled
1	CT1L	Compare Timer 1 output level 0 Compare timer 1 output level disabled 1 Compare timer 1 output level enabled
2	CT2L	Compare Timer 2 output level 0 Compare timer 2 output level disabled 1 Compare timer 2 output level enabled
3	CT3L	Compare Timer 3 output level 0 Compare timer 3 output level disabled 1 Compare timer 3 output level enabled
4	CT4L	Compare Timer 4 output level 0 Compare timer 4 output level disabled 1 Compare timer 4 output level enabled
5:31		Reserved

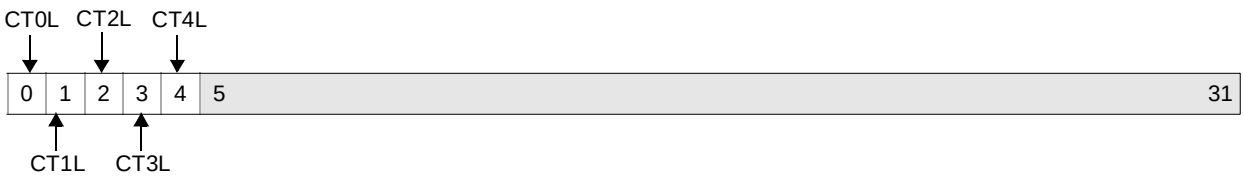


Figure 12-4. GPT Output Level Register (GPT0_OL)

0	CT0L	Compare Timer 0 Output Level 0 Compare timer 0 output level disabled 1 Compare timer 0 output level enabled
1	CT1L	Compare Timer 1 output level 0 Compare timer 1 output level disabled 1 Compare timer 1 output level enabled
2	CT2L	Compare Timer 2 output level 0 Compare timer 2 output level disabled 1 Compare timer 2 output level enabled
3	CT3L	Compare Timer 3 output level 0 Compare timer 3 output level disabled 1 Compare timer 3 output level enabled
4	CT4L	Compare Timer 4 output level 0 Compare timer 4 output level disabled 1 Compare timer 4 output level enabled
5:31		Reserved

12.5.2.4 GPT Interrupt Mask Register (GPT0_IM)

B [16:20] correspond to the compare timer interrupt masks.

Bits in this register mask both the setting of the corresponding GPT0_IS bits and the interrupt output signals, GPT_uicIntrpt. If masked, GPT0_IS bits are not set, and interrupt signals are not generated (even if the GPT0_IE bits are enabled).

For interrupt signals to be active, the GPT0_IM bits must be reset (not masked) and the GPT0_IE bits must be enabled. ~~Figure 12-5~~*Figure 12-5* describes GPT0_IM register bits.

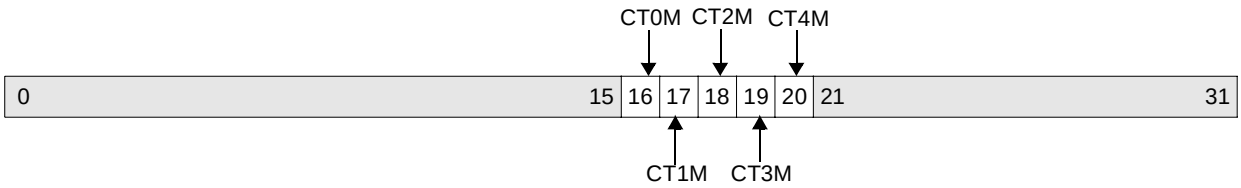


Figure 0-5. GPT Interrupt Mask Register (GPT0_IM)

0:15		Reserved
16	CT0M	Compare Timer 0 Interrupt Mask 0 Compare timer 0 interrupt mask disabled 1 Compare timer 0 interrupt mask enabled
17	CT1M	Compare Timer 1 Interrupt Mask 0 Compare timer 1 interrupt mask disabled 1 Compare timer 1 interrupt mask enabled
18	CT2M	Compare Timer 2 Interrupt Maskl 0 Compare timer 2 interrupt mask disabled 1 Compare timer 2 interrupt mask enabled
19	CT3M	Compare Timer 3 Interrupt Mask 0 Compare timer 3 interrupt mask disabled 1 Compare timer 3 interrupt mask enabled
20	CT4M	Compare Timer 4 Interrupt Mask 0 Compare timer 4 interrupt mask disabled 1 Compare timer 4 interrupt mask enabled
21:31		Reserved

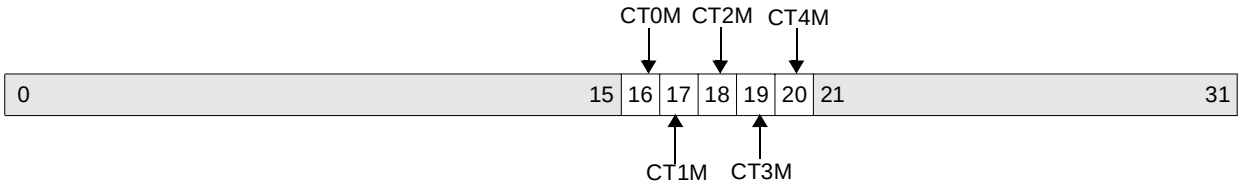


Figure 12-5. GPT Interrupt Mask Register (GPT0_IM)

0:15		Reserved
16	CT0M	Compare Timer 0 Interrupt Mask 0 Compare timer 0 interrupt mask disabled 1 Compare timer 0 interrupt mask enabled
17	CT1M	Compare Timer 1 Interrupt Mask 0 Compare timer 1 interrupt mask disabled 1 Compare timer 1 interrupt mask enabled
18	CT2M	Compare Timer 2 Interrupt Maskl 0 Compare timer 2 interrupt mask disabled 1 Compare timer 2 interrupt mask enabled
19	CT3M	Compare Timer 3 Interrupt Mask 0 Compare timer 3 interrupt mask disabled 1 Compare timer 3 interrupt mask enabled
20	CT4M	Compare Timer 4 Interrupt Mask 0 Compare timer 4 interrupt mask disabled 1 Compare timer 4 interrupt mask enabled
21:31		Reserved

12.5.2.5 GPT Interrupt Status Register (GPT0_ISS and GPT0_ISC)

Bits [16:20] correspond to the compare timer interrupt status.

The GPT Interrupt status bits for the compare timers are set when a valid comparison is made, and the compare interrupt is enabled.

GPT0_ISS can be accessed through address offset 1C, which provides a normal read access and a "Write-Set" access, allowing individual status bits to be set through a write access. Any status bits written to 1 are set (forced to 1), while bits written to 0 remain unchanged (0 or 1).

Offset 20 (GPT0_ISC) provides a normal read access and a "Write-Clear" access, which allows individual status bits to be reset through a write access. Any status bits written to 1 are cleared (forced to 0), while bits written to 0 remain unchanged (0 or 1).

All Interrupt Status Register bits reset to "0". ~~Figure 12-6~~*Figure 12-6* describes GPT0_ISS and GPT0_ISC bits.

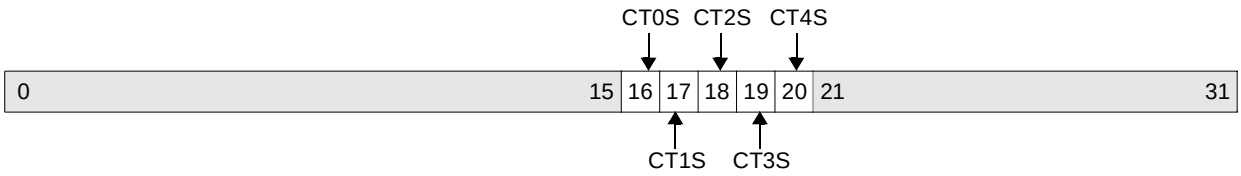


Figure 0-6. GPT Interrupt Status Register (GPT0_ISS and GPT0_ISC)

0:15		Reserved
16	CT0S	Compare Timer 0 Interrupt Status 0 Compare timer 0 interrupt status disabled 1 Compare timer 0 interrupt status enabled
17	CT1S	Compare Timer 1 Interrupt Status 0 Compare timer 1 interrupt status disabled 1 Compare timer 1 interrupt status enabled
18	CT2IS	Compare Timer 2 Interrupt Status 0 Compare timer 2 interrupt status disabled 1 Compare timer 2 interrupt status enabled
19	CT3S	Compare Timer 3 Interrupt Status 0 Compare timer 3 interrupt status disabled 1 Compare timer 3 interrupt status enabled
20	CT4S	Compare Timer 4 Interrupt Status 0 Compare timer 4 interrupt status disabled 1 Compare timer 4 interrupt status enabled
21:31		Reserved

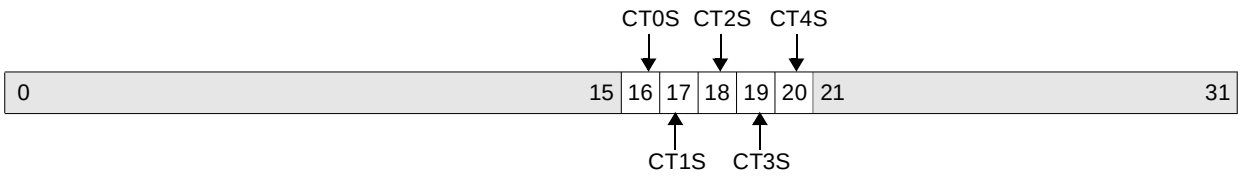


Figure 12-6. GPT Interrupt Status Register (GPT0_ISS and GPT0_ISC)

0:15		Reserved
16	CT0S	Compare Timer 0 Interrupt Status 0 Compare timer 0 interrupt status disabled 1 Compare timer 0 interrupt status enabled
17	CT1S	Compare Timer 1 Interrupt Status 0 Compare timer 1 interrupt status disabled 1 Compare timer 1 interrupt status enabled
18	CT2IS	Compare Timer 2 Interrupt Status 0 Compare timer 2 interrupt status disabled 1 Compare timer 2 interrupt status enabled
19	CT3S	Compare Timer 3 Interrupt Status 0 Compare timer 3 interrupt status disabled 1 Compare timer 3 interrupt status enabled
20	CT4S	Compare Timer 4 Interrupt Status 0 Compare timer 4 interrupt status disabled 1 Compare timer 4 interrupt status enabled
21:31		Reserved

12.5.2.6 GPT Interrupt Enable Register (GPT0_IE)

Bits [16:20], correspond to the compare timer interrupt enable bits.

When set, GPT0_IE bits prevent the corresponding compare interrupts from activating the GPT_uicIntrpt outputs, even if the interrupt mask (GPT0_IM) bits are set; however, these bits have no affect on the corresponding interrupt status (GPT0_IS) bits.

All bits reset to "1" (Interrupt enabled). ~~Figure 12-7~~*Figure 12-7* describes GPT0_IE register bits.

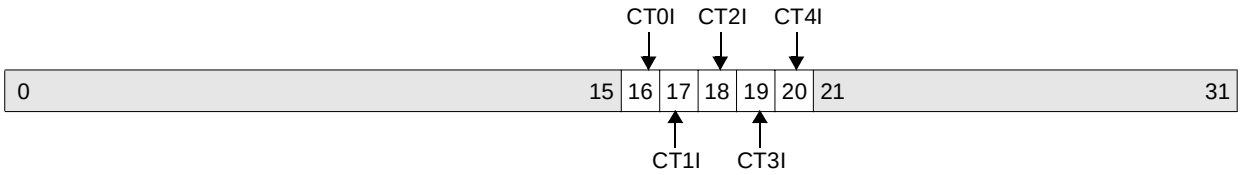


Figure 0-7. GPT Interrupt Enable Register (GPT0_IE)

0:15		Reserved
16	CT0I	Compare Timer 0 Interrupt Enable 0 Compare timer 0 interrupt enable disabled 1 Compare timer 0 interrupt enable enabled
17	CT1I	Compare Timer 1 Interrupt Enable 0 Compare timer 1 interrupt enable disabled 1 Compare timer 1 interrupt enable enabled
18	CT2I	Compare Timer 2 Interrupt Enable 0 Compare timer 2 interrupt enable disabled 1 Compare timer 2 interrupt enable enabled
19	CT3I	Compare Timer 3 Interrupt Enable 0 Compare timer 3 interrupt enable disabled 1 Compare timer 3 interrupt enable enabled
20	CT4I	Compare Timer 4 Interrupt Enable 0 Compare timer 4 interrupt enable disabled 1 Compare timer 4 interrupt enable enabled
21:31		Reserved

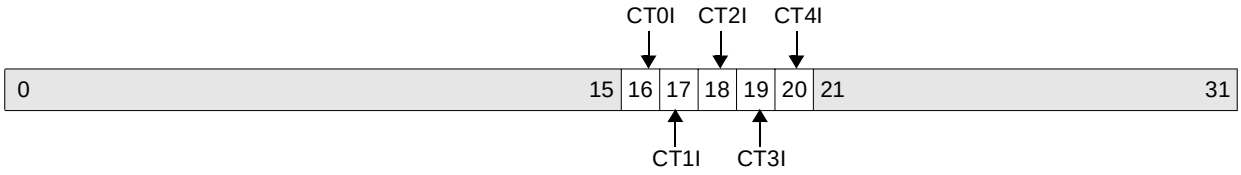


Figure 12-7. GPT Interrupt Enable Register (GPT0_IE)

0:15		Reserved
16	CT0I	Compare Timer 0 Interrupt Enable 0 Compare timer 0 interrupt enable disabled 1 Compare timer 0 interrupt enable enabled
17	CT1I	Compare Timer 1 Interrupt Enable 0 Compare timer 1 interrupt enable disabled 1 Compare timer 1 interrupt enable enabled
18	CT2I	Compare Timer 2 Interrupt Enable 0 Compare timer 2 interrupt enable disabled 1 Compare timer 2 interrupt enable enabled
19	CT3I	Compare Timer 3 Interrupt Enable 0 Compare timer 3 interrupt enable disabled 1 Compare timer 3 interrupt enable enabled
20	CT4I	Compare Timer 4 Interrupt Enable 0 Compare timer 4 interrupt enable disabled 1 Compare timer 4 interrupt enable enabled
21:31		Reserved

12.5.2.7 GPT Compare Timer Registers (GPT0_COMP0 - GPT0_COMP4)

Each GPT0_COMPn is programmed with the value that is continually compared to the TBC value, as filtered through each MASK register. The width of each GPT0_COMPn is 32 bits. ~~© Copyright IBM Corp. 2000, 2002~~*Figure 12-8* describes GPT0_COMPn register bits. All GPT0_COMPn bits reset to zero.

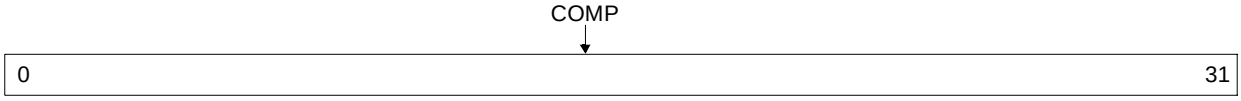


Figure 0-8. Compare Timer Register (GPT0_COMP0 - GPT0_COMP4)

0:31	COMP	Compare Timer

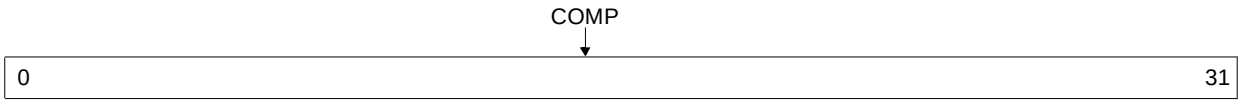


Figure 12-8. Compare Timer Register (GPT0_COMP0 - GPT0_COMP4)

0:31	COMP	Compare Timer

12.5.2.8 GPT Compare Mask Registers (GPT0_MASK0 - GPT0_MASK4)

The GPT0_MASKn bits are used by the compare timers to mask off the comparison (i.e., force a valid compare) of individual bits when the comparison function is performed. For bits that are set, a valid compare is always assumed, regardless of the actual value of these bits in the GPT0_COMPn or GPT0_TBC registers. The width of each implemented Mask Register is 32 bits. ~~Figure 12-9~~*Figure 12-9* describes GPT0_MASKn register bits.

All GPT0_MASKn bits reset to "0" (No mask).

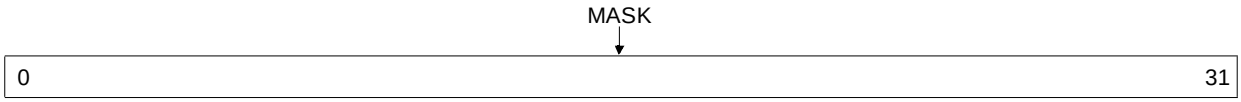


Figure 0-9. Compare Mask Register (GPT0_MASK0 - GPT0_MASK4)

0:31	MASK	Comparison Function 0 Comparison enabled 1 Comparison disabled	When set to 1, a valid comparison is assumed.

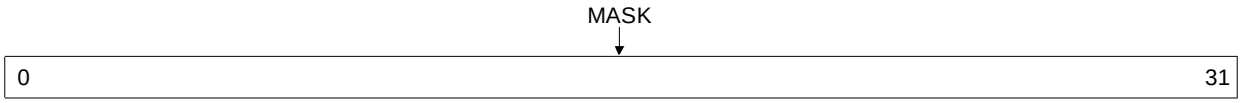


Figure 12-9. Compare Mask Register (GPT0_MASK0 - GPT0_MASK4)

0:31	MASK	Comparison Function 0 Comparison enabled 1 Comparison disabled	When set to 1, a valid comparison is assumed.

13. Clocking

Clocking in the PPC440GP is highly configurable and supports a wide range of clock ratios on the internal and external buses. PPC440GP system configuration registers (CPC0_SYS0 and CPC0_SYS1) allow for flexible power-on configuration using the IIC bootstrap controller, as well as for later adjustment after the PPC440GP has begun operation. Maximum performance at different operating frequencies is possible through the support of non integral relative frequencies for CPU and PLB clocks, including but not limited to ratios like 3:1, 4:1, 5:2 and 7:2.

Two PLLs are used in the PPC440GP which support both System and PCI clocking:

1. System PLL - source for CPU and PLB clocks, and indirectly the DDR SDRAM, OPB, serial and external bus clocks
2. PCI PLL - source for PCI clock used by the PCI interface

The control and configuration necessary for each PLL is described in the following sections. Be careful to properly filter both the analog Vdd and GND inputs used by each PLL as described in detail in PPC440GP datasheet.

PLL operation is controlled by several registers, including Forward Divide (Range A/B), Feedback Divide (FBDV), Tuning (TUNE), and both input reference and feedback clocks. The Voltage Controlled Oscillator (VCO) contained within each PLL is required to operate in the range from 500-1000+MHz. The exact value of Forward and Feedback divide ratios will affect the VCO operating frequency and the frequency generated at PLLOUTA and PLLOUTB.

13.1 System Clocking

Figure 13-1 illustrates the clocks related to general system operation of the PPC440GP. Notice that the CPU frequency is directly generated by PLLOUTA, while the PLB frequency is directly generated by PLLOUTB. When in Bypass mode, PLLOUTA and PLLOUTB are rebuffered versions of SysClk.

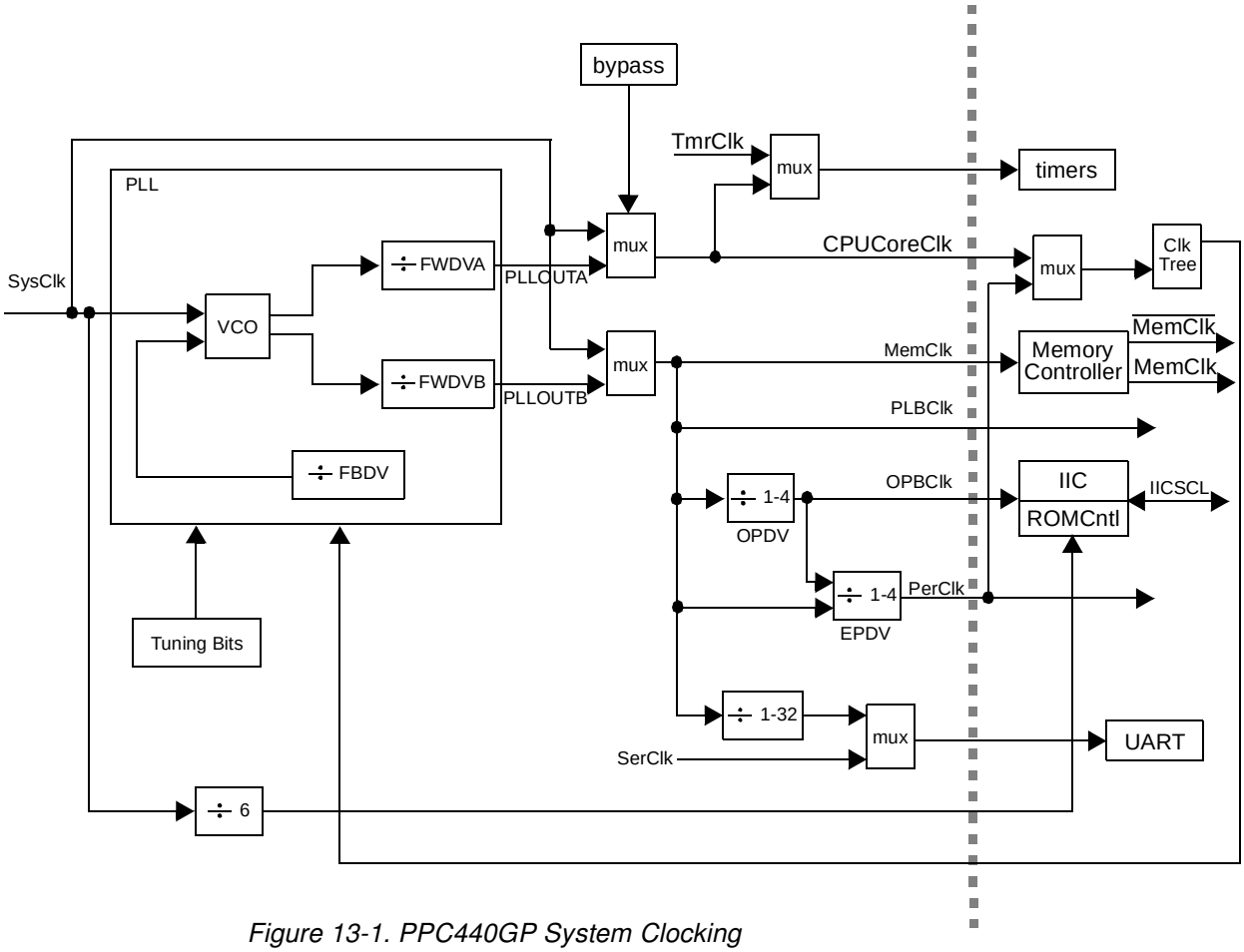


Figure 13-1. PPC440GP System Clocking

13.1.1 Input SysClk

The system clock provided to the PPC440GP must be a minimum of 25MHz in order for the on-chip PLL to achieve a stable lock. An input clock above 66.6MHz is not supported, and the provided clock must be clean and have relatively low jitter. See PPC440GP data sheet for details.

13.1.2 Feedback Selection

The PPC440GP system clocking logic is designed to use either of two feedback paths for the SYS PLL, using CPC0_SYS0[EXTSL]. Most users will find that CPU clock feedback provides the greatest flexibility in choosing CPU and PLB clock ratios. Users who require that the external bus clock, PerClk be both frequency and phase aligned with SysClk, however, will need to use PerClk for feedback. Phase alignment between

SysClk and ExtBus clock permits the use of the external master interface with external masters that cannot use the PPC440GP ExtBus clock for synchronization. These external masters will normally source the external bus clock themselves and provide it to the PPC440GP SysClk input.

For PerClk and SysClk to be phase aligned the clock dividers must be selected such that the frequencies are the same. Doing so requires that the CPC0_SYS0[FBDV] bits be set to a divide by one, and that PLB and other clocks derived from PLLOUTB are a multiple of PerClk. In the absence of any system requirement for phase alignment between PerClk and SysClk, users are encouraged to choose the CPU clock for feedback.

13.1.3 VCO Frequency and 'M' Value for SYS PLL

For any acceptable input SysClk frequency, the SYS PLL VCO frequency is set by the total product of divider circuits used in the path from VCO output back to VCO input, multiplied by the system clock frequency. The product of the divider circuits, both inside the PLL and in the external divider circuits like those which generate the PLB, OPB and PerClk, is referred to as the'M' multiplier value. Knowing this value is necessary in order to properly set the TUNE bits.

Note: The PerClk divider is only included in the M calculation for SYS PLL if PerClk feedback is selected. When CPU clock feedback is selected (default) there are no external dividers in the feedback path.

As an example of the effect various settings have on clock frequencies and the M multiplier, consider the System PLL configured first with CPU feedback as shown in Table 13-1 and then with PerClk feedback as shown in Table 13-2.

Table 13-1. System PLL Configuration Using CPU Feedback

PLL Settings	Resulting Configuration
SYSCLK = 33.3MHz	VCO = 800MHz
FWDVA = 2	CPU = 400MHz
FWDVB = 6	PLB = 133.3MHz
OPDV = don't care	OPB = don't care
EPDV = don't care	PerClk = don't care
FBDV = 12	M = 24
M = 24	TUNE = 1000111110

Table 13-2. System PLL Configuration Using PerClk Feedback

PLL Settings	Resulting Configuration
SYSCLK = 33.3MHz	VCO = 800MHz
FWDVA = 2	CPU = 400MHz
FWDVB = 6	PLB = 133.3MHz
OPDV = 2	OPB = 66.6 MHz
EPDV = 2	PerClk = 33.3 MHz
FBDV = 1 (must be 1)	M = 24
M = 24	TUNE = 1000111110

The M multiplier for the SYS PLL can be calculated using either of the following equations, depending upon the choice made for feedback as shown in Table 13-3.

Table 13-3. Clock Frequencies and M multiplier

Feedback Choice	M multiplier equation
CPU clock	M = FBDV x FWDVA
PerClk	M = EPDV x OPDV x FWDVB

13.1.4 SYS PLL TUNE Setting

The SYS PLL must be provided with different 'tune' bit settings based upon the VCO frequency and 'M' value which result from the system configuration. Table 13-4 describes how to determine the proper CPC0_SYS0[TUNE] bit settings for appropriate application.

Table 13-4. CPC0_SYS0[TUNE] Bit Settings

'M' Value Range	VCO Frequency Range	CPC0_SYS0[TUNE]									
		0	1	2	3	4	5	6	7	8	9
$6 < M \leq 10$		0	1	0	0	1	1	1	0	0	0
$10 < M \leq 14$	$500MHz < VCO \leq 800MHz$	0	1	0	0	1	1	1	1	0	0
$10 < M \leq 14$	$800MHz < VCO \leq 1000MHz$	0	1	1	0	1	1	1	1	0	0
$14 < M \leq 40$	$500MHz < VCO \leq 800MHz$	1	0	0	0	1	1	1	1	1	0
$14 < M \leq 40$	$800MHz < VCO \leq 1000MHz$	1	0	1	0	1	1	1	1	1	0
$14 < M \leq 40$	$1000MHz < VCO \leq 1333MHz$	1	0	1	1	1	1	1	1	1	0

13.1.5 IIC Bootstrap Controller Clocking

SysClk is divided down by 6 to provide the IIC bootstrap controller a clock with a maximum frequency not greater than 12.8 MHz. This clock is internal to the PPC440GP and cannot be adjusted. ~~See "IIC Bootstrap Controller" on page 8-1.~~ *IIC Bootstrap Controller* on page 283 In addition, the IIC bootstrap controller further divides its clock by 128 so that the maximum frequency at the serial ROM is 100 KHz.

13.1.6 Clocks For Off Chip Use

The DDR memory controller generates a differential MemClk to the DDR SDRAM chips (DQS) at PLB frequency. The PerClk is generated by the system clocking logic for use by an external bus master or other synchronous device. The UART serial clock can be generated either from an internal clock divider circuit or from a chip input. This clock is presented internally to the UART core, which can be further subdivided.

13.1.7 SYS PLL Strapping

System clocking is primarily controlled by the CPC0_SYS0 register. This register and the companion CPC0_SYS1 register are normally initialized at power on time using the IIC bootstrap controller and an external serial ROM. The CPC0_CUST0 and CPC0_CUST1 are also initialized during this process. If a serial ROM is not present at power on then the pre-programmed defaults will be used.

If it becomes necessary after power on to alter a setting in CPC0_SYS0 register, it is recommended that the setting of the CPC0_CR0[SWE] bit be changed in order to allow DCR writes. Normal DCR writes can then be used to update CPC0_SYS0, although bit fields which affect the SYSPLL will not go into effect until a chip reset is initiated. An interlock mechanism posts writes to the CPC0_SYS0 register so that they will only go into effect after a chip reset. A chip reset can be generated by using the debug control register (DBCR0[2:3]) bits

System reset always re-loads the CPC0_SYS0, CPC0_SYS1, CPC0_CUST0 and CPC0_CUST1 registers using the values programmed into the serial bootstrap ROM.

A number of different bit values in the CPC0_SYS0 register are required to correctly configure the chip in a manner that will allow reliable operation of SYS PLL and clock divider circuitry. When setting these values be careful to avoid accidentally configuring the PPC440GP in an unusable state. See the following section for assistance in selecting acceptable system clocking divider ratios.

13.1.8 PLL Bypass (Emulation Mode)

The CPC0_SYS0[BYPASS] bit is provided to disable the SYS PLL if it is intentionally being used outside of its operating range. The most common reason for using this mode is in emulation systems used in product development, where the SYS CLK input to PPC440GP is set to a frequency well below 25MHz, perhaps as low as 1MHz. Forcing the SYS PLL into bypass mode prevents the PLL from attempting to lock, and may allow the PPC440GP to operate well outside of its supported range. When in Bypass mode PLLOUTA and PLLOUTB are rebuffered versions of SysClk.

13.1.9 CPU / PLB Frequency N:1 Setting

The PPC440GP clocking logic allows for CPU:PLB clock frequencies which are in the ratio of N:X, where N = {1,2,...,etc} and X = {1,2}. The CPU clock is considered to be an integral multiple of PLB when X = 1, as in the example where CPU = 400 and PLB = 133 (N:X = 3:1). A non-integral CPU to PLB relationship can be seen when X = 2, as in the example where CPU = 333 and PLB =133 (N:X = 5:2). These ratios are set by the values chosen for the divisors in the clocking tree. See ~~"System Clock Ratio Examples" on page 13-6~~*System Clock Ratio Examples* on page 412 to determine valid CPU and PLB frequencies and their resultant CPU:PLB, N:X, ratio.Table 13-5 indicates the results of programming CPC0_SYS0[Nto1] to either 0 or 1, depending upon whether the CPU:PLB ratio is N:1.

Table 13-5. CPC0_SYS0[Nto1] Settings

CPU:PLB Ratio	CPC0_SYS0[Nto1] = 0	CPC0_SYS0[Nto1] = 1
CPU:PLB Ratio is N:1 (X={1})	Slight performance loss as unnecessary clock resynchronization occurs at CPU:PLB interface	Optimal performance for N to 1 mode
CPU:PLB Ratio is not N:1 (X={2})	Optimal performance for non N to 1 mode	System failure can occur

13.1.10 Choosing System Clock Ratios

Table 13-6 describes the rules that apply when choosing acceptable system clock ratios and frequencies for the PPC440GP. Please see a current datasheet as different PPC440GP revisions may have different limits than those described here.

Table 13-6. System Clock Rules

Rule	Value
CPU frequency must be less than or equal to	Max CPU frequency - see PPC440GP datasheet
PLB frequency must be less than or equal to	Max PLB frequency - see PPC440GP datasheet
CPU:PLB ratio is N:X with {N,X} being integer and X less than or equal to	2
VCO frequency must be greater than or equal to	500MHz
VCO frequency must be less than or equal to	Max supported VCO - see PPC440GP datasheet

Table 13-7 describes the equations that can be used to determine the resulting VCO, CPU, PLB frequencies based on selected FWDVA, FWDVB and FBDV settings in CPC0_SYS0 register.

Table 13-7. Equations to Determine VCO, CPU, PLB Frequency

Feedback Selection	Equations
CPU clock	M = FBDV x FWDVA VCO = SysClk x M CPU = VCO / FWDVA PLB = (VCO / FWDVB)
PerClk	M = EPDV x OPDV x FWDVB VCO = SysClk x M CPU = VCO / FWDVA PLB = VCO / FWDVB

13.1.11 System Clock Ratio Examples

Example tables are provided in the following sections for the case where SysClk = 33.3MHz and either CPU feedback or PerClk feedback are used. These tables describe all possible combinations of clocking ratios which can be achieved using the bit values in CPC0_SYS0.

13.1.11.1 CPU Feedback Example

As shown in Table 13-6, when using CPU feedback, M, VCO and CPU can be determined once specific values for the FWDVA and FBDV dividers are chosen. Therefore, the CPU feedback table is organized so that each row represents a specific combination of FWDVA and FBDV divider. From these specified values, M, VCO and CPU are known and listed.

For each combination of these dividers, there are up to seven different columns on the right hand of the table which indicate the possible PLB frequencies that result from the VCO frequency determined for that row, depending upon a specific FWDVB divider value.

Entries that result from obviously unusable configurations are left empty, including those that result in VCO frequencies below 500MHz or above 1333MHz, and excessive CPU and/or PLB frequencies. In several cases the table will indicate identical CPU and PLB frequencies, however in each case there will be a different VCO frequency that results.

Note: Table 13-8 provides clock ratio listing with CPU feedback, SysClk = 33.3MHz with the following frequencies: 500MHz = <VCO = <1000MHz, CPU = <400MHz, PLB = <133.3MHz for PPC440GP. Also since PPC440GP uses DDRSDRAM, the PLB must operate between 100MHz to 133.3MHz with supported CPU:PLB ratios of 5:2, 3:1, 7:2 and 4:1. See the PPC440GP data sheet for max PLB, CPU and VCO frequencies.

Table 13-8. Clock Ratio Listing With CPU Feedback, SysClk = 33.3MHz

FWDVB ====>					2	3	4	5	6	7	8
CPU:PLB Ratio====>					2:1	3:1	4:1	5:1	6:1	7:1	8:1
FWDVA	FBDV	M	VCO	CPU	PLB Clock Frequency (MHz)						
1	1-14	-	<500	-							
1	15-16	-	-	>400.0							
FWDVB ====>					2	3	4	5	6	7	8
CPU:PLB Ratio====>					1:1	3:2	2:1	5:2	3:1	7:2	4:1
FWDVA	FBDV	M	VCO	CPU	PLB Clock Frequency (MHz)						
2	1-7	-	<500	-							
2	8	16	533.3	266.7				106.7			
2	9	18	600.0	300.0				120.0	100.0		
2	10	20	666.7	333.3				133.3	111.1		
2	11	22	733.3	366.7					122.2	104.8	
2	12	24	800.0	400.0					133.3	114.3	100.0
2	13-16	-	-	>400							

13.1.11.2 PerClk Feedback Example

As shown in the System Clock Rules table, when using PerClk feedback, M, VCO and PLB can be determined once specific values for the FWDVB, EPDV and OPDV dividers are chosen. Therefore, the PerClk feedback table is organized so that each row represents a specific combination of FWDVB and EPDV/OPDV dividers. To simplify the table, the product EPDV x OPDV is presented as a single value. From these specified values, M, VCO and CPU are known and listed.

Since both EPDV and OPDV can each vary from a divide by 1 up to a divide by 4, the possible values are 1, 2, 3, 4, 6, 8, 9,12 and 16. Furthermore, when using external bus feedback, PerClk has the same frequency as SysClk. This in turn implies that PLB = SysClk x ExtDiv. Table 13-9 demonstrates this fact by listing possible PLB frequencies which are all multiples of SysClk (that is, 66.6MHz, 100MHz, and 133.3MHz, assuming SysClk = 33.3MHz.

For each combination of these dividers, there are up to eight different columns on the right hand of the table which indicate the possible CPU frequencies that result from the VCO frequency determined for that row, depending upon a specific FWDVA divider value.

Entries that result from obviously unusable configurations are left empty, including those that result in VCO frequencies below 500MHz or above 1333MHz, and excessive CPU and/or PLB frequencies. In several cases the table will indicate identical CPU and PLB frequencies, however in each case there will be a different VCO frequency that results.

Note: Table 13-9 provides clock ratio listing with PerClk feedback, SysClk = 33.3MHz. The values shown are valid for parts with the following maximum frequencies: CPU = 400MHz, VCO = 1000MHz, PLB =133MHz and OPB = 66MHz as described in PPC440GP data sheet. Similar tables can be constructed for other SysClk frequencies and different frequency maximums as described in PPC440GP data sheet.

Table 13-9. Clock Ratio Listing With PerClk Feedback, SysClk = 33.3MHz

FWDVA ====>					1	2	3	4	5	6	7	8
EPDVxOPDV	FWDVB	M	VCO	PLB	CPU Clk Frequency							
1	1-8	x	<500	-	-	-	-	-	-	-	-	-
2	1-7	x	<500	-	-	-	-	-	-	-	-	-
2	8	16	533	<100	-							
3	1-4	x	<500	-	-	-	-	-	-	-	-	-
3	5	15	500	100.0	-	250.0	166.7	125.0	100.0	-	-	-
3	6	18	600	100.0	-	300.0	200.0	150.0	120.0	100.0	-	-
3	7	21	700	100.0	-	350.0	233.3	175.0	140.0	116.7	100.0	-
3	8	24	800	100.0	-	400.0	266.7	200.0	160.0	133.3	114.3	100.0
4	1:3	x	<500	-	-	-	-	-	-	-	-	-
4	4	16	533	133.3	-	266.7	177.8	133.3	-	-	-	-
4	5	20	666.7	133.3	-	333.3	222.2	166.7	133.3	-	-	-
4	6	24	800	133.3	-	400.0	266.7	200.0	186.7	160.0	133.3	-
4	7	28	933.3	133.3	-		311.1	233.3	186.7	186.7	155.6	133.3-
4	8	32	<1000	-	-	-	-	-	-	-	-	-
6 8 9 12 16	x	x	x	>133.3	-	-	-	-	-	-	-	-

Table 13-10 shows the CPU:PLB ratios which results from the indicated FWDVA and FWDVB dividers when using PerClk feedback when SysClk is 33MHz. Several ratios are left blank because they are illegal.

Table 13-10. CPU:PLB Ratio

FWDVB	FWDVA Divider							
	1	2	3	4	5	6	7	8
1	1:1							
2	2:1	1:1						
3	3:1	3:2	1:1					
4	4:1	2:1		1:1				
5	5:1	5:2			1:1			
6	6:1	3:1	2:1	3:2		1:1		
7	7:1	7:2					1:1	
8	8:1	4:1		2:1				1:1

13.2 PCI Clocking

Figure 13-2 illustrates the PCI clock generation logic. The PCI PLL is used for phase alignment in all PCI modes except in conventional 0-33MHz, a mode where the PCI PLL is bypassed. Unlike system clocking logic, the PCI PLL divider circuitry cannot be directly controlled by the user. Instead, the PCI core logic determines the proper settings, including settings for selectable feedback delays, based on PCI protocol and the PCIX frequency selection (CPC0_SYS1[PXFS]) setting. The PCI PLL serves the purpose of compensating for the internal clock tree delay of the PCI clock (phase alignment) and not for clock multiplication or other frequency adjustment.

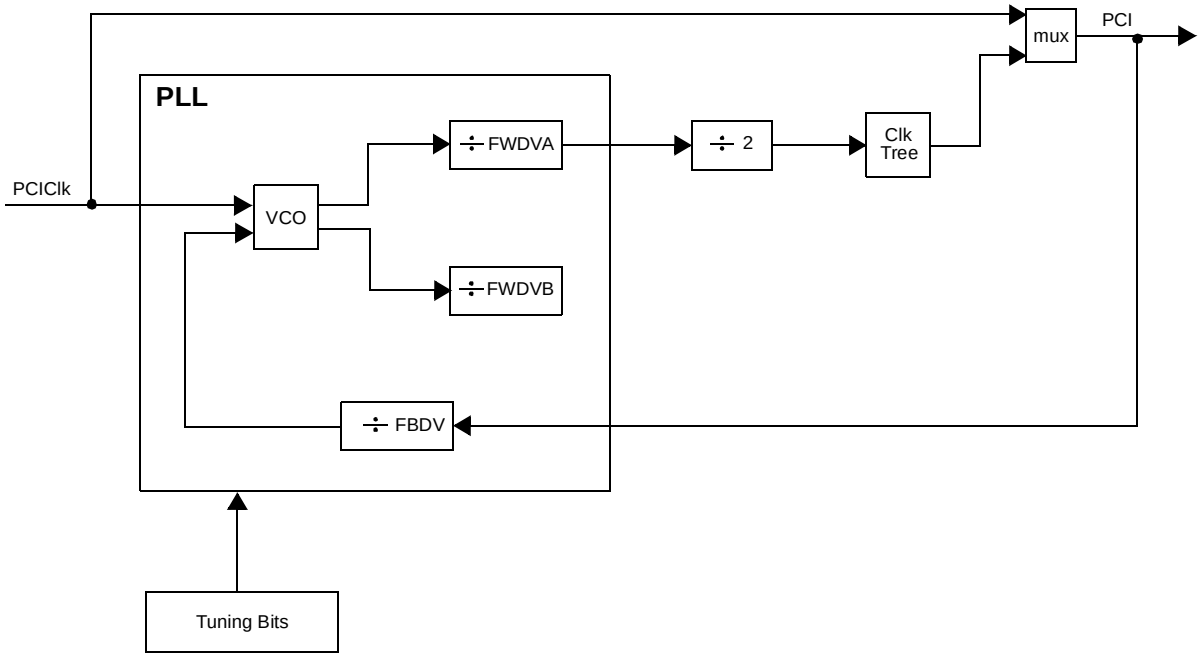


Figure 13-2. PCI Clocking

13.2.1 Input PCI CLK

The PCI clock provided to the PPC440GP must be in the range from 0 - 133MHz maximum. The provided clock must be clean and have relatively low jitter. See PPC440GP data sheet for details.

13.2.2 Support PCI Modes and 'M' value for PCI PLL

Table 13-11 describes the PCI modes supported in PPC440GP. The values in this table are automatically determined by the PCI core logic using PCI protocol during power-on reset, and cannot be configured by the user.

Table 13-11. Supported PCI Modes

Mode	Range (MHz)	M	VCO	FWD DIV	PLL OUTA	EXT DIV	PLL Feedback Type
Conventional Mode (1)	0-33	1	n/a	n/a	0-33	n/a	PLL pass thru
Conventional Mode (2)	33-66	16	533-1066	8	66-133	2	Worst case clock delay is (1.2ns to 1.8ns)
PCI-X (1)	50-66	16	800-1066	8	100-133	2	Worst case clock delay is (-0.3 ns to 0.3 ns)
PCI-X (2)	66-100	10	666-1000	5	133-200	2	Same as previous
PCI-X (3)	100-133	6	600-800	3	200-266	2	Same as previous

The 'M' multiplier is calculated for each PCI mode and is needed when setting the PCI PLL Tune bits.

13.2.3 PCI PLL TUNE Setting

The PCI PLL must be provided with different TUNE bit settings based upon the VCO frequency and 'M' value, both of which result from the mode calculation as determined in Table 13-11. By default, the PCI interface will dynamically determine appropriate TUNE bit settings during power-on as shown in Table 13-12.

Table 13-12. Dynamically Determined PCI TUNE Bit Settings

Mode	PCI PLL TUNE[0:9]
Conventional mode (1)	NA
Conventional mode (2)	10_1011_1010
PCI-X (1)	10_1011_1110
PCI-X (2)	01_1011_1000
PCI-X (3)	01_0011_0100

The CPC0_SYS1[TUNE] register field is available for overriding the default TUNE bits which were automatically calculated by the PCI interface. Figure 13-3 shows how CPC0_SYS1[TUNEEN] controls whether the calculated or override values are used by the PCI PLL.

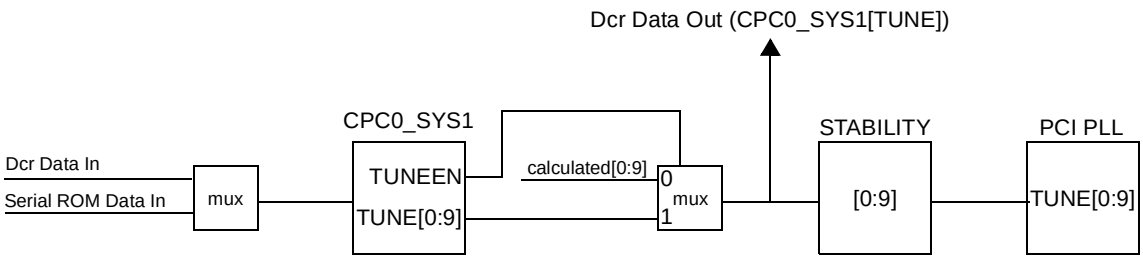


Figure 13-3. Overriding the Default TUNE Bits

In order to override the calculated TUNE bits, an appropriate TUNE bit value needs to be programmed into the serial ROM for CPC0_SYS1[TUNE], along with a '1' for CPC0_SYS1[TUNEEN]. Alternatively, once the PPC440GP is running, it is possible to override the TUNE bits with a DCR write for these same register bits followed by the execution of a chip reset to allow the changes to be propagated to the PCI PLL.

Table 13-13 can be used to select override TUNE bit values for the unlikely scenario where the calculated TUNE bits are not appropriate.

Table 13-13. CPC0_SYS1[TUNE] Bit Settings

'M' Value Range	VCO Frequency Range	CPC0_SYS0[TUNE]									
		0	1	2	3	4	5	6	7	8	9
$6 < M \leq 10$		0	1	0	0	1	1	1	0	0	0
$10 < M \leq 14$	500MHz < VCO ≤ 800MHz	0	1	0	0	1	1	1	1	0	0
$10 < M \leq 14$	800MHz < VCO ≤ 1000MHz	0	1	1	0	1	1	1	1	0	0
$14 < M \leq 40$	500MHz < VCO ≤ 800MHz	1	0	0	0	1	1	1	1	1	0
$14 < M \leq 40$	800MHz < VCO ≤ 1000MHz	1	0	1	0	1	1	1	1	1	0
$14 < M \leq 40$	1000MHz < VCO ≤ 1333MHz	1	0	1	1	1	1	1	1	1	0

13.3 Clocking Registers

Table 13-14 summarizes the device control registers that control clocking in PPC440GP.

Table 13-14. Clocking Control Registers

Mnemonic	Register	Address	Access	Page
CPC0_SYS0	System Configuration Register 0	0x0E0	R/W	~~13-12~~4 18
CPC0_SYS1	System Configuration Register 1	0x0E1	R/W	~~13-14~~4 20

13.3.1 System Configuration Register 0 (CPC0_SYS0)

CPC0_SYS0 is a 32-bit read/write version of CPC0_STRP0. All bits in this register at initial power on reset system are either default strap values or the values read from the bootstrap serial ROM shown in ~~"Default Strap Settings" on page 8-2~~*Default Strap Settings* on page 284.

Note: Writes to this register are ignored unless CPC0_CR0[SWE] = 1. Also writes to these fields do not affect either SYS or PCI PLL until a CHIP reset occurs.

Figure 13-4 describes CPC0_SYS0 bit definitions.

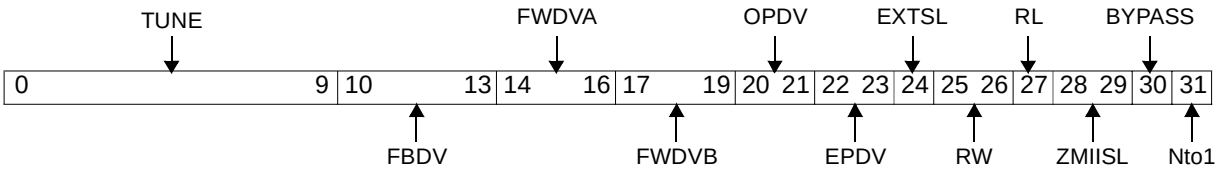


Figure 0-1. System Configuration Register 0 (CPC0_SYS0)

0:9	TUNE	System PLL TUNE bits	See Table 13-5, "CPC0_SYS0[Nto1] Settings," on page 13-5 for tune bit setting information.
10:13	FBDV	PLL Feedback Divisor 0000 PLL Feedback Divisor = 16 0001 PLL Feedback Divisor = 1 0010 PLL Feedback Divisor = 2 0011 PLL Feedback Divisor = 3 0100 PLL Feedback Divisor = 4 0101 PLL Feedback Divisor = 5 0110 PLL Feedback Divisor = 6 0111 PLL Feedback Divisor = 7 1000 PLL Feedback Divisor = 8 1001 PLL Feedback Divisor = 9 1010 PLL Feedback Divisor = 10 1011 PLL Feedback Divisor = 11 1100 PLL Feedback Divisor = 12 1101 PLL Feedback Divisor = 13 1110 PLL Feedback Divisor = 14 1111 PLL Feedback Divisor = 15	

14:16	FWDVA	PLL Forward Divisor A 000 PLL Forward Divisor A = 8 001 PLL Forward Divisor A = 7 010 PLL Forward Divisor A = 6 011 PLL Forward Divisor A = 5 100 PLL Forward Divisor A = 4 101 PLL Forward Divisor A = 3 110 PLL Forward Divisor A = 2 111 PLL Forward Divisor A = 1	
17:19	FWDVB	PLL Forward Divisor B 000 PLL Forward Divisor B = 8 001 PLL Forward Divisor B = 7 010 PLL Forward Divisor B = 6 011 PLL Forward Divisor B = 5 100 PLL Forward Divisor B = 4 101 PLL Forward Divisor B = 3 110 PLL Forward Divisor B = 2 111 PLL Forward Divisor B = 1	
20:21	OPDV	PLB to OPB Clock Divisor Ratio 00 PLB to OPB Clock Divisor = 1 01 PLB to OPB Clock Divisor = 2 10 PLB to OPB Clock Divisor = 3 11 PLB to OPB Clock Divisor = 4	The OPDV is the divisor ratio between the PLB and OPB clock. For example, if the PLB clock frequency is 133MHz, a divisor ratio of two sets the OPB clock frequency to 66MHz.
22:23	EPDV	EBCO to OPB Clock Divisor Ratio 00 EBCO to OPB Clock Divisor = 1 01 EBCO to OPB Clock Divisor = 2 10 EBCO to OPB Clock Divisor = 3 11 EBCO to OPB Clock Divisor = 4	The EPDV is the divisor ratio between the EBC0 (PerClk) and OPB clock. For example, if the OPB clock frequency is 66MHz, a divisor ratio of two sets the EBC0 clock (PerClk) frequency to 33MHz.
24	EXTSL	EBC0 clock (PerClk) Feedback Path Selection 0 CPU clock selected as PLL feedback path 1 EBC0 clock (PerClk) selected as PLL feedback path	Selecting the CPU clock as the PLL feedback path phase aligns PerClk with the CPU clock. Selecting the PerClk as the PLL feedback path phase aligns PerClk with SysClk.
25:26	RW	ROM Width 00 8-bit ROM 01 16-bit ROM 10 reserved 11 32-bit ROM	
27	RL	ROM Location 0 ROM connected to EBCO 1 ROM connected to PCI	If ROM is connected to EBCO, the real address of the reset vector is 0x1 FFFF FFFC. If ROM is connected to PCI, the real address of the reset vector is 0x2 FFFF FFFC.

28:29	ZMIISL	ZMII Selection 00 MII mode selected 01 SMII mode selected 10 RMII 10Mb mode selected 11 RMII 100Mb mode selected	See "EMAC-ZMII Bridge Interfaces" on page 21-3.
30	BYPASS	Bypass System PLL 0 System PLL enabled 1 Bypass system PLL	
31	Nto1	CPU:PLB N to 1 clock ratio 0 CPU:PLB clock ratio is N:P where P is greater than 1 1 CPU:PLB clock ratio is N:1	

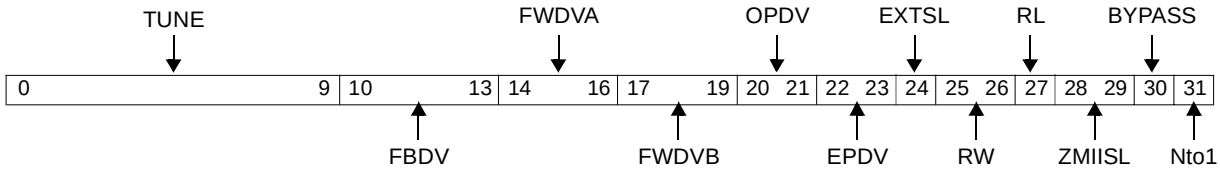


Figure 13-4. System Configuration Register 0 (CPC0_SYS0)

0:9	TUNE	System PLL TUNE bits	See Table 13-5, "CPC0_SYS0[Nto1] Settings," on page -411 for tune bit setting information.
10:13	FBDV	PLL Feedback Divisor 0000 PLL Feedback Divisor = 16 0001 PLL Feedback Divisor = 1 0010 PLL Feedback Divisor = 2 0011 PLL Feedback Divisor = 3 0100 PLL Feedback Divisor = 4 0101 PLL Feedback Divisor = 5 0110 PLL Feedback Divisor = 6 0111 PLL Feedback Divisor = 7 1000 PLL Feedback Divisor = 8 1001 PLL Feedback Divisor = 9 1010 PLL Feedback Divisor = 10 1011 PLL Feedback Divisor = 11 1100 PLL Feedback Divisor = 12 1101 PLL Feedback Divisor = 13 1110 PLL Feedback Divisor = 14 1111 PLL Feedback Divisor = 15	

14:16	FWDVA	PLL Forward Divisor A 000 PLL Forward Divisor A = 8 001 PLL Forward Divisor A = 7 010 PLL Forward Divisor A = 6 011 PLL Forward Divisor A = 5 100 PLL Forward Divisor A = 4 101 PLL Forward Divisor A = 3 110 PLL Forward Divisor A = 2 111 PLL Forward Divisor A = 1	
17:19	FWDVB	PLL Forward Divisor B 000 PLL Forward Divisor B = 8 001 PLL Forward Divisor B = 7 010 PLL Forward Divisor B = 6 011 PLL Forward Divisor B = 5 100 PLL Forward Divisor B = 4 101 PLL Forward Divisor B = 3 110 PLL Forward Divisor B = 2 111 PLL Forward Divisor B = 1	
20:21	OPDV	PLB to OPB Clock Divisor Ratio 00 PLB to OPB Clock Divisor = 1 01 PLB to OPB Clock Divisor = 2 10 PLB to OPB Clock Divisor = 3 11 PLB to OPB Clock Divisor = 4	The OPDV is the divisor ratio between the PLB and OPB clock. For example, if the PLB clock frequency is 133MHz, a divisor ratio of two sets the OPB clock frequency to 66MHz.
22:23	EPDV	EBCO to OPB Clock Divisor Ratio 00 EBCO to OPB Clock Divisor = 1 01 EBCO to OPB Clock Divisor = 2 10 EBCO to OPB Clock Divisor = 3 11 EBCO to OPB Clock Divisor = 4	The EPDV is the divisor ratio between the EBC0 (PerClk) and OPB clock. For example, if the OPB clock frequency is 66MHz, a divisor ratio of two sets the EBC0 clock (PerClk) frequency to 33MHz.
24	EXTSL	EBC0 clock (PerClk) Feedback Path Selection 0 CPU clock selected as PLL feedback path 1 EBC0 clock (PerClk) selected as PLL feedback path	Selecting the CPU clock as the PLL feedback path phase aligns PerClk with the CPU clock. Selecting the PerClk as the PLL feedback path phase aligns PerClk with SysClk.
25:26	RW	ROM Width 00 8-bit ROM 01 16-bit ROM 10 reserved 11 32-bit ROM	
27	RL	ROM Location 0 ROM connected to EBCO 1 ROM connected to PCI	If ROM is connected to EBCO, the real address of the reset vector is 0x1FFFFFFC. If ROM is connected to PCI, the real address of the reset vector is 0x2FFFFFFC.
28:29	ZMIISL	ZMII Selection 00 MII mode selected 01 SMII mode selected 10 RMII 10Mb mode selected 11 RMII 100Mb mode selected	See *EMAC-ZMII Bridge Interfaces* on page 721.
30	BYPASS	Bypass System PLL 0 System PLL enabled 1 Bypass system PLL	

31	Nto1	CPU:PLB N to 1 clock ratio 0 CPU:PLB clock ratio is N:P where P is greater than 1 1 CPU:PLB clock ratio is N:1

13.3.2 System Configuration Register 1 (CPC0_SYS1)

CPC0_SYS1 is a 32-bit read/write version of CPC0_STRP1. While most bits in this register at initial power on reset system are either default strap values or the values read from the bootstrap serial ROM shown in ~~"Default Strap Settings" on page 8-2~~*Default Strap Settings* on page 284, the TUNE bits are, by default, determined dynamically when the PCI interface calculates its mode (conventional 0-33, conventional 66, PCIX-1, PCIX-2 or PCIX-3).

Note: Writes to this register are ignored unless the CPC0_CR0[SWE] = 1. Also writes to these fields do not affect either SYS or PCI PLL until a CHIP reset occurs.

Figure 13-5 describes CPC0_SYS1 bit definitions.

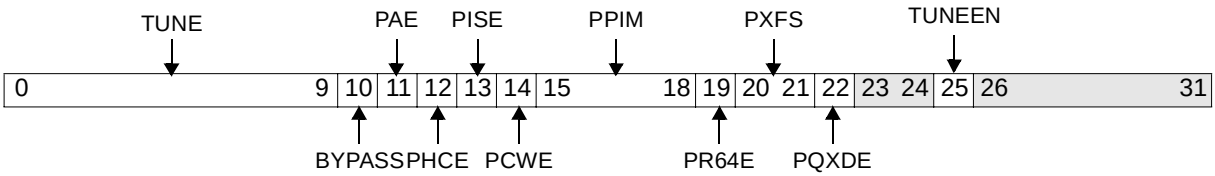


Figure 0-2. System Configuration Register 1 (CPC0_SYS1)

0:9	TUNE	CPC0_SYS1[TUNEEN] 0 CPC0_SYS1[TUNEEN] reads will return the tune bit value which was dynamically determined by the PCI interface at power-on (default). 1 CPC0_SYS1[TUNEEN] reads will return the current value programmed into CPC0_SYS1[TUNE]	CPC0_SYS1[TUNEEN] controls a mux which affects the value read from this field. Also see Table 13-13, "CPC0_SYS1[TUNE] Bit Settings," on page 13-11 for PCI PLL tune bit setting information.
10	BYPASS	Bypass PCI PLL 0 PCI PLL enabled 1 Bypass PCI PLL	
11	PAE	PCI arbiter enable 0 PCI arbiter disabled 1 PCI arbiter enabled	
12	PHCE	PCI host configuration enable 0 PCI host configuration disabled 1 PCI host configuration enabled	
13	PISE	PCI initial sequence enable 0 PCI initial sequence disabled 1 PCI initial sequence enabled	

14	PCWE	PCI local CPU wait enable 0 PCI local CPU wait disabled 1 PCI local CPU wait enabled	
15:18	PPIM	PCI Inbound Map (PIM) Settings 0000 PIM0 off, PIM1 off, PIM2 off 0001 PIM0 4k, PIM1 off, PIM2 off 0010 PIM0 1M, PIM1 off, PIM2 off 0011 PIM0 64M, PIM1 off, PIM2 off 0100 PIM0 4kp, PIM1 off, PIM2 off 0101 PIM0 1Mp, PIM1 off, PIM2 off 0110 PIM0 64Mp, PIM1 off, PIM2 off 0111 PIM0 64k, PIM1 off, PIM2 16k 1000 PIM0 1M, PIM1 off, PIM2 64k 1001 PIM0 64kp, PIM1 off, PIM2 16k 1010 PIM0 1Mp, PIM1 off, PIM2 64k 1011 PIM0 64k, PIM1 off, PIM2 64Kp 1100 PIM0 1M, PIM1 off, PIM2 1Mp 1101 PIM0 1Mp, PIM1 off, PIM2 1Mp 1110 PIM0 1M, PIM1 on, PIM2 off 1111 PIM0 1M, PIM1 on, PIM2 16k	These bits control the default settings for various PIM fields. See "PCI Inbound Map Setting" on page 8-3.
19	PR64E	PCI initialize Req64 Enable	
20:21	PXFS	PCIX Frequency Selection 00 100 - 133MHz frequency selection 01 66 - 100MHz frequency selection 10 50 - 66MHz frequency selection 11 reserved	
22	PQXDE	Quick PCIX capable detect enable 0 Quick PCIX capable detect disabled 1 Quick PCIX capable detect enabled	
23:24		Reserved	
25	TUNEEN	Custom PCI Tune Bit Enable 0 Use dynamically determined tune bits (default) 1 Use tune bits from CPC0_SYS1[TUNE]	Override the PCI PLL tune bit settings which are dynamically determined at power-on with the value programmed in CPC0_SYS1[TUNE]
26:31		Reserved	

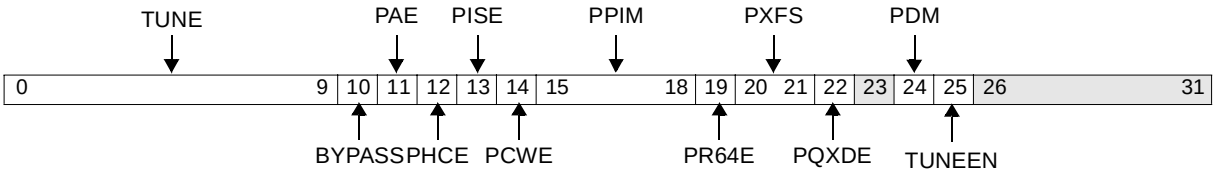


Figure 13-5. System Configuration Register 1 (CPC0_SYS1)

0:9	TUNE	CPC0_SYS1[TUNEEN] 0 CPC0_SYS1[TUNEEN] reads will return the tune bit value which was dynamically determined by the PCI interface at power-on (default). 1 CPC0_SYS1[TUNEEN] reads will return the current value programmed into CPC0_SYS1[TUNE]	CPC0_SYS1[TUNEEN] controls a mux which affects the value read from this field. Also see Table 13-13, "CPC0_SYS1[TUNE] Bit Settings," on page -417 for PCI PLL tune bit setting information.
10	BYPASS	Bypass PCI PLL 0 PCI PLL enabled 1 Bypass PCI PLL	
11	PAE	PCI arbiter enable 0 PCI arbiter disabled 1 PCI arbiter enabled	
12	PHCE	PCI host configuration enable 0 PCI host configuration disabled 1 PCI host configuration enabled	
13	PISE	PCI initial sequence enable 0 PCI initial sequence disabled 1 PCI initial sequence enabled	
14	PCWE	PCI local CPU wait enable 0 PCI local CPU wait disabled 1 PCI local CPU wait enabled	
15:18	PPIM	PCI Inbound Map (PIM) Settings 0000 PIM0 off, PIM1 off, PIM2 off 0001 PIM0 4k, PIM1 off, PIM2 off 0010 PIM0 1M, PIM1 off, PIM2 off 0011 PIM0 64M, PIM1 off, PIM2 off 0100 PIM0 4kp, PIM1 off, PIM2 off 0101 PIM0 1Mp, PIM1 off, PIM2 off 0110 PIM0 64Mp, PIM1 off, PIM2 off 0111 PIM0 64k, PIM1 off, PIM2 16k 1000 PIM0 1M, PIM1 off, PIM2 64k 1001 PIM0 64kp, PIM1 off, PIM2 16k 1010 PIM0 1Mp, PIM1 off, PIM2 64k 1011 PIM0 64k, PIM1 off, PIM2 64Kp 1100 PIM0 1M, PIM1 off, PIM2 1Mp 1101 PIM0 1Mp, PIM1 off, PIM2 1Mp 1110 PIM0 1M, PIM1 on, PIM2 off 1111 PIM0 1M, PIM1 on, PIM2 16k	These bits control the default settings for various PIM fields. See *PCI Inbound Map Setting* on page 285.

19	PR64E	PCI initialize Req64 Enable 0 PCI initialize Req64 disabled 1 PCI initialize Req64 enabled
20:21	PXFS	PCIX Frequency Selection 00 100 - 133MHz frequency selection 01 66 - 100MHz frequency selection 10 50 - 66MHz frequency selection 11 reserved
22	PQXDE	Quick PCIX capable detect enable 0 Quick PCIX capable detect disabled 1 Quick PCIX capable detect enabled
23		Reserved
24	PDM	PCIX driver mode enable 0 PCIX driver impedence is 20 ohms for multi point mode 1 PCIX driver impedence is 40 ohms for point to point mode
25	TUNEEN	Custom PCI Tune Bit Enable 0 Use dynamically determined tune bits (default) 1 Use tune bits from CPC0_SYS1[TUNE] Override the PCI PLL tune bit settings which are dynamically determined at power-on with the value programmed in CPC0_SYS1[TUNE]
26:31		Reserved

14. Clock and Power Management

The PPC440GP provides a clock and power management (CPM) controller that reduces power dissipation by stopping clocks in unused or dormant functional units. Use of the CPM controller requires careful programming and special consideration to avoid compromising system and functional unit integrity.

14.1 Overview

The CPM controller supports three different types of sleep interfaces to the functional units:

- In a CPM class 1 interface, the CPM_Sleep_N signal is asserted by the CPM controller when a register bit is set by software. The functional unit is unconditionally put to sleep. There is no other communication with the functional unit.
- In a CPM class 2 interface, the functional unit uses a combination of its internal state and external inputs to determine whether or not it can be put to sleep. If sleeping is permitted, the functional unit asserts the Sleep_Req signal to the CPM controller that responds by asserting CPM_Sleep_N if the enable for that unit is set. The CPM_Sleep_N signal to a class 2 unit is deasserted when the CPM controller enable bit for that unit is reset, or when the unit deasserts its Sleep_Req signal.
- The CPM class 3 interface has a CPM_SleepInit signal that is asserted by the CPM controller to request that a functional unit go to sleep. If the unit can sleep, it asserts the Sleep_Req signal to the CPM controller. The CPM_Sleep_N signal is then asserted by the CPM controller to shut off the class 3 clocks in the functional unit. The functional unit or the CPM controller can end the sleep state. If the CPM controller enable bit for the unit is reset, the CPM controller immediately deasserts CPM_SleepInit and CPM_Sleep_N.

14.2 CPM Registers

Table 14-1 lists the registers used to program the CPM controller.

Table 14-1. CPM Registers

Register	Description	DCR Address	Access	(Bits 0:31)
CPC0_ER	CPM Enable Register	0x0B1	Read/Write	0x000000000
CPC0_FR	CPM Force Register	0x0B2	Read/Write	0x000000000
CPC0_SR	CPM Status Register	0x0B0	Read Only	0xFFFFFFFF

14.2.1 CPM Enable Register (CPC0_ER)

The CPC0_ER bits enable the process of putting a functional unit to sleep. The class of a unit determines how its interface signals are controlled when the bit associated with the unit is set to 1.

Figure 14-1 describes CPC0_ER bit assignment and CPM class for each PPC440GP functional unit.

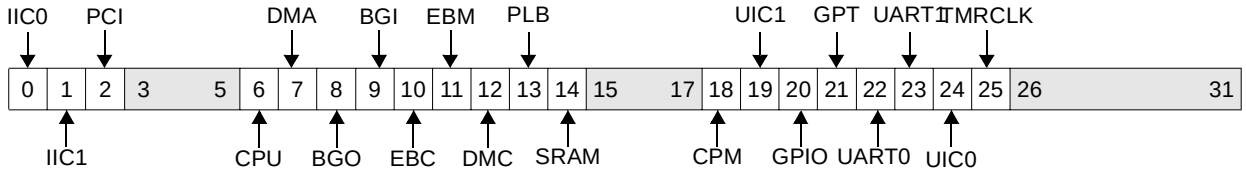


Figure 0-1. CPM Enable Register (CPC0_ER)

0	IIC0	Inter-Integrated Circuit 0	Class 3
1	IIC1	Inter-Integrated Circuit 1	Class 3
2	PCI	Peripheral Component Interconnect	Class 3
3:5		Reserved	
6	CPU	Processor Core	Class 2
7	DMA	Direct Memory Access Controller	Class 2
8	BGO	PLB to OPB Bridge	Class 2
9	BGI	OPB to PLB Bridge	Class 2
10	EBC	External Bus Controller	Class 2
11	EBM	External Bus Master Interface	Class 2
12	DMC	DDR SDRAM Controller	Class 2
13	PLB	PLB Arbiter	Class 2
14	SRAM	Internal SRAM Controller	Class 2
15:17		Reserved	
18	PPM	PLB Performance Monitor	Class 1
19	UIC1	Universal Interrupt Controller 1	Class 1
20	GPIO	General Purpose IO	Class 1
21	GPT	General Purpose Timer	Class 1
22	UART0	Universal Asynchronous Receiver/Transmitter 0	Class 1
23	UART1	Universal Asynchronous Receiver/Transmitter 1	Class 1
24	UIC0	Universal Interrupt Controller 0	Class 1
25	TMRCLK	CPU Timer	Class 1
26:31		Reserved	

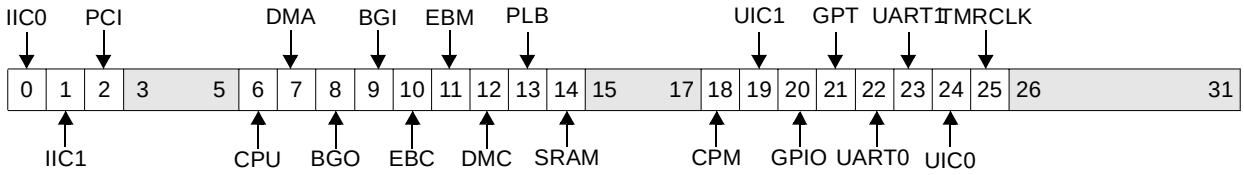


Figure 14-1. CPM Enable Register (CPC0_ER)

0	IIC0	Inter-Integrated Circuit 0	Class 3
1	IIC1	Inter-Integrated Circuit 1	Class 3
2	PCI	Peripheral Component Interconnect	Class 3
3:5		Reserved	
6	CPU	PPC440GP Processor Core	Class 2
7	DMA	Direct Memory Access Controller	Class 2
8	BGO	PLB to OPB Bridge	Class 2
9	BGI	OPB to PLB Bridge	Class 2
10	EBC	External Bus Controller	Class 2
11	EBM	External Bus Master Interface	Class 2
12	DMC	DDR SDRAM Controller	Class 2
13	PLB	PLB Arbiter	Class 2
14	SRAM	Internal SRAM Controller	Class 2
15:17		Reserved	
18	PPM	PLB Performance Monitor	Class 1
19	UIC1	Universal Interrupt Controller 1	Class 1
20	GPIO	General Purpose IO	Class 1
21	GPT	General Purpose Timer	Class 1
22	UART0	Universal Asynchronous Receiver/Transmitter 0	Class 1
23	UART1	Universal Asynchronous Receiver/Transmitter 1	Class 1
24	UIC0	Universal Interrupt Controller 0	Class 1
25	TMRCLK	CPU Timer	Class 1
26:31		Reserved	

-

Class 1	When an associated CPC0_ER bit is set to 1, the CPM_Sleep_N signal to the class 1 unit is asserted. When the bit is set to 0, CPM_Sleep_N is deasserted.
Class 2	When an associated CPC0_ER bit is set to 1, and the Sleep_Req signal from the class 2 unit is asserted (the unit is requesting sleep state), CPM_Sleep_N to the class 2 unit is asserted. When the bit is set to 0, the CPM_Sleep_N signal is deasserted.
Class 3	When an associated CPC0_ER bit is set to 1, the CPM_SleepInit signal to the class 3 unit is asserted (the CPM controller is requesting permission to put the unit to sleep). When the class 3 unit activating the Sleep_Req in response, (the unit is giving permission to be put to sleep), CPM_Sleep_N signal to the class 3 unit is asserted. When the bit is set to 0, CPM_SleepInit and CPM_Sleep_N are deasserted.

14.2.2 CPM Force Register (CPC0_FR)

Setting a CPC0_FR bit forces assertion of the CPM_Sleep_N signal to the functional unit. For a class 1 unit, this is equivalent to setting the CPC0_ER bit associated with the unit. For class 2 or class 3 units, CPM_Sleep_N is asserted regardless of the state of the Sleep_Req signal coming from the unit.

~~Figure 14-2~~Figure 14-2 describes CPC_FR bit assignment and CPM class for each PPC440GP functional unit~~.~~.

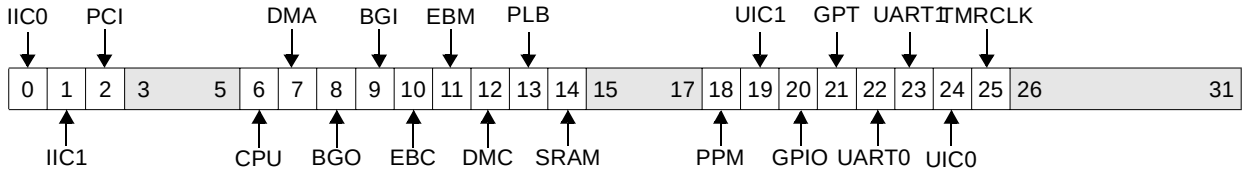


Figure 0-2. CPM Force Register (CPC0_FR)

0	IIC0	Inter-Integrated Circuit 0	Class 3
1	IIC1	Inter-Integrated Circuit 1	Class 3
2	PCI	Peripheral Component Interconnect	Class 3
3:5		Reserved	
6	CPU	Processor Core	Class 2
7	DMA	Direct Memory Access Controller	Class 2
8	BGO	PLB to OPB Bridge	Class 2
9	BGI	OPB to PLB Bridge	Class 2
10	EBC	External Bus Controller	Class 2
11	EBM	External Bus Master Interface	Class 2
12	DMC	DDR SDRAM Controller	Class 2
13	PLB	PLB Arbiter	Class 2
14	SRAM	Internal SRAM Controller	Class 2
15:17		Reserved	

Figure 0-2. CPM Force Register (CPC0_FR)

18	PPM	PLB Performance Monitor	Class 1
19	UIC1	Universal Interrupt Controller 1	Class 1
20	GPIO	General Purpose IO	Class 1
21	GPT	General Purpose Timer	Class 1
22	UART0	Universal Asynchronous Receiver/Transmitter 0	Class 1
23	UART1	Universal Asynchronous Receiver/Transmitter 1	Class 1
24	UIC0	Universal Interrupt Controller 0	Class 1
25	TMRCLK	CPU Timer	Class 1
26:31		Reserved	

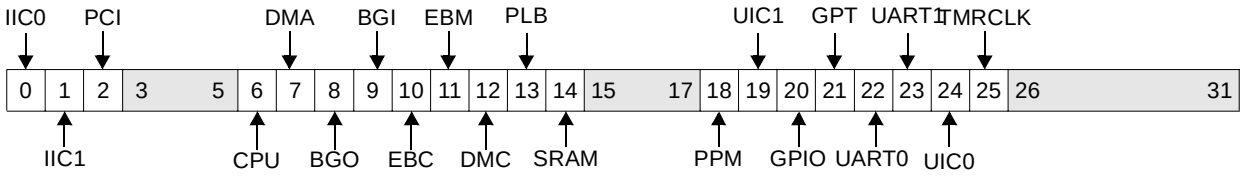


Figure 14-2. CPM Force Register (CPC0_FR)

0	IIC0	Inter-Integrated Circuit 0	Class 3
1	IIC1	Inter-Integrated Circuit 1	Class 3
2	PCI	Peripheral Component Interconnect	Class 3
3:5		Reserved	
6	CPU	PPC440GP Processor Core	Class 2
7	DMA	Direct Memory Access Controller	Class 2
8	BGO	PLB to OPB Bridge	Class 2
9	BGI	OPB to PLB Bridge	Class 2
10	EBC	External Bus Controller	Class 2
11	EBM	External Bus Master Interface	Class 2
12	DMC	DDR SDRAM Controller	Class 2
13	PLB	PLB Arbiter	Class 2
14	SRAM	Internal SRAM Controller	Class 2
15:17		Reserved	
18	PPM	PLB Performance Monitor	Class 1
19	UIC1	Universal Interrupt Controller 1	Class 1
20	GPIO	General Purpose IO	Class 1

Figure 14-2. CPM Force Register (CPC0_FR)

21	GPT	General Purpose Timer	Class 1
22	UART0	Universal Asynchronous Receiver/Transmitter 0	Class 1
23	UART1	Universal Asynchronous Receiver/Transmitter 1	Class 1
24	UIC0	Universal Interrupt Controller 0	Class 1
25	TMRCLK	CPU Timer	Class 1
26:31		Reserved	

14.2.3 CPM Status Register (CPC0_SR)

The read-only CPC0_SR shows the current state of all CPM_Sleep_N signals.

~~Figure 14-3~~Figure 14-3 describes CPC0_SR bit assignment and CPM class for each PPC440GP functional unit.

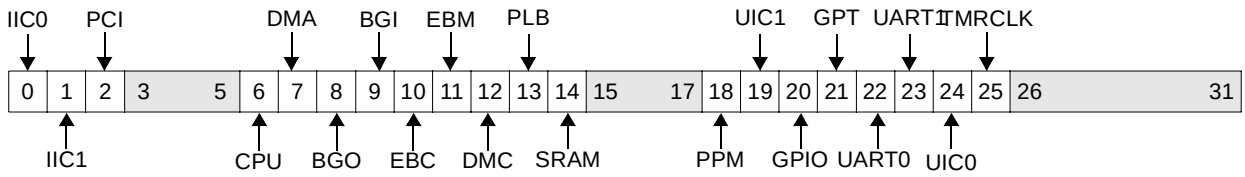


Figure 0-3. CPM Status Register (CPC0_SR)

0	IIC0	Inter-Integrated Circuit 0	Class 3
1	IIC1	Inter-Integrated Circuit 1	Class 3
2	PCI	Peripheral Component Interconnect	Class 3
3:5		Reserved	
6	CPU	Processor Core	Class 2
7	DMA	Direct Memory Access Controller	Class 2
8	BGO	PLB to OPB Bridge	Class 2
9	BGI	OPB to PLB Bridge	Class 2
10	EBC	External Bus Controller	Class 2
11	EBM	External Bus Master Interface	Class 2
12	DMC	DDR SDRAM Controller	Class 2
13	PLB	PLB Arbiter	Class 2
14	SRAM	Internal SRAM Controller	Class 2

Figure 0-3. CPM Status Register (CPC0_SR)

15:17		Reserved	
18	PPM	PLB Performance Monitor	Class 1
19	UIC1	Universal Interrupt Controller 1	Class 1
20	GPIO	General Purpose IO	Class 1
21	GPT	General Purpose Timer	Class 1
22	UART0	Universal Asynchronous Receiver/Transmitter 0	Class 1
23	UART1	Universal Asynchronous Receiver/Transmitter 1	Class 1
24	UIC0	Universal Interrupt Controller 0	Class 1
25	TMRCLK	CPU Timer	Class 1
26:31		Reserved	

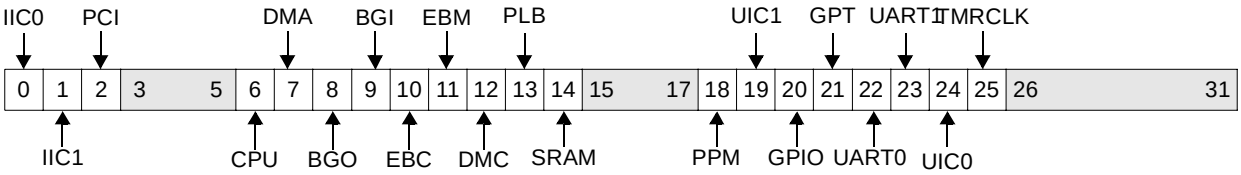


Figure 14-3. CPM Status Register (CPC0_SR)

0	IIC0	Inter-Integrated Circuit 0	Class 3
1	IIC1	Inter-Integrated Circuit 1	Class 3
2	PCI	Peripheral Component Interconnect	Class 3
3:5		Reserved	
6	CPU	PPC440GP Processor Core	Class 2
7	DMA	Direct Memory Access Controller	Class 2
8	BGO	PLB to OPB Bridge	Class 2
9	BGI	OPB to PLB Bridge	Class 2
10	EBC	External Bus Controller	Class 2
11	EBM	External Bus Master Interface	Class 2
12	DMC	DDR SDRAM Controller	Class 2
13	PLB	PLB Arbiter	Class 2
14	SRAM	Internal SRAM Controller	Class 2
15:17		Reserved	
18	PPM	PLB Performance Monitor	Class 1
19	UIC1	Universal Interrupt Controller 1	Class 1

Figure 14-3. CPM Status Register (CPC0_SR)

20	GPIO	General Purpose IO	Class 1
21	GPT	General Purpose Timer	Class 1
22	UART0	Universal Asynchronous Receiver/Transmitter 0	Class 1
23	UART1	Universal Asynchronous Receiver/Transmitter 1	Class 1
24	UIC0	Universal Interrupt Controller 0	Class 1
25	TMRCLK	CPU Timer	Class 1
26:31		Reserved	

15. Debug Facilities

The debug facilities of the PPC440GP include support for several debug modes for debugging during hardware and software development, as well as debug events that allow developers to control the debug process. Debug registers control these debug modes and debug events. The debug registers may be accessed either through software running on the processor or through the JTAG debug port of the PPC440GP. Access to the debug facilities through the JTAG debug port is typically provided by a debug tool such as the RISCWatch™ development tool from IBM. A trace port, which enables the tracing of code running in real time, is also provided.

15.1 Support for Development Tools

The RISCWatch product from IBM is an example of a development tool that uses external debug mode, debug events, and the JTAG debug port to implement a hardware and software development tool. The RISCTrace™ feature of RISCWatch is an example of a development tool that uses the real-time instruction trace capability of the PPC440GP.

15.2 Debug Interfaces

The PPC440GP provides JTAG and trace interfaces to support hardware and software test and debug. Typically, the JTAG interface connects to a debug port external to the PPC440GP; the JTAG debug port is typically connected to a JTAG connector on a processor board. The trace interface connects to a trace port external to the PPC440GP; the trace port is typically connected to a trace connector on a processor board.

15.2.1 IEEE 1149.1 Test Access Port (JTAG Debug Port)

The IEEE 1149.1 Test Access Port (TAP), commonly called the JTAG (Joint Test Action Group) debug port, is an architectural standard described in IEEE Std 1149.1–1990, *IEEE Standard Test Access Port and Boundary Scan Architecture*. The standard describes a method for accessing internal chip facilities using a four- or five-signal interface.

The JTAG debug port, originally designed to support scan-based board testing, is enhanced to support the attachment of debug tools. The enhancements, which comply with the IEEE 1149.1 specifications for vendor-specific extensions, are compatible with standard JTAG hardware for boundary-scan system testing.

JTAG Signals	The JTAG debug port implements the four required JTAG signals: TCK, TMS, TDI, and TDO, and the optional $\overline{\text{TRST}}$ signal.
JTAG Clock Requirements	The frequency of the TCK signal can range from DC to one-half of the internal chip clock frequency.
JTAG Reset Requirements	The JTAG debug port logic is reset at the same time as a system reset. Upon receiving $\overline{\text{TRST}}$, the JTAG debug port returns to the Test-Logic Reset state.

15.2.1.1 JTAG Connector

The PPC440GP implements a JTAG interface to support system debugging. The interface enables the connection of an external debug tool, such as RISCWatch. Detailed information on JTAG capabilities and how to connect an external debug tool is available in *RISCWatch Debugger User's Guide*.

15.2.1.2 JTAG Instructions

The JTAG debug port provides the standard *extest*, *idcode*, *sample/preload*, and *bypass* instructions and the optional *highz* and *clamp* instructions. Invalid instructions behave as the *bypass* instruction.

Table 15-1. JTAG Instructions

Instruction	Code	Comments
Extest	1111000	IEEE 1149.1 standard
	1111001	Reserved
Sample/Preload	1111010	IEEE 1149.1 standard
IDCode	1111011	IEEE 1149.1 standard
Private	xxxx100	Private instructions
HighZ	1111101	IEEE 1149.1a-1993 optional
Clamp	1111110	IEEE 1149.1a-1993 optional
Bypass	1111111	IEEE 1149.1 standard

15.2.1.3 JTAG Boundary Scan

Boundary Scan Description Language (BSDL), IEEE 1149.1b-1994, is a supplement to IEEE 1149.1-1990 and IEEE 1149.1a-1993 *Standard Test Access Port and Boundary-Scan Architecture*. BSDL, a subset of the IEEE 1076-1993 Standard VHSIC Hardware Description Language (VHDL), allows a rigorous description of testability features in components which comply with the standard. BSDL is used by automated test pattern generation tools for package interconnect tests and by electronic design automation (EDA) tools for synthesized test logic and verification. BSDL supports robust extensions that can be used for internal test generation and to write software for hardware debug and diagnostics.

The primary components of BSDL include the logical port description, the physical pin map, the instruction set, and the boundary register description.

The logical port description assigns symbolic names to the pins of a chip. Each pin has a logical type of in, out, inout, buffer, or linkage that defines the logical direction of signal flow.

The physical pin map correlates the logical ports of the chip to the physical pins of a specific package. A BSDL description can have several physical pin maps; each map is given a unique name.

Instruction set statements describe the bit patterns that must be shifted into the Instruction Register to place the chip in the various test modes defined by the standard. Instruction set statements also support descriptions of instructions that are unique to the chip.

The boundary register description lists each cell or shift stage of the Boundary Register. Each cell has a unique number: the cell numbered 0 is the closest to the Test Data Out (TDO) pin; the cell with the highest number is closest to the Test Data In (TDI) pin. Each cell contains additional information, including: cell type, logical port associated with the cell, logical function of the cell, safe value, control cell number, disable value, and result value.

15.2.1.4 JTAG ID Register (CPC0_JTAGID)__

CPC0_JTAGID is a Device Control Register that enables manufacturing, part number, and version information to be determined through the TAP. The **mfdcr** instruction is used to read this register.

Refer to ~~*PowerPC 440GP*~~ PPC440GP *Embedded Processor Data Sheet* for the values of the CPC0_JTAGID fields.

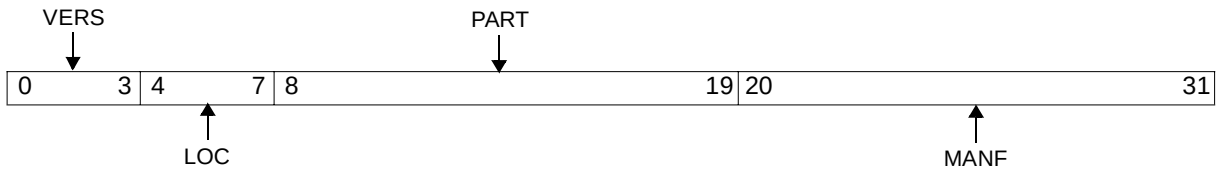


Figure 0-1. JTAG ID Register (CPC0_JTAGID)

0:3	VERS	Version
4:7	LOC	Developer Location
8:19	PART	Part Number
20:31	MANF	Manufacturer Identifier

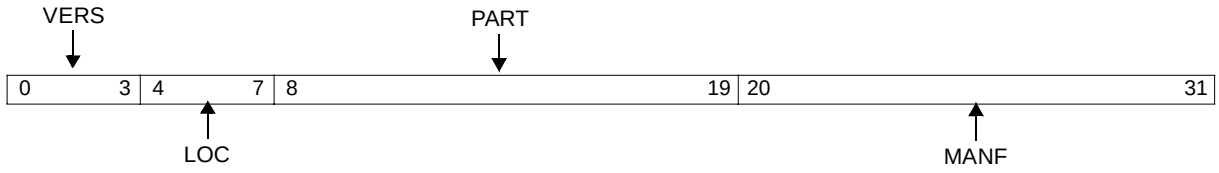


Figure 15-1. JTAG ID Register (CPC0_JTAGID)

0:3	VERS	Version
4:7	LOC	Developer Location
8:19	PART	Part Number
20:31	MANF	Manufacturer Identifier

15.2.2 Trace Port

The PPC440GP implements a trace status interface to support the tracing of code running in real time. This interface enables the connection of an external trace tool, such as RISCWatch, and allows for user-extended trace functions. A software tool with trace capability, such as RISCWatch with RISCTrace, can use the data

collected from this port to trace code running on the processor. The result is a trace of the code executed, including code executed out of the instruction cache if it was enabled. Information on trace capabilities, how trace works, and how to connect the external trace tool is available in *RISCWatch Debugger User's Guide*.

15.3 Debug Modes

The following sections describe the various debug modes supported by the PPC440GP. Each of these debug modes supports a particular type of debug tool or debug task commonly used in embedded systems development. For all debug modes, the various debug event types are enabled by the setting of corresponding fields in Debug Control Register 0 (DBCR0), and upon their occurrence are recorded in the Debug Status Register (DBSR).

There are four debug modes:

- Internal debug mode
- External debug mode
- Debug wait mode
- Trace debug mode

The PowerPC Book-E architecture specification deals only with internal debug mode, and the relationship of Debug interrupts to the rest of the interrupt architecture. Internal debug mode is the mode which involves debug software running on the processor itself, typically in the form of the Debug interrupt handler. The other debug modes, on the other hand, are outside the scope of the architecture, and involve special-purpose debug hardware external to the PPC440GP core, connected either to the JTAG interface (for external debug mode and debug wait mode) or the trace interface (for trace debug mode). Details of these interfaces and their operation are beyond the scope of this manual.

15.3.1 Internal Debug Mode

Internal debug mode provides access to architected processor resources and supports setting hardware and software breakpoints and monitoring processor status. In this mode, debug events are considered exceptions, which, in addition to recording their status in the DBSR, generate Debug interrupts if and when such interrupts are enabled (Machine State Register (MSR) DE field is 1; see ~~Chapter 10, "Interrupts and Exceptions"~~*Interrupts and Exceptions* on page 337 for a description of the MSR and Debug interrupts). When a Debug interrupt occurs, special debugger software at the interrupt handler can check processor status and other conditions related to the debug event, as well as alter processor resources using all of the instructions defined for the PPC440GP.

Internal debug mode relies on this interrupt handling software at the Debug interrupt vector to debug software problems. This mode, used while the processor executes instructions, enables debugging of both application programs and operating system software, including all of the non-critical class interrupt handlers.

In this mode, the debugger software can communicate with the outside world through a communications port, such as a serial port, external to the processor core.

To enable internal debug mode, the IDM field of DBCR0 must be set to 1 (DBCR0[IDM] = 1). This mode can be enabled in combination with external debug mode (see ~~"External Debug Mode"~~*External Debug Mode* below) and/or debug wait mode (see ~~"Debug Wait Mode" on page 15-5~~*Debug Wait Mode* on page 433).

15.3.2 External Debug Mode

External debug mode provides access to architected processor resources and supports stopping, starting, and stepping the processor, setting hardware and software breakpoints, and monitoring processor status. In this mode, debug events record their status in the DBSR and then cause the processor to enter the *stop state*, in which normal instruction execution stops and architected processor resources and memory can be accessed and altered via the JTAG interface. While in the stop state, interrupts are temporarily disabled.

Storage access control by a memory management unit (MMU) remains in effect while in external debug mode; the debugger may need to modify MSR or TLB values to access protected memory.

External debug mode relies only on internal processor resources, and no Debug interrupt handling software, so it can be used to debug both system hardware and software problems. This mode can also be used for software development on systems without a control program, or to debug control program problems, including problems within the Debug interrupt handler itself, or within any other critical class interrupt handlers.

External debug mode is enabled by setting DBCR0[EDM] to 1. This mode can be enabled in combination with internal debug mode (see ~~"Internal Debug Mode" on page 15-4~~*Internal Debug Mode* on page 432) and/or debug wait mode (see ~~"Debug Wait Mode"~~*Debug Wait Mode* below). External debug mode takes precedence over internal debug mode however, in that debug events will first cause the processor to enter stop state rather than generating a Debug interrupt, although a Debug interrupt may be pending while the processor is in the stop state.

15.3.3 Debug Wait Mode

Debug wait mode is similar to external debug mode in that debug events cause the processor to enter the stop state. However, interrupts are still enabled while in debug wait mode, such that if and when an exception occurs for which the associated interrupt type is enabled, the processor will leave the stop state and generate the interrupt. This mode is useful for real-time hardware environments which cannot tolerate interrupts being disabled for an extended period of time. In such environments, if external debug mode were to be used, various I/O devices could operate incorrectly due to not being serviced in a timely fashion when they assert an interrupt request to the processor, if the processor happened to be in stop state at the time of the interrupt request.

When in debug wait mode, as with external debug mode, access to the architected processor resources and memory is via the JTAG interface.

Debug wait mode is enabled by setting both MSR[DWE] and the debug wait mode enable within the JTAG controller to 1. Since MSR[DWE] is automatically cleared upon any interrupt, debug wait mode is temporarily disabled upon an interrupt, and then can be automatically re-enabled when returning from the interrupt due to the restoration of the MSR value upon the execution of an **rfi** or **rfci** instruction.

While debug wait mode can be enabled in combination with external debug mode, external debug mode takes precedence and interrupts are temporarily disabled, thereby effectively nullifying the effect of debug wait mode. Similarly, debug wait mode can be enabled in combination with internal debug mode. However, if Debug interrupts are enabled (MSR[DE] is 1), then any debug event will lead to an exception and a corresponding Debug interrupt, which takes precedence over the stop state associated with debug wait mode. On the other hand, if Debug interrupts are disabled (MSR[DE] is 0), then debug wait mode will take effect and a debug event will cause the processor to enter stop state.

15.3.4 Trace Debug Mode

Trace debug mode is simply the *absence* of each of the other modes. That is, if internal debug mode, external debug mode, and debug wait mode are all disabled, then the processor is in trace debug mode. While in trace debug mode, all debug events are simply recorded in the DBSR, and are indicated over the trace interface from the PPC440GP core. The processor does not enter the stop state, nor does a Debug interrupt occur.

15.4 Debug Events

There are several different kinds of debug events, each of which is enabled by a field in DBCR0 (except for the Unconditional debug event) and recorded in the DBSR. ~~"Debug Modes" on page 15-3~~*Debug Modes* on page 431 describes the operation that results when a debug event occurs while operating in any of the debug modes.

Table 15-2 lists the various debug events recognized by the PPC440GP. Detailed explanations of each debug event type follow the table.

Table 15-2. Debug Events

Event	Description
Instruction Address Compare (IAC)	Caused by the attempted execution of an instruction for which the address matches the conditions specified by DBCR0, DBCR1, and the IAC–IAC4 registers.
Data Address Compare (DAC)	Caused by the attempted execution of a load, store, or cache management instruction for which the data storage address matches the conditions specified by DBCR0, DBCR2, and the DAC–DAC2 registers.
Data Value Compare (DVC)	Caused by the attempted execution of a load, store, or cache management instruction for which the data storage address matches the conditions specified by DBCR0, DBCR2, and the DAC1–DAC2 registers, and for which the referenced data matches the value specified by the DVC1–DVC2 registers.
Branch Taken (BRT)	Caused by the attempted execution of a branch instruction for which the branch conditions are met (that is, for a branch instruction that results in the re-direction of the instruction stream).
Trap (TRAP)	Caused by the attempted execution of a **tw** or **twi** instruction for which the trap conditions are met.
Return (RET)	Caused by the attempted execution of an **rfi** or **rfci** instruction.
Instruction Complete (ICMP)	Caused by the successful completion of the execution of any instruction.
Interrupt (IRPT)	Caused by the generation of an interrupt.
Unconditional (UDE)	Caused by the assertion of an unconditional debug event request from the JTAG interface to the PPC440GP.

15.4.1 Instruction Address Compare (IAC) Debug Event

IAC debug events occur when execution is attempted of an instruction for which the instruction address and other parameters match the IAC conditions specified by DBCR0, DBCR1, and the IAC registers. There are four IAC registers on the PPC440GP, IAC1–IAC4. Depending on the IAC mode specified by DBCR1, these IAC registers can be used to specify four independent, exact IAC addresses, or they can be configured in pairs (IAC1/IAC2 and IAC3/IAC4) in order to specify *ranges* of instruction addresses for which IAC debug events should occur.

15.4.1.1 IAC Debug Event Fields

There are several fields in DBCR0 and DBCR1 which are used to specify the IAC conditions, as follows:

IAC Event Enable Field

DBCR0[IAC1, IAC2, IAC3, IAC4] are the individual IAC event enables for each of the four IAC events: IAC1, IAC2, IAC3, and IAC4. For a given IAC event to occur, the corresponding IAC event enable bit in DBCR0 must be set. When a given IAC event occurs, the corresponding DBSR[IAC1, IAC2, IAC3, IAC4] bit is set.

IAC Mode Field

DBCR1[IAC12M, IAC34M] control the comparison mode for the IAC1/IAC2 and IAC3/IAC4 events, respectively. There are three comparison modes supported by the PPC440GP:

- Exact comparison mode (DBCR1[IAC12M/IAC34M] = 0b00)

  In this mode, the instruction address is compared to the value in the corresponding IAC register, and the IAC event occurs only if the comparison is an exact match.

- Range inclusive comparison mode (DBCR1[IAC12M/IAC34M] = 0b10)

  In this mode, the IAC1 or IAC2 event occurs only if the instruction address is *within* the range defined by the IAC1 and IAC2 register values, as follows: IAC1 ≤ address < IAC2. Similarly, the IAC3 or IAC4 event occurs only if the instruction address is *within* the range defined by the IAC3 and IAC4 register values, as follows: IAC3 ≤ address < IAC4.

  For a given IAC1/IAC2 or IAC3/IAC4 pair, when the instruction address falls within the specified range, either one or both of the corresponding IAC debug event bits will be set in the DBSR, as determined by which of the two corresponding IAC event enable bits are set in DBCR0. For example, when the IAC1/IAC2 pair are set to range inclusive comparison mode, and the instruction address falls within the defined range, then DBCR1[IAC1, IAC2] will determine whether one or the other or both of DBSR[IAC1, IAC2] are set. It is a programming error to set either of the IAC pairs to a range comparison mode (either inclusive or exclusive) without also enabling at least one of the corresponding IAC event enable bits in DBCR0.

  Note that the IAC range auto-toggle mechanism can "switch" the IAC range mode from inclusive to exclusive, and vice-versa. See ~~"IAC Range Mode Auto-Toggle Field" on page 15-8~~*IAC Range Mode Auto-Toggle Field* on page 436.

- Range exclusive comparison mode (DBCR1[IAC12M/IAC34M] = 0b11)

  In this mode, the IAC1 or IAC2 event occurs only if the instruction address is *outside* the range defined by the IAC1 and IAC2 register values, as follows: address < IAC1 or address ≥ IAC2. Similarly, the IAC3 or IAC4 event occurs only if the instruction address is *outside* the range defined by the IAC3 and IAC4 register values, as follows: address < IAC3 or address ≥ IAC4.

  For a given IAC1/IAC2 or IAC3/IAC4 pair, when the instruction address falls outside the specified range, either one or both of the corresponding IAC debug event bits will be set in the DBSR, as determined by which of the two corresponding IAC event enable bits are set in DBCR0. For example, when the IAC1/IAC2 pair are set to range exclusive comparison mode, and the instruction address falls outside the defined range, then DBCR1[IAC1, IAC2] will determine whether one or the other or both of DBSR[IAC1, IAC2] are set. It is a programming error to set either of the IAC pairs to a range comparison mode (either inclusive or exclusive) without also enabling at least one of the corresponding IAC event enable bits in DBCR0.

Note that the IAC range auto-toggle mechanism can "switch" the IAC range mode from inclusive to exclusive, and vice-versa. See ~~"IAC Range Mode Auto-Toggle Field" on page 15-8~~*IAC Range Mode Auto-Toggle Field* on page 436.

The PowerPC Book-E architecture defines DBCR1[IAC12M/IAC34M] = 0b01 as IAC address bit mask mode, but that mode is not supported by the PPC440GP, and that value of the IAC12M/IAC34M fields is reserved.

IAC User/Supervisor Field

DBCR1[IAC1US, IAC2US, IAC3US, IAC4US] are the individual IAC user/supervisor fields for each of the four IAC events. The IAC user/supervisor fields specify what operating mode the processor must be in order for the corresponding IAC event to occur. The operating mode is determined by the Problem State field of the Machine State Register (MSR[PR]; see ~~"User and Supervisor Modes" on page 4-61~~*User and Supervisor Modes* on page 199). When the IAC user/supervisor field is 0b00, the operating mode does not matter; the IAC debug event may occur independent of the state of MSR[PR]. When this field is 0b10, the processor must be operating in supervisor mode (MSR[PR] = 0). When this field is 0b11, the processor must be operating in user mode (MSR[PR] = 1). The IAC user/supervisor field value of 0b01 is reserved.

If a pair of IAC events (IAC1/IAC2 or IAC3/IAC4) are operating in range inclusive or range exclusive mode, it is a programming error (and the results of any instruction address comparison are undefined) if the corresponding pair of IAC user/supervisor fields are not set to the same value. For example, if IAC1/IAC2 are operating in one of the range modes, then both DBCR1[IAC1US] and DBCR1[IAC2US] must be set to the same value.

IAC Effective/Real Address Field

DBCR1[IAC1ER, IAC2ER, IAC3ER, IAC4ER] are the individual IAC effective/real address fields for each of the four IAC events. The IAC effective/real address fields specify whether the instruction address comparison should be performed using the effective, virtual, or real address (see ~~Chapter 6, "Memory Management"~~*Memory Management* on page 233) for an explanation of these different types of addresses). When the IAC effective/real address field is 0b00, the comparison is performed using the effective address only—the IAC debug event may occur independent of the instruction address space (MSR[IS]). When this field is 0b10, the IAC debug event occurs only if the effective address matches the IAC conditions and is in virtual address space 0 (MSR[IS] = 0). Similarly, when this field is 0b11, the IAC debug event occurs only if the effective address matches the IAC conditions and is in virtual address space 1 (MSR[IS] = 1). Note that in these latter two modes, in which the virtual address space of the instruction is considered, it is not the entire virtual address which is considered. The Process ID, which forms the final part of the virtual address, is not considered. Finally, the IAC effective/real address field value of 0b01 is reserved, and corresponds to the PowerPC Book-E architected real address comparison mode, which is not supported by the PPC440GP.

If a pair of IAC events (IAC1/IAC2 or IAC3/IAC4) are operating in range inclusive or range exclusive mode, it is a programming error (and the results of any instruction address comparison are undefined) if the corresponding pair of IAC effective/real address fields are not set to the same value. For example, if IAC1/IAC2 are operating in one of the range modes, then both DBCR1[IAC1ER] and DBCR1[IAC2ER] must be set to the same value.

IAC Range Mode Auto-Toggle Field

DBCR1[IAC12AT, IAC34AT] control the auto-toggle mechanism for the IAC1/IAC2 and IAC3/IAC4 events, respectively. When the IAC mode for one of the pairs of IAC debug events is set to one of the range modes (either range inclusive or range exclusive), then the IAC range

mode auto-toggle field corresponding to that pair of IAC debug events controls whether or not the range mode will automatically "toggle" from inclusive to exclusive, and vice-versa. When the IAC range mode auto-toggle field is set to 1, this automatic toggling is enabled; otherwise it is disabled. It is a programming error (and the results of any instruction address comparison are undefined) if an IAC range mode auto-toggle field is set to 1 without the corresponding IAC mode field being set to one of the range modes.

When auto-toggle is enabled for a pair of IAC debug events, then upon each occurrence of an IAC debug event within that pair the value of the corresponding auto-toggle status field in the DBSR (DBSR[IAC12ATS, IAC34ATS]) is reversed. That is, if the auto-toggle status field is 0 before the occurrence of the IAC debug event, then it will be changed to 1 at the same time that the IAC debug event is recorded in the DBSR. Conversely, if the auto-toggle status field is 1 before the occurrence of the IAC debug event, then it will be changed to 0 at the same time that the IAC debug event is recorded in the DBSR.

Furthermore, when auto-toggle is enabled, the auto-toggle status field of the DBSR affects the interpretation of the IAC mode field of DBCR1. If the auto-toggle status field is 0, then the IAC mode field is interpreted in the normal fashion, as defined in ~~"IAC Mode Field" on page 15-7~~*IAC Mode Field* on page 434. That is, the IAC mode field value of 0b10 selects range inclusive mode, whereas the value of 0b11 selects range exclusive mode. On the other hand, when the auto-toggle status field is 1, then the interpretation of the IAC mode field is "reversed". That is, the IAC mode field value of 0b10 selects range exclusive mode, whereas the value of 0b11 selects range inclusive mode.

The relationship of the IAC mode, IAC range mode auto-toggle, and IAC range mode auto-toggle status fields is summarized in Table 15-3

Table 15-3. IAC Range Mode Auto-Toggle Summary

DBCR1 IAC12M/IAC34M	DBCR1 IAC12AT/IAC34AT	DBSR IAC12ATS/IAC34ATS	IAC Mode
0b10	0	—	Range Inclusive
0b10	1	0	Range Inclusive
0b10	1	1	Range Exclusive
0b11	0	—	Range Exclusive
0b11	1	0	Range Exclusive
0b11	1	1	Range Inclusive

The affect of the auto-toggle mechanism is to cause the IAC mode to switch back and forth between range inclusive mode and range exclusive mode, as each IAC range mode debug event occurs. For example, if the IAC mode is set to range inclusive, and auto-toggle is enabled, and the auto-toggle status field is 0, then the first IAC debug event will be a range inclusive event. Upon that event, the DBSR auto-toggle status field will be set to 1, and the next IAC debug event will then be a range exclusive event. Upon this next event, the DBSR auto-toggle status field will be set back to 0, such that the next IAC debug event will again be a range inclusive event.

This auto-toggling between range inclusive and range exclusive IAC modes is particularly helpful when enabling IAC range mode debug events in trace debug mode. A common debug operation is to detect when the instruction stream enters a particular region of the instruction address space (range inclusive mode). Once having entered the region of interest (a range inclusive event), it is common for the debugger to then want to be informed when that region is exited (a range exclusive event). By automatically toggling to range exclusive mode upon the occurrence of the range inclusive IAC debug event, this particular debug operation is facilitated. Furthermore, by

not remaining in range inclusive mode upon entry to the region of interest, the debugger avoids a continuous stream of range inclusive IAC debug events while the processor continues to execute instructions within that region, which can often be for a very long series of instructions.

15.4.1.2 IAC Debug Event Processing

When operating in external debug mode or debug wait mode, the occurrence of an IAC debug event is recorded in the corresponding bit of the DBSR and causes the instruction execution to be suppressed. The processor then enters the stop state and ceases the processing of instructions. The program counter will contain the address of the instruction which caused the IAC debug event. Similarly, when operating in internal debug mode with Debug interrupts enabled (MSR[DE] = 1), the occurrence of an IAC debug event is recorded in the DBSR and causes the instruction execution to be suppressed. A Debug interrupt then occurs with Critical Save/Restore Register 0 (CSRR0) set to the address of the instruction which caused the IAC debug event.

When operating in internal debug mode (and not also in external debug mode nor debug wait mode) with Debug interrupts disabled (MSR[DE] = 0), the behavior of IAC debug events depends on the IAC mode. If the IAC mode is set to exact comparison, then an IAC debug event can occur and will set the corresponding IAC field of the DBSR, along with the Imprecise Debug Event (IDE) field of the DBSR. The instruction execution is not suppressed, as no Debug interrupt will occur immediately. Instead, instruction execution continues, and a Debug interrupt will occur if and when MSR[DE] is set to 1, thereby enabling Debug interrupts, assuming software has not cleared the IAC debug event status from the DBSR in the meantime. Upon such a "delayed" interrupt, the Debug interrupt handler software may query the DBSR[IDE] field to determine that the Debug interrupt has occurred imprecisely. On the other hand, if the IAC mode is set to either range inclusive or range exclusive mode, then IAC debug events cannot occur when operating in internal debug mode with MSR[DE] = 0, unless external debug mode and/or debug wait mode is also enabled.

When operating in trace mode, the occurrence of an IAC debug event simply sets the corresponding IAC field of the DBSR and is indicated over the trace interface, and instruction execution continues.

15.4.2 Data Address Compare (DAC) Debug Event

DAC debug events occur when execution is attempted of a load, store, or cache management instruction for which the data storage address and other parameters match the DAC conditions specified by DBCR0, DBCR2, and the DAC registers. There are two DAC registers on the PPC440GP, DAC1 and DAC2. Depending on the DAC mode specified by DBCR2, these DAC registers can be used to specify two independent, exact DAC addresses, or they can be configured to operate as a pair. When operating as a pair, then can specify either a *range* of data storage addresses for which DAC debug events should occur, or a combination of an address and an *address bit mask* for selective comparison with the data storage address.

Note that for integer load and store instructions, and for cache management instructions, the address that is used in the DAC comparison is the starting data address calculated as part of the instruction execution. As explained in the instruction definitions for the cache management instructions, the target operand of these instructions is an aligned cache block, which on the PPC440GP is 32 bytes. Therefore, the storage reference for these instructions effectively ignores the low-order five bits of the calculated data address, and the entire aligned 32-byte cache block—which starts at the calculated data address as modified with the low-order five bits set to 0b00000—is accessed. However, the DAC comparison does not take into account this implicit 32-byte alignment of the storage reference of a cache management instruction, and instead the DAC comparison considers the entire data address, as calculated according to the instruction definition.

On the other hand, for auxiliary processor load and store instructions, the AP interface can specify that the PPC440GP should *force* the storage access to be aligned on an operand-size boundary, by zeroing the appropriate number of low-order address bits. In such a case, the DAC comparison is performed against this modified, alignment-forced address, rather than the original address as calculated according to the instruction definition.

15.4.2.1 DAC Debug Event Fields

There are several fields in DBCR0 and DBCR2 which are used to specify the DAC conditions, as follows:

DAC Event Enable Field

DBCR0[DAC1R, DAC1W, DAC2R, DAC2W] are the individual DAC event enables for the two DAC events DAC1 and DAC2. For each of the two DAC events, there is one enable for DAC read events, and another for DAC write events. Load, **dcbt**, **dcbtst**, **icbi**, and **icbt** instructions may cause DAC read events, while store, **dcbst**, **dcbf**, **dcbi**, and **dcbz** instructions may cause DAC write events (see ~~"DAC Debug Events Applied to Various Instruction Types" on page 15-15~~*DAC Debug Events Applied to Various Instruction Types* on page 442 for more information on these instructions and the types of DAC debug events they may cause). For a given DAC event to occur, the corresponding DAC event enable bit in DBCR0 for the particular operation type must be set. When a DAC event occurs, the corresponding DBSR[DAC1R, DAC1W, DAC2R, DAC2W] bit is set. These same DBSR bits are shared by DVC debug events (see ~~"Data Value Compare (DVC) Debug Event" on page 15-16~~*Data Value Compare (DVC) Debug Event* on page 444).

DAC Mode Field

DBCR2[DAC12M] controls the comparison mode for the DAC1 and DAC2 events. There are four comparison modes supported by the PPC440GP:

- Exact comparison mode (DBCR2[DAC12M] = 0b00)

  In this mode, the data address is compared to the value in the corresponding DAC register, and the DAC event occurs only if the comparison is an exact match.

- Address bit mask mode (DBCR2[DAC12M] = 0b01)

  In this mode, the DAC1 or DAC2 event occurs only if the data address matches the value in the DAC1 register, as masked by the value in the DAC2 register. That is, the DAC1 register specifies an address value, and the DAC2 register specifies an *address bit mask* which determines which bit of the data address should participate in the comparison to the DAC1 value. For every bit set to 1 in the DAC2 register, the corresponding data address bit must match the value of the same bit position in the DAC1 register. For every bit set to 0 in the DAC2 register, the corresponding address bit comparison does not affect the result of the DAC event determination.

  This comparison mode is useful for detecting accesses to a particular byte address, when the accesses may be of various sizes. For example, if the debugger is interested in detecting accesses to byte address 0x00000003, then these accesses may occur due to a byte access to that specific address, or due to a halfword access to address 0x00000002, or due to a word access to address 0x00000000. By using address bit mask mode and specifying that the low-order two bits of the address should be ignored (that is, setting the address bit mask in DAC2 to 0xFFFFFFFC), the debugger can detect each of these types of access to byte address 0x00000003.

When the data address matches the address bit mask mode conditions, either one or both of the DAC debug event bits corresponding to the operation type (read or write) will be set in the DBSR, as determined by which of the corresponding two DAC event enable bits are set in DBCR0. That is, when an address bit mask mode DAC debug event occurs, the setting of DBCR2[DAC1R, DAC1W, DAC2R, DAC2W] will determine whether one or the other or both of the DBSR[DAC1R, DAC1W, DAC2R, DAC2W] bits corresponding to the operation type are set. It is a programming error to set the DAC mode field to address bit mask mode without also enabling at least one of the four DAC event enable bits in DBCR0.

- Range inclusive comparison mode (DBCR2[DAC12M] = 0b10)

  In this mode, the DAC1 or DAC2 event occurs only if the data address is *within* the range defined by the DAC1 and DAC2 register values, as follows: DAC1 ≤ address < DAC2.

  When the data address falls within the specified range, either one or both of the DAC debug event bits corresponding to the operation type (read or write) will be set in the DBSR, as determined by which of the corresponding two DAC event enable bits are set in DBCR0. That is, when a range inclusive mode DAC debug event occurs, the setting of DBCR2[DAC1R, DAC1W, DAC2R, DAC2W] will determine whether one or the other or both of the DBSR[DAC1R, DAC1W, DAC2R, DAC2W] bits corresponding to the operation type are set. It is a programming error to set the DAC mode field to a range comparison mode (either inclusive or exclusive) without also enabling at least one of the four DAC event enable bits in DBCR0.

- Range exclusive comparison mode (DBCR2[DAC12M] = 0b11)

  In this mode, the DAC1 or DAC2 event occurs only if the data address is *outside* the range defined by the DAC1 and DAC2 register values, as follows: address < DAC1 or address ≥ DAC2.

  When the data address falls outside the specified range, either one or both of the DAC debug event bits corresponding to the operation type (read or write) will be set in the DBSR, as determined by which of the corresponding two DAC event enable bits are set in DBCR0. That is, when a range exclusive mode DAC debug event occurs, the setting of DBCR2[DAC1R, DAC1W, DAC2R, DAC2W] will determine whether one or the other or both of the DBSR[DAC1R, DAC1W, DAC2R, DAC2W] bits corresponding to the operation type are set. It is a programming error to set the DAC mode field to a range comparison mode (either inclusive or exclusive) without also enabling at least one of the four DAC event enable bits in DBCR0.

DAC User/Supervisor Field

DBCR2[DAC1US, DAC2US] are the individual DAC user/supervisor fields for the two DAC events. The DAC user/supervisor fields specify what operating mode the processor must be in order for the corresponding DAC event to occur. The operating mode is determined by the Problem State field of the Machine State Register (MSR[PR]; see ~~"User and Supervisor Modes" on page 4-61~~*User and Supervisor Modes* on page 199). When the DAC user/supervisor field is 0b00, the operating mode does not matter—the DAC debug event may occur independent of the state of MSR[PR]. When this field is 0b10, the processor must be operating in supervisor mode (MSR[PR] = 0). When this field is 0b11, the processor must be operating in user mode (MSR[PR] = 1). The DAC user/supervisor field value of 0b01 is reserved.

If the DAC mode is set to one of the "paired" modes (address bit mask mode, or one of the two range modes), it is a programming error (and the results of any data address comparison are undefined) if DBCR2[DAC1US] and DBCR2[DAC2US] are not set to the same value.

DAC Effective/Real Address Field

DBCR2[DAC1ER, DAC2ER] are the individual DAC effective/real address fields for the two DAC events. The DAC effective/real address fields specify whether the instruction address comparison should be performed using the effective, virtual, or real address (see ~~Chapter 6, "Memory Management"~~*Memory Management* on page 233) for an explanation of these different types of addresses). When the DAC effective/real address field is 0b00, the comparison is performed using the effective address only; the DAC debug event may occur independent of the data address space (MSR[DS]). When this field is 0b10, the DAC debug event occurs only if the effective address matches the DAC conditions and is in virtual address space 0 (MSR[DS] = 0). Similarly, when this field is 0b11, the DAC debug event occurs only if the effective address matches the DAC conditions and is in virtual address space 1 (MSR[DS] = 1). Note that in these latter two modes, in which the virtual address space of the data is considered, it is not the entire virtual address which is considered. The Process ID, which forms the final part of the virtual address, is not considered. Finally, the DAC effective/real address field value of 0b01 is reserved, and corresponds to the PowerPC Book-E architected real address comparison mode, which is not supported by the PPC440GP.

If the DAC mode is set to one of the "paired" modes (address bit mask mode, or one of the two range modes), it is a programming error (and the results of any data address comparison are undefined) if DBCR2[DAC1ER] and DBCR2[DAC2ER] are not set to the same value.

DVC Byte Enable Field

DBCR2[DVC1BE, DVC2BE] are the individual data *value* compare (DVC) byte enable fields for the two DVC events. These fields must be disabled (by being set to 4b0000) in order for the corresponding DAC debug event to be enabled. In other words, when any of the DVC byte enable field bits for a given DVC event are set to 1, the corresponding DAC event is disabled, and the various DAC field conditions are used in conjunction with the DVC field conditions to determine whether a DVC event should occur. See ~~"Data Value Compare (DVC) Debug Event" on page 15-16~~*Data Value Compare (DVC) Debug Event* on page 444 for more information on DVC events.

15.4.2.2 DAC Debug Event Processing

The behavior of the PPC440GP upon a DAC debug event depends on the setting of DBCR2[DAC12A]. This field of DBCR2 controls whether DAC debug events are processed in a *synchronous* (DBCR2[DAC12A] = 0) or an *asynchronous* (DBCR2[DAC12A] = 1) fashion.

DBCR2[DAC12A] = 0 (Synchronous Mode)

When operating in external debug mode or debug wait mode, the occurrence of a DAC debug event is recorded in the corresponding bit of the DBSR and causes the instruction execution to be suppressed. The processor then enters the stop state and ceases the processing of instructions. The program counter will contain the address of the instruction which caused the DAC debug event. Similarly, when operating in internal debug mode with Debug interrupts enabled (MSR[DE] = 1), the occurrence of a DAC debug event is recorded in the DBSR and causes the instruction execution to be suppressed. A Debug interrupt will occur with CSRR0 set to the address of the instruction which caused the DAC debug event.

When operating in internal debug mode (and not also in external debug mode nor debug wait mode) with Debug interrupts disabled (MSR[DE] = 0), then a DAC debug event will set the corresponding DAC field of the DBSR, along with the Imprecise Debug Event (IDE) field of the DBSR. The instruction execution is not suppressed, as no Debug interrupt will occur immediately.

Instead, instruction execution continues, and a Debug interrupt will occur if and when MSR[DE] is set to 1, thereby enabling Debug interrupts, assuming software has not cleared the DAC debug event status from the DBSR in the meantime. Upon such a "delayed" interrupt, the Debug interrupt handler software may query the DBSR[IDE] field to determine that the Debug interrupt has occurred imprecisely.

When operating in trace mode, the occurrence of a DAC debug event simply sets the corresponding DAC field of the DBSR and is indicated over the trace interface, and instruction execution continues. DBCR2[DAC12A] does not affect the processing of DAC debug events when operating in trace mode.

Engineering Note: When DAC debug events are enabled in any debug mode other than trace mode, and DBCR2[DAC12A] is set to 0 (synchronous mode), in order for the PPC440GP to deal with a DAC-related Debug interrupt in a synchronous fashion, the processing of all potential DAC debug event-causing instructions (loads, stores, and cache management instructions) is impacted by one processor cycle. This one cycle impact occurs whether or not the instruction is actually causing a DAC debug event. Overall processor performance is thus significantly impacted if synchronous mode DAC debug events are enabled. In order to maintain normal processor performance while DAC debug events are enabled *and in the absence of any actual DAC debug events*, software should set DBCR2[DAC12A] to 1.

DBCR2[DAC12A] = 1 (Asynchronous Mode)

When operating in external debug mode or debug wait mode, the occurrence of a DAC debug event is recorded in the corresponding bit of the DBSR and causes the processor to enter stop state and cease processing instructions. However, the determination and processing of the DAC debug event is not handled synchronously with respect to the instruction execution. That is, the processor may process the DAC debug event and enter the stop state either before or after the completion of the instruction causing the event. If the DAC debug event is processed *before* the completion of the instruction causing the event, then upon entering the stop state the program counter will contain the address of that instruction, and that instruction's execution will have been suppressed. Conversely, if the DAC debug event is processed *after* the completion of the instruction causing the event, then the program counter will contain the address of some instruction after the one which caused the event. Whether or not the DAC debug event processing occurs before or after the completion of the instruction depends on the particular circumstances surrounding the instruction's execution, ~~and is~~ the details of which are generally beyond the scope of this document.

Similarly, when operating in internal debug mode with Debug interrupts enabled (MSR[DE] = 1), the occurrence of a DAC debug event is recorded in the DBSR and will generate a Debug interrupt with CSRR0 set to the address of the instruction which caused the DAC debug event, or to the address of some subsequent instruction, depending upon whether the event is processed before or after the instruction completes.

When operating in internal debug mode (and not also in external debug mode nor debug wait mode) with Debug interrupts disabled (MSR[DE] = 0), then a DAC debug event will set the corresponding DAC field of the DBSR, along with the Imprecise Debug Event (IDE) field of the DBSR. Instruction execution continues, and a Debug interrupt will occur if and when MSR[DE] is set to 1, thereby enabling Debug interrupts, assuming software has not cleared the DAC debug event status from the DBSR in the meantime. Upon such a "delayed" interrupt, the Debug interrupt handler software may query the DBSR[IDE] field to determine that the Debug interrupt

has occurred imprecisely.

When operating in trace mode, the occurrence of a DAC debug event simply sets the corresponding DAC field of the DBSR and is indicated over the trace interface, and instruction execution continues. DBCR2[DAC12A] does not affect the processing of DAC debug events when operating in trace mode.

15.4.2.3 DAC Debug Events Applied to Instructions that Result in Multiple Storage Accesses

Certain misaligned load and store instructions are handled by making multiple, independent storage accesses. Similarly, load and store multiple and string instructions which access more than one register result in more than one storage access. ~~"Load and Store Alignment" on page 5-19~~*Load and Store Alignment* on page 222 provides a detailed description of the circumstances that lead to such multiple storage accesses being made as the result of the execution of a single instruction.

Whenever the execution of a given instruction results in multiple storage accesses, the data address of each access is independently considered for whether or not it will cause a DAC debug event.

15.4.2.4 DAC Debug Events Applied to Various Instruction Types

Various special cases apply to the cache management instructions, the store word conditional indexed (**stwcx.**) instruction, and the load and store string indexed (**lswx**, **stswx**) instructions, with regards to DAC debug events. These special cases are as follows:

dcbz, **dcbi**

The **dcbz** and **dcbi** instructions are considered "stores" with respect to both storage access control and DAC debug events. The **dcbz** instruction directly changes the contents of a given storage location, whereas the **dcbi** instruction can indirectly change the contents of a given storage location by invalidating data which has been modified within the data cache, thereby "restoring" the value of the location to the "old" contents of memory. As "store" operations, they may cause DAC write debug events.

dcbst, **dcbf**

The **dcbst** and **dcbf** instructions are considered "loads" with respect to storage access control, since they do not change the contents of a given storage location. They may merely cause the data at that storage location to be moved from the data cache out to memory. However, in a debug environment, the fact that these instructions may lead to write operations on the external interface is typically the event of interest. Therefore, these instructions are considered "stores" with respect to DAC debug events, and may cause DAC write debug events.

dcbt, **dcbtst**, **icbt**

The *touch* instructions are considered "loads" with respect to both storage access control and DAC debug events. However, these instructions are treated as no-ops if they reference caching inhibited storage locations, or if they cause Data Storage or Data TLB Miss exceptions. Consequently, if a *touch* instruction is being treated as a no-op for one of these reasons, then it does not cause a DAC read debug event. However, if a *touch* instruction is not being treated as a no-op for one of these reasons, it may cause a DAC read debug event.

dcba

The **dcba** instruction is treated as a no-op by the PPC440GP, and thus will not cause a DAC debug event.

icbi

The **icbi** instruction is considered a "load" with respect to both storage access control and DAC debug events, and thus may cause a DAC read debug event.

dccci, **dcread**, **iccci**, **icread**

The **dccci** and **iccci** instructions do not generate an address, but rather they affect the entire data and instruction cache, respectively. Similarly, the **dcread** and **icread** instructions do not generate an address, but rather an "index" which is used to select a particular location in the respective cache, without regard to the storage address represented by that location. Therefore, none of these instructions cause DAC debug events.

stwcx.

If the execution of a **stwcx.** instruction would otherwise have caused a DAC write debug event, but the processor does not have the reservation from a **lwarx** instruction, then the DAC write debug event does not occur since the storage location does not get written.

lswx, **stswx**

DAC debug events do not occur for **lswx** or **stswx** instructions with a length of 0 (XER[TBC] = 0), since these instructions do not actually access storage.

15.4.3 Data Value Compare (DVC) Debug Event

DVC debug events occur when execution is attempted of a load, store, or **dcbz** instruction for which the data storage address and other parameters match the DAC conditions specified by DBCR0, DBCR2, and the DAC registers, and for which the data accessed matches the DVC conditions specified by DBCR2 and the DVC registers. In other words, in order for a DVC debug event to occur, the conditions for a DAC debug event must first be met, and then the data must also match the DVC conditions. ~~"Data Address Compare (DAC) Debug Event" on page 15-10~~*Data Address Compare (DAC) Debug Event* on page 438 describes the DAC conditions. In addition to the DAC conditions, there are two DVC registers on the PPC440GP, DVC1 and DVC2. The DVC registers can be used to specify two independent, 4-byte data values, which are selectively compared against the data being accessed by a given load, store, or cache management instruction.

When a DVC event occurs, the corresponding DBSR[DAC1R, DAC1W, DAC2R, DAC2W] bit is set. These same DBSR bits are shared by DAC debug events.

15.4.3.1 DVC Debug Event Fields

In addition to the DAC debug event fields described in ~~"DAC Debug Event Fields" on page 15-11~~*DAC Debug Event Fields* on page 438, and the DVC registers themselves, there are two fields in DBCR2 which are used to specify the DVC conditions, as follows:

DVC Byte Enable Field

DBCR2[DVC1BE, DVC2BE] are the individual DVC byte enable fields for the two DVC events. When one or the other (or both) of these fields is disabled (by being set to 4b0000), the corresponding DVC debug event is disabled (the corresponding DAC debug event may still be

enabled, as determined by the DAC debug event enable field of DBCR0). When either one or both of these fields is enabled (by being set to a non-zero value), then the corresponding DVC debug event is enabled.

Each bit of a given DVC byte enable field corresponds to a byte position within an aligned word of memory. For a given aligned word of memory, the byte offsets (or "byte lanes") within that word are numbered 0, 1, 2, and 3, starting from the left-most (most significant) byte of the word. Accordingly, bits 0:3 of a given DVC byte enable field correspond to bytes 0:3 of an aligned word of memory being accessed.

For an access to "match" the DVC conditions for a given byte, the access must be actually transferring data on that given byte position *and* the data must match the corresponding byte value within the DVC register.

For each storage access, the DVC comparison is made against the bytes that are being accessed within the aligned word of memory containing the starting byte of the transfer. For example, consider a load word instruction with a starting data address of x01. The four bytes from memory are located at addresses 0x01–0x04, but the aligned word of memory containing the starting byte consists of addresses 0x00–0x03. Thus the only bytes being accessed within the aligned word of memory containing the starting byte are the bytes at addresses 0x01–0x03, and only these bytes are considered in the DVC comparison. The byte transferred from address 0x04 is not considered.

DVC Mode Field

DBCR2[DVC1M, DVC2M] are the individual DVC mode fields for the two DVC events. Each one of these fields specifies the particular data value comparison mode for the corresponding DVC debug event. There are three comparison modes supported by the PPC440GP:

- AND comparison mode (DBCR2[DVC1M, DVC2M] = 0b01)

  In this mode, all data byte lanes enabled by a DVC byte enable field must be being accessed and must match the corresponding byte data value in the corresponding DVC1 or DVC2 register.

- OR comparison mode (DBCR2[DVC1M, DVC2M] = 0b10)

  In this mode, at least one data byte lane that is enabled by a DVC byte enable field must be being accessed and must match the corresponding byte data value in the corresponding DVC1 or DVC2 register.

- AND-OR comparison mode (DBCR2[DVC1M, DVC2M] = 0b11)

  In this mode, the four byte lanes of an aligned word are divided into two pairs, with byte lanes 0 and 1 being in one pair, and byte lanes 2 and 3 in the other pair. The DVC comparison mode for each pair of byte lanes operates in AND mode, and then the results of these two AND mode comparisons are ORed together to determine whether a DVC debug event occurs. In other words, a DVC debug event occurs if either one or both of the pairs of byte lanes satisfy the AND mode comparison requirements.

  This mode may be used to cause a DVC debug event upon an access of a particular halfword data value in either of the two halfwords of a word in memory.

15.4.3.2 DVC Debug Event Processing

The behavior of the PPC440GP upon a DVC debug event depends on the setting of DBCR2[DAC12A]. This field of DBCR2 controls whether DVC debug events are processed in a *synchronous* (DBCR2[DAC12A] = 0) or an *asynchronous* (DBCR2[DAC12A] = 1) fashion. The processing of DVC debug events is the same as it is for DAC debug events. See ~~"DAC Debug Event Processing" on page 15-13~~*DAC Debug Event Processing* on page 441 for more information.

15.4.3.3 DVC Debug Events Applied to Instructions that Result in Multiple Storage Accesses

Certain misaligned load and store instructions are handled by making multiple, independent storage accesses. Similarly, load and store multiple and string instructions which access more than one register result in more than one storage access. ~~"Load and Store Alignment" on page 5-19~~*Load and Store Alignment* on page 222 provides a detailed description of the circumstances that lead to such multiple storage accesses being made as the result of the execution of a single instruction.

Whenever the execution of a given instruction results in multiple storage accesses, the address and data of each access is independently considered for whether or not it will cause a DVC debug event.

15.4.3.4 DVC Debug Events Applied to Various Instruction Types

Various special cases apply to the cache management instructions, the store word conditional indexed (**stwcx.**) instruction, and the load and store string indexed (**lswx**, **stswx**) instructions, with regards to DVC debug events. These special cases are as follows:

dcbz

The **dcbz** instruction is the only cache management instruction which can cause a DVC debug event. **dcbz** is the only such instruction which actually writes new data to a storage location (in this case, an entire 32-byte data cache line is written to zeroes).

stwcx.

If the execution of a **stwcx.** instruction would otherwise have caused a DVC write debug event, but the processor does not have the reservation from a **lwarx** instruction, then the DVC write debug event does not occur since the storage location does not get written.

lswx, **stswx**

DVC debug events do not occur for **lswx** or **stswx** instructions with a length of 0 (XER[TBC] = 0), since these instructions do not actually access storage.

15.4.4 Branch Taken (BRT) Debug Event

BRT debug events occur when BRT debug events are enabled (DBCR0[BRT] = 1) and execution is attempted of a branch instruction for which the branch condition(s) are satisfied, such that the instruction stream will be redirected to the target address of the branch.

When operating in external debug mode or debug wait mode, the occurrence of a BRT debug event is recorded in DBSR[BRT] and causes the instruction execution to be suppressed. The processor then enters the stop state and ceases the processing of instructions. The program counter will contain the address of the branch instruction which caused the BRT debug event. Similarly, when operating in internal debug mode with

Debug interrupts enabled (MSR[DE] = 1), the occurrence of a BRT debug event is recorded in DBSR[BRT] and causes the instruction execution to be suppressed. A Debug interrupt will occur with CSRR0 set to the address of the branch instruction which caused the BRT debug event.

When operating in internal debug mode (and not also in external debug mode nor debug wait mode) with Debug interrupts disabled (MSR[DE] = 0), then BRT debug events cannot occur. Since taken branches are a very common operation and thus likely to be frequently executed within the critical class interrupt handlers (which typically have MSR[DE] set to 0), allowing BRT debug events under these conditions would lead to an undesirable number of delayed (and hence imprecise) Debug interrupts.

When operating in trace mode, the occurrence of a BRT debug event is simply recorded in DBSR[BRT] and is indicated over the trace interface, and instruction execution continues.

15.4.5 Trap (TRAP) Debug Event

TRAP debug events occur when TRAP debug events are enabled (DBCR0[TRAP] = 1) and execution is attempted of a trap (**tw**, **twi**) instruction for which the trap condition is satisfied.

When operating in external debug mode or debug wait mode, the occurrence of a TRAP debug event is recorded in DBSR[TRAP] and causes the instruction execution to be suppressed. The processor then enters the stop state and ceases the processing of instructions. The program counter will contain the address of the trap instruction which caused the TRAP debug event. Similarly, when operating in internal debug mode with Debug interrupts enabled (MSR[DE] = 1), the occurrence of a TRAP debug event is recorded in DBSR[TRAP] and causes the instruction execution to be suppressed. A Debug interrupt will occur with CSRR0 set to the address of the trap instruction which caused the TRAP debug event.

When operating in internal debug mode (and not also in external debug mode nor debug wait mode) with Debug interrupts disabled (MSR[DE] = 0), the occurrence of a TRAP debug event will set DBSR[TRAP], along with the Imprecise Debug Event (IDE) field of the DBSR. Although a Debug interrupt will not occur immediately, the instruction execution is suppressed as a Trap exception type Program interrupt will occur instead. A Debug interrupt will also occur later, if and when MSR[DE] is set to 1, thereby enabling Debug interrupts, assuming software has not cleared the TRAP debug event status from the DBSR in the meantime. Upon such a "delayed" interrupt, the Debug interrupt handler software may query the DBSR[IDE] field to determine that the Debug interrupt has occurred imprecisely.

When operating in trace mode, the occurrence of a TRAP debug event is simply recorded in DBSR[TRAP] and is indicated over the trace interface, and instruction execution continues.

15.4.6 Return (RET) Debug Event

RET debug events occur when RET debug events are enabled (DBCR0[RET] = 1) and execution is attempted of a return (**rfi**, **rfci**) instruction.

When operating in external debug mode or debug wait mode, the occurrence of a RET debug event is recorded in DBSR[RET] and causes the instruction execution to be suppressed. The processor then enters the stop state and ceases the processing of instructions. The program counter will contain the address of the return instruction which caused the RET debug event. Similarly, when operating in internal debug mode with Debug interrupts enabled (MSR[DE] = 1), the occurrence of a RET debug event is recorded in DBSR[RET] and causes the instruction execution to be suppressed. A Debug interrupt will occur with CSRR0 set to the address of the return instruction which caused the RET debug event.

When operating in internal debug mode (and not also in external debug mode nor debug wait mode) with Debug interrupts disabled (MSR[DE] = 0), then RET debug events can occur only for **rfi** instructions, and not for **rfci** instructions. Since the **rfci** instruction is typically used to return from a critical class interrupt handler (including the Debug interrupt itself), and MSR[DE] is typically 0 at the time of the return, the **rfci** must not be allowed to cause a RET debug event under these conditions, or else it would not be possible to return from the critical class interrupts.

For the **rfi** instruction only, if a RET debug event occurs under these conditions (internal debug mode enabled, external debug mode and debug wait mode disabled, and MSR[DE] = 0), then DBSR[RET] is set, along with the Imprecise Debug Event (IDE) field of the DBSR. The instruction execution is not suppressed, as no Debug interrupt will occur immediately. Instead, instruction execution continues, and a Debug interrupt will occur if and when MSR[DE] is set to 1, thereby enabling Debug interrupts, assuming software has not cleared the RET debug event status from the DBSR in the meantime. Upon such a "delayed" interrupt, the Debug interrupt handler software may query the DBSR[IDE] field to determine that the Debug interrupt has occurred imprecisely.

When operating in trace mode, the occurrence of a RET debug event is simply recorded in DBSR[RET] and is indicated over the trace interface, and instruction execution continues.

15.4.7 Instruction Complete (ICMP) Debug Event

ICMP debug events occur when ICMP debug events are enabled (DBCR0[ICMP] = 1) and the PPC440GP completes the execution of any instruction.

When operating in external debug mode or debug wait mode, the occurrence of an ICMP debug event is recorded in DBSR[ICMP] and causes the processor to enter the stop state and cease processing instructions. The program counter will contain the address of the instruction which would have executed next, had the ICMP debug event not occurred. Note that if the instruction whose completion caused the ICMP debug event was a branch instruction (and the branch conditions were satisfied), then upon entering the stop state the program counter will contain the target of the branch, and not the address of the instruction that is sequentially after the branch. Similarly, if the ICMP debug event is caused by the execution of a return (**rfi**, **rfci**) instruction, then upon entering the stop state the program counter will contain the address being *returned to*, and not the address of the instruction which is sequentially after the return instruction.

When operating in internal debug mode with Debug interrupts enabled (MSR[DE] = 1), the occurrence of an ICMP debug event is recorded in DBSR[ICMP] and a Debug interrupt will occur with CSRR0 set to the address of the instruction which would have executed next, had the ICMP debug event not occurred. Note that there is a special case of MSR[DE] = 1 at the time of the execution of the instruction causing the ICMP debug event, but that instruction itself sets MSR[DE] to 0. This special case is described in more detail in ~~"Debug Interrupt" on page 10-35~~*Debug Interrupt* on page 369, in the subsection on the setting of CSRR0.

When operating in internal debug mode (and not also in external debug mode nor debug wait mode) with Debug interrupts disabled (MSR[DE] = 0), then ICMP debug events cannot occur. Since the code at the beginning of the critical class interrupt handlers (including the Debug interrupt itself) must execute at least temporarily with MSR[DE] = 0, there would be no way to avoid causing additional ICMP debug events and setting DBSR[IDE], if ICMP debug events were allowed to occur under these conditions.

The PPC440GP does not support the use of the ICMP debug event when operating in trace mode. Software must not enable ICMP debug events unless one of the other debug modes is enabled as well.

15.4.8 Interrupt (IRPT) Debug Event

IRPT debug events occur when IRPT debug events are enabled (DBCR0[IRPT] = 1) and an interrupt occurs.

When operating in external debug mode or debug wait mode, the occurrence of an IRPT debug event is recorded in DBSR[IRPT] and causes the processor to enter the stop state and cease processing instructions. The program counter will contain the address of the instruction which would have executed next, had the IRPT debug event not occurred. Since the IRPT debug event is caused by the occurrence of an interrupt, by definition this address is that of the first instruction of the interrupt handler for the interrupt type which caused the IRPT debug event.

When operating in internal debug mode with external debug mode and debug wait mode both disabled (and regardless of the value of MSR[DE]), an IRPT debug event can only occur due to a non-critical class interrupt. Critical class interrupts (Machine Check, Critical Input, Watchdog Timer, and Debug interrupts) cannot cause IRPT debug events in internal debug mode (unless also in external debug mode or debug wait mode), as otherwise the Debug interrupt which would occur as the result of the IRPT debug event would by necessity always be imprecise, since the critical class interrupt which would be causing the IRPT debug event would itself be causing MSR[DE] to be set to 0.

For a non-critical class interrupt which is causing an IRPT debug event while internal debug mode is enabled and external debug mode and debug wait mode are both disabled, the occurrence of the IRPT debug event is recorded in DBSR[IRPT]. If MSR[DE] is 1 at the time of the IRPT debug event, then a Debug interrupt occurs with CSRR0 set to the address of the instruction which would have executed next, had the IRPT debug event not occurred. Since the IRPT debug event is caused by the occurrence of some other interrupt, by definition this address is that of the first instruction of the interrupt handler for the interrupt type which caused the IRPT debug event. If MSR[DE] is 0 at the time of the IRPT debug event, then the Imprecise Debug Event (IDE) field of the DBSR is also set and a Debug interrupt does not occur immediately. Instead, instruction execution continues, and a Debug interrupt will occur if and when MSR[DE] is set to 1, thereby enabling Debug interrupts, assuming software has not cleared the IRPT debug event status from the DBSR in the meantime. Upon such a "delayed" interrupt, the Debug interrupt handler software may query the DBSR[IDE] field to determine that the Debug interrupt has occurred imprecisely.

When operating in trace mode, the occurrence of an IRPT debug event is simply recorded in DBSR[IRPT] and is indicated over the trace interface, and instruction execution continues.

15.4.9 Unconditional Debug Event (UDE)

UDE debug events occur when a debug tool asserts the unconditional debug event request via the JTAG interface. The UDE debug event is the only event which does not have a corresponding enable field in DBCR0.

When operating in external debug mode or debug wait mode, the occurrence of a UDE debug event is recorded in DBSR[UDE] and causes the processor to enter the stop state and cease processing instructions. The program counter will contain the address of the instruction which would have executed next, had the UDE debug event not occurred. Similarly, when operating in internal debug mode with Debug interrupts enabled (MSR[DE] = 1), the occurrence of a UDE debug event is recorded in DBSR[UDE] and a Debug interrupt will occur with CSRR0 set to the address of the instruction which would have executed next, had the UDE debug event not occurred.

When operating in internal debug mode (and not also in external debug mode nor debug wait mode) with Debug interrupts disabled (MSR[DE] = 0), the occurrence of a UDE debug event will set DBSR[UDE], along with the Imprecise Debug Event (IDE) field of the DBSR. The Debug interrupt will not occur immediately.

Instead, instruction execution continues, and a Debug interrupt will occur if and when MSR[DE] is set to 1, thereby enabling Debug interrupts, assuming software has not cleared the UDE debug event status from the DBSR in the meantime. Upon such a "delayed" interrupt, the Debug interrupt handler software may query the DBSR[IDE] field to determine that the Debug interrupt has occurred imprecisely.

When operating in trace mode, the occurrence of a UDE debug event simply sets DBSR[UDE] and is indicated over the trace interface, and instruction execution continues.

15.4.10 Debug Event Summary

Table 15-4 summarizes each of the debug event types, and the effect of debug mode and MSR[DE] on their occurrence.

Table 15-4. Debug Event Summary

External Debug Mode	Debug Wait Mode	Internal Debug Mode	MSR DE	Debug Events								
				IAC	DAC	DVC	BRT	TRAP	RET	ICMP	IRPT	UDE
Enabled	—	—	—	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
—	Enabled	—	—	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Disabled	Disabled	Enabled	1	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Note 1	Yes
Disabled	Disabled	Enabled	0	Note 2	Yes	Yes	No	Yes	Note 3	No	Note 1	Yes
Disabled	Disabled	Disabled	—	Yes	Yes	Yes	Yes	Yes	Yes	Note 4	yes	Yes

Table Notes

1. IRPT debug events may only occur for non-critical class interrupts when operating in internal debug mode with external debug mode and debug wait mode both disabled.
2. IAC debug events may not occur in internal debug mode with MSR[DE] = 0 and with external debug mode and debug wait mode both disabled, and the IAC mode set to range inclusive or range exclusive. They *may* occur if the IAC mode is set to exact.
3. RET debug events may not occur for **rfci** instructions when operating in internal debug mode with MSR[DE] = 0 and with external debug mode and debug wait mode both disabled. They may only occur in this mode for the **rfi** instruction.
4. ICMP debug events are not permitted when operating in trace debug mode. Software must not enable ICMP debug events unless one of the other debug modes is enabled.

15.5 Debug Reset

Software can initiate an immediate reset operation by setting DBCR0[RST] to a non-zero value. The results of a reset operation within the PPC440GP core are described in ~~Chapter 7, "Reset and Initialization"~~ *Reset and Initialization* on page 259. The results of a reset operation on the rest of the chip and/or system is dependent on the particular type of reset operation (core, chip, or system reset), and on the particular chip and system implementation. See the chip user's manual for details.

15.6 Debug Timer Freeze

In order to maintain the semblance of "real time" operation while a system is being debugged, DBCR0[FT] can be set to 1, which will cause all of the timers within the PPC440GP core to stop incrementing or decrementing for as long as a debug event bit is set in the DBSR, or until DBCR0[FT] is set to 0. See ~~Chapter 11, "Timer Facilities"~~ *Timer Facilities* on page 383 for more information on the operation of the PPC440GP core timers.

15.7 Debug Registers

Various Special Purpose Registers (SPRs) are used to enable the debug modes, to configure and record debug events, and to communicate with debug tool hardware and software. These debug registers may be accessed either through software running on the processor or through the JTAG debug port of the PPC440GP.

Programming Note: It is the responsibility of software to synchronize the context of any changes to the debug facility registers. Specifically, when changing the contents of any of the debug facility registers, software must execute an **isync** instruction both *before* and *after* the changes to these registers, to ensure that all preceding instructions use the *old* values of the registers, and that all succeeding instructions use the *new* values. In addition, when changing any of the debug facility register fields related to the DAC and/or DVC debug events, software must execute an **msync** instruction *before* making the changes, to ensure that all storage accesses complete using the *old* context of these register fields.

15.7.1 Debug Control Register 0 (DBCR0)

DBCR0 is an SPR that is used to enable debug modes and events, reset the processor, and control timer operation when debugging. DBCR0 can be written from a GPR using **mtspr**, and can be read into a GPR using **mfspr**.

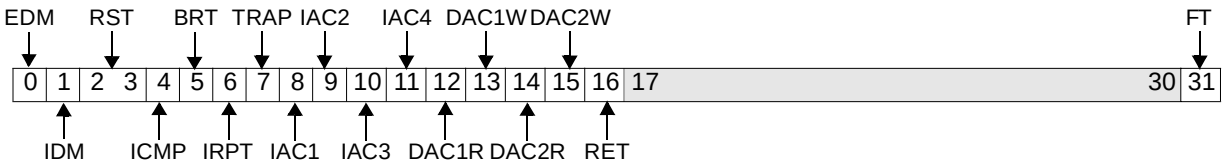


Figure 0-2. Debug Control Register 0 (DBCR0)

0	EDM	External Debug Mode 0 Disable external debug mode. 1 Enable external debug mode.
1	IDM	Internal Debug Mode 0 Disable internal debug mode. 1 Enable internal debug mode.

2:3	RST	Reset 00 No action 01 Core reset 10 Chip reset 11 System reset	
		Attention: Writing 01, 10, or 11 to this field causes a processor reset to occur.	
4	ICMP	Instruction Completion Debug Event 0 Disable instruction completion debug event. 1 Enable instruction completion debug event.	Instruction completions do not cause instruction completion debug events if MSR[DE] = 0 in internal debug mode, unless also in external debug mode or debug wait mode.
5	BRT	Branch Taken Debug Event 0 Disable branch taken debug event. 1 Enable branch taken debug event.	Taken branches do not cause branch taken debug events if MSR[DE] = 0 in internal debug mode, unless also in external debug mode or debug wait mode.
6	IRPT	Interrupt Debug Event 0 Disable interrupt debug event. 1 Enable interrupt debug event.	Critical interrupts do not cause interrupt debug events in internal debug mode, unless also in external debug mode or debug wait mode.
7	TRAP	Trap Debug Event 0 Disable trap debug event. 1 Enable trap debug event.	
8	IAC1	Instruction Address Compare (IAC) 1 Debug Event 0 Disable IAC 1 debug event. 1 Enable IAC 1 debug event.	
9	IAC2	IAC 2 Debug Event 0 Disable IAC 2 debug event. 1 Enable IAC 2 debug event.	
10	IAC3	IAC 3 Debug Event 0 Disable IAC 3 debug event. 1 Enable IAC 3 debug event.	
11	IAC4	IAC 4 Debug Event 0 Disable IAC 4 debug event. 1 Enable IAC 4 debug event.	
12	DAC1R	Data Address Compare (DAC) 1 Read Debug Event 0 Disable DAC 1 read debug event. 1 Enable DAC 1 read debug event.	
13	DAC1W	DAC 1 Write Debug Event 0 Disable DAC 1 write debug event. 1 Enable DAC 1 write debug event.	

14	DAC2R	DAC 2 Read Debug Event 0 Disable DAC 2 read debug event. 1 Enable DAC 2 read debug event.	
15	DAC2W	DAC 2 Write Debug Event 0 Disable DAC 2 write debug event. 1 Enable DAC 2 write debug event.	
16	RET	Return Debug Event 0 Disable return (**rfi/rfci**) debug event. 1 Enable return (**rfi/rfci**) debug event.	**rfci** does not cause a return debug event if MSR[DE] = 0 in internal debug mode, unless also in external debug mode or debug wait mode.
17:30		Reserved	
31	FT	Freeze timers on debug event 0 Timers are not frozen. 1 Freeze timers if a DBSR field associated with a debug event is set.	

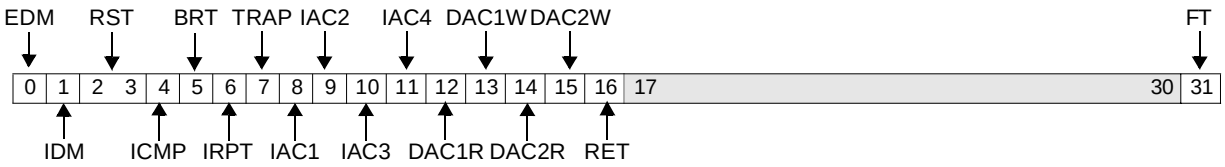


Figure 15-2. Debug Control Register 0 (DBCR0)

0	EDM	External Debug Mode 0 Disable external debug mode. 1 Enable external debug mode.	
1	IDM	Internal Debug Mode 0 Disable internal debug mode. 1 Enable internal debug mode.	
2:3	RST	Reset 00 No action 01 Core reset 10 Chip reset 11 System reset	
		Attention: Writing 01, 10, or 11 to this field causes a processor reset to occur.	
4	ICMP	Instruction Completion Debug Event 0 Disable instruction completion debug event. 1 Enable instruction completion debug event.	Instruction completions do not cause instruction completion debug events if MSR[DE] = 0 in internal debug mode, unless also in external debug mode or debug wait mode.
5	BRT	Branch Taken Debug Event 0 Disable branch taken debug event. 1 Enable branch taken debug event.	Taken branches do not cause branch taken debug events if MSR[DE] = 0 in internal debug mode, unless also in external debug mode or debug wait mode.

6	IRPT	Interrupt Debug Event 0 Disable interrupt debug event. 1 Enable interrupt debug event.	Critical interrupts do not cause interrupt debug events in internal debug mode, unless also in external debug mode or debug wait mode.
7	TRAP	Trap Debug Event 0 Disable trap debug event. 1 Enable trap debug event.	
8	IAC1	Instruction Address Compare (IAC) 1 Debug Event 0 Disable IAC 1 debug event. 1 Enable IAC 1 debug event.	
9	IAC2	IAC 2 Debug Event 0 Disable IAC 2 debug event. 1 Enable IAC 2 debug event.	
10	IAC3	IAC 3 Debug Event 0 Disable IAC 3 debug event. 1 Enable IAC 3 debug event.	
11	IAC4	IAC 4 Debug Event 0 Disable IAC 4 debug event. 1 Enable IAC 4 debug event.	
12	DAC1R	Data Address Compare (DAC) 1 Read Debug Event 0 Disable DAC 1 read debug event. 1 Enable DAC 1 read debug event.	
13	DAC1W	DAC 1 Write Debug Event 0 Disable DAC 1 write debug event. 1 Enable DAC 1 write debug event.	
14	DAC2R	DAC 2 Read Debug Event 0 Disable DAC 2 read debug event. 1 Enable DAC 2 read debug event.	
15	DAC2W	DAC 2 Write Debug Event 0 Disable DAC 2 write debug event. 1 Enable DAC 2 write debug event.	
16	RET	Return Debug Event 0 Disable return (**rfi/rfci**) debug event. 1 Enable return (**rfi/rfci**) debug event.	**rfci** does not cause a return debug event if MSR[DE] = 0 in internal debug mode, unless also in external debug mode or debug wait mode.
17:30		Reserved	
31	FT	Freeze timers on debug event 0 Timers are not frozen. 1 Freeze timers if a DBSR field associated with a debug event is set.	

15.7.2 Debug Control Register 1 (DBCR1)

DBCR1 is an SPR that is used to configure IAC debug events. DBCR1 can be written from a GPR using **mtspr**, and can be read into a GPR using **mfspr**.

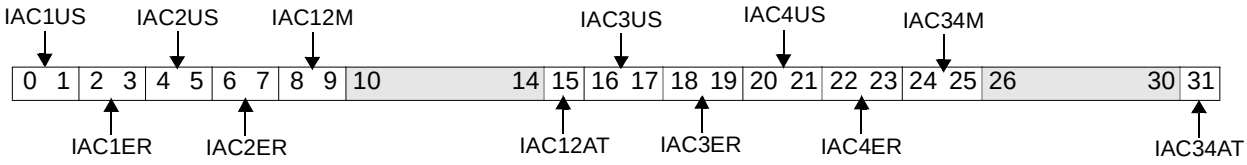


Figure 0-3. Debug Control Register 1 (DBCR1)

0:1	IAC1US	Instruction Address Compare (IAC) 1 User/Supervisor 00 Both 01 Reserved 10 Supervisor only (MSR[PR] = 0) 11 User only (MSR[PR] = 1)
2:3	IAC1ER	IAC 1 Effective/Real 00 Effective (MSR[IS] = don't care) 01 Reserved 10 Virtual (MSR[IS] = 0 11 Virtual (MSR[IS] = 1)
4:5	IAC2US	IAC 2 User/Supervisor 00 Both 01 Reserved 10 Supervisor only (MSR[PR] = 0) 11 User only (MSR[PR] = 1)
6:7	IAC2ER	IAC 2 Effective/Real 00 Effective (MSR[IS] = don't care) 01 Reserved 10 Virtual (MSR[IS] = 0 11 Virtual (MSR[IS] = 1)
8:9	IAC12M	IAC 1/2 Mode 00 Exact match — Match if address[0:29] = IAC 1/2[0:29]; two independent compares 01 Reserved 10 Range inclusive — Match if IAC1 ≤ address < IAC2 11 Range exclusive — Match if address < IAC1 OR address ≥ IAC2
10:14		Reserved
15	IAC12AT	IAC 1/2 Auto-Toggle Enable 0 Disable IAC 1/2 auto-toggle 1 Enable IAC 1/2 auto-toggle

16:17	IAC3US	IAC 3 User/Supervisor 00 Both 01 Reserved 10 Supervisor only (MSR[PR] = 0) 11 User only (MSR[PR] = 1)
18:19	IAC3ER	IAC 3 Effective/Real 00 Effective (MSR[IS] = don't care) 01 Reserved 10 Virtual (MSR[IS] = 0 11 Virtual (MSR[IS] = 1)
20:21	IAC4US	IAC 4 User/Supervisor 00 Both 01 Reserved 10 Supervisor only (MSR[PR] = 0) 11 User only (MSR[PR] = 1)
22:23	IAC4ER	IAC 4 Effective/Real 00 Effective (MSR[IS] = don't care) 01 Reserved 10 Virtual (MSR[IS] = 0 11 Virtual (MSR[IS] = 1)
24:25	IAC34M	IAC 3/4 Mode 00 Exact match — Match if address[0:29] = IAC 3/4[0:29]; two independent compares 01 Reserved 10 Range inclusive — Match if IAC3 ≤ address < IAC4 11 Range exclusive — Match if address < IAC3 OR address ≥ IAC4
26:30		Reserved
31	IAC34AT	IAC3/4 Auto-Toggle Enable 0 Disable IAC 3/4 auto-toggle 1 Enable IAC 3/4 auto-toggle

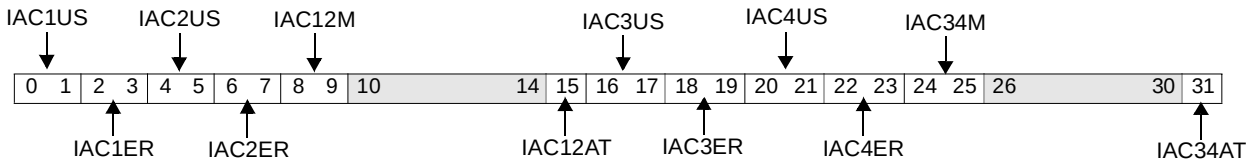


Figure 15-3. Debug Control Register 1 (DBCR1)

0:1	IAC1US	Instruction Address Compare (IAC) 1 User/Supervisor 00 Both 01 Reserved 10 Supervisor only (MSR[PR] = 0) 11 User only (MSR[PR] = 1)

2:3	IAC1ER	IAC 1 Effective/Real 00 Effective (MSR[IS] = don't care) 01 Reserved 10 Virtual (MSR[IS] = 0) 11 Virtual (MSR[IS] = 1)
4:5	IAC2US	IAC 2 User/Supervisor 00 Both 01 Reserved 10 Supervisor only (MSR[PR] = 0) 11 User only (MSR[PR] = 1)
6:7	IAC2ER	IAC 2 Effective/Real 00 Effective (MSR[IS] = don't care) 01 Reserved 10 Virtual (MSR[IS] = 0) 11 Virtual (MSR[IS] = 1)
8:9	IAC12M	IAC 1/2 Mode 00 Exact match — Match if address[0:29] = IAC 1/2[0:29]; two independent compares 01 Reserved 10 Range inclusive — Match if IAC1 ≤ address < IAC2 11 Range exclusive — Match if address < IAC1 OR address ≥ IAC2
10:14		Reserved
15	IAC12AT	IAC 1/2 Auto-Toggle Enable 0 Disable IAC 1/2 auto-toggle 1 Enable IAC 1/2 auto-toggle
16:17	IAC3US	IAC 3 User/Supervisor 00 Both 01 Reserved 10 Supervisor only (MSR[PR] = 0) 11 User only (MSR[PR] = 1)
18:19	IAC3ER	IAC 3 Effective/Real 00 Effective (MSR[IS] = don't care) 01 Reserved 10 Virtual (MSR[IS] = 0) 11 Virtual (MSR[IS] = 1)
20:21	IAC4US	IAC 4 User/Supervisor 00 Both 01 Reserved 10 Supervisor only (MSR[PR] = 0) 11 User only (MSR[PR] = 1)
22:23	IAC4ER	IAC 4 Effective/Real 00 Effective (MSR[IS] = don't care) 01 Reserved 10 Virtual (MSR[IS] = 0) 11 Virtual (MSR[IS] = 1)
24:25	IAC34M	IAC 3/4 Mode 00 Exact match — Match if address[0:29] = IAC 3/4[0:29]; two independent compares 01 Reserved 10 Range inclusive — Match if IAC3 ≤ address < IAC4 11 Range exclusive — Match if address < IAC3 OR address ≥ IAC4
26:30		Reserved

31	IAC34AT	IAC3/4 Auto-Toggle Enable 0 Disable IAC 3/4 auto-toggle 1 Enable IAC 3/4 auto-toggle

15.7.3 Debug Control Register 2 (DBCR2)

DBCR2 is an SPR that is used to configure DAC and DVC debug events. DBCR2 can be written from a GPR using **mtspr**, and can be read into a GPR using **mfspr**.

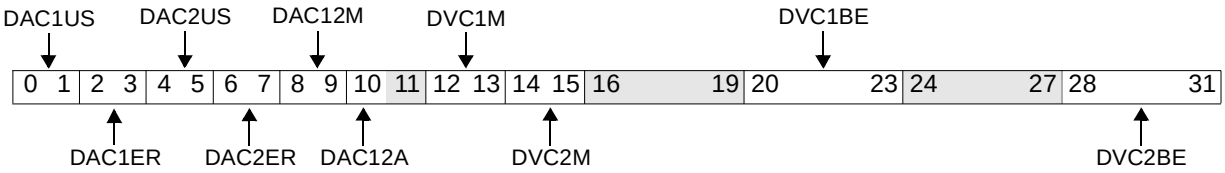


Figure 0-4. Debug Control Register 2 (DBCR2)

0:1	DAC1US	Data Address Compare (DAC) 1 User/Supervisor 00 Both 01 Reserved 10 Supervisor only (MSR[PR] = 0) 11 User only (MSR[PR] = 1)
2:3	DAC1ER	DAC 1 Effective/Real 00 Effective (MSR[DS] = don't care) 01 Reserved 10 Virtual (MSR[DS] = 0) 11 Virtual (MSR[DS] = 1)
4:5	DAC2US	DAC 2 User/Supervisor 00 Both 01 Reserved 10 Supervisor only (MSR[PR] = 0) 11 User only (MSR[PR] = 1)
6:7	DAC2ER	DAC 2 Effective/Real 00 Effective (MSR[DS] = don't care) 01 Reserved 10 Virtual (MSR[DS] = 0) 11 Virtual (MSR[DS] = 1)
8:9	DAC12M	DAC 1/2 Mode 00 Exact match — Match if address[0:31] = DAC 1/2[0:31]; two independent compares 01 Address bit mask — Match if address = DAC1; only compare bits corresponding to 1 bits in DAC2 10 Range inclusive — Match if DAC1 ≤ address < DAC2 11 Range exclusive — Match if address < DAC1 OR address ≥ DAC2
10	DAC12A	DAC 1/2 Asynchronous 0 Debug interrupt caused by DAC1/2 exception will be synchronous 1 Debug interrupt caused by DAC1/2 exception will be asynchronous
11		Reserved

12:13	DVC1M	Data Value Compare (DVC) 1 Mode 00 Reserved 01 AND all bytes enabled by DVC1BE 10 OR all bytes enabled by DVC1BE 11 AND-OR pairs of bytes enabled by DVC1BE (0 AND 1) OR (2 AND 3)
14:15	DVC2M	DVC 2 Mode 00 Reserved 01 AND all bytes enabled by DVC2BE 10 OR all bytes enabled by DVC2BE 11 AND-OR pairs of bytes enabled by DVC2BE (0 AND 1) OR (2 AND 3)
16:19		Reserved
20:23	DVC1BE	DVC 1 Byte Enables 0:3
24:27		Reserved
28:31	DVC2BE	DVC 2 Byte Enables 0:3

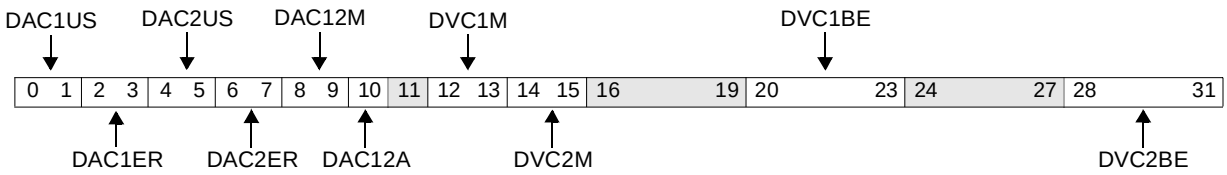


Figure 15-4. Debug Control Register 2 (DBCR2)

0:1	DAC1US	Data Address Compare (DAC) 1 User/Supervisor 00 Both 01 Reserved 10 Supervisor only (MSR[PR] = 0) 11 User only (MSR[PR] = 1)
2:3	DAC1ER	DAC 1 Effective/Real 00 Effective (MSR[DS] = don't care) 01 Reserved 10 Virtual (MSR[DS] = 0) 11 Virtual (MSR[DS] = 1)
4:5	DAC2US	DAC 2 User/Supervisor 00 Both 01 Reserved 10 Supervisor only (MSR[PR] = 0) 11 User only (MSR[PR] = 1)
6:7	DAC2ER	DAC 2 Effective/Real 00 Effective (MSR[DS] = don't care) 01 Reserved 10 Virtual (MSR[DS] = 0) 11 Virtual (MSR[DS] = 1)

8:9	DAC12M	DAC 1/2 Mode 00 Exact match 01 Address bit mask 10 Range inclusive 11 Range exclusive	 Match if address[0:31] = DAC 1/2[0:31]; two independent compares Match if address = DAC1; only compare bits corresponding to 1 bits in DAC2 Match if DAC1 ≤ address < DAC2 Match if address < DAC1 OR address ≥ DAC2
10	DAC12A	DAC 1/2 Asynchronous 0 Debug interrupt caused by DAC1/2 exception will be synchronous 1 Debug interrupt caused by DAC1/2 exception will be asynchronous	
11		Reserved	
12:13	DVC1M	Data Value Compare (DVC) 1 Mode 00 Reserved 01 AND all bytes enabled by DVC1BE 10 OR all bytes enabled by DVC1BE 11 AND-OR pairs of bytes enabled by DVC1BE	 (0 AND 1) OR (2 AND 3)
14:15	DVC2M	DVC 2 Mode 00 Reserved 01 AND all bytes enabled by DVC2BE 10 OR all bytes enabled by DVC2BE 11 AND-OR pairs of bytes enabled by DVC2BE	 (0 AND 1) OR (2 AND 3)
16:19		Reserved	
20:23	DVC1BE	DVC 1 Byte Enables 0:3	
24:27		Reserved	
28:31	DVC2BE	DVC 2 Byte Enables 0:3	

15.7.4 Debug Status Register (DBSR)

The DBSR contains status on debug events as well as information on the type of the most recent reset. The status bits are set by the occurrence of debug events, while the reset type information is updated upon the occurrence of any of the three reset types.

The DBSR is read into a GPR using **mfspr**. Clearing the DBSR is performed using **mtspr** by placing a 1 in the GPR source register in all bit positions which are to be cleared in the DBSR, and a 0 in all other bit positions. The data written from the GPR to the DBSR is not direct data, but a mask. A 1 clears the bit and a 0 leaves the corresponding DBSR bit unchanged.

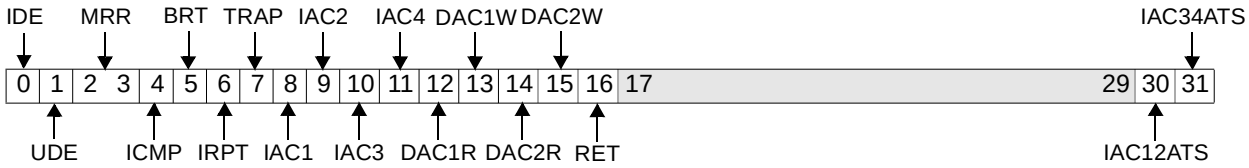


Figure 0-5. Debug Status Register (DBSR)

0	IDE	Imprecise Debug Event 0 Debug event ocurred while MSR[DE] = 1 1 Debug event occurred while MSR[DE] = 0	For synchronous debug events in internal debug mode, this field indicates whether the corresponding Debug interrupt occurs precisely or imprecisely
1	UDE	Unconditional Debug Event 0 Event didn't occur 1 Event occurred	
2:3	MRR	Most Recent Reset 00 No reset has occurred since this field was last cleared by software. 01 Core reset 10 Chip reset 11 System reset	This field is set upon any processor reset to a value indicating the type of reset.
4	ICMP	Instruction Completion Debug Event 0 Event didn't occur 1 Event occurred	
5	BRT	Branch Taken Debug Event 0 Event didn't occur 1 Event occurred	
6	IRPT	Interrupt Debug Event 0 Event didn't occur 1 Event occurred	
7	TRAP	Trap Debug Event 0 Event didn't occur 1 Event occurred	
8	IAC1	IAC 1 Debug Event 0 Event didn't occur 1 Event occurred	
9	IAC2	IAC 2 Debug Event 0 Event didn't occur 1 Event occurred	
10	IAC3	IAC 3 Debug Event 0 Event didn't occur 1 Event occurred	
11	IAC4	IAC 4 Debug Event 0 Event didn't occur 1 Event occurred	

12	DAC1R	DAC 1 Read Debug Event 0 Event didn't occur 1 Event occurred
13	DAC1W	DAC 1 Write Debug Event 0 Event didn't occur 1 Event occurred
14	DAC2R	DAC 2 Read Debug Event 0 Event didn't occur 1 Event occurred
15	DAC2W	DAC 2 Write Debug Event 0 Event didn't occur 1 Event occurred
16	RET	Return Debug Event 0 Event didn't occur 1 Event occurred
17:29		Reserved
30	IAC12ATS	IAC 1/2 Auto-Toggle Status 0 Range is not reversed from value specified in DBCR1[IAC12M] 1 Range is reversed from value specified in DBCR1[IAC12M]
31	IAC34ATS	IAC 3/4 Auto-Toggle Status 0 Range is not reversed from value specified in DBCR1[IAC34M] 1 Range is reversed from value specified in DBCR1[IAC34M]

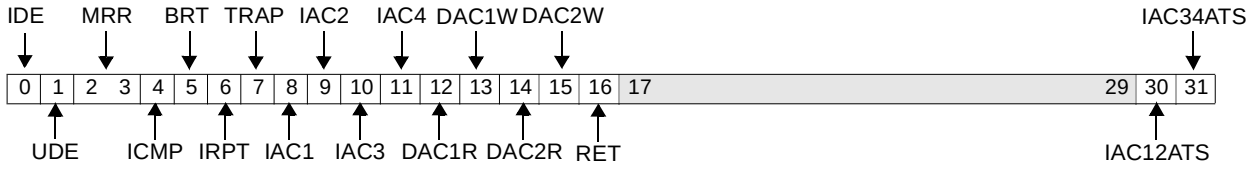


Figure 15-5. Debug Status Register (DBSR)

0	IDE	Imprecise Debug Event 0 Debug event ocurred while MSR[DE] = 1 1 Debug event occurred while MSR[DE] = 0	For synchronous debug events in internal debug mode, this field indicates whether the corresponding Debug interrupt occurs precisely or imprecisely
1	UDE	Unconditional Debug Event 0 Event didn't occur 1 Event occurred	
2:3	MRR	Most Recent Reset 00 No reset has occurred since this field was last cleared by software. 01 Core reset 10 Chip reset 11 System reset	This field is set upon any processor reset to a value indicating the type of reset.

4	ICMP	Instruction Completion Debug Event 0 Event didn't occur 1 Event occurred
5	BRT	Branch Taken Debug Event 0 Event didn't occur 1 Event occurred
6	IRPT	Interrupt Debug Event 0 Event didn't occur 1 Event occurred
7	TRAP	Trap Debug Event 0 Event didn't occur 1 Event occurred
8	IAC1	IAC 1 Debug Event 0 Event didn't occur 1 Event occurred
9	IAC2	IAC 2 Debug Event 0 Event didn't occur 1 Event occurred
10	IAC3	IAC 3 Debug Event 0 Event didn't occur 1 Event occurred
11	IAC4	IAC 4 Debug Event 0 Event didn't occur 1 Event occurred
12	DAC1R	DAC 1 Read Debug Event 0 Event didn't occur 1 Event occurred
13	DAC1W	DAC 1 Write Debug Event 0 Event didn't occur 1 Event occurred
14	DAC2R	DAC 2 Read Debug Event 0 Event didn't occur 1 Event occurred
15	DAC2W	DAC 2 Write Debug Event 0 Event didn't occur 1 Event occurred
16	RET	Return Debug Event 0 Event didn't occur 1 Event occurred
17:29		Reserved
30	IAC12ATS	IAC 1/2 Auto-Toggle Status 0 Range is not reversed from value specified in DBCR1[IAC12M] 1 Range is reversed from value specified in DBCR1[IAC12M]
31	IAC34ATS	IAC 3/4 Auto-Toggle Status 0 Range is not reversed from value specified in DBCR1[IAC34M] 1 Range is reversed from value specified in DBCR1[IAC34M]

15.7.5 Instruction Address Compare Registers (IAC1–IAC4)

The four IAC registers specify the addresses upon which IAC debug events should occur. Each of the IAC registers can be written from a GPR using **mtspr**, and can be read into a GPR using **mfspr**.

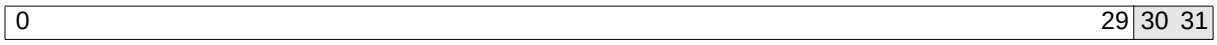


Figure 0-6. Instruction Address Compare Registers (IAC1–IAC4)

0:29		Instruction Address Compare (IAC) word address
30:31		Reserved

Figure 15-6. Instruction Address Compare Registers (IAC1–IAC4)

0:29		Instruction Address Compare (IAC) word address
30:31		Reserved

15.7.6 Data Address Compare Registers (DAC1–DAC2)

The two DAC registers specify the addresses upon which DAC (and/or DVC) debug events should occur. Each of the DAC registers can be written from a GPR using **mtspr**, and can be read into a GPR using **mfspr**.

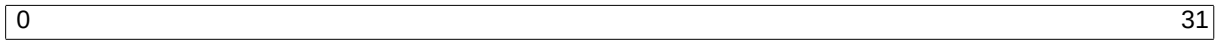


Figure 0-7. Data Address Compare Registers (DAC1–DAC2)

0:31		Data Address Compare (DAC) byte address

Figure 15-7. Data Address Compare Registers (DAC1–DAC2)

0:31		Data Address Compare (DAC) byte address

15.7.7 Data Value Compare Registers (DVC1–DVC2)

The DVC registers specify the data values upon which DVC debug events should occur. Each of the DVC registers can be written from a GPR using **mtspr**, and can be read into a GPR using **mfspr**.

Figure 0-8. Data Value Compare Registers (DVC1–DVC2)

0:31		Data value to compare

Figure 15-8. Data Value Compare Registers (DVC1–DVC2)

0:31		Data value to compare

15.7.8 Debug Data Register (DBDR)

The DBDR can be used for communication between software running on the processor and debug tool hardware and software. The DBDR can be written from a GPR using **mtspr**, and can be read into a GPR using **mfspr**.

Figure 0-9. Debug Data Register (DBDR)

0:31		Debug Data

Figure 15-9. Debug Data Register (DBDR)

0:31		Debug Data

Part IV. PPC440GP External Interfaces

16. Internal SRAM Controller

The Internal SRAM Controller (ISC) transfers data between the SRAM and the Processor Local Bus (PLB). The Internal SRAM Controller (ISC) is the 128-bit interface to the SRAM. The SRAM is configured as a single 128-bit wide 8-KB bank, that is, 512 x 128 bits.

The Internal SRAM Controller supports the following:

- 1 bank of 8 KB configurable to either 4 KB or 8 KB (128 bits wide)
- 128-bit slave attachment addressable by any PLB master
- Transfers by PLB slave cycles:
  - Single-beat read and write (1 to 8 bytes for 64-bit masters, 1 to 16 bytes for 128-bit masters)
  - 4-word line read and write
  - 8-word line read and write
  - Doubleword read and write bursts for 64-bit masters
  - Quadword read and write bursts for 128-bit masters
  - Slave-terminated doubleword and quadword fixed length bursts
  - Master-terminated variable length bursts
- Guarded Memory Access on 4 KB boundaries
- Data Parity Checking
- 1 x PLB clock

16.1 Accessing Internal SRAM Controller Registers

The Internal SRAM Controller (ISC) contains registers that are accessible on the Device Control Register (DCR) bus using move to device control register (**mtdcr**) and move from device control register (**mfdcr**).

16.1.1 Address Map

A complete list of registers that affect operation is found in ~~Table 16-1.~~ *Table 16-1.* Default values after reset are included in ~~Table 16-2~~ *Table 16-2*.

Table 16-1. SRAM Registers

Mnemonic	Name	DCR Number	Access	Page
SRAM0_SB0CR	SRAM Bank 0 Configuration Register	0x20	R/W	~~16-3~~465
SRAM0_SB1CR	SRAM Bank 1 Configuration Register	0x21	R/W	~~16-3~~465
SRAM0_SB2CR	SRAM Bank 2 Configuration Register (Reserved for the PPC440GP)	0x22	R/W	~~16-3~~465
SRAM0_SB3CR	SRAM Bank 3 Configuration Register (Reserved for the PPC440GP)	0x23	R/W	~~16-3~~465
SRAM0_BEAR	SRAM Bus Error Address Register	0x24	R/W	~~16-4~~466
SRAM0_BESR0	Bus Error Status Register 0	0x25	R/W	~~16-4~~466
SRAM0_BESR1	Bus Error Status Register 1	0x26	R/W	~~16-6~~468
SRAM0_PMEG	Power Management	0x27	R/W	~~16-8~~470
SRAMx_CID	Core ID	0x28	Read	~~16-9~~470
SRAMx_REVID	Core Revision ID	0x29	Read	~~16-9~~471
SRAM0_DPC	Data Parity Check Register	0x2A	R/W	~~16-9~~471

Table 16-2. Register Contents After Reset

Mnemonic	Fields	Reset Value	Comments	Page
SRAM0_SB0CR	0:31	0x80000380		~~16-3~~465
SRAM0_SB1CR	0:31	0x80000380		~~16-3~~465
SRAM0_BEAR	0:31	0x00000000		~~16-4~~466
SRAM0_BESR0	0:31	0x00000000		~~16-4~~466
SRAM0_BESR1	0:31	0x00000000		~~16-6~~468
SRAM0_PMEG	0:31	0x01E00000		~~16-8~~470
Core ID	0:31	—		~~16-9~~470
Core Revision ID	0:31	—		~~16-9~~471
SRAM0_DPC	0:31	0x80000000		~~16-9~~471

16.1.2 Register Descriptions

16.1.2.1 SRAM Bank Configuration Registers (SRAM0_SB0CR - SRAM0_SB3CR)

To ensure proper address decode and translation by the internal SRAM controller, the base address is truncated to the twenty most significant bits and written to SRAM0_SBnCR[0:19]. The default base address is x80000.

SB0CR[20:22] is configurable to either 000 for 4 KB or 001 for 8 KB. The default is 8 KB.

See the section 16.2 for additional programming notes.

SRAM0_SB0CR - SRAM0_SB3CR bit definitions:

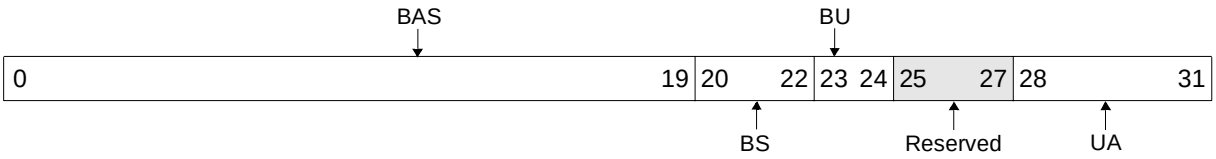


Figure 0-1. Memory Configuration (SRAM0_SB0CR - SRAM0_SB3CR)

0: 19	BAS	Base Address Select	Specifies the starting address of the SRAM bank.
20:22	BS	Bank Size 000 4 KB 001 8 KB 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved	Specifies the size of the logical bank address range.
23:24	BU	Bank Usage 00 Disabled 01 Bank is valid for read only (RO) 10 Reserved 11 Bank is valid for read/write (R/W)	
25:27		Reserved	
28:31	UA	Four least significant bits of the upper 32-bit address.	

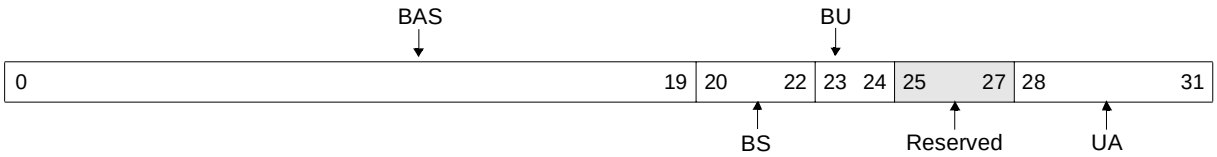


Figure 16-1. Memory Configuration (SRAM0_SB0CR - SRAM0_SB3CR)

0: 19	BAS	Base Address Select	Specifies the starting address of the SRAM bank.
20:22	BS	Bank Size 000 4 KB 001 8 KB 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved	Specifies the size of the logical bank address range.
23:24	BU	Bank Usage 00 Disabled 01 Bank is valid for read only (RO) 10 Reserved 11 Bank is valid for read/write (R/W)	
25:27		Reserved	
28:31	UA	Four least significant bits of the upper 32-bit address.	

There are no decodes of 0 KB for the Bank Size field; therefore, do not skip over any SRAM0_SBnCR register. Configure these registers successively, starting from SRAM0_SB0CR. Putting 00 into the Bank Usage field only disables that bank of SRAM; it does not move physical memory further to the next successive bank.

- **The BAS field (Base Address Select, bits 0:19)** sets the base address for an SRAM range. The BAS field is compared to bits 0:19 of the effective address. If the effective address is within the range of the starting address plus bank size, the associated bank is enabled for the transaction.
- **The BS field (Bank Size, bits 20:22)** sets the number of bytes which the bank may access, beginning with the base address set in the BAS field.
- **The BU field (Bank Usage, bits 23:24)** protects banks of physical devices from read or write accesses.
- **The UA field (Upper Address, bits 28:31)** sets the four least significant upper address bits.

  When an attempt is made to write access to an address within the range of the BAS field and the bank is designated as read-only, an SRAM controller protection error occurs.

16.1.2.2 Bus Error Address Register (SRAM0_BEAR)

The Bus Error Address Register (SRAM0_BEAR) is a 32-bit register which contains the address of the access where a data bus error has occurred. The BEAR is written when a data access error occurs and its contents may be locked until the BEAR lock bit in the Bus Error Status Register 0 (BESR0) is cleared, depending on the state of the LOCK ERROR signal when the transaction was accepted. The contents of the BEAR can be accessed using the move from device control register (**mfdcr**) and move to device control register **(mtdcr**) instructions.

BEAR bit definitions:

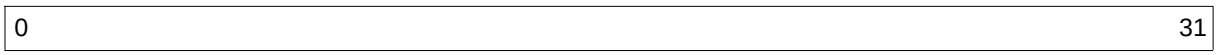


Figure 0-2. Bus Error Address Register (SRAM0_BEAR)

0:31		Address of Bus Error (asynchronous)

Figure 16-2. Bus Error Address Register (SRAM0_BEAR)

0:31		Address of Bus Error (asynchronous)

16.1.2.3 Bus Error Status Register 0 (SRAM0_BESR0)

The Bus Error Status Register 0 (SRAM0_BESR0) records the occurrence and type of errors for transactions attempted on behalf of each PLB master. The contents of the BESR0-1 can be accessed using the move from device control register (**mfdcr**) and move to device control register (**mtdcr**) instructions.

Note: The field lock bit protects fields ETn, RWSn, FLn, and ALn for master n. Upon error detection, any of master n fields may be overwritten if the field lock bit for that master is zero. If the field lock bit for a particular master has been set to zero by the DCR master, the BEAR lock bit for that master will be overwritten when the next PLB error is detected, regardless of the status of the BEAR lock prior to the error. If the BEAR lock is overwritten as a zero, the BEAR will be overwritten on the next error detected from that master, but will not be overwritten if the BEAR lock is due to an error caused by another master.

BESR0 bit definitions:

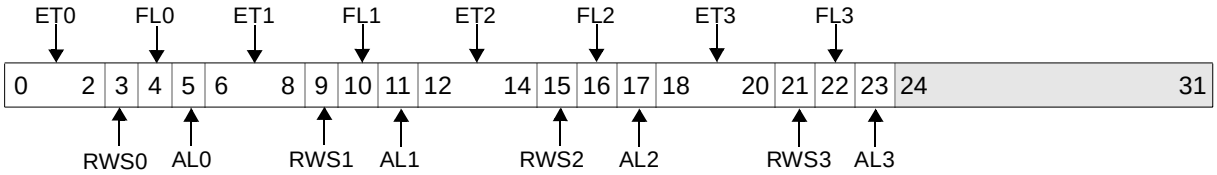


Figure 0-3. Bus Error Status Register 0 (SRAM0_BESR0)

0:2	ET0	Error type for master 0 (CPU ICU Read Operation) 000 No error 001 Reservd 010 Parity error 011 Reserved 100 Protection error 101 Reserved 110 Reserved 111 Reserved
3	RWS0	Read/write status for master 0 (CPU ICU Read Operation) 0 Error operation was a write operation 1 Error operation was a read operation
4	FL0	Field lock for master 0 (CPU ICU Read Operation) 0 Fields are unlocked 1 Fields are locked
5	AL0	BEAR address lock for master 0 (CPU ICU Read Operation) 0 BEAR address unlocked 1 BEAR address locked
6:8	ET1	Error type for master 1 (CPU DCU Read Operation) 000 No error 001 Reserved 010 Parity error 011 Reserved 100 Protection error 101 Reserved 110 Reserved 111 Reserved
9	RWS1	Read/write status for master 1 (CPU DCU Read Operation) 0 Error operation was a write operation 1 Error operation was a read operation
10	FL1	Field lock for master 1 (CPU DCU Read Operation) 0 Fields are unlocked 1 Fields are locked
11	AL1	BEAR address lock for master 1 (CPU DCU Read Operation) 0 BEAR address unlocked 1 BEAR address locked

12:14	ET2	Error type for master 2 000 No error 001 Reserved 010 Parity error 011 Reserved 100 Protection error 101 Reserved 110 Reserved 111 Reserved
15	RWS2	Read/write status for master 2 (CPU DCU Write Operation) 0 Error operation was a write operation 1 Error operation was a read operation
16	FL2	Field lock for master 2 (CPU DCU Write Operation) 0 Fields are unlocked 1 Fields are locked
17	AL2	BEAR address lock for master 2 (CPU DCU Write Operation) 0 BEAR address unlocked 1 BEAR address locked
18:20	ET3	Error type for master 3 (PCI-X) 000 No error 001 Reserved 010 Parity error 011 Reserved 100 Protection error 101 Reserved 110 Reserved 111 Reserved
21	RWS3	Read/write status for master 3 (PCI-X) 0 Error operation was a write operation 1 Error operation was a read operation
22	FL3	Field lock for master (PCI-X)3 0 Fields are unlocked 1 Fields are locked
23	AL3	BEAR address lock for master 3 (PCI-X) 0 BEAR address unlocked 1 BEAR address locked
24:31		Reserved

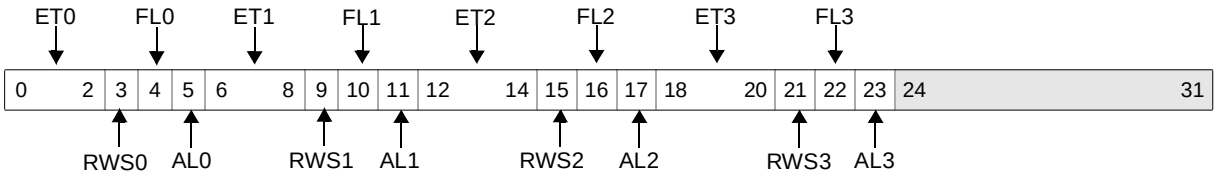


Figure 16-3. Bus Error Status Register 0 (SRAM0_BESR0)

0:2	ET0	Error type for master 0 (CPU ICU Read Operation) 000 No error 001 Reservd 010 Parity error 011 Reserved 100 Protection error 101 Reserved 110 Reserved 111 Reserved
3	RWS0	Read/write status for master 0 (CPU ICU Read Operation) 0 Error operation was a write operation 1 Error operation was a read operation
4	FL0	Field lock for master 0 (CPU ICU Read Operation) 0 Fields are unlocked 1 Fields are locked
5	AL0	BEAR address lock for master 0 (CPU ICU Read Operation) 0 BEAR address unlocked 1 BEAR address locked
6:8	ET1	Error type for master 1 (CPU DCU Read Operation) 000 No error 001 Reserved 010 Parity error 011 Reserved 100 Protection error 101 Reserved 110 Reserved 111 Reserved
9	RWS1	Read/write status for master 1 (CPU DCU Read Operation) 0 Error operation was a write operation 1 Error operation was a read operation
10	FL1	Field lock for master 1 (CPU DCU Read Operation) 0 Fields are unlocked 1 Fields are locked
11	AL1	BEAR address lock for master 1 (CPU DCU Read Operation) 0 BEAR address unlocked 1 BEAR address locked
12:14	ET2	Error type for master 2 000 No error 001 Reserved 010 Parity error 011 Reserved 100 Protection error 101 Reserved 110 Reserved 111 Reserved

15	RWS2	Read/write status for master 2 (CPU DCU Write Operation) 0 Error operation was a write operation 1 Error operation was a read operation
16	FL2	Field lock for master 2 (CPU DCU Write Operation) 0 Fields are unlocked 1 Fields are locked
17	AL2	BEAR address lock for master 2 (CPU DCU Write Operation) 0 BEAR address unlocked 1 BEAR address locked
18:20	ET3	Error type for master 3 (PCI-X) 000 No error 001 Reserved 010 Parity error 011 Reserved 100 Protection error 101 Reserved 110 Reserved 111 Reserved
21	RWS3	Read/write status for master 3 (PCI-X) 0 Error operation was a write operation 1 Error operation was a read operation
22	FL3	Field lock for master (PCI-X)3 0 Fields are unlocked 1 Fields are locked
23	AL3	BEAR address lock for master 3 (PCI-X) 0 BEAR address unlocked 1 BEAR address locked
24:31		Reserved

16.1.2.4 Bus Error Status Register 1 (SRAM0_BESR1)

The Bus Error Status Register 1 (SRAM0_BESR1) has identical functions as SRAM0_BESR0, except that SRAM0_BESR1 is used for masters 4, 5, 6, and 7.

BESR1 bit definitions:

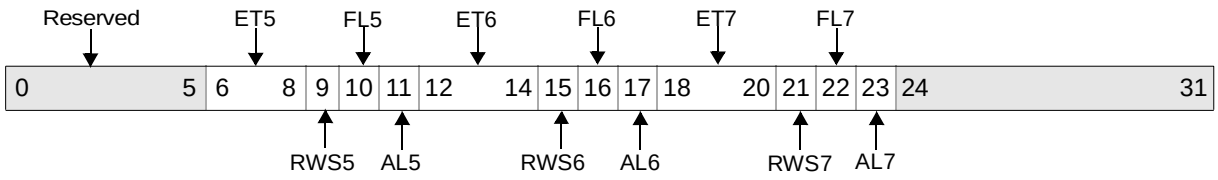


Figure 0-4. Bus Error Status Register 1 (SRAM0_BESR1)

0:5		Reserved

6:8	ET5	Error type for master 5 (MAL) 000 No error 001 Reserved 010 Parity error 011 Reserved 100 Protection error 101 Reserved 110 Reserved 111 Reserved
9	RWS5	Read/write status for master 5 (MAL) 0 Error operation was a write operation 1 Error operation was a read operation
10	FL5	Field lock for master 5 (MAL) 0 Fields are unlocked 1 Fields are locked
11	AL5	BEAR address lock for master 5 (MAL) 0 BEAR address unlocked 1 BEAR address locked
12:14	ET6	Error type for master 6 (DMA) 000 No error 001 Reserved 010 Parity error 011 Reserved 100 Protection error 101 Reserved 110 Reserved 111 Reserved
15	RWS6	Read/write status for master 6 (DMA) 0 Error operation was a write operation 1 Error operation was a read operation
16	FL6	Field lock for master 6 (DMA) 0 Fields are unlocked 1 Fields are locked
17	AL6	BEAR address lock for master 6 (DMA) 0 BEAR address unlocked 1 BEAR address locked
18:20	ET7	Error type for master 7 (BGI) 000 No error 001 Reserved 010 Parity error 011 Reserved 100 Protection error 101 Reserved 110 Reserved 111 Reserved
21	RWS7	Read/write status for master 7 (BGI) 0 Error operation was a write operation 1 Error operation was a read operation

22	FL7	Field lock for master 7 (BGI) 0 Fields are unlocked 1 Fields are locked
23	AL7	BEAR address lock for master 7 (BGI) 0 BEAR address unlocked 1 BEAR address locked
24:31		Reserved

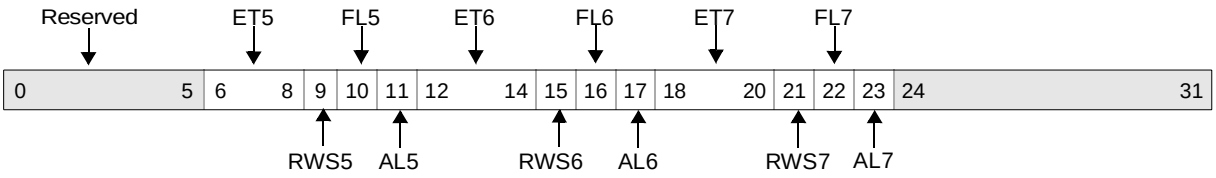


Figure 16-4. Bus Error Status Register 1 (SRAM0_BESR1)

0:5		Reserved
6:8	ET5	Error type for master 5 (MAL) 000 No error 001 Reserved 010 Parity error 011 Reserved 100 Protection error 101 Reserved 110 Reserved 111 Reserved
9	RWS5	Read/write status for master 5 (MAL) 0 Error operation was a write operation 1 Error operation was a read operation
10	FL5	Field lock for master 5 (MAL) 0 Fields are unlocked 1 Fields are locked
11	AL5	BEAR address lock for master 5 (MAL) 0 BEAR address unlocked 1 BEAR address locked
12:14	ET6	Error type for master 6 (DMA) 000 No error 001 Reserved 010 Parity error 011 Reserved 100 Protection error 101 Reserved 110 Reserved 111 Reserved
15	RWS6	Read/write status for master 6 (DMA) 0 Error operation was a write operation 1 Error operation was a read operation
16	FL6	Field lock for master 6 (DMA) 0 Fields are unlocked 1 Fields are locked

17	AL6	BEAR address lock for master 6 (DMA) 0 BEAR address unlocked 1 BEAR address locked
18:20	ET7	Error type for master 7 (BGI) 000 No error 001 Reserved 010 Parity error 011 Reserved 100 Protection error 101 Reserved 110 Reserved 111 Reserved
21	RWS7	Read/write status for master 7 (BGI) 0 Error operation was a write operation 1 Error operation was a read operation
22	FL7	Field lock for master 7 (BGI) 0 Fields are unlocked 1 Fields are locked
23	AL7	BEAR address lock for master 7 (BGI) 0 BEAR address unlocked 1 BEAR address locked
24:31		Reserved

16.1.2.5 Power Management Register (SRAM0_PMEG)

The Power Management Register (SRAM0_PMEG) enables the sleep function for the SRAM controller core. PM_En (bit 0) enables or disables the sleep mode. PM_CNT (bits 1:6) contains the multiplier n. If PM_EN is enabled to 1, the SRAM controller will go to sleep after being idle for (n x 16) clock cycles. PM_DFLT (bits 7:10) is hardwired to 1111 and is not writable.

SRAM0_PMEG bit definitions:

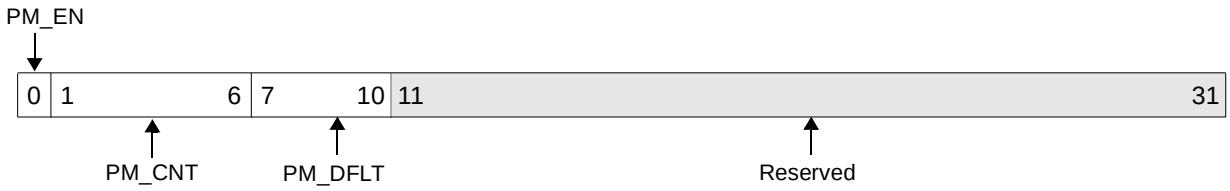


Figure 0-5. Power Management Register (SRAM0_PMEG)

0	PM_EN	Power Management enable: 0 Sleep mode disabled 1 Sleep mode enabled	
1:6	PM_CNT	Power Management Counter	The value (n) programmed into these register bits are the multiples of 16 clock cycles (n x 16). If PM_EN is set to 1, SRAM controller will go to sleep mode after being idle for (n x 16) clock cycles.

7:10	PM_DFLT	Power Management Default Wait Interval Hardwired to 1111.	The default value of 16 clock cycles. If PM_EN is set to 1, SRAM controller will go to sleep mode after being idle for 16 clock cycles.
11:31		Reserved	

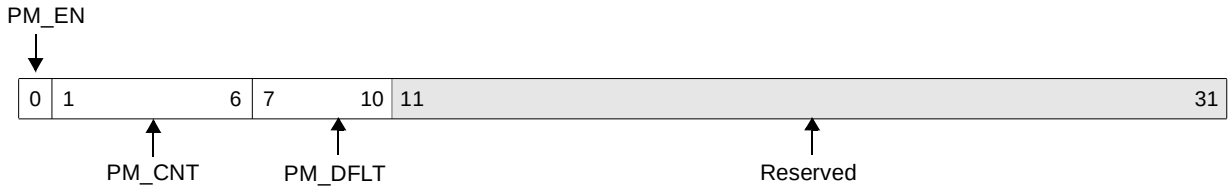


Figure 16-5. Power Management Register (SRAM0_PMEG)

0	PM_EN	Power Management enable: 0 Sleep mode disabled 1 Sleep mode enabled	
1:6	PM_CNT	Power Management Counter	The value (n) programmed into these register bits are the multiples of 16 clock cycles (n x 16). If PM_EN is set to 1, SRAM controller will go to sleep mode after being idle for (n x 16) clock cycles.
7:10	PM_DFLT	Power Management Default Wait Interval Hardwired to 1111.	The default value of 16 clock cycles. If PM_EN is set to 1, SRAM controller will go to sleep mode after being idle for 16 clock cycles.
11:31		Reserved	

16.1.2.6 Core ID Register (SRAM0_CID)

The Core ID Register (SRAM0_CID) identifies the core number for the SRAM controller core.

SRAM0_CID bit definitions:

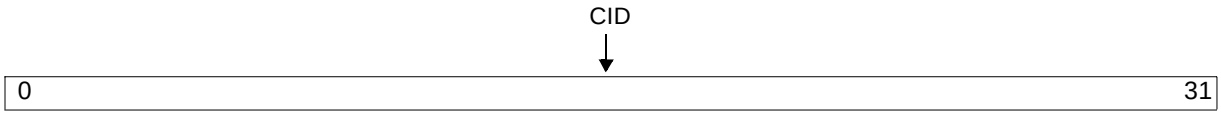


Figure 0-6. SRAM Internal Core Device ID Register (SRAM0_CID)

0:31	CID	Internal Core Device ID	H322B0000 (Read only).

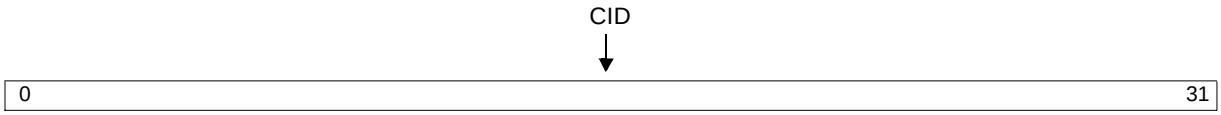


Figure 16-6. SRAM Internal Core Device ID Register (SRAM0_CID)

0:31	CID	Internal Core Device ID	H322B0000 (Read only).

16.1.2.7 Revision ID Register (SRAM0_REVID)

The Revision ID Register (SRAM0_REVID) identifies the revision project, SCCS, and netlist numbers for the SRAM controller core.

SRAM0_REVID bit definitions:

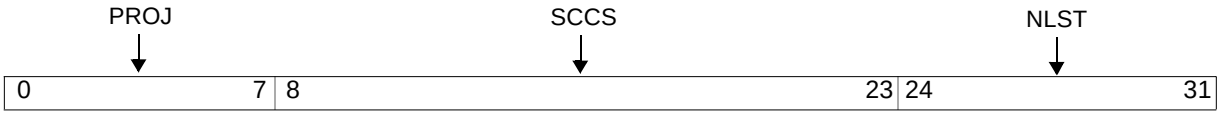


Figure 0-7. SRAM Internal Core Revision ID Register (SRAM0_REVID)

0:7	PROJ	Project Number	Read only.
8:23	SCCS	SCCS Version	Read only.
24:31	NLST	Netlist Number	Read only

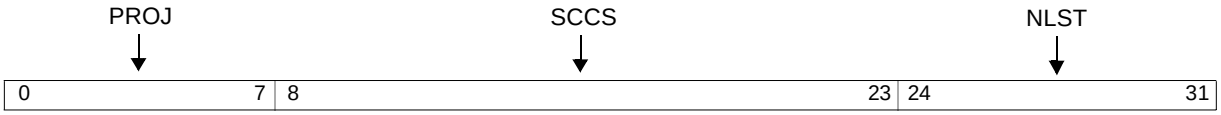


Figure 16-7. SRAM Internal Core Revision ID Register (SRAM0_REVID)

0:7	PROJ	Project Number	Read only.
8:23	SCCS	SCCS Version	Read only.
24:31	NLST	Netlist Number	Read only

16.1.2.8 Data Parity Checking Register (SRAM0_DPC)

The Data Parity Checking Register (SRAM0_DPC) enables data parity generation and checking if the data parity function is implemented. It controls data parity generation during write transactions and data parity checking during read transactions.

Note: During an alternated mode reset (SYS_plbReset = 1 and SYS_altMode = 1) the value of pin SYS_data_parity will be loaded into the DPC register.

SRAM0_DPC bit definitions:

Figure 0-8. Data Parity Checking Register (SRAM0_DPC)

0	DPC	Data Parity Check: 0 Disabled 1 Enabled	When set to 1, this bit enables data parity generation during write transactions and read transactions. When set to 0, this bit disables both functions.
1:31		Reserved	

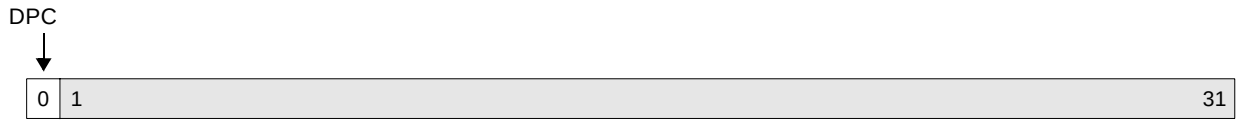


Figure 16-8. Data Parity Checking Register (SRAM0_DPC)

0	DPC	Data Parity Check: 0 Disabled 1 Enabled	When set to 1, this bit enables data parity generation during write transactions and read transactions. When set to 0, this bit disables both functions.
1:31		Reserved	

16.2 Errors

The internal SRAM controller monitors two types of errors when executing PLB transfers: protect errors and data parity errors.

16.2.1 Protect Error

A protect error occurs when the requested read or write operation violates the bank usage programming, for example, when a write is attempted on read-only bank.

When the SRAM detects an error, it will report this error condition to the owning master using a unique ERROR line. When a master requests a transfer, it also provides a unique master ID number that is encoded in the PLB_masterID. This value is latched in a register and when an error occurs during this master's tenure,

then it also owns the error and, thus the internal SRAM controller will drive the corresponding ERROR signal back to this master. The internal SRAM controller logs the type of error into the appropriate BESR and the address at which this error occurred into the BEAR. There is one error field for each master, but only one BEAR.

BESR Field

If a master does not assert the transfer qualifier LOCK ERROR, then any error occurring for this master will overwrite any previously logged error in the BESR field for this master. The BEAR always has the address of the most recently detected error if lockerror has not been asserted and no master has its own field address bit locked.

16.2.2 Data Parity Error

A data parity error occurs when the parity of data during a read transaction does not match the parity that is generated and stored for the specific data. Generation of stored parity occurs during write transactions. The operation of parity error is identical to protect error. The SRAM0_BEAR contains the address at which parity error occurred. Unlike with the protect error, the address captured in the SRAM0_BEAR for parity error could be one address beyond where the parity error occurred. SRAM0_BESRx fields will also log parity errors for the particular master, depending on the state of the lock error signal.

17. DDR SDRAM Controller

The double data rate (DDR) SDRAM memory controller supports x8 DDR SDRAMs, consists of a PLB slave interface and a DCR interface, and can provide a 32- or 64-bit interface to SDRAM memory with optional Error Checking and Correction (ECC). The DDR SDRAM can be relocated in the memory address ~~range~~ range, but relocation is generally not needed.

The controller supports page mode operation with bank interleaving always active and can maintain up to eight open pages. The controller supports up to four 512 MB logical banks in limited configurations, providing memory up to 2 GB. Global memory timings, address and bank sizes, and memory addressing modes are programmable. System power can be reduced by placing the DDR SDRAM controller in sleep and/or self-refresh mode.

17.1 Interface Signals

In many systems the memory controller can directly interface to DDR SDRAM; however, since each application is different, a detailed timing analysis should always be performed.

The DDR SDRAM uses PowerPC bit ordering (most significant bit = 0); however, the DDR SDRAM address signals (MemAddr12:0 and BA1:0) and external data bus use industry standard bit ordering. Interface signals then appear on the external DDR SDRAM memory interface as MemData0:63. Coming from the core, a general purpose register data bit 0, for example, is the most significant bit, but on the external interface, MemData0 is the least significant bit; thus, the internal PLB/data bit 0 corresponds to the External DDR interface bit 0, the internal bit 1 to the external bit 1, and so forth to bit/31/63; however, the key point to remember is that when examining the DDR memory interface, external data is bit reversed from internal data, so that, for example, an internal register bit 0 value of 0xF753 corresponds to a bit 0 value of 0xCAEF on the external memory data bus.

Figure 17-1 illustrates the DDR SDRAM signal I/Os.

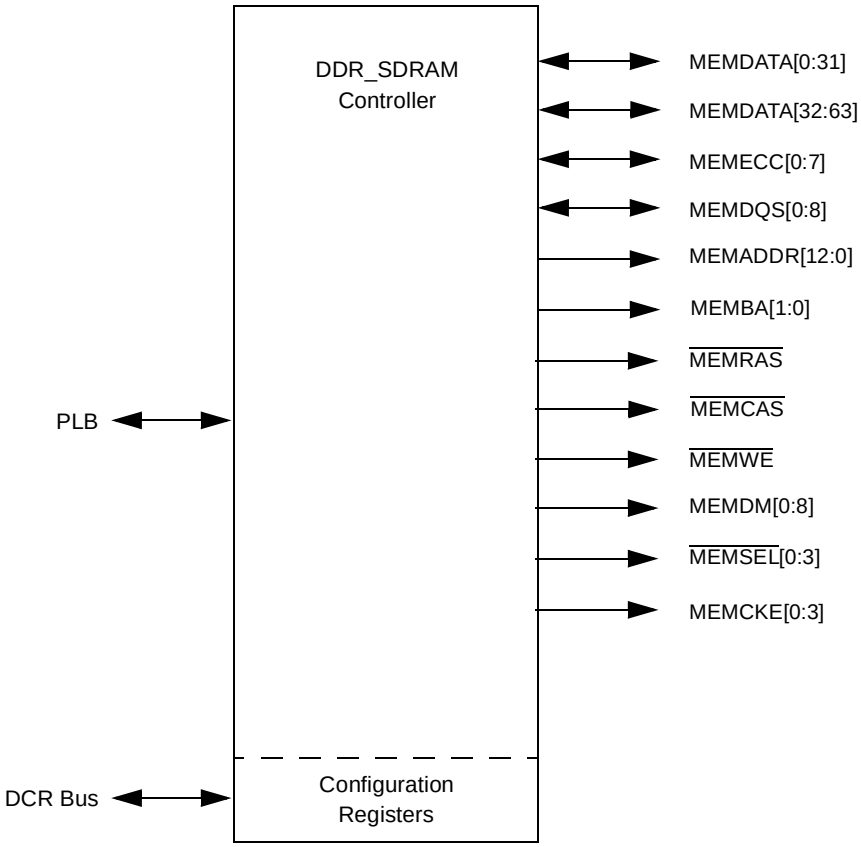


Figure 0-1. DDR SDRAM Controller Signals

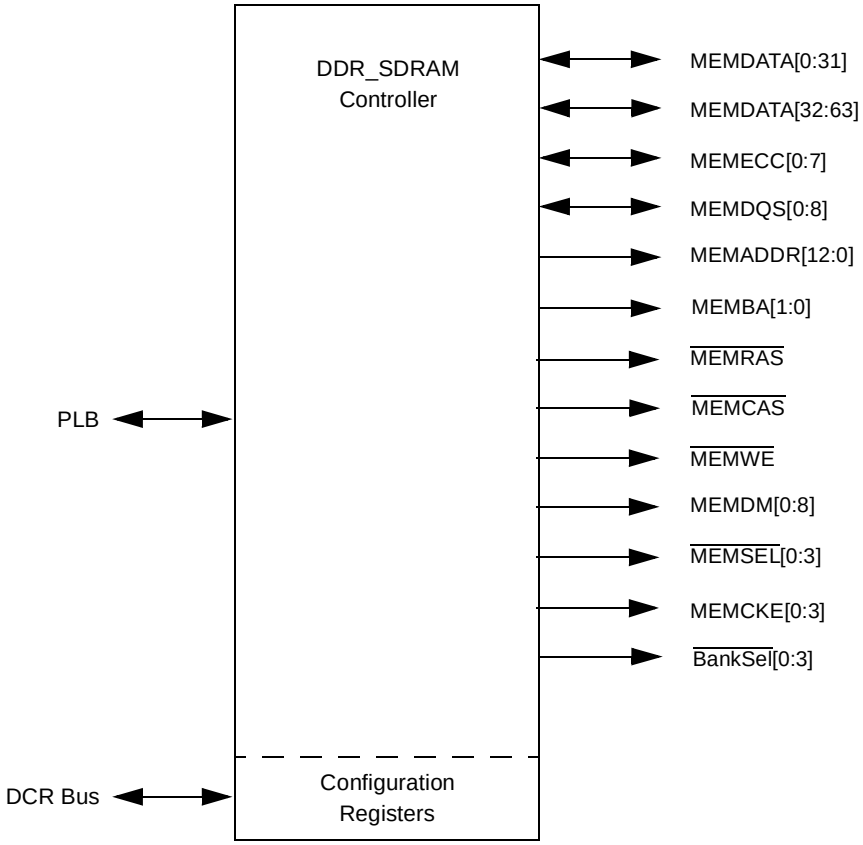


Figure 17-1. DDR SDRAM Controller Signals

The usage and signal state after a chip or system reset for each of the following signals is shown in Table 17-1.

Table 0-1. DDR SDRAM Signal Usage and State During/Following Reset

Signal	During Reset State	Following Reset State	Usage
MEMCKE[0:31]	0s	0s	Clock enable.
MEMBA[1:0]	0s	0s	Bank address. Used to select an internal SDRAM bank in dual- and quad-bank SDRAM devices.
MEMADDR[12:0]	0s	0s	Memory address. See section Section 17.8, "SDRAM Commands and Operations," on page 17-53 for additional details.
MEMSEL[0:3]	1s	1s	Chip select.
MEMRAS	1	1	Row Address Strobe.
MEMCAS	1	1	Column Address Strobe.
MEMWE	1	1	Write Enable.
MEMDATA[0:31]	Unknown	High -Z	Data input/output. MemData0 is the least significant bit.

Table 0-1. DDR SDRAM Signal Usage and State During/Following Reset (continued)

Signal	During Reset State	Following Reset State	Usage
MEMDATA[32:63]	Unknown	High -Z	Data input/output. MemData32 is the most significant bit.
MEMDQS[0:8]	Unknown	High -Z	Byte lane strobe.
MEMECC[0:7]	Unknown	High -Z	ECC check bits.
MEMDM[0:8]	1s	1s	Byte lane mask.

Table 17-1. DDR SDRAM Signal Usage and State During/Following Reset

Signal	During Reset State	Following Reset State	Usage
MEMCKE[0:3]	0s	0s	Clock enable.
MEMBA[1:0]	0s	0s	Bank address. Used to select an internal SDRAM bank in dual- and quad-bank SDRAM devices.
MEMADDR[12:0]	0s	0s	Memory address. See *SDRAM Commands and Operations* on page 522 for additional details.
$\overline{\text{MEMSEL}}$[0:3]	1s	1s	Chip select.
$\overline{\text{MEMRAS}}$	1	1	Row Address Strobe.
$\overline{\text{MEMCAS}}$	1	1	Column Address Strobe.
$\overline{\text{MEMWE}}$	1	1	Write Enable.
MEMDATA[0:31]	Unknown	High -Z	Data input/output. MemData0 is the least significant bit.
MEMDATA[32:63]	Unknown	High -Z	Data input/output. MemData32 is the most significant bit.
MEMDQS[0:8]	Unknown	High -Z	Byte lane strobe.
MEMECC[0:7]	Unknown	High -Z	ECC check bits.
MEMDM[0:8]	1s	1s	Byte lane mask.
BankSel[0:3]	1s	1s	Select up to four external DDR SDRAM banks

17.2 DDR SDRAM Configuration Registers

17.2.1 Accessing DDR SDRAM Registers

All DDR SDRAM configuration and status registers are accessed using the **mtdcr** and **mfdcr** instructions. To address a DDR SDRAM register, the DCR address offset of the register is placed in the configuration address register, SDRAM0_CFGADDR, using the **mtdcr** command. To read or write the register addressed by the configuration address, the data is accessed from the configuration data register, SDRAM0_CFGDATA, using the **mfdcr** or **mtdcr** instruction. Table 17-3 and Table 17-4 list the DCR addresses and DCR address offsets for all DDR SDRAM registers.

Note: Reserved fields in the DDR SDRAM controller configuration registers must not change when the register is written.

To modify bit fields within a register, read the register, use a mask to clear or set the target bits, or in the new field value, and then write the result back. A subsequent read of the target register is recommended to ensure that the targeted bit fields were updated properly.

Table 17-2. SDRAM Controller DCR Addresses

Mnemonic	Register	Address	Access
SDRAM0_CFGADDR	DDR SDRAM Controller Address Register	0x010	R/W
SDRAM0_CFGDATA	DDR SDRAM Controller Data Register	0x011	R/W

The following PowerPC code illustrates how to access a DDR SDRAM register by writing the SDRAM0_CFGDATA register and then reading back the written value.

```
lir3,register_offset! address offset of DDR_SDRAM register
lisr4,<config upper>! upper half of configuration data
orir4,r4,<config lower>! lower half of configuration data
mtdcrSDRAM0_CFGADDR,r3! set offset address
mtdcrSDRAM0_CFGDATA,r4! write configuration data
mfdcrr5, SDRAM0_CFGDATA! read back configuration data
```

17.2.2 DDR SDRAM Configuration Register Address Map

Table 17-3 lists the DDR SDRAM controller set. Refer to the indicated pages for detailed register descriptions.

Table 17-3. Address Map for Device Configuration Registers

Mnemonic	Register	Offset	Access	Page
SDRAM0_BESR0	Bus Error Status Register 0	0x00 0x04	R/W	~~17-9~~480
SDRAM0_BESR1	Bus Error Status Register 1	0x08 0x0C	R/W	~~17-11~~482
SDRAM0_BEAR	Bus Error Address Register	0x10	Read only	~~17-13~~484
SDRAM0_MIRQ	Master Write Interrupt	0x11 0x12	R/W	~~17-14~~485
SDRAM0_SLI0	PLB Slave Interface Options	0x18	R/W	~~17-16~~486
SDRAM0_CFG0	DDR SDRAM Controller Options 0	0x20	R/W	~~17-17~~487
SDRAM0_CFG1	DDR SDRAM Controller Options 1	0x21	R/W	~~17-20~~490
SDRAM0_DEVOPT	DDR SDRAM Device Options	0x22	R/W	~~17-21~~491
SDRAM0_MCSTS	DDR SDRAM Memory Controller Status	0x24	Read only	~~17-22~~492
SDRAM0_RTR	Refresh Timer Register	0x30	R/W	~~17-23~~493
SDRAM0_PMIT	Power Management Idle Timer	0x34	R/W	~~17-24~~494
SDRAM0_UABBA	PLB UA Bus Base Address	0x38	R/W	~~17-25~~495
SDRAM0_B0CR	DDR SDRAM Bank 0 Configuration	0x40	R/W	~~17-26~~496

Table 17-3. Address Map for Device Configuration Registers

SDRAM0_B1CR	DDR SDRAM Bank 1 Configuration	0x44	R/W	17-264 96
SDRAM0_B2CR	DDR SDRAM Bank 2 Configuration	0x48	R/W	17-264 96
SDRAM0_B3CR	DDR SDRAM Bank 3 Configuration	0x4C	R/W	17-264 96
SDRAM0_TR0	DDR SDRAM Timing Register 0	0x80	R/W	17-284 98
SDRAM0_TR1	DDR SDRAM Timing Register 1	0x81	R/W	17-305 00
SDRAM0_CLKTR	DDR SDRAM Clock Timing Register	0x82	R/W	17-375 07
SDRAM0_WDDCTR	Write Data, DQS, DM Clock Timing Register	0x83	R/W	17-405 09
SDRAM0_DLYCAL	Delay Line Calibration Register	0x84	R/W	17-445 13
SDRAM0_ECCESR	ECC Error Status	0x98	R/W	17-455 14
SDRAM0_CID	Controller ID Register	0xA4	Read only	17-465 15
SDRAM0_RID	Revision ID Register	0xA8	Read only	17-475 16

Table 17-4 lists the register contents after a reset.

Table 17-4. Register Contents After Reset

Mnemonic	Register	DCR Number	Access	Reset Value	Page
SDRAM0_BESR0	Bus Error Status Register0	0x00 0x04	R/W	x0000_0000	17-948 0
SDRAM0_BESR1	Bus Error Status Register 1	0x08 0x0C	R/W	x0000_0000	17-114 82
SDRAM0_BEAR	Bus Error Address Register	0x10	Read only	x0000_0000	17-134 84
SDRAM0_MIRQ	Master Write Interrupt	0x11 0x12	R/W	x0000_0000	17-144 85
SDRAM0_SLIO	Slave Interface Options	0x18	R/W	x0000_0000	17-164 86
SDRAM0_CFG0	DDR SDRAM Options 0	0x20	R/W	x0200_0000	17-174 87
SDRAM0_CFG1	DDR SDRAM Options 1	0x21	R/W	x0000_0000	17-204 90
SDRAM0_DEVOPT	DDR SDRAM Device Options	0x22	R/W	x0000_0000	17-214 91
SDRAM0_MCSTS	Memory Controller Status	0x24	Read only	x2000_0000	17-224 92
SDRAM0_RTR	Refresh Timer Register	0x30	R/W	x07E0_0000	17-234 93
SDRAM0_PMIT	Power Management Idle Timer	0x34	R/W	x07C0_0000	17-244 94
SDRAM0_UABBA	PLB UA Bus Base Address	0x38	R/W	x0000_0000	17-254 95
SDRAM0_B0CR	DDR SDRAM Bank 0 Configuration	0x40	R/W	x0000_0000	17-264 96

Table 17-4. Register Contents After Reset

SDRAM0_B1CR	DDR SDRAM Bank 1 Configuration	0x44	R/W	x0000_0000	17-264 96
SDRAM0_B2CR	DDR SDRAM Bank 2 Configuration	0x48	R/W	x0000_0000	17-264 96
SDRAM0_B3CR	DDR SDRAM Bank 3 Configuration	0x4C	R/W	x0000_0000	17-264 96
SDRAM0_TR0	DDR SDRAM Timing Register 0	0x80	R/W	x0089_4012	17-284 98
SDRAM0_TR1	DDR SDRAM Timing Register 1	0x81	R/W	x4040_0000	17-305 00
SDRAM0_CLKTR	DDR SDRAM Clock Timing Register	0x82	R/W	x0000_0000	17-375 07
SDRAM0_WDDCTR	Write Data, DQS, DM Clock Timing Register	0x83	R/W	x0000_0000	17-405 09
SDRAM0_DLYCAL	Delay Line Calibration Register	0x84	R/W	x0000_0000	17-445 13
SDRAM0_ECCESR	ECC Error Status	0x98	R/W	x0000_0000	17-455 14
SDRAM0_CID	controller ID Register	0xA4	Read only	x320B_0000	17-465 15
SDRAM0_RID	Revision ID Register	0xA8	Read only	See the CDID, RCID, and BRID bits of the SDRAM0_RID.	17-475 16

17.2.3 Device Configuration

The following registers apply globally to the memory subsystem and are used to configure its operation.

Note: Reserved fields in the memory configuration registers must not change when the register is written. Changing reserved fields can cause DDR SDRAM controller malfunction.

The following table lists the complete set of configuration registers and the effects of reading and writing to each register with SDRAM0_CFG0[DCEN] asserted/deasserted.

Table 17-5. DCRs with Dependencies on CFG0[DCEN]

Mnemonic	Register	Offset	CFG0 [DCEN]	Transaction	Result
SDRAM0_BESR0	Bus Error Status Register 0	x00	X[a]	Write	Clear
		x04	X	Write	Set
SDRAM0_BESR1	Bus Error Status Register 1	x08	X	Write	Clear
		x0C	X	Write	Set
SDRAM0_BEAR	Bus Error Address Register	x10	X	Write	No Change
SDRAM0_MIRQ	Bus Master Interrupt	x11	X	Write	Clear
		x12	X	Write	Set
SDRAM0_SLI0	DDR SDRAM Slave Interface Options	x18	X	Write	Update
SDRAM0_CFG0	DDR SDRAM Options 0	x20	X	Write	Update
SDRAM0_CFG1	DDR SDRAM Options 1	x21	X	Write	Update

Table 17-5. DCRs with Dependencies on CFG0[DCEN] (continued)

SDRAM0_DEVOPT	DDR SDRAM Device Options	x22	0	Write	Update
			1	Write	No Change
SDRAM0_MCSTS	Memory Controller Status	x24	X	Write	No Change
SDRAM0_RTR	Refresh Timer Register	x30	0	Write	Update
			1	Write	No Change
SDRAM0_PMIT	Power Management Idle Timer	x34	0	Write	Update
			1	Write	No Change
SDRAM0_UABBA	PLB UABus Base Address	x38	0	Write	Update
			1	Write	No Change
SDRAM0_B0CR	DDR SDRAM Bank 0 Configuration	x40	0	Write	Update
			1	Write	No Change
SDRAM0_B1CR	DDR SDRAM Bank 1 Configuration	x44	0	Write	Update
			1	Write	No Change
SDRAM0_B2CR	DDR SDRAM Bank 2 Configuration	x48	0	Write	Update
			1	Write	No Change
SDRAM0_B3CR	DDR SDRAM Bank 3 Configuration	x4C	0	Write	Update
			1	Write	No Change
SDRAM0_TR0	SDRAM Timing Register 0	x80	0	Write	Update
			1	Write	No Change
SDRAM0_TR1	SDRAM Timing Register 1	x81	X	Write[b]	Update
SDRAM0_CLKTR	DDR Clock Timing Register	x82	X	Write[2]	Update
SDRAM0_WDDCTR	Write Data/DM/DQS Clock Timing Register	x83	X	Write[2]	Update
SDRAM0_DLYCAL	Delay Line Calibration Register	x84	X	Write	Update
SDRAM0_ECCESR	ECC Error Status	x98	X	Write	No Change
any	any	any	X	Read	Read Data

a. X indicates don't care.
b. Modification of SDRAM0_TR1, SDRAM0_CLKTR, and SDRAM0_WDDCTR after enabling SDRAM0_CFG0[0] is allowed by the logic; however, software must guarantee that the effects of doing so does not affect the operation of the DDR SDRAM controller. The DDR SDRAM should be idle and have no pending requests. Also, when changing SDRAM0_CLKTR, the downstream effects (such as minimum lock time, and re-establishing lock) on the system/chip level PLLs must be accounted for before attempting to access the memory. Modification of SDRAM0_CLKTR after setting SDRAM0_CFG0[0] is not recommended.

17.2.3.1 Initial Configuration Following Power-On-Reset

Note: Configuration of all SDRAM-specific registers must be performed prior to enabling the DDR SDRAM ~~controller, according to section Section 17.3, "Initialization," on page 17-47~~controller as described in *Initialization* on page 516.

All registers must be configured prior to setting SDRAM0_CFG0[DCEN] = 1. Any DCR write to the above SDRAM registers while SDRAM0_CFG0[DCEN] = 1 will complete on the DCR bus; however, no update to target register(s) will be performed.

Write access to SDRAM0_CFG0 is allowed, independent of the SDRAM0_CFG0[DCEN] value. Software must ensure that the DDR SDRAM controller is idle when updating/changing SDRAM0_CFG0. This is required to guarantee that the register update will not affect the operation of any in-progress or preceding PLB-to-memory accesses.

Write access to the DDR SDRAM controller error status registers (SDRAM0_BESR0, SDRAM0_BESR1, SDRAM0_BEAR, SDRAM0_MIRQ, and SDRAM0_ECCESR) are independent of the SDRAM0_CFG0 value. DCR writes to the error status registers are given the highest priority in the event that a DCR error status register write occurs coincident with an error status register update.

DCR reads will complete normally with the contents of the targeted register returned independent of the SDRAM0_CFG0[0] value.

17.2.3.2 Re-Configuration Following Initial Configuration

Re-configuration of the SDRAM-specific registers may be performed at any time following the initial configuration. Before re-configuring, all pending and queued requests targeting the DDR SDRAM controller must be allowed to complete. The following details the specific sequence required:

1. Guarantee all pending and queued memory requests are complete.
2. Disable DDR SDRAM controller - write SDRAM0_CFG0[DCEN]=0.
3. Update SDRAM specific timing registers.
4. Enable DDR SDRAM controller - Write SDRAM0_CFG0[DCEN]=1.

Once enabled, the DDR SDRAM controller will re-initialize the SDRAM devices by performing the necessary auto refresh/MRS/auto refresh sequence as described in ~~section Section 17.3, "Initialization," on page 17-47~~*Initialization* on page 516.

17.2.4 SDRAM Register Descriptions

Table 17-6 gives the master numbers and names used in the following register descriptions.

Table 17-6. DDR SDRAM PLB Masters

Master Number	Master Name
0	440 CPU Instruction Cache Controller
1	440 CPU Data Cache Controller
2	440 CPU Data Cache Controller
3	PLB-PCIX Bridge
4	Reserved
5	MAL
6	DMA Controller
7	OPB to PLB Bridge

17.2.4.1 Bus Error Syndrome Register 0 (SDRAM0_BESR0)

The SDRAM0_BESR0 tracks errors encountered during PLB Master[0-3] accesses to the DDR SDRAM (see ~~Table 17-6, "DDR SDRAM PLB Masters," on page 17-8~~*Table 17-6 DDR SDRAM PLB Masters* on page 479). Bits in SDRAM0_BESR0 are cleared by writing a 32-bit value to SDRAM0_BESR0 with a 1 in any bit position that is to be cleared and 0 in all other bit positions.

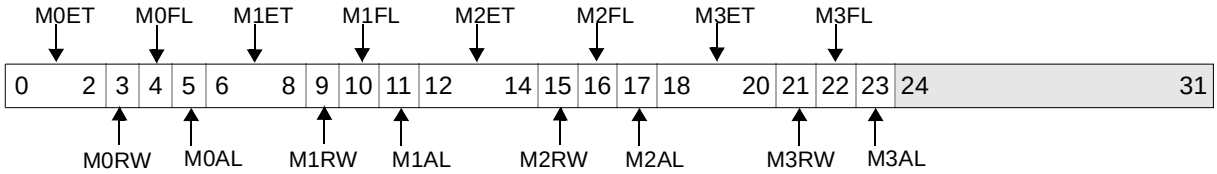


Figure 0-2. Bus Error Syndrome Register 0 (SDRAM0_BESR0)

0:2	M0ET	Error Type for 440 CPU Instruction Cache Controller 000 No Error 001 Reserved 01x ECC Uncorrectable Error 1xx Reserved
3	M0RW	Read/Write Status for 440 CPU Instruction Cache Controller (Read Access only) 0 Write Error 1 Read Error
4	M0FL	BESR Field Lock for 440 CPU Instruction Cache Controller 0 BESR0 Unlocked 1 BESR0 Locked
5	M0AL	BEAR Address Lock for 440 CPU Instruction Cache Controller 0 BEAR Not Locked by 440 CPU Instruction Cache Controller 1 BEAR Locked by 440 CPU Instruction Cache Controller
6:8	M1ET	Error Type for 440 CPU Data Cache Read Controller 000 No Error 001 Reserved 01x ECC Uncorrectable Error 1xx Reserved
9	M1RW	Read/Write Status for 440 CPU Data Cache Controller (Read Access only) 0 Reserved 1 Read Error
10	M1FL	BESR Field Lock for 440 CPU Data Cache Controller 0 BESR1 Unlocked 1 BESR1 Locked
11	M1AL	BEAR Address Lock for 440 CPU Data Cache Controller 0 BEAR Not Locked by 440 CPU Data Cache Controller 1 BEAR Locked by 440 CPU Data Cache Controller
12:14	M2ET	Error Type for 440 CPU Data Cache Write Controller 000 No Error 001 Reserved 01x ECC Uncorrectable Error 1xx Reserved

15	M2RW	Read/Write Status for 440 CPU Data Cache Controller (Write Access only) 0 Write Error 1 Reserved
16	M2FL	BESR Field Lock for 440 CPU Data Cache Controller 0 BESR2 Unlocked 1 BESR2 Locked
17	M2AL	BEAR Address Lock for 440 CPU Data Cache Controller 0 BEAR Not Locked 1 BEAR Locked
18:20	M3ET	Error Type for PLB-PCIX Bridge 000 001 Reserved 01x ECC Uncorrectable Error 1xx Reserved
21	M3RW	Read/Write Status for PLB-PCIX Bridge 0 Write Error 1 Read Error
22	M3FL	BESR Field Lock for Read/Write Status for PLB-PCIX Bridge 0 BESR3 Unlocked 1 BESR3 Locked
23	M3AL	BEAR Address Lock for Read/Write Status for PLB-PCIX Bridge 0 BEAR Not Locked by Read/Write Status for PLB-PCIX Bridge 1 BEAR Locked by Read/Write Status for PLB-PCIX Bridge
24:31		Reserved

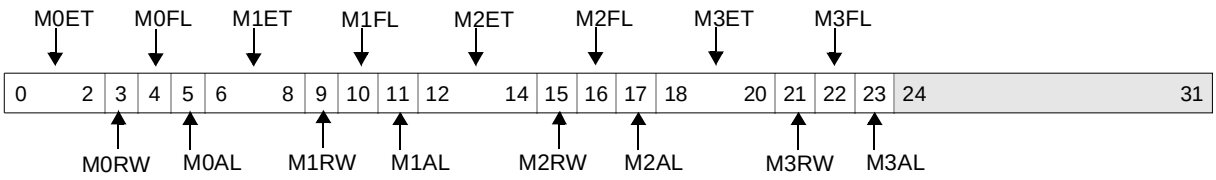


Figure 17-2. Bus Error Syndrome Register 0 (SDRAM0_BESR0)

0:2	M0ET	Error Type for 440 CPU Instruction Cache Controller 000 No Error 001 Reserved 01x ECC Uncorrectable Error 1xx Reserved
3	M0RW	Read/Write Status for 440 CPU Instruction Cache Controller (Read Access only) 0 Write Error 1 Read Error
4	M0FL	BESR Field Lock for 440 CPU Instruction Cache Controller 0 BESR0 Unlocked 1 BESR0 Locked

5	M0AL	BEAR Address Lock for 440 CPU Instruction Cache Controller 0 BEAR Not Locked by 440 CPU Instruction Cache Controller 1 BEAR Locked by 440 CPU Instruction Cache Controller
6:8	M1ET	Error Type for 440 CPU Data Cache Read Controller 000 No Error 001 Reserved 01x ECC Uncorrectable Error 1xx Reserved
9	M1RW	Read/Write Status for 440 CPU Data Cache Controller (Read Access only) 0 Reserved 1 Read Error
10	M1FL	BESR Field Lock for 440 CPU Data Cache Controller 0 BESR1 Unlocked 1 BESR1 Locked
11	M1AL	BEAR Address Lock for 440 CPU Data Cache Controller 0 BEAR Not Locked by 440 CPU Data Cache Controller 1 BEAR Locked by 440 CPU Data Cache Controller
12:14	M2ET	Error Type for 440 CPU Data Cache Write Controller 000 No Error 001 Reserved 01x ECC Uncorrectable Error 1xx Reserved
15	M2RW	Read/Write Status for 440 CPU Data Cache Controller (Write Access only) 0 Write Error 1 Reserved
16	M2FL	BESR Field Lock for 440 CPU Data Cache Controller 0 BESR2 Unlocked 1 BESR2 Locked
17	M2AL	BEAR Address Lock for 440 CPU Data Cache Controller 0 BEAR Not Locked 1 BEAR Locked
18:20	M3ET	Error Type for PLB-PCIX Bridge 000 001 Reserved 01x ECC Uncorrectable Error 1xx Reserved
21	M3RW	Read/Write Status for PLB-PCIX Bridge 0 Write Error 1 Read Error
22	M3FL	BESR Field Lock for Read/Write Status for PLB-PCIX Bridge 0 BESR3 Unlocked 1 BESR3 Locked
23	M3AL	BEAR Address Lock for Read/Write Status for PLB-PCIX Bridge 0 BEAR Not Locked by Read/Write Status for PLB-PCIX Bridge 1 BEAR Locked by Read/Write Status for PLB-PCIX Bridge
24:31		Reserved

17.2.4.2 Bus Error Syndrome Register 1 (SDRAM0_BESR1)

The SDRAM0_BESR1 tracks errors encountered during PLB Master [5-7] accesses to system memory (see ~~Table 17-6, "DDR SDRAM PLB Masters," on page 17-8~~*Table 17-6 DDR SDRAM PLB Masters* on page 479). This register can be accessed at offset 0x08 for the purpose of reading or clearing error status bits. Offset 0x0C is available for test purposes only and allows error bits to be set. A write access to 0x08 should provide a 32-bit mask, **where each 1 in the mask will clear the corresponding bit in the error status register**. A write access to 0x0C should provide a 32-bit mask, where each 1 in the mask will set the corresponding bit in the error status register.

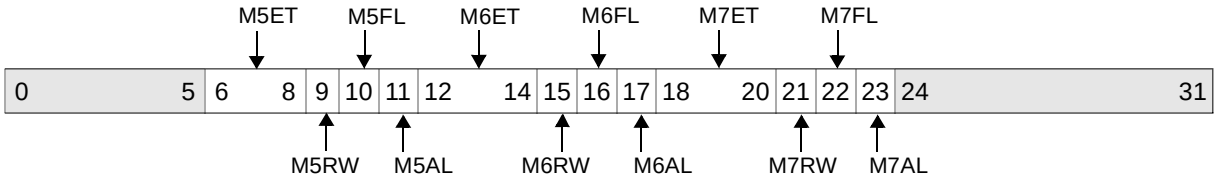


Figure 0-3. Bus Error Syndrome Register 1 (SDRAM0_BESR1)

0:5		Reserved
6:8	M5ET	Error Type for Memory Access Layer (MAL) 000 No Error 001 Reserved 01x ECC Uncorrectable Error 1xx Reserved
9	M5RW	Read/Write Status for MAL 0 Write Error 1 Read Error
10	M5FL	BESR Field Lock for MAL 0 BESR5 Unlocked 1 BESR5 Locked
11	M5AL	BEAR Address Lock for MAL 0 BEAR Not Locked by MAL 1 BEAR Locked by MAL
12:14	M6ET	Error Type for DMA Controller 000 No Error 001 Reserved 01x ECC Uncorrectable Error 1xx Reserved
15	M6RW	Read/Write Status for DMA Controller 0 Write Error 1 Read Error
16	M6FL	BESR Field Lock for DMA Controller 0 BESR6 Unlocked 1 BESR6 Locked
17	M6AL	BEAR Address Lock for DMA Controller 0 BEAR Not Locked by DMA Controller 1 BEAR Locked by DMA Controller

18:20	M7ET	Error Type for OPB to PLB Bridge 000 No Error 001 Reserved 01x ECC Uncorrectable Error 1xx Reserved
21	M7RW	Read/Write Status for OPB to PLB Bridge 0 Write Error 1 Read Error
22	M7FL	BESR Field Lock for OPB to PLB Bridge 0 BESR7 Unlocked 1 BESR7 Locked
23	M7AL	BEAR Address Lock for OPB to PLB Bridge 0 BEAR Not Locked by OPB to PLB Bridge 1 BEAR Locked by OPB to PLB Bridge
24:31		Reserved

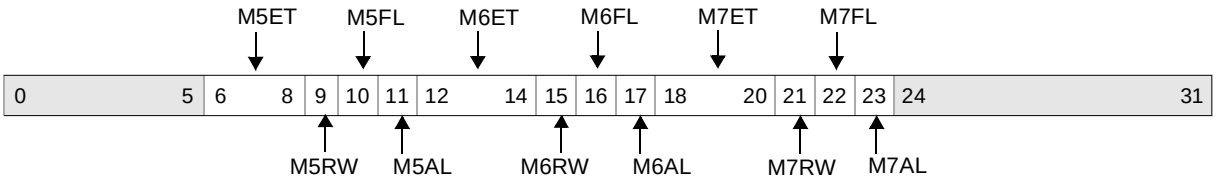


Figure 17-3. Bus Error Syndrome Register 1 (SDRAM0_BESR1)

0:5		Reserved
6:8	M5ET	Error Type for Memory Access Layer (MAL) 000 No Error 001 Reserved 01x ECC Uncorrectable Error 1xx Reserved
9	M5RW	Read/Write Status for Memory Access Layer (MAL) 0 Write Error 1 Read Error
10	M5FL	BESR Field Lock for Memory Access Layer (MAL) 0 BESR5 Unlocked 1 BESR5 Locked
11	M5AL	BEAR Address Lock for Memory Access Layer (MAL) 0 BEAR Not Locked by MAL 1 BEAR Locked by MAL
12:14	M6ET	Error Type for DMA Controller 000 No Error 001 Reserved 01x ECC Uncorrectable Error 1xx Reserved

15	M6RW	Read/Write Status for DMA Controller 0 Write Error 1 Read Error
16	M6FL	BESR Field Lock for DMA Controller 0 BESR6 Unlocked 1 BESR6 Locked
17	M6AL	BEAR Address Lock for DMA Controller 0 BEAR Not Locked by DMA Controller 1 BEAR Locked by DMA Controller
18:20	M7ET	Error Type for OPB to PLB Bridge 000 No Error 001 Reserved 01x ECC Uncorrectable Error 1xx Reserved
21	M7RW	Read/Write Status for OPB to PLB Bridge 0 Write Error 1 Read Error
22	M7FL	BESR Field Lock for OPB to PLB Bridge 0 BESR7 Unlocked 1 BESR7 Locked
23	M7AL	BEAR Address Lock for OPB to PLB Bridge 0 BEAR Not Locked by OPB to PLB Bridge 1 BEAR Locked by OPB to PLB Bridge
24:31		Reserved

17.2.4.3 Master Bus Error Address Register (SDRAM0_BEAR)

The SDRAM Bus Error Address Register (SDRAM0_BEAR) is a 32-bit register containing the address of the access where an uncorrectable ECC error occurred. If the master that initiated the transfer requested error locking, and the SDRAM0_BEAR is not already locked, the contents of SDRAM0_BEAR are locked until the lock bit in one of the Peripheral Bus Error Registers (SDRAM0_BESR0 or SDRAM0_BESR1) is cleared. The content of the SDRAM0_BEAR is accessed indirectly through the SDRAM0_CFGADDR and SDRAM0_CFGDATA registers using the **mfdcr** and **mtdcr** instructions.

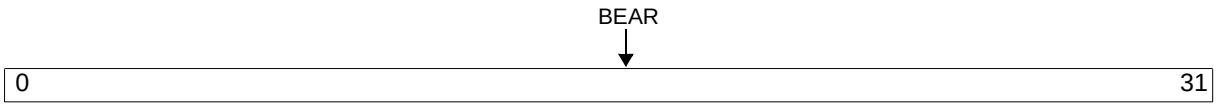


Figure 0-4. Bus Error Address Register (SDRAM0_BEAR)

0:31	BEAR	Address of Bus Error

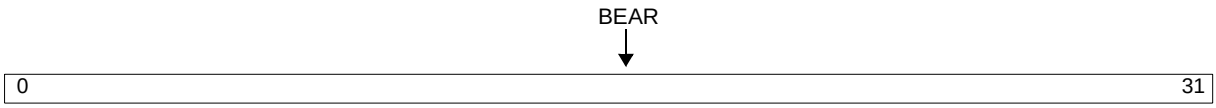


Figure 17-4. Bus Error Address Register (SDRAM0_BEAR)

0:31	BEAR	Address of Bus Error

17.2.4.4 Master Write Interrupt (SDRAM0_MIRQ)

The SDRAM0_MIRQ register tracks errors encountered during Master [0:3 and 5:7] write accesses to system memory (see ~~Table 17-6, "DDR SDRAM PLB Masters," on page 17-8~~*Table 17-6 DDR SDRAM PLB Masters* on page 479). This register can be accessed at offset 0x11 for the purpose of reading or clearing error status bits. A write access to 0x11 provides a 32-bit mask, where each 1 in the mask clears the corresponding bit in the error status register. Offset 0x12 is available for test purposes only and allows error bits to be set. A write access to 0x12 provides a 32-bit mask, where each 1 in the mask sets the corresponding bit in the error status register.

This register is set when a posted write error occurs for the associated PLB master. The correspond-ing DDR SDRAM output will assert and remain asserted for the associated master's transfer until the corresponding register bit is cleared by a DCR write to offset 0x11.

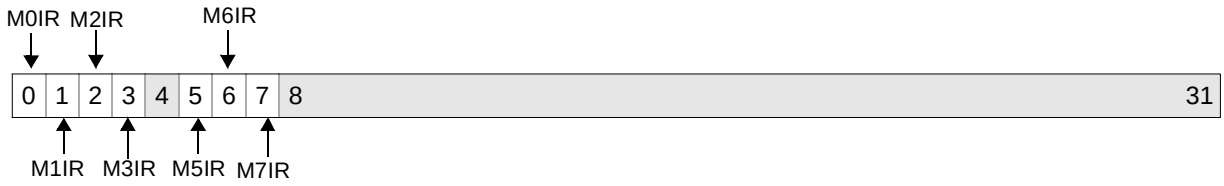


Figure 0-5. Master Write Interrupt (SDRAM0_MIRQ)

0	M0IR	440 CPU Instruction Cache Controller Write Interrupt 0 No Error 1 Write Error
1	M1IR	440 CPU Data Cache Controller Write Interrupt 0 No Error 1 Write Error
2	M2IR	440 CPU Data Cache Controller Write Interrupt 0 No Error 1 Write Error
3	M3IR	PLB_PCIX Bridge Write Interrupt 0 No Error 1 Write Error
4		Reserved
5	M5IR	MAL Write Interrupt 0 No Error 1 Write Error
6	M6IR	DMA Controller Write Interrupt 0 No Error 1 Write Error

7	M7IR	OPB to PLB Bridge Write Interrupt 0 No Error 1 Write Error
8:31		Reserved

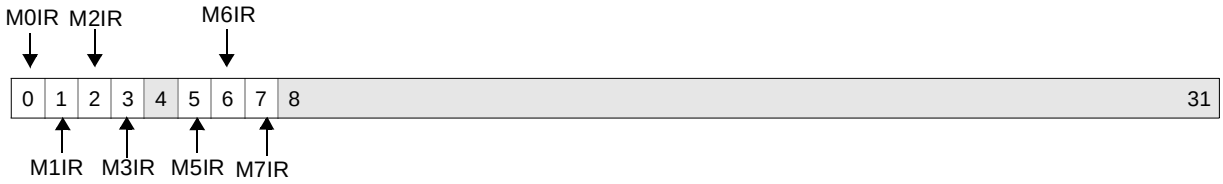


Figure 17-5. Master Write Interrupt (SDRAM0_MIRQ)

0:1		Reserved
0	M0IR	440 CPU Instruction Cache Controller Write Interrupt 0 No Error 1 Write Error
1	M1IR	440 CPU Data Cache Controller Write Interrupt 0 No Error 1 Write Error
2	M2IR	440 CPU Data Cache Controller Write Interrupt 0 No Error 1 Write Error
3	M3IR	PLB_PCIX Bridge Write Interrupt 0 No Error 1 Write Error
4		Reserved
5	M5IR	MAL Write Interrupt 0 No Error 1 Write Error
6	M6IR	DMA Controller Write Interrupt 0 No Error 1 Write Error
7	M7IR	OPB to PLB Bridge Write Interrupt 0 No Error 1 Write Error
8:31		Reserved

17.2.4.5 PLB Slave Interface Options (SDRAM0_SLIO)

This register enables the configuration of the PLB Slave-specific interface options and determines how the interface responds on the PLB, based on the status of the address queue and buffer queues.

See the associated sections of the PLB slave interface for a detailed description of the behavior of each of these options.

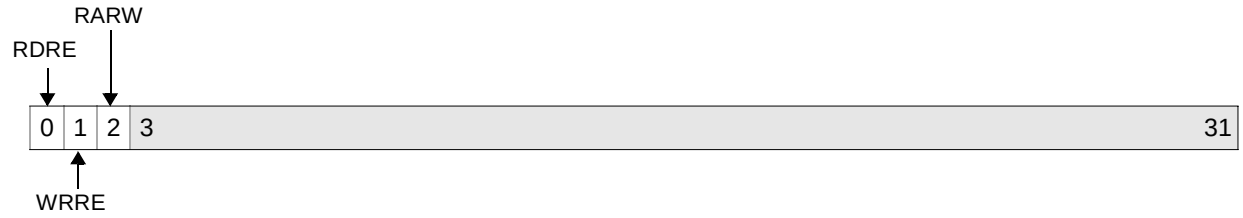


Figure 0-6. PLB Slave Interface Options (SDRAM0_SLIO)

0	RDRE	PLB Slave Read Rearbitrate Enable 0 Disable 1 Enable
1	WRRE	PLB Slave Write Rearbitrate Enable 0 Disable 1 Enable
2	RARW	PLB Read Around Write Enable 0 Disable 1 Enable
3:31		Reserved

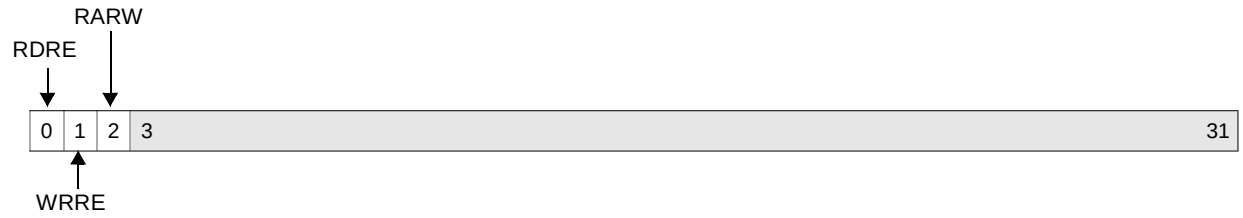


Figure 17-6. PLB Slave Interface Options (SDRAM0_SLIO)

0	RDRE	PLB Slave Read Rearbitrate Enable 0 Disable 1 Enable
1	WRRE	PLB Slave Write Rearbitrate Enable 0 Disable 1 Enable

2	RARW	PLB Read Around Write Enable 0 Disable 1 Enable
3:31		Reserved

17.2.4.6 Memory Controller Options 0 (SDRAM0_CFG0)

This register enables the configuration of the DDR SDRAM controller specific options. See the associated DDR SDRAM sections of this document for specific details and resultant behavior.

All application-specific options selected using SDRAM0_CFG0 should be configured prior to enabling the DDR SDRAM controller.

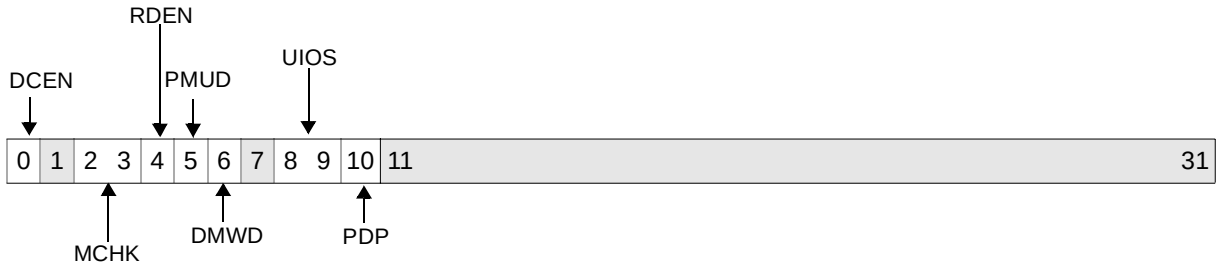


Figure 0-7. Memory Controller Options 0 (SDRAM0_CFG0)

0	DCEN	DDR SDRAM Controller Enable 0 Disable 1 Enable	Once enabled, DDR SDRAM will be initialized using the power-on sequence and subsequently available for access. All DDR SDRAM controller configuration registers should be initialized and valid when this is enabled.
1		Reserved	
2:3	MCHK	Memory Data Error Checking 00 None 01 Reserved 10 ECC generation only (no checking or correction) 11 ECC checking and correction	
4	RDEN	Registered DIMM Enable 0 Disable 1 Enable	
5	PMU	Page Management Unit 0 Enable 1 Disable	When PMU is enabled, up to 8 open pages will be maintained for subsequent accesses. When PMU is disabled, the accessed page will be opened for a given access, remain open for the duration of the access (allowing page hits within PLB sequential bursts) and be precharged (using read/write with autoprecharge command) upon completion of the access.

6	DMWD	DDR SDRAM Width 0 32-bit 1 64-bit	
7		Reserved	
8:9	UIOS[1]	Unused I/O State 00 Active (High-Z on Read, Driven on Write) -switching DQS 01 Active (High-Z on Read, Driven on Write) -static values 10 Static (Always Driven) - static values 11 High-Z	This field can be used to control the state of the unused I/O for various configurations through the corresponding I/O driver data and enable signals. Unused I/O are defined as the I/O associated with a function that is not enabled. Example: CB[0:7], DM[8], and DQS[8] would be unused I/O when ECC is not enabled. Likewise, DATA[32:63], DM[4:7], and DQS[4:7] would be unused I/O in 32-bit mode. All unused I/O receivers will be gated off.
10	PDP	Page Deallocation Policy 0 Pseudo Least Recently Allocated 1 Least Recently Used	This field selects the page deallocation policy when page mode is enabled. Page deallocation occurs when the PMU has 8 open pages and an access does not target a page or bank address associated with one of those open pages. A PMU entry must be deallocated or replaced (and the corresponding page closed). With PMU_DP=0, the PMU entry that has been open longest will be deallocated. With PMU_DP=1, the page that was least recently used (least recently accessed) will be deallocated. **Note:** This distinction is only relevant for memory subsystems employing more than two physical banks (chip selects) of memory.
11:31		Reserved	

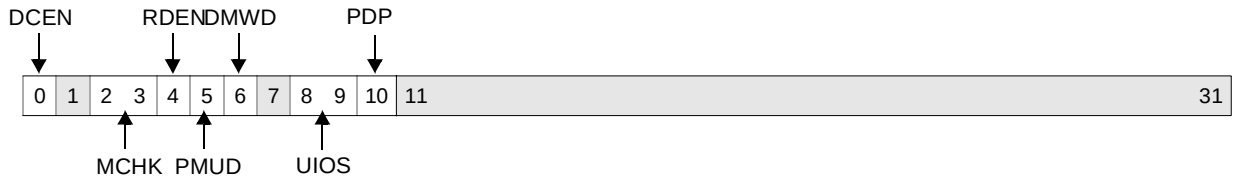


Figure 17-7. Memory Controller Options 0 (SDRAM0_CFG0)

0	DCEN	DDR SDRAM Controller Enable 0 Disable 1 Enable	Once enabled, DDR SDRAM will be initialized using the power-on sequence and subsequently available for access. All DDR SDRAM controller configuration registers should be initialized and valid when this is enabled.
1		Reserved	
2:3	MCHK	Memory Data Error Checking 00 None 01 Reserved 10 ECC generation only (no checking or correction) 11 ECC checking and correction	
4	RDEN	Registered DIMM Enable 0 Disable 1 Enable	
5	PMU	Page Management Unit 0 Enable 1 Disable	When PMU is enabled, up to 8 open pages will be maintained for subsequent accesses. When PMU is disabled, the accessed page will be opened for a given access, remain open for the duration of the access (allowing page hits within PLB sequential bursts) and be precharged (using read/write with autoprecharge command) upon completion of the access.
6	DMWD	DDR SDRAM Width 0 32-bit 1 64-bit	
7		Reserved	
8:9	UIOS[1]	Unused I/O State 00 Active (High-Z on Read, Driven on Write) - switching DQS 01 Active (High-Z on Read, Driven on Write) -static values 10 Static (Always Driven) - static values 11 High-Z	This field can be used to control the state of the unused I/O for various configurations through the corresponding I/O driver data and enable signals. Unused I/O are defined as the I/O associated with a function that is not enabled. Example: CB[0:7], DM[8], and DQS[8] would be unused I/O when ECC is not enabled. Likewise, DATA[32:63], DM[4:7], and DQS[4:7] would be unused I/O in 32-bit mode. All unused I/O receivers will be gated off.

10	PDP	Page Deallocation Policy 0 Pseudo Least Recently Allocated 1 Least Recently Used	This field selects the page deallocation policy when page mode is enabled. Page deallocation occurs when the PMU has 8 open pages and an access does not target a page or bank address associated with one of those open pages. A PMU entry must be deallocated or replaced (and the corresponding page closed). With PMU_DP=0, the PMU entry that has been open longest will be deallocated. With PMU_DP=1, the page that was least recently used (least recently accessed) will be deallocated. **Note: Note:** This distinction is only relevant for memory subsystems employing more than two physical banks (chip selects) of memory.
11:31		Reserved	

Note 1: Effect of UUIOS

The UUIOS settings enable the user to configure the behavior of the unused I/O, if any, at the chip level. Based on the selected UUIOS setting, the DDR SDRAM output connections to the chip level I/O (Z, RG, A, and TS) inputs are controlled to provide the described behavior.

There are two potential unused I/O groupings as follows:

1. CB[0:7], DM[8], and DQS[8] for ECC disabled.
2. DATA[32:63], DM[4:7], and DQS[4:7] for 32-bit mode enabled.

The behavior of these I/O groupings, based on the selected UUIOS setting, is as follows:

Table 0-2. Unused IO - ECC Disabled

Unused IO	UUIOS			
	00	01	10	11
	Write/Read	Write/Read	Write/Read	Write/Read
DM[8]	1 / 1	1 / 1	1/1	1/1
DQS[8]	T / Hi-Z[1]	0 / Hi-Z	0/0	Hi-Z/Hi-Z
CB[0:7]	0s / Hi-Z	0s / Hi-Z	0s/0s	Hi-Z/Hi-Z

Table 0-3. Unused IO - 32-bit Mode Enabled

Unused IO	UUIOS			
	00	01	10	11
	Write/Read	Write/Read	Write/Read	Write/Read
DM[4:7]	1s / 1s	1s / 1s	1s/1s	1s/1s
DQS[4:7]	T / Hi-Z[1]	0s / Hi-Z	0s/0s	Hi-Z/Hi-Z
DATA[32:63]	0's / Hi-Z	0s / Hi-Z	0s/0s	Hi-Z/Hi-Z

Unused IO	UUIOS			
	00	01	10	11
	Write/Read	Write/Read	Write/Read	Write/Read
DM[8]	1 / 1	1 / 1	1/1	1/1
DQS[8]	T / Hi-Z[1]	0 / Hi-Z	0/0	Hi-Z/Hi-Z
CB[0:7]	0s / Hi-Z	0s / Hi-Z	0s/0s	Hi-Z/Hi-Z

Unused IO	UUIOS			
	00	01	10	11
	Write/Read	Write/Read	Write/Read	Write/Read
DM[4:7]	1s / 1s	1s / 1s	1s/1s	1s/1s
DQS[4:7]	T / Hi-Z[1]	0s / Hi-Z	0s/0s	Hi-Z/Hi-Z
DATA[32:63]	0's / Hi-Z	0s / Hi-Z	0s/0s	Hi-Z/Hi-Z

T: Toggles on Write Cycles

Hi-Z: High Impedance

17.2.4.7 Memory Controller Options 1 (SDRAM0_CFG1)

This register enables the configuration of the DDR SDRAM controller power management and self-refresh features. See the associated DDR SDRAM sections of this document for specific details and resultant behavior.

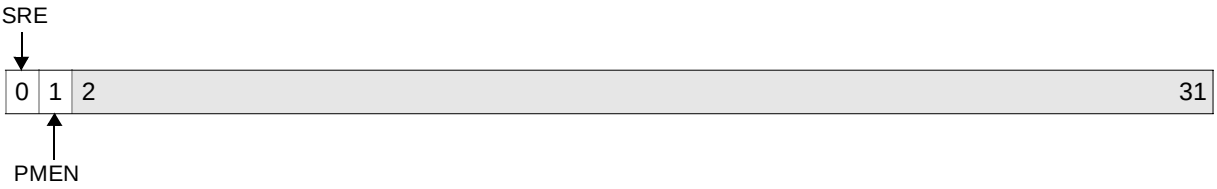


Figure 0-8. Memory Controller Options 1 (SDRAM0_CFG1)

0	SRE	Self-Refresh Entry 0 Exit 1 Enter	This bit may be set by software (using DCR write). Once set, the SDRAM is maintained in self-refresh mode until this bit is reset by software. The DDR_SDRAM controller responds to this bit independent of the state of SDRAM0_CFG0[DCEN].
1	PMEN	Power Management Enable 0 Disable 1 Enable	Refer to section on "Power Management" of this document for specific details.
2:31		Reserved	

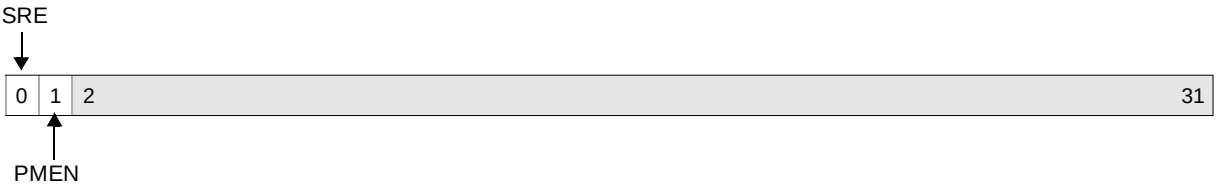


Figure 17-8. Memory Controller Options 1 (SDRAM0_CFG1)

0	SRE	Self-Refresh Entry 0 Exit 1 Enter	This bit may be set by software (using DCR write). Once set, the SDRAM is maintained in self-refresh mode until this bit is reset by software. The DDR_SDRAM controller responds to this bit independent of the state of SDRAM0_CFG0[DCEN].
1	PMEN	Power Management Enable 0 Disable 1 Enable	Refer to section on "Power Management" of this document for specific details.
2:31		Reserved	

17.2.4.8 DDR SDRAM Device Options (SDRAM0_DEVOPT)

This registers allows for the configuration of the defined DDR SDRAM device-specific options. Users should consult the vendor-specific DDR SDRAM device specification for a detailed description of these options.

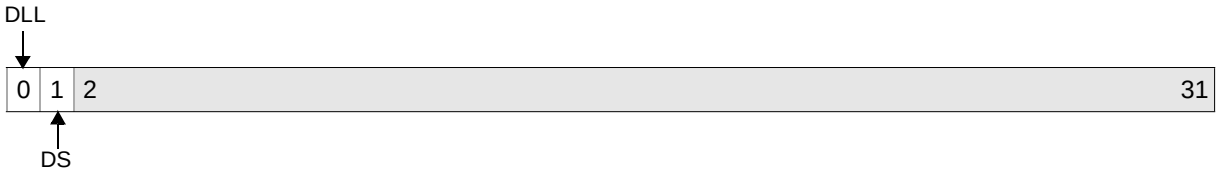


Figure 0-9. DDR_SDRAM Device Options (SDRAM0_DEVOPT)

0	DLL	DDR_SDRAM Device DLL Disable 0 Enable 1 Disable	For normal operation, enable DLL. The DLL setting controls the state of address bit during the initialization of the Extended Mode Register.
1	DS	DDR_SDRAM Device I/O Drive Strength 0 Normal 1 Weak	For normal operation, clear DS. The DS setting controls the state of address bit MemAddr1 during the initialization of the Extended Mode Register.
2:31		Reserved	

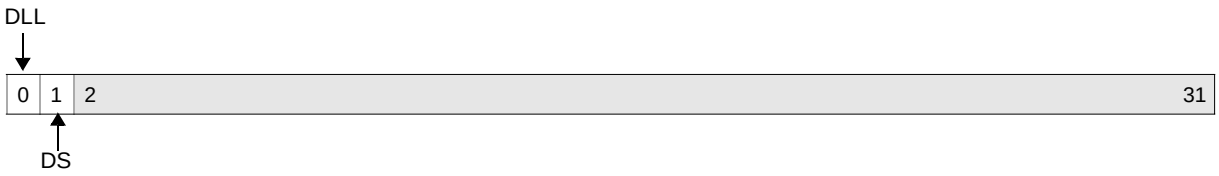


Figure 17-9. DDR_SDRAM Device Options (SDRAM0_DEVOPT)

0	DLL	DDR_SDRAM Device DLL Disable 0 Enable 1 Disable	For normal operation, enable DLL. The DLL setting controls the state of address bit during the initialization of the Extended Mode Register.
1	DS	DDR_SDRAM Device I/O Drive Strength 0 Normal 1 Weak	For normal operation, clear DS. The DS setting controls the state of address bit MemAddr1 during the initialization of the Extended Mode Register.
2:31		Reserved	

17.2.4.9 Memory Controller Status (SDRAM0_MCSTS)

This register provides real-time status to the system software on the operation of the DDR SDRAM controller. The information returned indicates the DDR SDRAM status at the time of the DCR read.

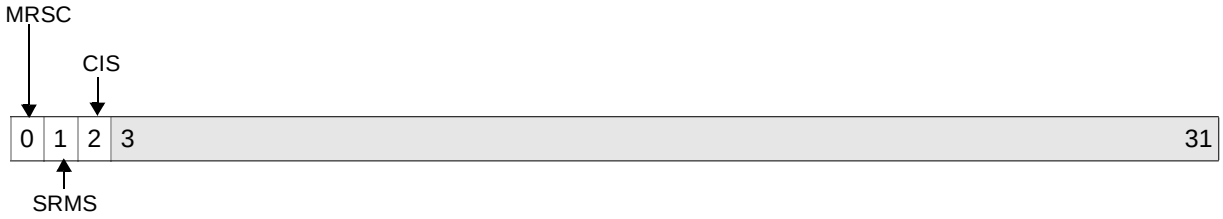


Figure 0-10. Memory Controller Status (SDRAM0_MCSTS)

0	MRSC	MRS Command Complete 0 MRS Not Complete 1 MRS Complete	This bit will be set to "1" once the DDR SDRAM controller has successfully completed the Mode Register Set Command, which results from setting DC_EN in SDRAM0_CFG0. Clearing DC_EN will clear this bit in the following cycle.
1	SRMS	Self-Refresh Mode Status 0 SDRAM is not in Self-Refresh Mode 1 SDRAM is in Self-Refresh Mode	This bit will be set upon the successful completion of Self-Refresh Mode entry, that is, the SDRAM device has been placed in Self-Refresh Mode. This bit will be cleared when Self-Refresh Mode has been exited. This bit applies to the software-initiated Self-Refresh Mode using SDRAM0_CFG0[SRE].
2	CIS	Core Idle Status 0 Busy 1 Idle	Indicates that there are no current or pending queued read/write accesses.
3:31		Reserved	

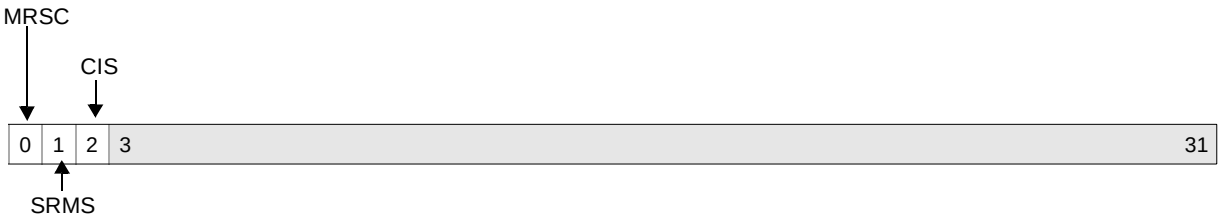


Figure 17-10. Memory Controller Status (SDRAM0_MCSTS)

0	MRSC	MRS Command Complete 0 MRS Not Complete 1 MRS Complete	This bit will be set to "1" once the DDR SDRAM controller has successfully completed the Mode Register Set Command, which results from setting DC_EN in SDRAM0_CFG0. Clearing DC_EN will clear this bit in the following cycle.
1	SRMS	Self-Refresh Mode Status 0 SDRAM is not in Self-Refresh Mode 1 SDRAM is in Self-Refresh Mode	This bit will be set upon the successful completion of Self-Refresh Mode entry, that is, the SDRAM device has been placed in Self-Refresh Mode. This bit will be cleared when Self-Refresh Mode has been exited. This bit applies to the software-initiated Self-Refresh Mode using SDRAM0_CFG0[SRE].
2	CIS	Core Idle Status 0 Busy 1 Idle	Indicates that there are no current or pending queued read/write accesses.
3:31		Reserved	

17.2.4.10 Refresh Timer Register (SDRAM0_RTR)

The 16-bit field of RTR determines the memory refresh rate for the DDR SDRAM. The internal counter runs at the controller clock frequency; thus, at 100/133 MHz, a value of 0x5F0/0x7E0 produces a refresh interval of 15.20µs (1520 times 10ns equals 15.20µs/2016 times 7.5nS equals 15.12µs). This register must be programmed to accommodate the DDR SDRAM device-specific refresh interval for the target operating frequency.

Note: Bit 0 is the Most Significant Bit and bit 15 is the Least Significant Bit.

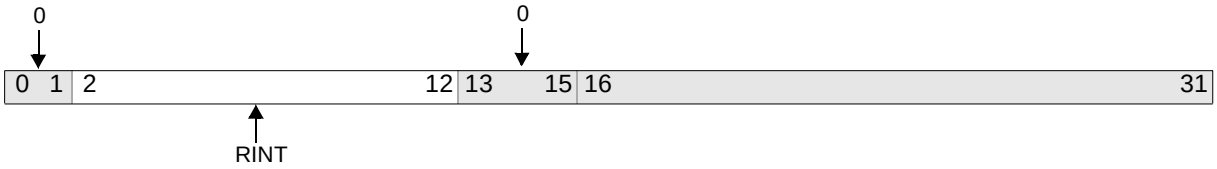


Figure 0-11. Refresh Timing Register (SDRAM0_RTR)

0:1		Reserved	Hardcoded to zero
2:12	RINT	Refresh Interval	Programmable
13:15		Reserved	Hardcoded to zero
16:31		Reserved	

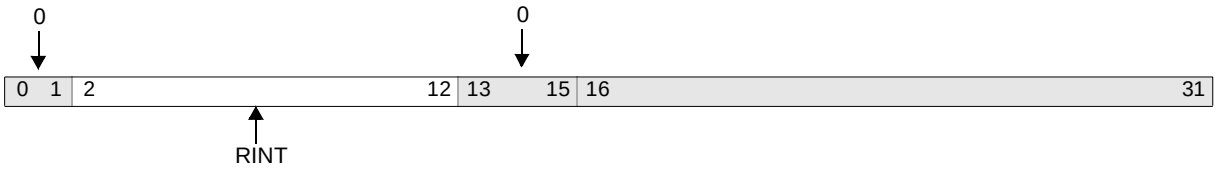


Figure 17-11. Refresh Timing Register (SDRAM0_RTR)

0:1		Reserved	Hardcoded to zero
2:12	RINT	Refresh Interval	Programmable
13:15		Reserved	Hardcoded to zero
16:31		Reserved	

17.2.4.11 Power Management Idle Timer (SDRAM0_PMIT)

The 10-bit field of the SDRAM0_PMIT determines the number of controller clock cycles that the DDR SDRAM controller must be idle before making a sleep request to the Clock and Power Management controller (CPM), when power management is enabled using SDRAM0_CFG1[PMEN]. The reset value initializes this timer to 32-clock cycles.

Note 1: The minimum granularity of the idle timer is 32-clock cycles.

Note 2: Bit 0 is the most significant bit and bit 4 is the least significant bit.

Figure 0-12. Power Management Idle Timer (SDRAM0_PMIT)

0:4	PM_C	Power Management Count	Programmable
5:9		Reserved	Hardcoded to ones
10:31		Reserved	Hardcoded to zero

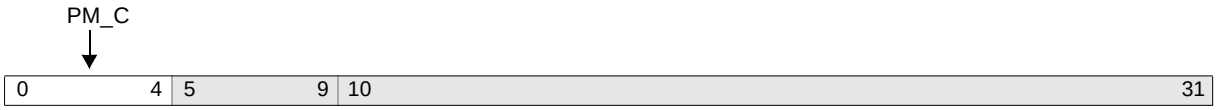


Figure 17-12. Power Management Idle Timer (SDRAM0_PMIT)

0:4	PM_C	Power Management Count	Programmable
5:9		Reserved	Hardcoded to ones
10:31		Reserved	Hardcoded to zero

17.2.4.12 PLB UABus Base Address (SDRAM0_UABBA)

The UABBA must be configured in conjunction with the SDRAM0_BnCR registers to define the base address of each memory bank.

To maintain default system memory map, the SDRAM0_UABBA[28:31] should stay at zero. Contact PowePC support if there is a need to change it.

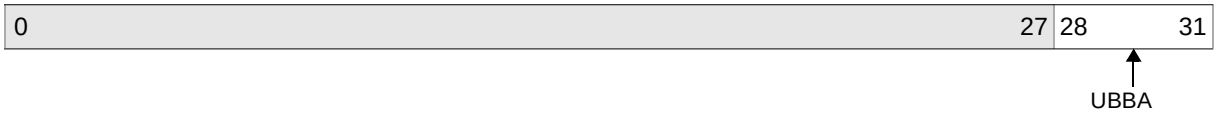


Figure 0-13. PLB UABus Base Address (SDRAM0_UABBA)

0:27		Reserved	
28:31	UBBA	PLB UABus Base Address	Defines the 4 GB aligned region in which the physical memory is located.

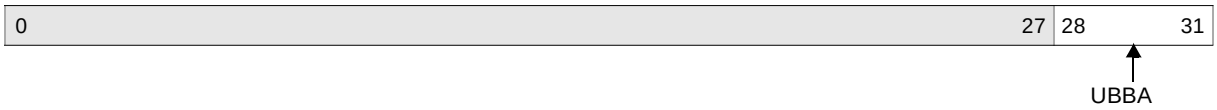


Figure 17-13. PLB UABus Base Address (SDRAM0_UABBA)

0:27		Reserved	
28:31	UBBA	PLB UABus Base Address	Defines the 4 GB aligned region in which the physical memory is located.

17.2.4.13 Memory 0 - 3 Configuration (SDRAM0_B0CR-SDRAM0_B3CR)

The DDR SDRAM memory controller provides support for four logical banks. The SDRAM0_ BnCR registers configure and enable memory in each respective logical bank. Only logical banks are enabled, SDRAM0_BnCR[BE]=1, are initialized when the DDR SDRAM controller is enabled, SDRAM0_CFG[DCE]=1. Since the SDRAM0_BnCR registers cannot be modified when SDRAM0_CFG[DCE]=1, adding or removing memory banks requires that the DDR SDRAM controller be disabled and then reinitialized.

Note: The base address must be aligned on a boundary that matches the size of the region as defined by the SDSZ field. For example, a 16-MB region must begin on an address that is divisible by 16 MB.

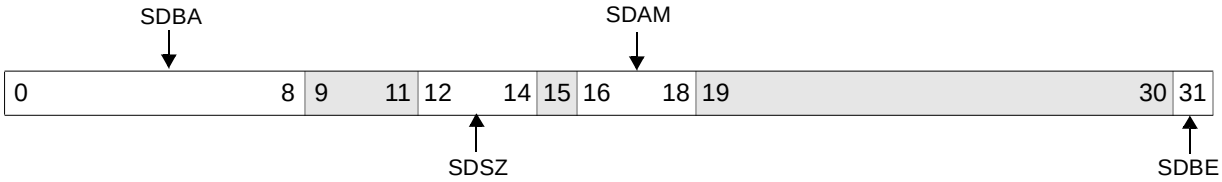


Figure 0-14. Memory 0-3 Configuration (SDRAM0_B0CR-SDRAM0_B3CR)

0:8	SDBA	Base Address	
9:11		Reserved	
12:14	SDSZ	Size 000 Reserved 001 8 MB 010 16 MB 011 32 MB 100 64 MB 101 128 MB 110 256 MB 111 512 MB	
15		Reserved	
16:18	SDAM	Addressing Mode 000 Mode 1 001 Mode 2 010 Mode 3 011 Mode 4 1xx Reserved	See the table on "DDR SDRAM Addressing Modes" on next page for details.
19:30		Reserved	
31	SDBE	Memory Bank Enable	

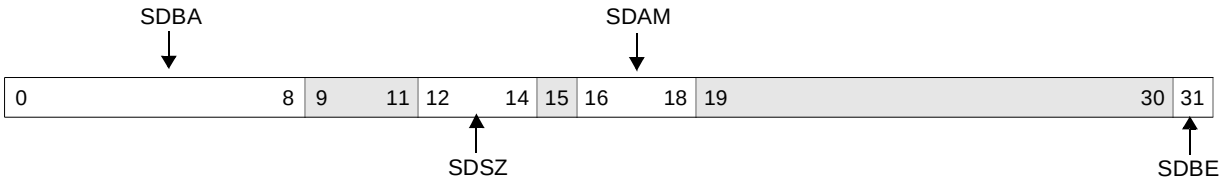


Figure 17-14. Memory 0-3 Configuration (SDRAM0_B0CR-SDRAM0_B3CR)

0:8	SDBA	Base Address	
9:11		Reserved	
12:14	SDSZ	Size 000 Reserved 001 8 MB 010 16 MB 011 32 MB 100 64 MB 101 128 MB 110 256 MB 111 512 MB	
15		Reserved	
16:18	SDAM	Addressing Mode 000 Mode 1 001 Mode 2 010 Mode 3 011 Mode 4 1xx Reserved	See the table on "DDR SDRAM Addressing Modes" on next page for details.
19:30		Reserved	
31	SDBE	Memory Bank Enable	

Table 17-7. DDR SDRAM Addressing Modes

Addressing Mode	DDR SDRAM Memory Organization[1]
Mode 1	11 x 8 - 4 Bank 12 x 8 - 4 Bank 13 x 8 - 4 Bank
Mode 2	11 x 9 - 4 Bank 12 x 9 - 4 Bank 13 x 9 - 4 Bank
Mode 3	11 x10 - 4 Bank 12 x10 - 4 Bank 13 x10 - 4 Bank
Mode 4	11 x11 - 4 Bank 12 x11 - 4 Bank 13 x11 - 4 Bank

1. Row x Column - Number of physical banks contained in the DDR SDRAM module.

17.2.4.14 SDRAM Timing Register 0 (SDRAM0_TR0)

This register must be programmed to configure the DDR SDRAM device-specific timing parameters, expressed in units of clock cycles, as defined by the selected DDR SDRAM device specification.

Settings below the minimums defined by the device specifications and the minimums, in terms of clock cycles, calculated from the formulas provided herein are not supported.

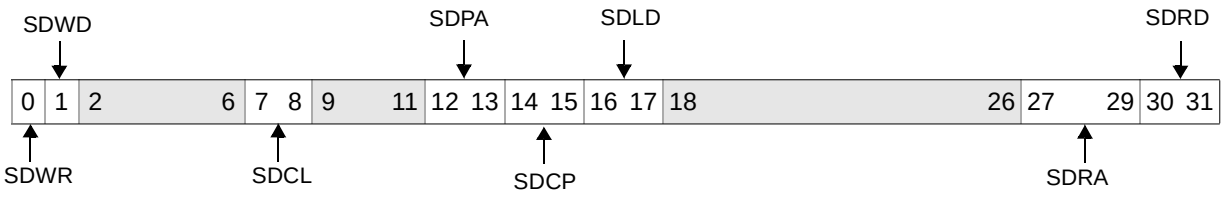


Figure 0-15. SDRAM Timing Register 0 (SDRAM0_TR0)

0	SDWR	DDR SDRAM Write Recovery 0 2 CLK 1 3 CLK	T_{wr}
1	SDWD	DDR SDRAM Write to Write delay when crossing a chip select boundary 0 0 CLK 1 1 CLK	This field configures the delay between back to back write cycles that cross chip select boundaries. A setting of 0 enables seamless writes when crossing chip select boundaries, while a setting of 1 will insert a single clock cycle delay between back-to-back writes when crossing a chip select boundary.
2:6		Reserved	
7:8	SDCL	DDR SDRAM CAS_ Latency 00 Reserved 01 2 CLK 10 2.5 CLK 11 3 CLK	T_{aa} This field indicates the DDR SDRAM device CAS latency (independent of SDRAM0_CFG0[R_DIMM_EN]) and is used directly during the SDRAM device mode set command to configure the DDR SDRAM. See the section on "Registered DIMM Support" for the supported configurations and CAS_ Latency settings. **Note:** Setting SD_CL to 2.5 CLK generally requires that SDRAM0_TR1[RDCD] be set to 1, SDRAM0_TR1[RDSS] be set to T2 Sample, and SDRAM0_TR1[RDSL] be set to Stage 3.

9:11		Reserved	
12:13	SDPA	DDR SDRAM CBR Precharge Command to next Activate Command minimum 00 Reserved 01 2 CLK 10 3 CLK 11 4 CLK	T_{rp}
14:15	SDCP	DDR SDRAM Read/Write Command to Precharge Command 00 2 CLK 01 3 CLK 10 4 CLK 11 5 CLK	T_{ras} - T_{rcd} The minimum value is calculated by first determining the values of T_{ras} and T_{rcd} in units of clock cycles (divide the respective device-specific AC timing specifications by the clock cycle time and round up to the nearest whole clock). Then apply the formula provided. SDCP settings that violate this rule (that is, settings less than the minimum calculated above for the device-specific timing parameters), independent of the programmed value of SDRD, are not supported.
	SDLD	DDR SDRAM Command Leadoff 00 1 CLK 01 2 CLK 10 Reserved 11 Reserved	
18:26		Reserved	
27:29	SDRA	DDR SDRAM CBR Refresh Command to next Activate Command minimum 000 6 CLK 001 7 CLK 010 8 CLK 011 9 CLK 100 10 CLK 101 11 CLK 110 12 CLK 111 13 CLK	T_{rfc}
30:31	SDRD	DDR SDRAM RAS to CAS Delay 00 Reserved 01 2 CLK 10 3 CLK 11 4 CLK	T_{rcd}

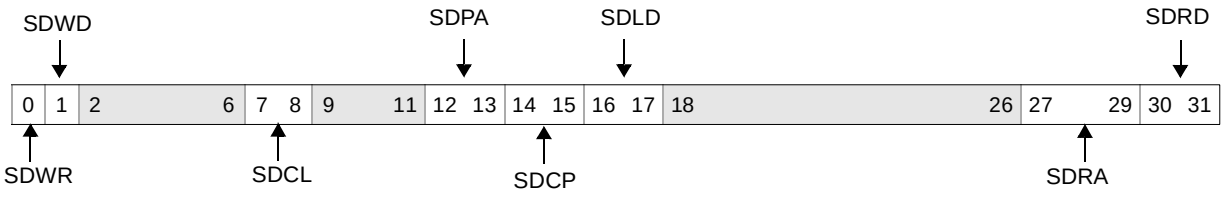


Figure 17-15. SDRAM Timing Register 0 (SDRAM0_TR0)

0	SDWR	DDR SDRAM Write Recovery 0 2 CLK 1 3 CLK	T_{wr}
1	SDWD	DDR SDRAM Write to Write delay when crossing a chip select boundary 0 0 CLK 1 1 CLK	This field configures the delay between back to back write cycles that cross chip select boundaries. A setting of 0 enables seamless writes when crossing chip select boundaries, while a setting of 1 will insert a single clock cycle delay between back-to-back writes when crossing a chip select boundary.
2:6		Reserved	
7:8	SDCL	DDR SDRAM CAS_ Latency 00 Reserved 01 2 CLK 10 2.5 CLK 11 3 CLK	T_{aa} This field indicates the DDR SDRAM device CAS latency (independent of SDRAM0_CFG0[R_DIMM_EN]) and is used directly during the SDRAM device mode set command to configure the DDR SDRAM. See the section on "Registered DIMM Support" for the supported configurations and CAS_ Latency settings. **Note:** Setting SD_CL to 2.5 CLK generally requires that SDRAM0_TR1[RDCD] be set to 1, SDRAM0_TR1[RDSS] be set to T2 Sample, and SDRAM0_TR1[RDSL] be set to Stage 3.
9:11		Reserved	
12:13	SDPA	DDR SDRAM CBR Precharge Command to next Activate Command minimum 00 Reserved 01 2 CLK 10 3 CLK 11 4 CLK	T_{rp}

14:15	SDCP	DDR SDRAM Read/Write Command to Precharge Command 00 2 CLK 01 3 CLK 10 4 CLK 11 Reserved	T_{ras} - T_{rcd} The minimum value is calculated by first determining the values of T_{ras} and T_{rcd} in units of clock cycles (divide the respective device-specific AC timing specifications by the clock cycle time and round up to the nearest whole clock). Then apply the formula provided. SDCP settings that violate this rule (that is, settings less than the minimum calculated above for the device-specific timing parameters), independent of the programmed value of SDRD, are not supported.
16:17	SDLD	DDR SDRAM Command Leadoff 00 1 CLK 01 2 CLK 10 Reserved 11 Reserved	
18:26		Reserved	
27:29	SDRA	DDR SDRAM CBR Refresh Command to next Activate Command minimum 000 6 CLK 001 7 CLK 010 8 CLK 011 9 CLK 100 10 CLK 101 11 CLK 110 12 CLK 111 13 CLK	T_{rfc}
30:31	SDRD	DDR SDRAM RAS to CAS Delay 00 Reserved 01 2 CLK 10 3 CLK 11 4 CLK	T_{rcd}

17.2.4.15 SDRAM Timing Register 1 (SDRAM0_TR1)

SDRAM0_TR1 controls the various aspects of the DDR SDRAM read data path.

DDR memory is source synchronous in that a read data strobe/clock (DQS) is propagated with the DDR read data. Read data is initially sampled in the DDR DQS clock domain and must be synchronized with the controller internal clock domain for transfer on the PLB Bus. Because the read Data and DQS are propagated from the DDR memory to the DDR SDRAM controller and derived from the DDR Memory clock, which is generated and propagated from the controller to the DDR memory, there is the potential for a varying (across different implementations) range of read Data/DQS arrival times relative to the fixed internal controller clock domains. Note that the read Data/DQS arrival is dependent on the DDR memory clock, and the memory clock is affected by the configuration of SDRAM0_CLKTR. The effects of such configuration must be understood and accounted for in determining the appropriate SDRAM0_TR1 settings.

As such, the DDR SDRAM controller read data path is structured to enable, in conjunction with SDRAM0_TR1, configurable support for a wide range of DDR read Data and read DQS arrival times with respect to the controller internal clock domains. The discussion that follows refers to determinations that should be made based on a complete and thorough memory subsystem timing budget analysis. The variations and options discussed are assumed to be static, once determined, for the specific application. Dynamically varying SDRAM0_TR1 during in-progress read accesses is not supported and doing so may yield unpredictable behavior and/or DDR SDRAM malfunction.

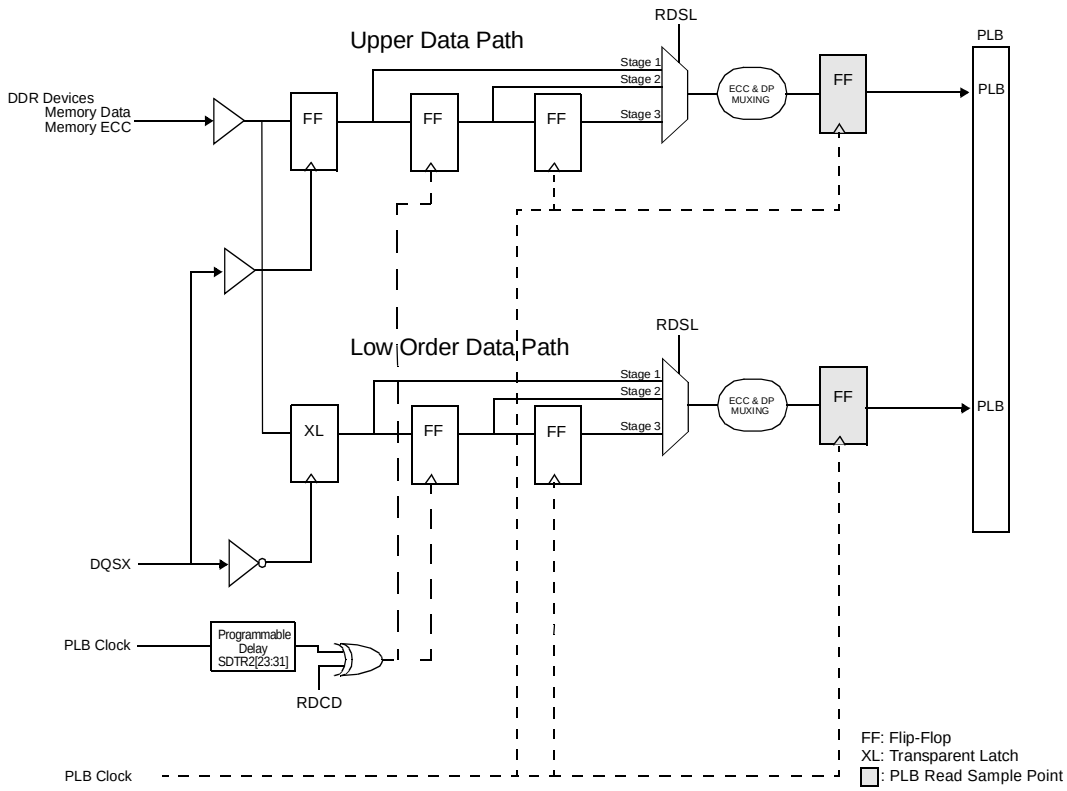


Figure 17-16. DDR SDRAM Controller Read Data Path Structure The DDR SDRAM read data path structure is composed of three stages of registers for both the high order (rising edge of DQS sample) data and ECC checkbits and the low order (falling edge of DQS sample) data and ECC checkbits. The logic elements are referred to as the "upper" data path and the "lower" data path in the following sections. Stages 1, 2, and 3 are placed physically adjacent to each other and to the IO cell for each associated bit of the read data bus.

Stage 1 consists of a flip-flop (FF) for the upper data path and a transparent latch (XL) for the lower data path. The upper data is captured into the FF with the rising edge of read DQS, while the lower data flushes through the XL when the read DQS is high and captured and held with the falling edge of read DQS.

Stage 2 consists of a FF for both the upper and lower data path. Stage 2 has programmable delay capability in one coarse delay of 1/2 cycle (SDRAM0_TR1[20]) as well as finer delay capability using the Read Clock Delay Tuning Bits (SDRAM0_TR1[23:31]). The coarse half cycle delay is provided primarily for CAS Latency 2.5 support and enables read Data to be captured into Stage 2 while compensating for the 1/2 cycle latency adder and allowing proper synchronization with the internal DDR SDRAM clock domain. The delay effect of the Read Clock Delay Tuning Bits (SDRAM0_TR1[23:31]) is dependent on many factors, including delay line physical placement, ASIC process variation, and voltage/temperature variations. The Delay Line Calibration Register (SDRAM0_DLYCAL) may be used to calibrate the tuning bits.

Stage 3 consists of a FF for both the upper and the lower data path. Stage 3 is clocked by the main DDR SDRAM 1xPLB clock, which also clocks the DDR SDRAM internal registers, read and write data buffers, as well as the PLB interface logic. There is no programmable delay capability for this clock.

The PLB Read Sample Point register is the final stage of relevance in the read data path structure. It is at this stage that the final relative clock domain synchronization occurs and that the DDR memory read data is sampled and driven back to the PLB read data bus in response to a given PLB Master read request.

SDRAM0_TR1 [RDSL]/[RDCD]/[RDCT]

Within SDRAM0_TR1, it is the SDRAM0_TR1[RDSL] setting that determines which of the three Stages will be the source of the data steered to the input of the Read Sample Point register. The critical path within the memory subsystem timing budget for a DDR application will be the lower read data; that is, the read data launched and propagated with the falling edge of read DQS, captured and synchronized to the internal clock domain(s).

In order to use Stage 1, lower read Data/DQS must arrive early enough so that the data can flush through the XL downstream and meet the setup and hold time requirements at the PLB Read Sample Point register. Note that the setup/hold time requirements at the PLB Read Sample Point register will differ for ECC enabled vs. ECC disabled.

If the lower read Data/DQS arrival is such that the downstream setup time requirements cannot be met at the PLB Read Sample Point register along the Stage 1 path, Stage 2 may be used. This is provided that the lower read Data arrival is such that the setup/hold time requirements at Stage 2 can be met. In the event that the lower read Data/DQS arrival is such that the read Data does not meet the setup time requirements at Stage 2, the Stage 2 sample clock may be delayed so that the setup/hold time requirements can be met to enable the Data to be captured.

Sample Stage 3 is provided as a final synchronization option for circumstances (read Data/DQS arrival) where the Stage 2 sample clock is delayed to the point where the register to register timing path between Stage 2 and the downstream PLBS Read Sample Point register can no longer be met. Such a circumstance may, for example, be running a CAS Latency 2.5 device with the Stage 2 read sample clock delay by 1/2

cycle or more (through the combination of SDRAM0_TR1[RDCD] and SDRAM0_TR1[RDCT] settings) with ECC enabled at a high frequency. If necessary, data can be captured directly into Stage 3 and then passed to the Read Sample Point register in the following clock cycle.

SDRAM0_TR1 [RDSS]

The SDRAM0_TR1[RDSS] setting determines the clock cycle, relative to the DDR read command, in which the internal logic should sample the read Data at the PLB Read Sample Point register. The relative cycles for the defined sample points are CAS Latency dependent. Both unbuffered (R_DIMM_EN=0) and registered (R_DIMM_EN=1) interfaces (such as Registered DIMMs) are depicted, as the configured sample point (and the arrival of the DDR read Data) differs based on these two types of memory interfaces. RDSS must be configured, in conjunction with SDRAM0_TR1[RDSL], to select the source and sample point for the incoming read Data.

CAS Latency 2 Devices

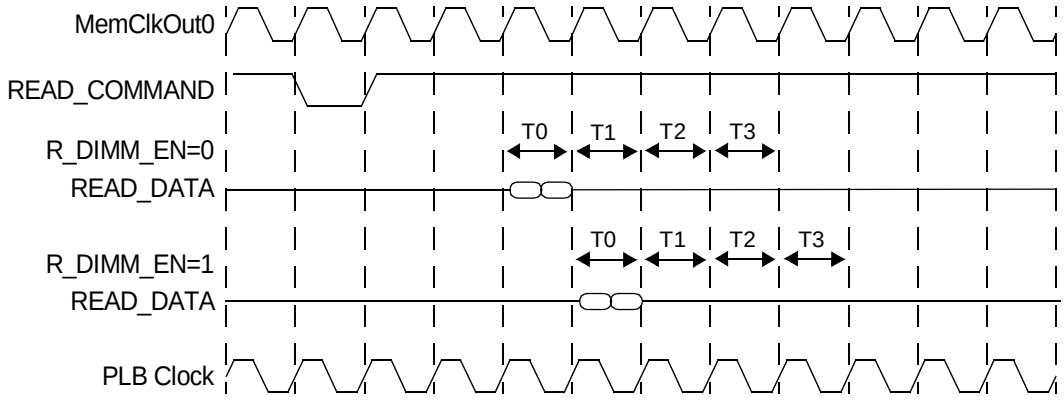


Figure 17-17. Read Sample Cycle: CL = 2 The DDR memory devices launch read data nominally coincident with the second rising edge of its Memory Clock following the cycle in which the read command was sampled by the device.

A T0 Sample configures the DDR SDRAM controller to sample incoming read data into the PLB Read Sample Point register at the end of cycle T0. This is the end of the first cycle in which data, both upper and lower, was launched from the DDR device. A T0 Sample should only be used in conjunction with Read Sample Stage 1 and requires that the timing budget support the use of Stage 1. A T0 Sample should never be used if SDRAM0_TR1[RDSL] is configured for Stages 2 or 3.

A T1 Sample configures the DDR SDRAM controller to sample incoming read data into the PLB Read Sample Point register at the end of cycle T1. This is the end of the second cycle, relative to the cycle in which data, both upper and lower, was launched from the DDR device. A T1 Sample will generally be used in conjunction with Read Sample Stage 2 and requires that the timing budget support the use of Stage 2.

A T2 Sample configures the DDR SDRAM controller to sample incoming read data into the PLB Read Sample Point register at the end of cycle T2. This is the end of the third cycle, relative to the cycle in which data, both upper and lower, was launched from the DDR device. A T2 Sample will generally be used in conjunction with Read Sample Stage 3 and requires that the timing budget support the use of Stage 3. It is expected that CAS Latency 2 applications would never need to use a T2 Sample or Stage 3.

A T3 Sample configures the DDR SDRAM controller to sample incoming read data into the PLB Read Sample Point register at the end of cycle T3. This is the end of the fourth cycle, relative to the cycle in which data, both upper and lower, was launched from the DDR device. It is expected that CAS Latency 2 applications would never need to use a T3 Sample and, as such, should not be configured to do so.

CAS Latency 2.5 Devices

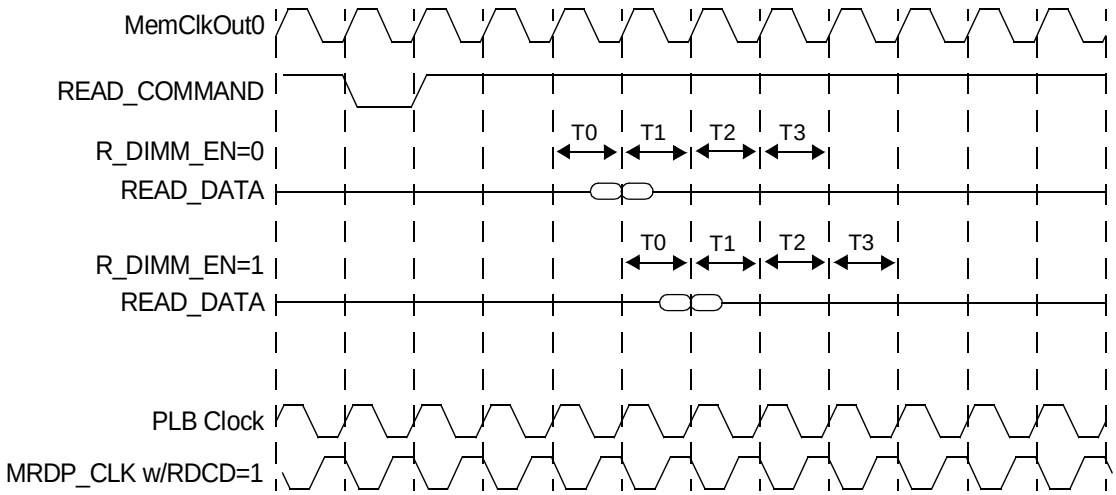


Figure 17-18. Read Sample Cycle: CL = 2.5 For CAS Latency 2.5 devices, the DDR device launches read data nominally coincident with the third *falling* edge of it's Memory Clock following the cycle in which the read command was sampled by the device. Note that a half cycle delay (SDRAM0_TR1[RDCD]=1) on MRDP_CLK enables read Data capture into Stage 2 to compensate for the 1/2 latency adder (associated with CL=2.5 devices) and the resultant mismatch with the internal read sample clock. The half cycle delay also allows proper synchronization with the internal DDR SDRAM clock domain.

A T0 Sample configures the DDR SDRAM controller to sample incoming read data into the PLB Read Sample Point register at the end of cycle T0. This is the end of the first *half* cycle from which data was launched from the DDR device. A T0 Sample should only be used in conjunction with Read Sample Stage 1 and requires that the timing budget support the use of Stage 1. A T0 Sample should never be used if SDRAM0_TR1[RDSL] is configured for Stages 2 or 3. It is expected that the use of T0 Sample for CL=2.5 applications would be unlikely given the half cycle latency adder of the devices. That said, it may be possible, through SDRAM0_CLKTR configuration, to recover the half cycle latency adder provided that the memory subsystem timing budget supports such configuration, thereby enabling a T0 Sample. In such a configuration, the appropriate setting for SDRAM0_TR1[RDCD] must be determined.

A T1 Sample configures the DDR SDRAM controller to sample incoming read data into the PLB Read Sample Point register at the end of cycle T1. This is the end of the third *half* cycle, relative to the *half* cycle from which data was launched from the DDR device. A T1 Sample will generally be used in conjunction with Read Sample Stage 2 and requires that the timing budget support the use of Stage 2.

A T2 Sample configures the DDR SDRAM controller to sample incoming read data into the PLB Read Sample Point register at the end of cycle T2. This is the end of the fifth *half* cycle, relative to the *half* cycle from which data was launched from the DDR device. A T2 Sample will generally be used in conjunction with a Read Sample Stage 3 and requires that the timing budget support the use of Stage 3 if configured as such.

Depending of the read Data/DQS arrival, if needed, it is possible to configure a T2 Sample in conjunction with Read Sample Stage 2. Such a configuration requires that the timing budget supports/requires the use of Stage 2 and the appropriate setting SDRAM0_TR1[RDCD] must be determined.

A T3 Sample configures the DDR SDRAM controller to sample incoming read data into the PLB Read Sample Point register at the end of cycle T3. This is the end of the seventh *half* cycle, relative to the *half* cycle from which data was launched from the DDR device. It is expected that CAS Latency 2.5 applications would never use a T3 Sample. Depending of the read Data/DQS arrival, if needed, it is possible to configure a T3 Sample in conjunction with Read Sample Stage 3. Such a configuration requires that the timing budget support/require the use of Stage 3 and the appropriate setting SDRAM0_TR1[RDCD] must be determined.

CAS Latency 3 Devices

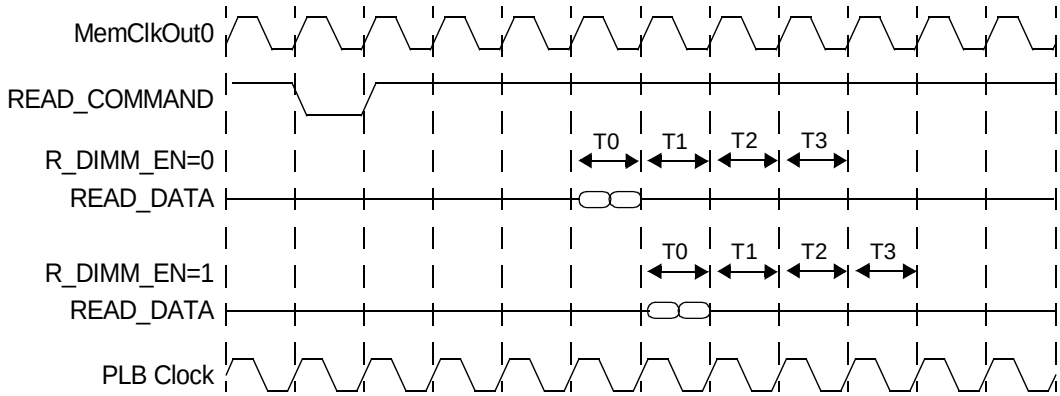


Figure 17-19. Read Sample Cycle: CL = 3 For CAS Latency 3 devices, the DDR device launches read data nominally coincident with the third rising edge of its Memory Clock following the cycle in which the read command was sampled by the device.

A T0 Sample configures the DDR SDRAM controller to sample incoming read data into the PLB Read Sample Point register at the end of cycle T0. This is the end of the first cycle in which data, both upper and lower, was launched from the DDR device. A T0 Sample should only be used in conjunction with Read Sample Stage 1 and requires that the timing budget support the use of Stage 1. A T0 Sample should never be used if SDRAM0_TR1[RDSL] is configured for Stages 2 or 3.

A T1 Sample configures the DDR SDRAM controller to sample incoming read data into the PLB Read Sample Point register at the end of cycle T1. This is the end of the second cycle, relative to the cycle in which data, both upper and lower, was launched from the DDR device. A T1 Sample will generally be used in conjunction with Read Sample Stage 2 and requires that the timing budget support the use of Stage 2.

A T2 Sample configures the DDR SDRAM controller to sample incoming read data into the PLB Read Sample Point register at the end of cycle T2. This is the end of the third cycle, relative to the cycle in which data, both upper and lower, was launched from the DDR device. A T2 Sample will generally be used in conjunction with Read Sample Stage 3 and requires that the timing budget support the use of Stage 3. It is expected that CAS Latency 3 applications would never need to use a T2 Sample or Stage 3.

A T3 Sample configures the DDR SDRAM controller to sample incoming read data into the PLB Read Sample Point register at the end of cycle T3. This is the end of the fourth cycle, relative to the cycle in which data, both upper and lower, was launched from the DDR device. It is expected that CAS Latency 3 applications would never use a T3 Sample and, as such, should not be configured to do so.

SDRAM0_TR1 Recommendations/Considerations

Within the logic, the SDRAM0_TR1[RDSS] configuration setting is independent of the SDRAM0_TR1[RDSL] setting. As such, the two can be configured independently and in such a way as to prevent successful capture of the DDR read data. Care must be taken to understand the memory subsystem clocking, timing budget, and delays before modifying SDRAM0_TR1.

In general, it is recommended that the SDRAM0_TR1[RDSS] setting equal the SDRAM0_TR1[RDSL] setting, unless the memory subsystem timing budget dictates otherwise. It is also recommended that the default SDRAM0_TR1 settings be used for CAS Latency 2 and 3 devices and the SDRAM0_TR1[RDCD] be set to 1 for CAS Latency 2.5 devices and SDRAM0_TR1[RDCT] be configured if appropriate or necessary.

The combination of Read Sample Stage 1 (SDRAM0_TR1[RDSL]=00) and a T0 Sample Cycle (SDRAM0_TR1[RDSS]=00) constitutes the "low latency" path and should only be used if the memory subsystem timing budget has sufficient margin to do so at the target operating frequency for the specific application. In general, the low latency path is more suited for non-ECC applications running at lower frequencies.

It is expected that ECC enabled applications would not use the low latency path. That said, it is up to the DDR SDRAM controller user to make the determination for the specific application and memory subsystem timing budget.

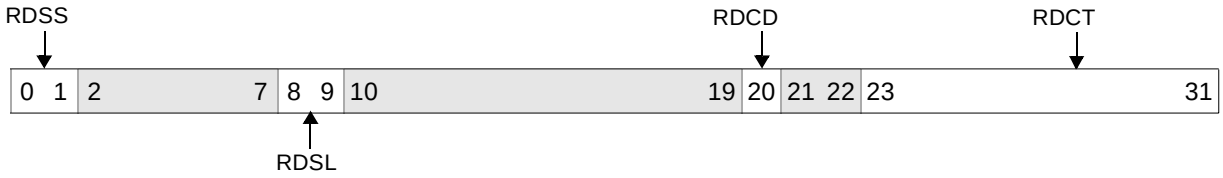


Figure 0-16. SDRAM Timing Register 1 (SDRAM0_TR1)

0:1	RDSS	Read Sample Cycle Select 00 T0 Sample 01 T1 Sample 10 T2 Sample 11 T3 Sample	**Note:** Set RDSS to T2 Sample for CAS latency 2.5 devices.
2:7		Reserved	
8:9	RDSL	Read Sample Stage Select 00 Stage 1 01 Stage 2 10 Stage 3 11 Reserved	**Note:** Set RDSL to Stage 3 for CAS latency 2.5 devices.
10:19		Reserved	
20	RDCD	Read Clock Delay - Stage 2 0 0 clock delay 1 1/2 clock delay	**Note:** Set RDCD to 1 for CAS latency 2.5 devices.
21:22		Reserved	

23:31	RDCT	Read Clock Delay Tuning Bits	Bit 23 is Most Significant Bit; bit 31 is Least Significant Bit. Bits 30:31, select increments of 1/4 delay element. Bits 23:29, select increments of 1 full delay element. This field should never be programmed to exceed a delay equivalent to 1/2 the Read Clock frequency.

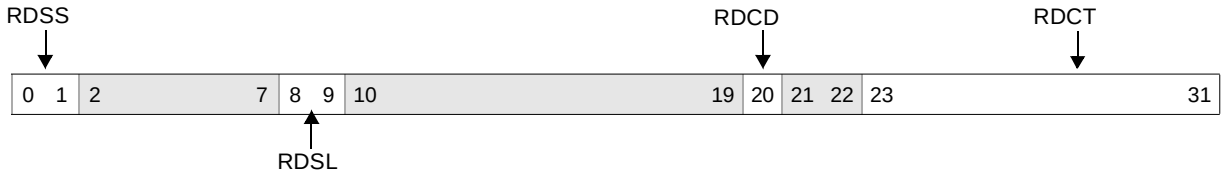


Figure 17-20. SDRAM Timing Register 1 (SDRAM0_TR1)

0:1	RDSS	Read Sample Cycle Select 00 T0 Sample 01 T1 Sample 10 T2 Sample 11 T3 Sample	**Note:** Set RDSS to T2 Sample for CAS latency 2.5 devices.
2:7		Reserved	
8:9	RDSL	Read Sample Stage Select 00 Stage 1 01 Stage 2 10 Stage 3 11 Reserved	**Note:** Set RDSL to Stage 3 for CAS latency 2.5 devices.
10:19		Reserved	
20	RDCD	Read Clock Delay - Stage 2 0 0 clock delay 1 1/2 clock delay	**Note:** Set RDCD to 1 for CAS latency 2.5 devices.
21:22		Reserved	
23:31	RDCT	Read Clock Delay Tuning Bits	Bit 23 is Most Significant Bit; bit 31 is Least Significant Bit. Bits 30:31, select increments of 1/4 delay element. Bits 23:29, select increments of 1 full delay element. This field should never be programmed to exceed a delay equivalent to 1/2 the Read Clock frequency.

17.2.4.16 DDR SDRAM Clock Timing Register (SDRAM0_CLKTR)

CLKTR Phase Advance and Delay

The DDR SDRAM controller provides the capability to change the timing and phase relationship of the controller-generated DDR Memory clock. This feature allows the user to effectively adjust the DDR memory interface timings, should it be necessary.

Note: Any modification to the SDRAM0_CLKTR register affects the MemClkOut0 signal. As such, this register should only be modified before initialization, and system software must guarantee adequate time for the "ripple" effects (such as allowing down stream PLLs to re-lock) to stabilize. It is the responsibility of the DDR SDRAM controller user to guarantee that the effect(s) of such modifications on the memory interface timing and timing budget are valid and do not create timing relationships that violate the DDR device specifications, the memory subsystem specific timing budget, or conflict with the overall DDR interface clocking methodology selected.

Note: A complete and thorough understanding of DDR Memory interface protocol and timing, as well as the target memory subsystem timing budget, is an absolute must when considering use of the capabilities provided by the SDRAM0_CLKTR.

The delay effect of the DDR Clock Delay Tuning Bits (SDRAM0_CLKTR[23:31]) is dependent on may factors, including delay line physical placement, ASIC process variation, and voltage/temperature variations. The Delay Line Calibration Register (SDRAM0_DLYCAL) may be used to calibrate the tuning bits.

Figure 17-21 depicts the configurable phase relationships available with SDRAM0_CLKTR. Note that all depicted relationships are with respect to the indicated DDR SDRAM outputs and are independent of any effects of inline path elements such as on-chip or off-chip PLLs, IO driver delays, board wiring, etc. As such, the effects of any inline path elements must be considered when determining the timing relationship of the resultant signals at the DDR memory devices.

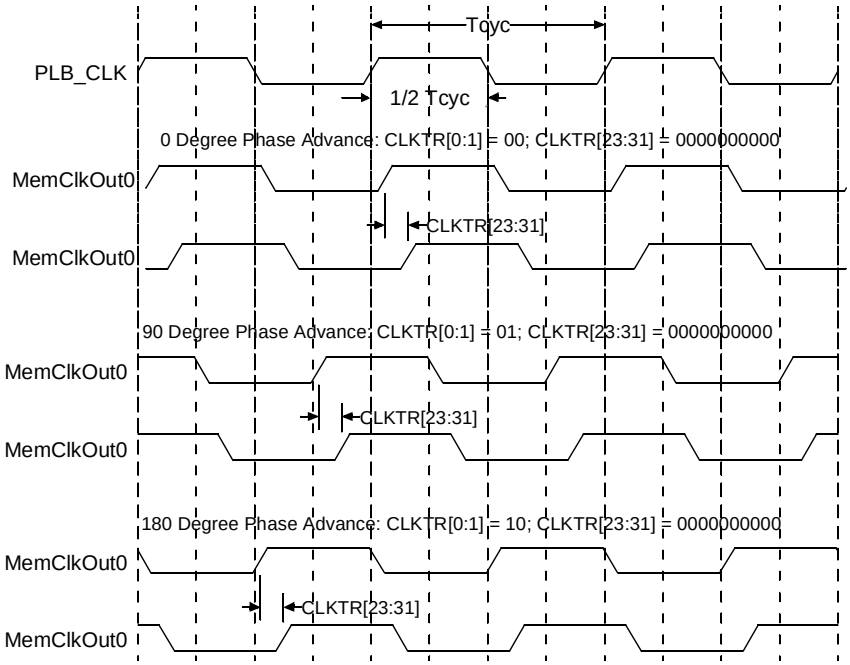


Figure 17-21. CLKTR Phase Advance and Delay

The MemClkOut0 signal may be advanced in two coarse increments of 90 and 180 degrees as configured via SDRAM0_CLKTR[0:1]. The indicated phase advance is effectively with respect to the PLB clock signal, which is used to clock the DDR SDRAM controller memory command interface and controller internal registers including the PLB read and write data path interfaces.

With SDRAM0_CLKTR[0:1] = 00, the MemClkOut0 signal is generated in phase, and slightly delayed, with respect to the PLB clock signal. The DDR Clock Delay Tuning bits (SDRAM0_CLKTR[23:31]) provide the ability to add delay, in fine increments, to further adjust the MemClkOut0 signal.

With SDRAM0_CLKTR[0:1] = 01, the MemClkOut0 signal is generated phase advanced by 90 degrees (1/4 Tcyc), and slightly delayed, with respect to the PLB clock signal. The DDR Clock Delay Tuning bits (SDRAM0_CLKTR[23:31]) provide the ability to add delay, in fine increments, to further adjust the MemClkOut0 signal.

With SDRAM0_CLKTR[0:1] = 10, the MemClkOut0 signal is generated phase advanced by 180 degrees (1/2 Tcyc), and slightly delayed, with respect to the PLB clock signal. The DDR Clock Delay Tuning bits (SDRAM0_CLKTR[23:31]) provide the ability to add delay, in fine increments, to further adjust the MemClkOut0 signal.

SDRAM0_CLKTR Phase Advance and Delay Considerations

As indicated previously, a complete and thorough understanding of the DDR device interface protocol and timing parameters is recommended. The following information is provided to highlight some of the issues and effects to be evaluated when considering adjusting the DDR Memory clock.

1.) Command Interface setup and hold time. Advancing (or not advancing) the DDR memory clock will affect the command interface setup and hold times guaranteed at the device inputs.

2.) DDR device Write DQSS minimum/maximum window. Advancing (or not advancing) the DDR memory clock will affect the timing relationship of write DM, DQS, and DATA with respect to the DDR memory clock. The SDRAM0_WDDCTR register provides compatible configuration options to allow this relationship to be adjusted to compensate for SDRAM0_CLKTR modifications. See ~~Section 17.2.4.17, "Write Data/DM/DQS Clock Timing Register (SDRAM0_WDDCTR)," on page 17-40~~*Write Data/DM/DQS Clock Timing Register (SDRAM0_WDDCTR)* on page 509 for more details.

3.) DDR device Read Access time. DDR Read Data and DQS are driven nominally phase aligned with the rising edge of the DDR Memory clock. Advancing the DDR memory clock, with respect to the internal DDR SDRAM controller read data sample clock, advances the read data access time. This is only true if the DDR SDRAM controller read data sample clock is derived from within the ASIC clock generation logic. In general, this is not true if the read data sample clock is derived from an external feedback read data clock derived from the external memory clock.

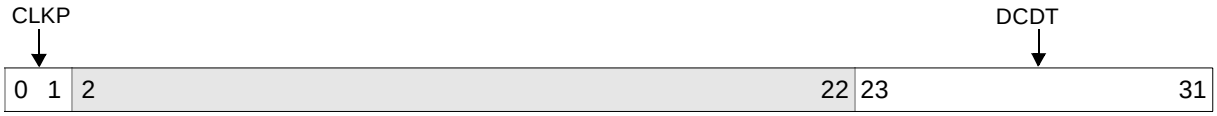


Figure 0-17. DDR_SDRAM Clock Timing Register (SDRAM0_CLKTR)

0:1	CLKP	Write Clock Phase 00 Advance 0 degrees 01 Advance 90 degrees 10 Advance 180 degrees 11 Reserved	In most applications, CLKP should be set to 90 degrees phase advance.
2:22		Reserved	
23:31	DCDT	DDR Clock Delay Tuning Bit 23 is Most Significant Bit; bit 31 is Least Significant Bit. Bits 23:29 select increments of 1 full delay element. Bits 30:31 select increments of 1/4 delay element.	This field should never be programmed to exceed a delay equivalent to 1/4 the MemClkOut0 frequency.

4.) For the majority of DDR SDRAM systems, SDRAMx0_CLKTR[0:1] should be set to 90 degrees phase advance.

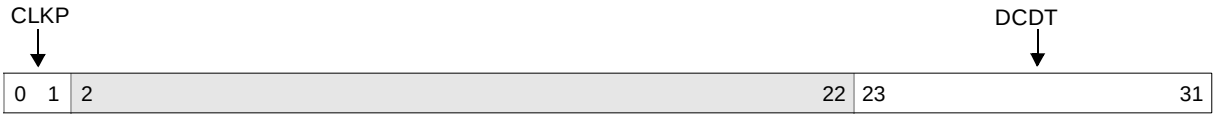


Figure 17-22. DDR_SDRAM Clock Timing Register (SDRAM0_CLKTR)

0:1	CLKP	Write Clock Phase 00 Advance 0 degrees 01 Advance 90 degrees 10 Advance 180 degrees 11 Reserved	In most applications, CLKP should be set to 90 degrees phase advance.
2:22		Reserved	
23:31	DCDT	DDR Clock Delay Tuning Bit 23 is Most Significant Bit; bit 31 is Least Significant Bit. Bits 23:29 select increments of 1 full delay element. Bits 30:31 select increments of 1/4 delay element.	This field should never be programmed to exceed a delay equivalent to 1/4 the MemClkOut0 frequency.

17.2.4.17 Write Data/DM/DQS Clock Timing Register (SDRAM0_WDDCTR)
WDDCTR Phase Advance and Delay

The DDR SDRAM controller provides the capability to change the timing and phase relationship of the controller generated DDR Memory write **data**, write data **mask**, and write data **strobe** signals. The afore mentioned group of signals will be collectively referred to as the "write data and control group" in the following paragraphs. This feature allows the user to effectively adjust the DDR memory interface timings for T_{dqss}, should it be necessary. The DDR device T_{dqss} window is defined with respect to the DDR Memory clock at the device. As such, the SDRAM0_WDDCTR register, in conjunction with the SDRAM0_CLKTR register, can be used to configure the relationship of the "write data and control group outputs" with respect to the MemClkOut0 signal.

It is the responsibility of the chip user to guarantee that the effect(s) of such modifications on the memory interface timing and timing budget are valid and do not create timing relationships that violate the DDR device specifications, the memory subsystem specific timing budget, or conflict with the overall DDR interface clocking methodology selected.

A complete and thorough understanding of DDR Memory interface protocol and timing, as well as the target memory subsystem timing budget is an absolute must when considering use of the capabilities provided by SDRAM0_WDDCTR.

WDDCTR/CLKTR Relationships

The following figures depict the range of timing relationships for the "write data and control group" outputs that may be configured through SDRAM0_WDDCTR. The relationships are depicted with respect to the MemClkOut0 signal and the corresponding setting for SDRAM0_CLKTR. The effects of a non-zero value setting for SDRAM0_CLKTR[23:31] or SDRAM0_WDDCTR[23:31] are not explicitly depicted; however, the effects of such settings must be considered when configuring the relative relationships depicted. Also, all depicted relationships are with respect to the indicated chip outputs and are independent of any effects of

inline path elements such as on-chip or off-chip PLLs, IO driver delays, board wiring, etc. As such, the effects of any inline path elements must be considered when determining the timing relationship of the resultant signals at the DDR memory devices.

The "write data and control group" signals may be advanced in two coarse increments of 90 and 180 degrees as configured via SDRAM0_WDDCTR[0:1]. The indicated phase advance is effective with respect to the PLB clock, which is used to clock the DDR SDRAM controller's memory command interface and controller internal registers, including the PLB read and write data path interfaces.

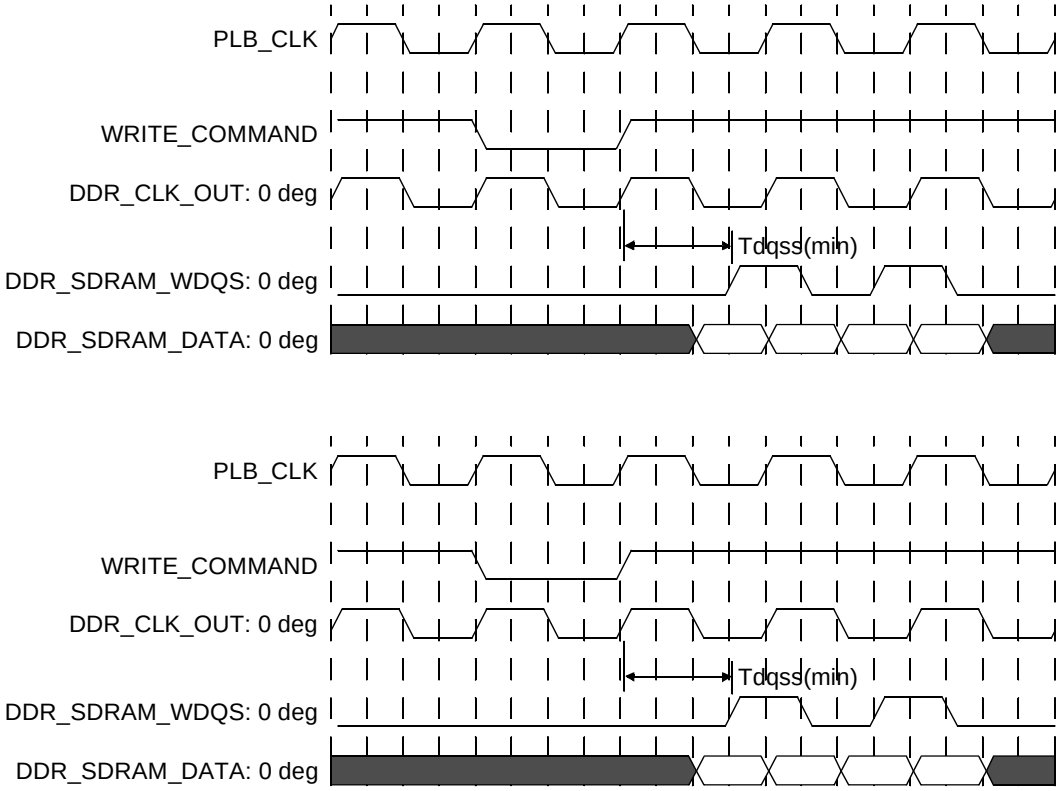


Figure 17-23. CLKTR 0 Degree Phase Advance and WDDCTR 0 Degree Phase Advance Figure 17-23 depicts the default configuration setting with no phase advance on SDRAM0_CLKTR or SDRAM0_WDDCTR. This effectively configures the DDR SDRAM controller to launch the "write data and control group" coincident with the $T_{dqss(min)}$ timing reference defined by the DDR device specification.

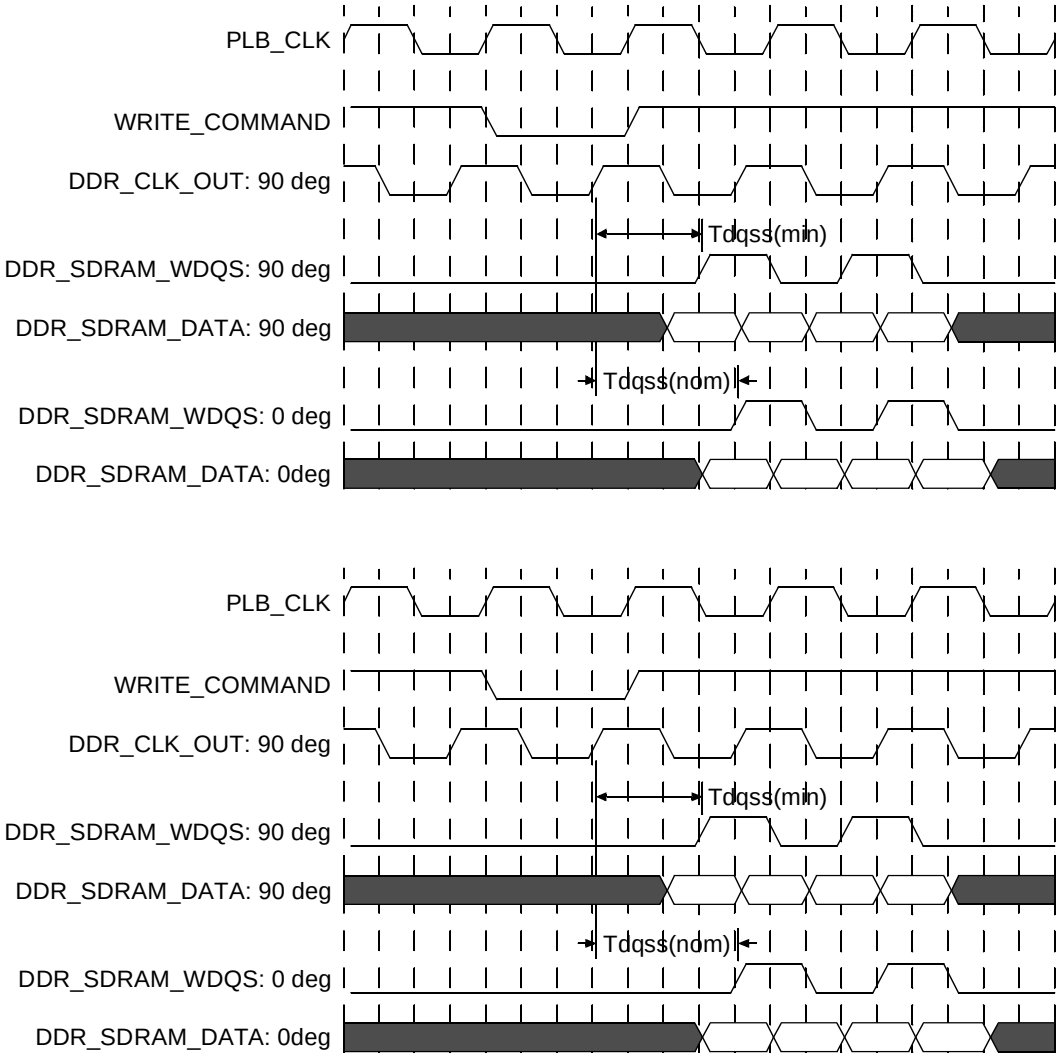


Figure 17-24. CLKTR 90 Degree Phase Advance and WDDCTR 90/0 Degree Phase Advance Figure 17-24 depicts a 90 degree phase advance on SDRAM0_CLKTR. With SDRAM0_WDDCTR configured for a 90 degree phase advance the "write data and control group" are launched coincident with the $T_{dqss(min)}$ timing reference defined by the DDR device specification. With SDRAM0_WDDCTR configured for a 0 degree phase advance the "write data and control group" are launched coincident with the $T_{dqss(nom)}$ timing reference defined by the DDR device specification.

Note the effect (setup/hold) on the DDR command interface (WRITE_COMMAND) of the SDRAM0_CLKTR 90 degree phase advance.

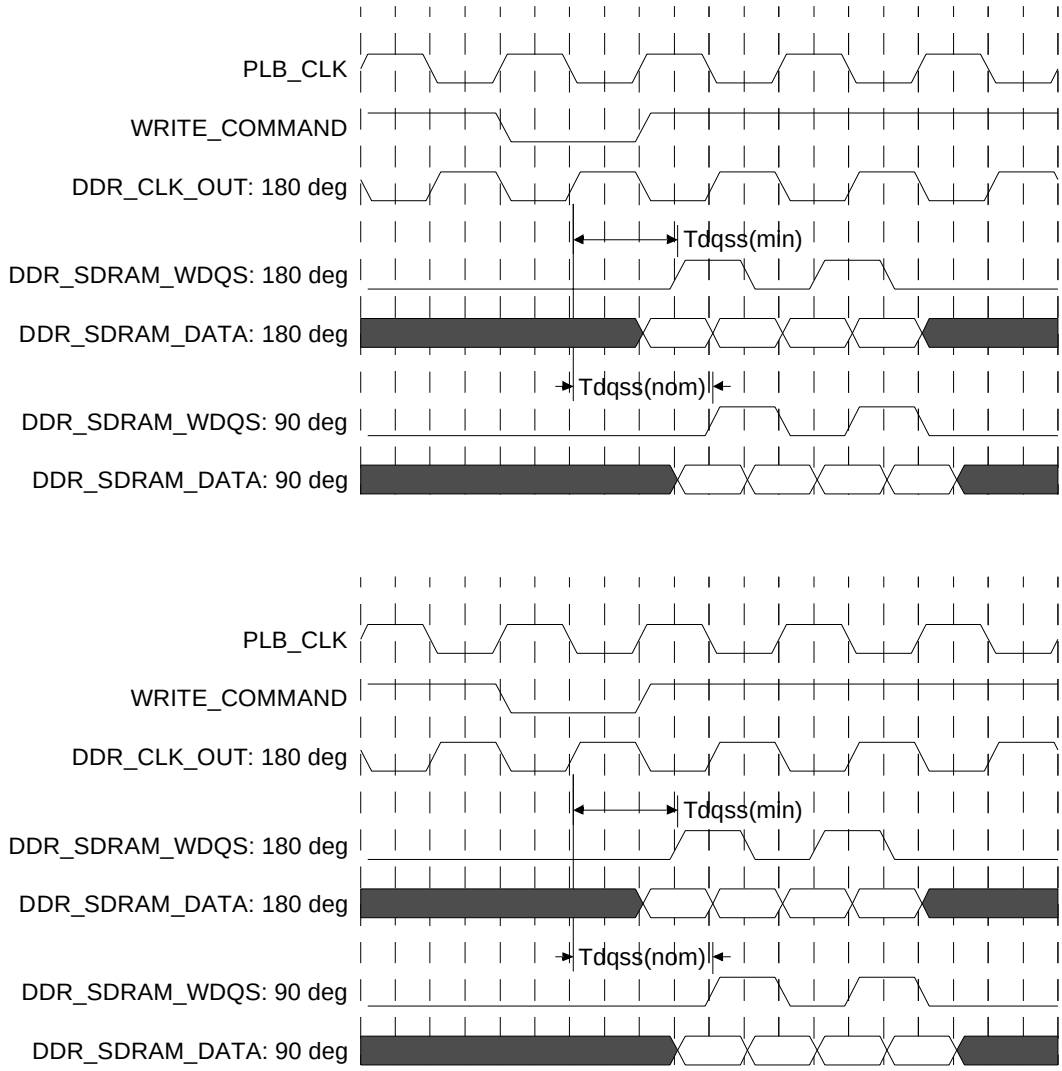


Figure 17-25. CLKTR 180 Degree Phase Advance and WDDCTR 180/90 Degree Phase Advance Figure 17-25 depicts a 180 degree phase advance on SDRAM0_CLKTR. With SDRAM0_WDDCTR configured for a 180 degree phase advance the "write data and control group" are launched coincident with the $T_{dqss(min)}$ timing reference defined by the DDR device specification. With WDDCTR configured for a 90 degree phase advance the "write data and control group" are launched coincident with the $T_{dqss(nom)}$ timing reference defined by the DDR device specification.

Note the effect (setup/hold) on the DDR command interface of the SDRAM0_CLKTR 180 degree phase advance.

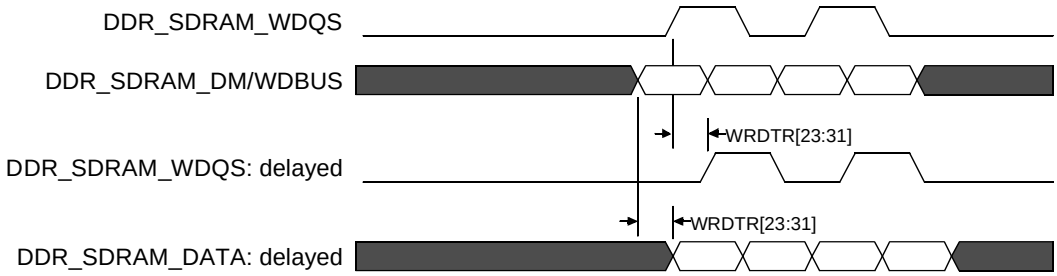


Figure 17-26. SDRAM0_WDDCTR Delay As illustrated in Figure 17-26, the DDR Write Data/DM/DQS Clock Delay Tuning Bits (SDRAM0_WDDCTR[23:31]) provide the ability to further adjust the "write data and control group" by adding delay in fine increments.The delay effect of the DDR Write Data/DM/DQS Clock Delay Tuning Bits (SDRAM0_WDDCTR[23:31]) is dependent on may factors including delay line physical placement, ASIC process variation, and voltage/temperature variations. The Delay Line Calibration Register (SDRAM0_DLYCAL) may be used to calibrate the tuning bits.

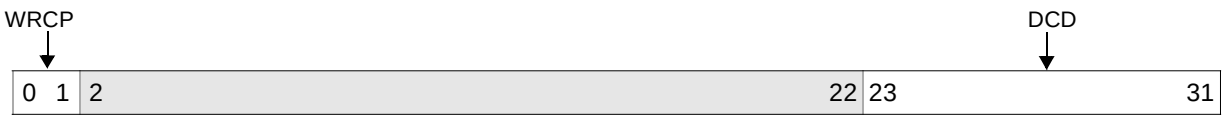


Figure 0-18. Write Data/DM/DQS Clock Timing Register (SDRAM0_WDDCTR)

0:1	WRCP	Write Clock Phase 00 Advance 0 degrees 01 Advance 90 degrees 10 Advance 180 degrees 11 Reserved
2:22		Reserved
23:31	DCD	DDR Write Data/DM/DQS Clock Delay Tuning Bits Bit 23 is Most Significant Bit; bit 31 is Least Significant Bit. Bits 23:29 select increments of 1 full delay element. Bits 30:31 select increments of 1/4 delay element.

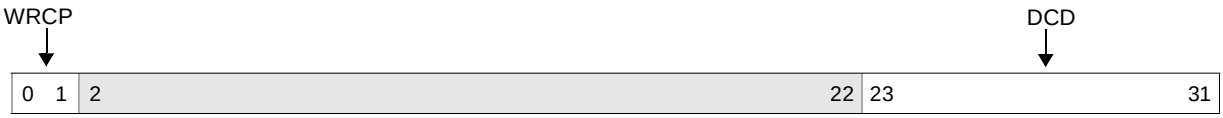


Figure 17-27. Write Data/DM/DQS Clock Timing Register (SDRAM0_WDDCTR)

0:1	WRCP	Write Clock Phase 00 Advance 0 degrees 01 Advance 90 degrees 10 Advance 180 degrees 11 Reserved
2:22		Reserved
23:31	DCD	DDR Write Data/DM/DQS Clock Delay Tuning Bits Bit 23 is Most Significant Bit; bit 31 is Least Significant Bit. Bits 23:29 select increments of 1 full delay element. Bits 30:31 select increments of 1/4 delay element.

17.2.4.18 Delay Line Calibration Register (SDRAM0_DLYCAL)

This register provides software with a hardware-based method of calibrating the various programmable delay lines available for use in the DDR SDRAM controller.

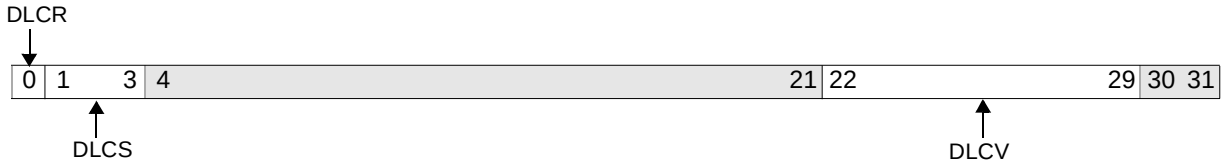


Figure 0-19. Delay Line Calibration Register (SDRAM0_DLYCAL)

0	DLCR	Delay Line Calibration Request 0 Delay Line Calibration not requested 1 Delay Line Calibration requested	Setting DLCR bit initiates the delay line calibration. During the re-initialization process, the DLCR bit is cleared. The delay line calibration occurs automatically following SysReset or upon request by setting the DLCR bit. The results of the calibration are provided to assist in the setting of SDRAM0_CLKTR, SDRAM0_WRDTR, and SDRAM0_SDTR2.
1:3	DLCS	Delay Line Calibration Status 000 Delay Line Calibration not run 001 Delay Line Calibration in progress 010 Delay Line Calibration complete 100 Delay Line Calibration error	This two bit field indicates the real-time status of the calibration process at the time of the register read. Once complete (DLCS = 010), the DLY_VAL field is valid. A delay line calibration status of "error" indicates that the calibration process completed without successfully determining the value of DLCV. Sufficient delay stages have been included in the calibration delay line such that this completion status should never occur within the supported operating frequency range.
4:21		Reserved	
22:29	DLCV	Delay Line Calibration Value	This 8-bit binary encoded value indicates the number of delay elements in a single 2x clock cycle or a single 1x half clock cycle. Bit 22 is Most Significant Bit; bit 29 is Least Significant Bit.
30:31		Reserved	

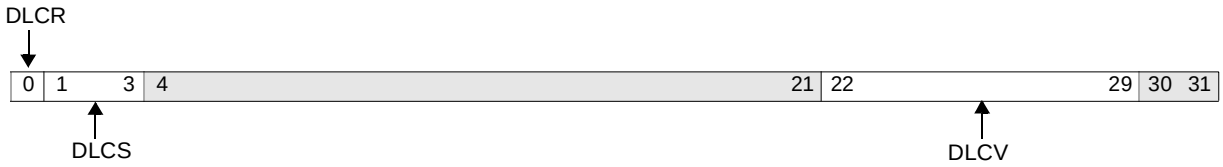


Figure 17-28. Delay Line Calibration Register (SDRAM0_DLYCAL)

0	DLCR	Delay Line Calibration Request 0 Delay Line Calibration not requested 1 Delay Line Calibration requested	Setting DLCR bit initiates the delay line calibration. During the re-initialization process, the DLCR bit is cleared. The delay line calibration occurs automatically following SysReset or upon request by setting the DLCR bit. The results of the calibration are provided to assist in the setting of SDRAM0_CLKTR, SDRAM0_WRDTR, and SDRAM0_SDTR2.
1:3	DLCS	Delay Line Calibration Status 000 Delay Line Calibration not run 001 Delay Line Calibration in progress 010 Delay Line Calibration complete 100 Delay Line Calibration error	This two bit field indicates the real-time status of the calibration process at the time of the register read. Once complete (DLCS = 010), the DLY_VAL field is valid. A delay line calibration status of "error" indicates that the calibration process completed without successfully determining the value of DLCV. Sufficient delay stages have been included in the calibration delay line such that this completion status should never occur within the supported operating frequency range.
4:21		Reserved	
22:29	DLCV	Delay Line Calibration Value	This 8-bit binary encoded value indicates the number of delay elements in a single 2x clock cycle or a single 1x half clock cycle. Bit 22 is Most Significant Bit; bit 29 is Least Significant Bit.
30:31		Reserved	

17.2.4.19 ECC Error Status Register (SDRAM0_ECCESR)

The ECC Error Status Register (SDRAM0_ECCESR) tracks ECC related errors encountered during SDRAM memory PLB master accesses to system memory. Bits in SDRAM0_ECCESR are cleared by writing a 32-bit value to SDRAM0_ECCESR with a "1" in any bit position that is to be cleared and "0" in all other bit positions.

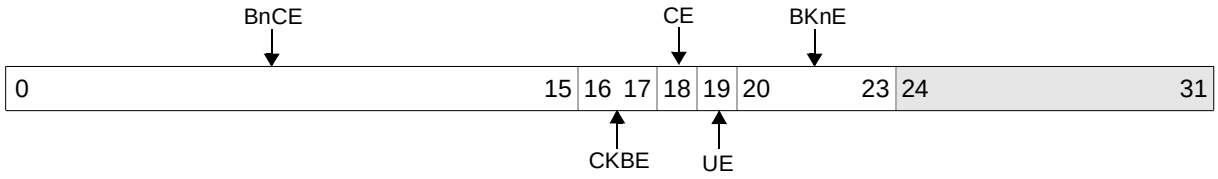


Figure 0-20. ECC Error Status Register (SDRAM0_ECCESR)

0:15	BnCE	Byte Lane n Corrected Error (where n=0-15) 0 No Error 1 Error Occurred in Byte Lane n
16:17	CKBE	Error Detected in Checkbits 64-bit Mode 00 No Error 01 Reserved 10 Error in Checkbits 11 Reserved 32-bit Mode 00 No Error 01 Error in Lower Checkbits 10 Error in Upper Checkbits 11 Error in Upper and Lower Checkbits
18	CE	Correctable Error
19	UE	Uncorrectable Error
20:23	BKnE	Bank n Error (where n=0-3) 0 No Error 1 Error Occurred in Bank n
24:31		Reserved

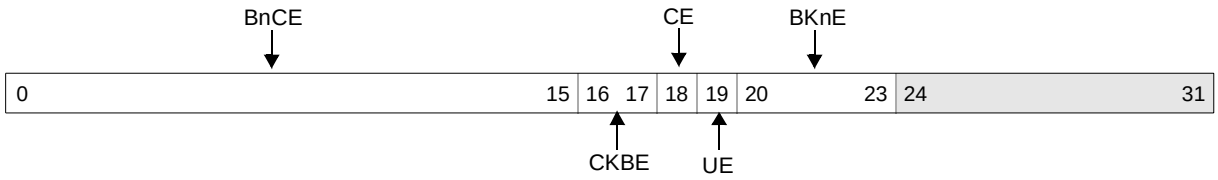


Figure 17-29. ECC Error Status Register (SDRAM0_ECCESR)

0:15	BnCE	Byte Lane n Corrected Error (where n=0-15) 0 No Error 1 Error Occurred in Byte Lane n

16:17	CKBE	Error Detected in Checkbits 64-bit Mode 00 No Error 01 Reserved 10 Error in Checkbits 11 Reserved 32-bit Mode 00 No Error 01 Error in Lower Checkbits 10 Error in Upper Checkbits 11 Error in Upper and Lower Checkbits
18	CE	Correctable Error
19	UE	Uncorrectable Error
20:23	BKnE	Bank n Error (where n=0-3) 0 No Error 1 Error Occurred in Bank n
24:31		Reserved

17.2.4.20 Controller ID Register (SDRAM0_CID)

The Controller ID Register is a "Read Only" register implemented to identify the DDR SDRAM controller and its history.

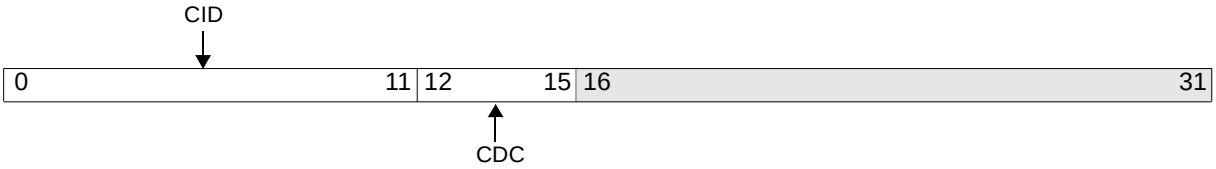


Figure 0-21. Controller ID Register (SDRAM0_CID)

0:11	CID	Core ID identifies the IBM specific core number associated with the DDR SDRAM.
12:15	CDC	Core Derivative Code B Base Library Core D Derivative of Base Library Core
16:31		Reserved

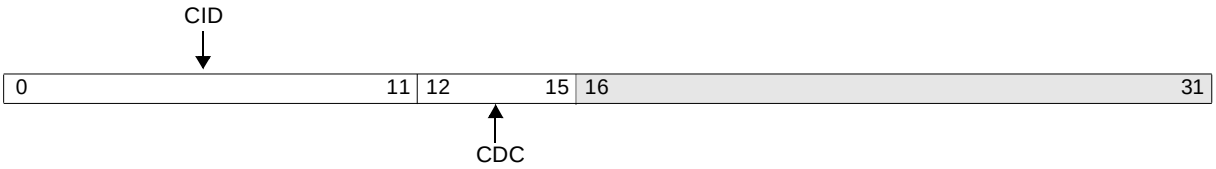


Figure 17-30. Controller ID Register (SDRAM0_CID)

0:11	CID	Core ID identifies the IBM specific core number associated with the DDR SDRAM.
12:15	CDC	Core Derivative Code B Base Library Core D Derivative of Base Library Core
16:31		Reserved

17.2.4.21 Revision ID Register (SDRAM0_RID)

The Revision ID Register is a "Read Only" register implemented to identify the DDR SDRAM controller revision level and revision history.

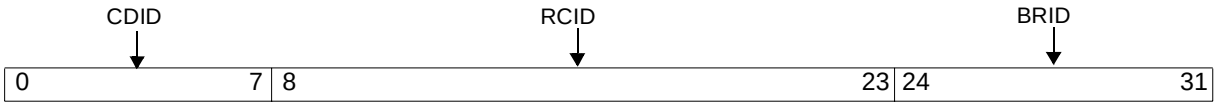


Figure 0-22. Revision ID Register (SDRAM0_RID)

0:7	CDID	Controller Derivative ID nn = controller derivative ID number nn = 00 indicates the Base Library Controller (CDC = B) nn = A non-zero values indicates the specific controller derivative ID number and is tracted by development (CDC = D).
8:23	RCID	Controller Revision Control ID cccc indicates the controller SCCS revision control ID associated with the specific version of the controller.
24:31	BRID	Controller Branch Revision Control ID bb indicates the core SCCS revision control branch ID associated with the specific version of the controller. This field is used to associate a specific Netlist with the corresponding RTL branch in the event of a modification at the Netlist level. bb = 00 indicates the an unmodified (post synthesis) netlist bb = non-zero value indicates the branch ID for the RTL and the corresponding netlist modification.

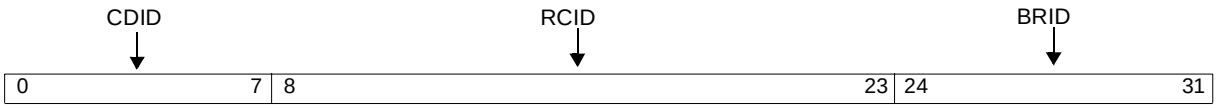


Figure 17-31. Revision ID Register (SDRAM0_RID)

0:7	CDID	Controller Derivative ID nn = controller derivative ID number nn = 00 indicates the Base Library Controller (CDC = B) nn = A non-zero values indicates the specific controller derivative ID number and is tracted by development (CDC = D).

8:23	RCID	Controller Revision Control ID cccc indicates the controller SCCS revision control ID associated with the specific version of the controller.
24:31	BRID	Controller Branch Revision Control ID bb indicates the core SCCS revision control branch ID associated with the specific version of the controller. This field is used to associate a specific Netlist with the corresponding RTL branch in the event of a modification at the Netlist level. bb = 00 indicates the an unmodified (post synthesis) netlist bb = non-zero value indicates the branch ID for the RTL and the corresponding netlist modification.

The Controller ID Register and the Revision ID Register, in combination, provide important tracking information about the specific DDR SDRAM version implemented.

17.3 Initialization

The following two steps are needed to begin proper initialization.

1. After reset is deactivated, pause the amount of time indicated in the DDR SDRAM memory device specification. No requests to memory controller can occur during the pause period. Configure DDR SDRAM configuration registers (can occur during pause period). If modifying SDRAM0_CLKTR, sufficient time must be allowed for any internal and/or external in-line PLLs to stabilize and lock before proceeding to Step 2.

Note: Step 2 can only occur after SDRAM configuration is completed in step 1.

2. Set SDRAM0_CFG0[DCEN] = 1 to enable the SDRAM controller.

The following steps are then performed by the DDR SDRAM controller to complete initialization:

1. Asserts CKE High.
2. Issues precharge command to all banks.
3. Waits minimum number of clock cycles as defined by SD_PTA.
4. Issues Extend Mode Register Set command (EMRS) to Enable DDR SDRAM DLL.
5. Issues Mode Register Set command (MRS1) to load mode register and to reset DDR SDRAM DLL.
6. Issues precharge command to all banks.
7. Performs two auto refresh cycles (each separated by SD_RFTA clock cycles).
8. Issues Mode Register Set (MRS2) command to enable normal operation.
9. Waits for a minimum count of 200 cycles from step 4 to expire.
10. Makes DDR SDRAM available for access.

17.4 DDR to Memory Address Decode

The DDR SDRAM supports the extended addressing capability of the PLB and can be located within any 4-GB aligned region of the 64-GB address space available given the 36-bit PLB address. The range of addresses to which the DDR SDRAM responds within the programmed 4-GB aligned region can be further limited to the specific range of addresses for which physical memory is installed.

Note: Moving the memory map may cause overlap with other mapped cores.

The specific range of addresses to which the PLB slave will respond is defined and configured by the combination of the PLB UABus Base Address Register (SDRAM0_UUABA) and the Memory Configuration Registers (SDRAM0_BnCR). For example, the base address for memory bank 0 is defined by SDRAM0_UUABA[28:31] and SDRAM0_B0CR[0:7]; likewise, the base address for memory bank 1 is defined by SDRAM0_UUABA[28:31] and SDRAM0_B0CR[0:7]. The PLB slave interface will respond to any PLB access that targets a defined base address and lies within the size of the physical memory region defined for the enabled memory bank provided that the access type is supported.

17.5 DDR Slave Interface Options

The DDR SDRAM PLB slave interface has several configurable options to allow users to change the way in which the DDR SDRAM responds to read and write requests for various conditions such as buffer full, address queue full, and read/write priority. These options are explained in the following sections and configured via the PLB Slave Interface Options (SDRAM0_SLIO) register.

17.5.1 Read Rearbitration

The DDR SDRAM will immediately accept any pending read request provided that an address queue entry is available. If the address queue is full, the default behavior of the DDR SDRAM is to WAIT the request until an entry in the address queue is available. The WAIT signal conveys to the PLB arbiter that the DDR SDRAM is unable to accept the current read request and that the PLB arbiter should maintain the current request assertion until the DDR SDRAM is able to accept the read request. This default behavior can be changed by setting SDRAM0_SLIO[0] (RDRE) to enable PLB slave read rearbitration. When RD_RE is set, any read request received when the address queue is full (PLB slave not able to accept immediately) will trigger a REARBITRATE signal and allow the PLB arbiter to re-arbitrate the PLB.

17.5.2 Write Rearbitration

The DDR SDRAM will immediately accept any pending write request provided that an address queue entry is available. If the address queue is full, the default behavior of the DDR SDRAM is to WAIT the request until an entry in the address queue is available. The WAIT signal conveys to the PLB arbiter that the DDR SDRAM is not able to accept the current write request and that the PLB arbiter should maintain the current request assertion until the DDR SDRAM is able to accept the write request. This default behavior can be changed by setting SDRAM0_SLIO[1] (WRRE) to enable PLB slave write rearbitration. When WR_RE is set, any write request received when the address queue is full or if there is less than 32 bytes of free space in the write buffer will trigger a REARBITRATE signal and allow the PLB arbiter to re-arbitrate the PLB Bus.

17.5.3 Read Around Write

The default behavior of the DDR SDRAM is to complete all read and write requests in the order in which each request was accepted.

Setting SDRAM0_SLIO[3] (RARW) enables the "read around write" feature which enables queued, non-burst read requests to bypass previously posted writes as long as the target address range of the read does not conflict with the target address range of the posted writes that precede the read request. In this mode, all writes will be completed in order with respect to all other writes and all reads will be completed in order with respect to all other reads. Also, to insure that the write pipe does not stall when in this mode, the read around write operations will be interleaved with the previously posted write cycles. For example, if there are 3 write requests queued followed by 3 non-burst, non-conflicting reads, the sequence to the memory would be write-read-write-read-write-read. Burst read requests are not allowed to bypass or read around any previously posted write cycle.

The minimum granularity of the address check for read around write candidates is on a 32-byte aligned boundary. When queued, ending address of each write is calculated based on the size of the transfer. Subsequent read request addresses are compared to the write starting address (bits [0:26]) and the write ending address (bits [0:26]). Non-conflicting (based on address compare), queued reads are marked for bypass and do so at the earliest available opportunity.

17.6 Basic DDR SDRAM Memory Requirements

The DDR SDRAM controller supports 100 and 133 MHz, quad-bank DDR SDRAMs in a x32 or x64 implementation (x40/x72 supported if ECC) with one DQS per byte and uses fully programmable timing parameters. The DDR SDRAM controller supports the *Double Data Rate (DDR)SDRAM Specification*, JESD79, JEDEC Standard.

17.7 Page Management

The DDR SDRAM paging policy is configurable in that the Page Management Unit (PMU) may be enabled or disabled and the page deallocation policy set to least recently used or least recently allocated.

17.7.1 PMU Enabled

The DDR SDRAM controller supports page mode operation with bank interleaving and maintains up to eight open pages in the memory subsystem. Subsequent accesses that target an open page result in a page hit.

The DDR SDRAM controller page management unit (PMU) tracks memory accesses (activate, read/write, precharge, and refresh) and maintains a directory of up to eight open pages. All PMU entries are allocated and deallocated based on current and pending accesses. Allocated entries are checked against the address of the pending access, and a page hit is signaled to the DDR SDRAM control logic when a match occurs. All PMU directory entries are deallocated when the DDR SDRAM control logic issues a precharge all command to the DDR SDRAM in preparation for an auto refresh cycle.

The PMU supports activating and accessing a ninth page for a non-conflicting address in the memory subsystem before closing one of the eight open pages. A non-conflicting address is any address whose chip select and bank address are not currently allocated in the PMU. Access to this ninth page (activate and read

or write command) occurs followed by a precharge command to one of the other eight open pages. If access to the ninth page conflicts with an open page, the DDR SDRAM controller issues a precharge command to the bank before performing the access.

In the event that all 8 PMU entries are allocated and an access occurs that does not target one of the open allocated pages in memory (page miss), the PMU entry must be deallocated and the PMU directory updated with the page tag information for the new access. The PMU entry deallocated depends on the configured page deallocation policy.

If configured for Pseudo Least Recently Allocated (PMU_DP = 0), PMU entries are allocated in a circular queue with each new page accessed being allocated in the next queue entry. If an access conflicts with an allocated entry, the PMU directory entry of the conflicting page closed by the corresponding precharge command is updated with the new page address. Subsequent accesses (page hits) to previously allocated entries do not affect the position of an allocated entry in the queue. Once all PMU entries are allocated (8 open pages), an access to a non-conflicting address will trigger a deallocation and replacement of the next PMU entry indexed by the queue pointer.

If configured for Least Recently Used (or Least Recently Allocated) (PMU_DP = 1), the PMU directory is maintained as a stack that is updated with each access such that the page most recently accessed is on top of the stack (entry 1) and the page least recently accessed is the highest numbered allocated entry in the stack (entry 8 in the case of a fully allocated PMU directory). Each new page allocated is pushed onto the stack at entry 1. Accesses that target a conflicting address result in that entry being deallocated and the corresponding page in memory precharged, with the new page address being pushed onto the stack. For example, if an access occurs that conflicts with PMU directory entry 5, the page associated with entry 5 is deallocated, entries 1 through 4 are shifted to locations 2 through 5 respectively, and the new page is allocated in entry 1 (entries 6 through 8 are unaffected), maintaining the LRU ordering. Accesses that page hit, result in the corresponding PMU entry being moved to the top of the stack and all prior entries being pushed to that point into the stack, again maintaining the LRU ordering. Once all PMU entries are allocated (8 open pages), an access to a non-conflicting address will trigger a deallocation of entry 8 (least recently accessed page), with entries 1 through 7 being shifted to entries 2 through 8, respectively, and the new page being allocated at the top of the stack (entry 1).

Open pages can be spread across the system memory array on multiple chip selects or be contained in a single chip select, depending on the memory access sequences and the memory subsystem implementation. For a single bank (single chip select) memory subsystem, the number of open pages is limited to 4, which is the number of internal banks associated with the SDRAM device in that bank. In this case, the maximum of four open pages is a limitation of the memory subsystem implementation, not the DDR SDRAM controller.

The SDRAM page size for page hits varies, depending on programmed addressing mode which is derived from the DDR SDRAM organization. The following tables details the relationship of the address mode to the page size:

Table 17-8. 32-Bit SDRAM Page Size

Address Mode	Page Size
1	1 KB
2	2 KB
3	4 KB
4	8 KB

Table 17-9. 64-Bit SDRAM Page Size

Address Mode	Page Size
1	2 KB
2	4 KB
3	8 KB
4	16 KB

17.7.2 PMU Disabled

The DDR SDRAM controller also supports disabling the PMU for applications with random memory access sequences or for any application that will not benefit from the multiple open pages maintained when the PMU is enabled. With the PMU enabled, random and non-page local accesses may alter the PMU directory, yielding the worst case latency associated with a page miss with conflict. This necessitates a precharge-activate-read/write sequence. This additional precharge overhead can be eliminated by disabling the PMU.

With the PMU disabled, no pages are kept open in the memory subsystem. Each individual PLB request will result in the DDR SDRAM controller performing an activate command, followed by N read/write commands (where N=1 for non-Burst PLB accesses and N>=1 for Burst PLB accesses), where the final (or only) access is a read/write with auto-precharge command.

Note: Allowing page hits for the duration of the sequential accesses of a PLB Burst enables this mode to sustain the bandwidth without incurring the activate penalty for each individual access of the burst while satisfying a singular PLB burst read/write request.

In this mode, the memory latency for any PLB access will be the "Page Idle" latency as no pages are maintained open for subsequent accesses; thus, there can never be a page hit, nor can there be a page miss with conflict for any PLB access.

17.7.3 Physical Address to Memory Address Mapping

The DDR SDRAM controller supports various quad-bank DDR SDRAM device densities including 64,128, 256, and 512 Mbit in x8, x16, and x32 organizations, and it uses one DQS signal per byte. The addressing configuration associated with the specific device(s) organization and density determines the corresponding addressing mode(s), independent of the packaging of those devices (DIMM, SODIMM, or Planar).

Table 17-10 summarizes the supported addressing modes for DDR SDRAM densities and configurations required.

Row and Column addresses are generated based on the organization of the SDRAM banks as defined in the memory timing registers (SDRAM0_BnCR). Table 17-10 and Table 17-11 show the mapping of the physical address to memory address for all memory accesses based on the configured mode for the associated bank; the DDR SDRAM device organization is row × column (internal banks).

N = 11, 12, or 13

Table 17-10. 32-Bit DDR SDRAM Addressing Modes

Mode	SDRAM Config		BA1	BA0	MA 12	MA 11	MA10/AP	MA 9	MA 8	MA 7	MA 6	MA 5	MA 4	MA 3	MA 2	MA 1	MA 0
1	Nx8 (4)	Row	20	21	7	8	9	10	11	12	13	14	15	16	17	18	19
		Col	20	21			AP			22	23	24	25	26	27	28	29
2	Nx9 (4)	Row	19	20	6	7	8	9	10	11	12	13	14	15	16	17	18
		Col	19	20			AP		21	22	23	24	25	26	27	28	29
3	Nx10 (4)	Row	18	19	5	6	7	8	9	10	11	12	13	14	15	16	17
		Col	18	19			AP	20	21	22	23	24	25	26	27	28	29
4	Nx11 (4)	Row	17	18	4	5	6	7	8	9	10	11	12	13	14	15	16
		Col	17	18		19*	AP	20	21	22	23	24	25	26	27	28	29

Table 17-11. 64-Bit DDR SDRAM Addressing Modes

Mode	SDRAM Config		BA1	BA0	MA 12	MA 11	MA10/AP***	MA 9	MA 8	MA 7	MA 6	MA 5	MA 4	MA 3	MA 2	MA 1	MA 0
1	Nx8 (4)	Row	19	20	6	7	8	9	10	11	12	13	14	15	16	17	18
		Col	19	20	**		AP			21	22	23	24	25	26	27	28
2	Nx9 (4)	Row	18	19	5	6	7	8	9	10	11	12	13	14	15	16	17
		Col	18	19			AP		20	21	22	23	24	25	26	27	28
3	Nx10 (4)	Row	17	18	4	5	6	7	8	9	10	11	12	13	14	15	16
		Col	17	18			AP	19	20	21	22	23	24	25	26	27	28
4	Nx11 (4)	Row	16	17	3	4	5	6	7	8	9	10	11	12	13	14	15
		Col	16	17		18*	AP	19	20	21	22	23	24	25	26	27	28

*****Column address bit 10 sent out on MA11 for 13 x 11 (4) parts.

**Blank entries are "don't care" for an addressing mode.

***AP is driven based on the specific type of read or write access occurring. With page mode enabled, AP=0 for all read and write commands. With page mode disabled, AP=0 for reads/writes that are part of a sequential PLB burst access and AP=1 for the last read/write of any access.

17.8 SDRAM Commands and Operations

The DDR SDRAM controller supports the following commands:

- Read
- Write

- Activate
- Single-Bank Precharge
- Read/Write with Auto-Precharge
- Precharge All Banks
- Auto CAS Before RAS (CBR) Refresh
- Self-Refresh (Hardware and Software Initiated)

17.8.1 DDR SDRAM Timing Parameters

Programmable memory timing support provides for flexibility at the system level. As a result, memory timing parameters can be adjusted to support faster SDRAM technologies as they become available, and maximize the performance of the SDRAM memory subsystem.

Note: All SDRAM timing diagrams illustrate cycle-based programmable timing parameters only. AC specific timing information should not be inferred from the timing diagrams.

Table 17-12 summarizes the SDRAM memory timing parameters.

Table 17-12. SDRAM Memory Timing Parameters

Name	Function	Description
SD_RCD	Activate to Read/Write Command	Minimum number of clock cycles from an Activate Command to a Read or Write Command. Corresponds to DRAM $\overline{RAS}$ to $\overline{CAS}$ assertion delay.
SD_RFTA	Refresh to Activate	Minimum number of clock cycles from a CBR Refresh Command to the next Activate Command.
SD_CTP	Command to Precharge	Sets the minimum number of clock cycles from a Read or Write Command to the Precharge Command.
SD_PTA	Precharge to Activate	Minimum number of clock cycles required to wait following a Precharge Command to issue the next Activate Command.
SD_RRD	Bank A Activate to Bank B Activate	Module Bank-to-Bank Activate Delay. Hard coded to 2-clock cycles. Pertains to internal banks only. This parameter is fixed and not programmable.
SD_SRX	Self Refresh Exit Delay	Minimum number of clock cycle until access to DDR SDRAM allowed following self refresh exit. 200+ clocks to allow for DLL to lock. This parameter is fixed and not programmable.
SD_CASL	$\overline{CAS}$ Latency	$\overline{CAS}$ access latency.
SD_LDF	Command Leadoff Delay	Number of clock cycles from address/command assertion to chip select assertion.

The following timing diagrams illustrate the relationship of the programmable timings for activate, read, write, and precharge commands:

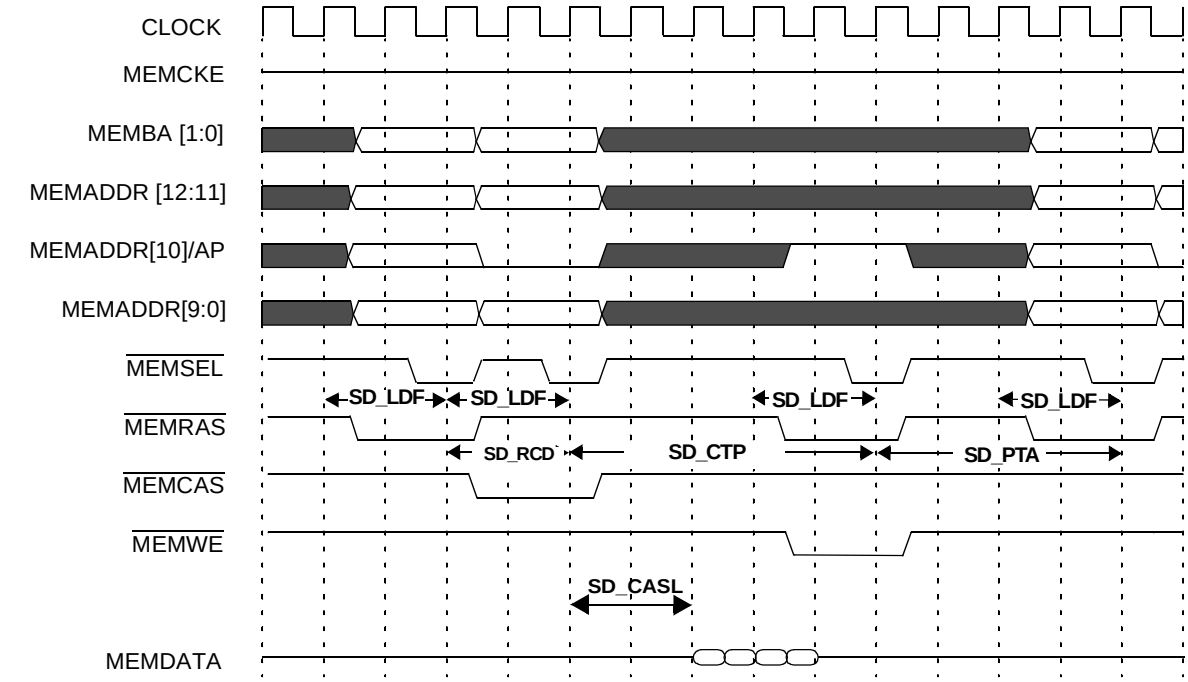


Figure 17-32. Activate - Read - Precharge - Activate

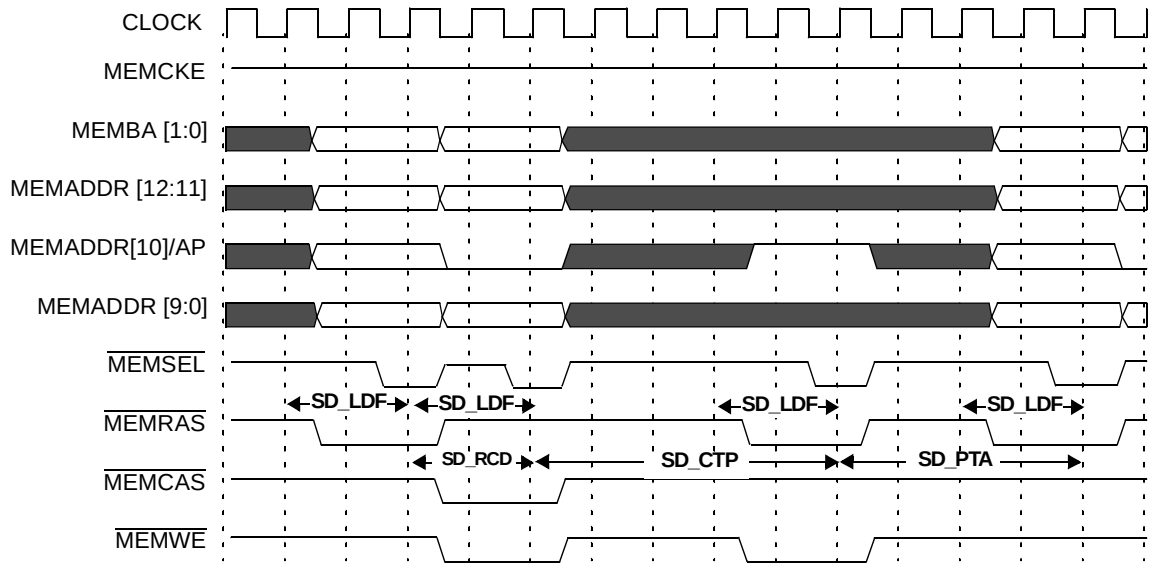
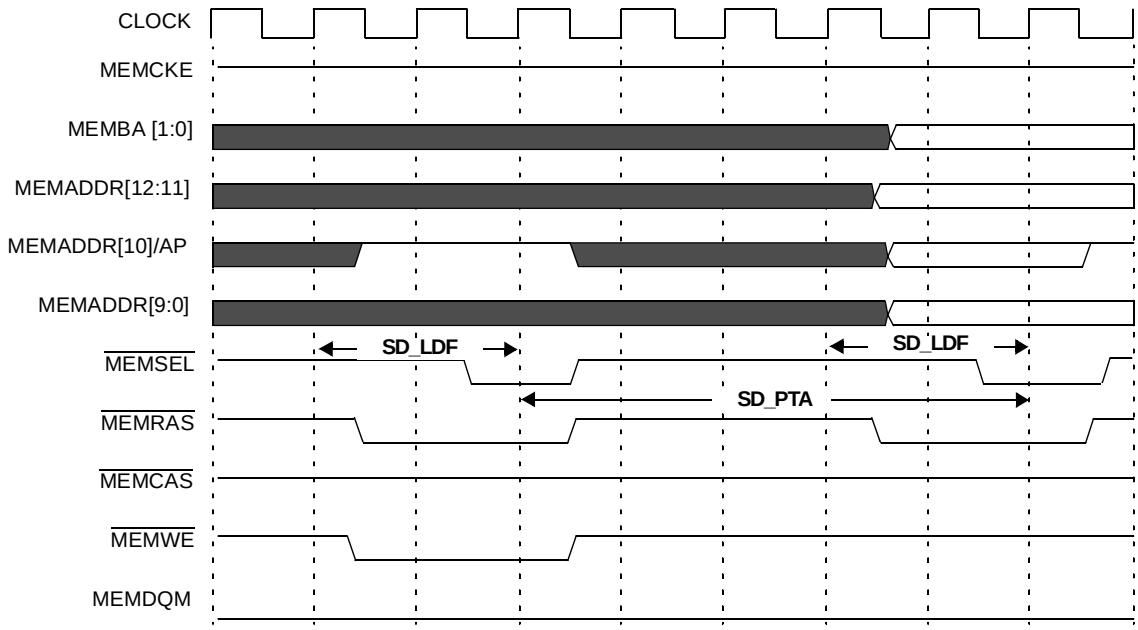


Figure 17-33. Activate - Write - Precharge - Activate



re 17-34. Precharge All - Activate

17.8.2 Precharge Command

The precharge command instructs the SDRAM to precharge some or all banks and is generated by the DDR SDRAM controller. The memory address bus contains the command associated with the precharge cycle, which is formatted as shown in Table 17-13.

Table 17-13. Precharge Command

Command	BA1	BA0	MA12	MA11	MA10/ AP	MA9	MA8	MA7	MA6	MA5	MA4	MA3	MA2	MA1	MA0
Precharge All Banks					b'1'										
Single Bank Precharge for Chip Select N	BA1_N	BA0_N			b'0'										

Note: Shaded entries in Table 17-13 are "don't care" and are driven with a stable value for every clock cycle.

17.8.3 Refresh

Refresh of odd memory banks is staggered from the refresh of even memory banks. Only banks enabled in the memory bank configuration register are initialized following reset and refreshed during normal operation. Once the memory controller is enabled and the initialization sequence has completed, the refresh mechanism starts automatically with refreshing of the memory continuing independent of SDRAM0_CFG0[DCEN].

Refresh requests are generated internally when the refresh timer expires. The refresh interval is programmable using the Refresh Timer Register (SDRAM0_RTR). During refresh, all SDRAM accesses are delayed until the current or pending refresh cycle completes.

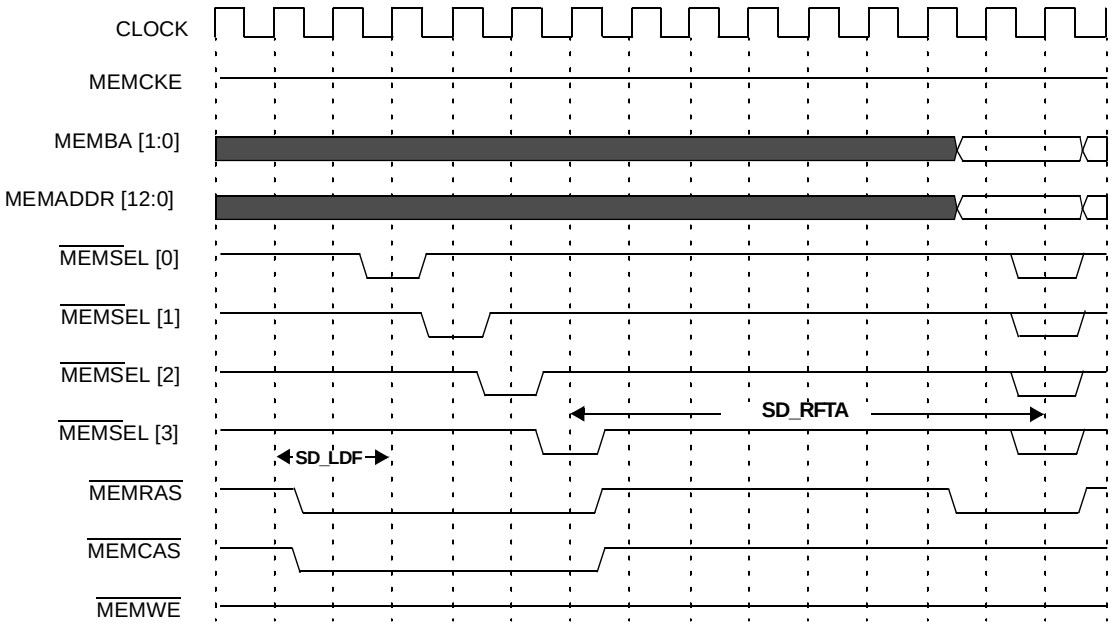


Figure 17-35. Auto (CBR) Refresh

17.8.4 Self-Refresh Operation

The DDR SDRAM controller supports software-initiated self-refresh. The self-refresh exit trigger mechanism is controlled by resetting (once set) SDRAM0_CFG1[SRE].

Note: Only the SDRAM memory is affected by the self-refresh operation.

Although not required, it is recommended that all pending and previously queued requests targeting the DDR SDRAM controller be allowed to complete before entering software-initiated self-refresh. Any pending or previously queued requests which have not completed prior to entering software-initiated self-refresh will stall (including the PLB handshake if in progress) until self-refresh is exited by clearing SDRAM0_CFG1[SRE] or by asserting RESET.

17.8.4.1 Software-Initiated Self-Refresh Operation

The SDRAM Controller supports self-refresh operation for applications desiring lower power.

Entry

Self-refresh entry by software is initiated by setting SDRAM0_CFG1[SRE]. When set, the DDR SDRAM controller will perform the following:

1. Complete the current request.
2. Perform precharge all to close all open pages.

3. Perform an auto-refresh cycle.
4. Enter self-refresh mode and set SDRAM0_MCSTS[SRMS].

The DDR SDRAM controller will maintain the SDRAM in self-refresh mode, independent of any pending memory access request, until SDRAM0_CFG1[SRE] is cleared.

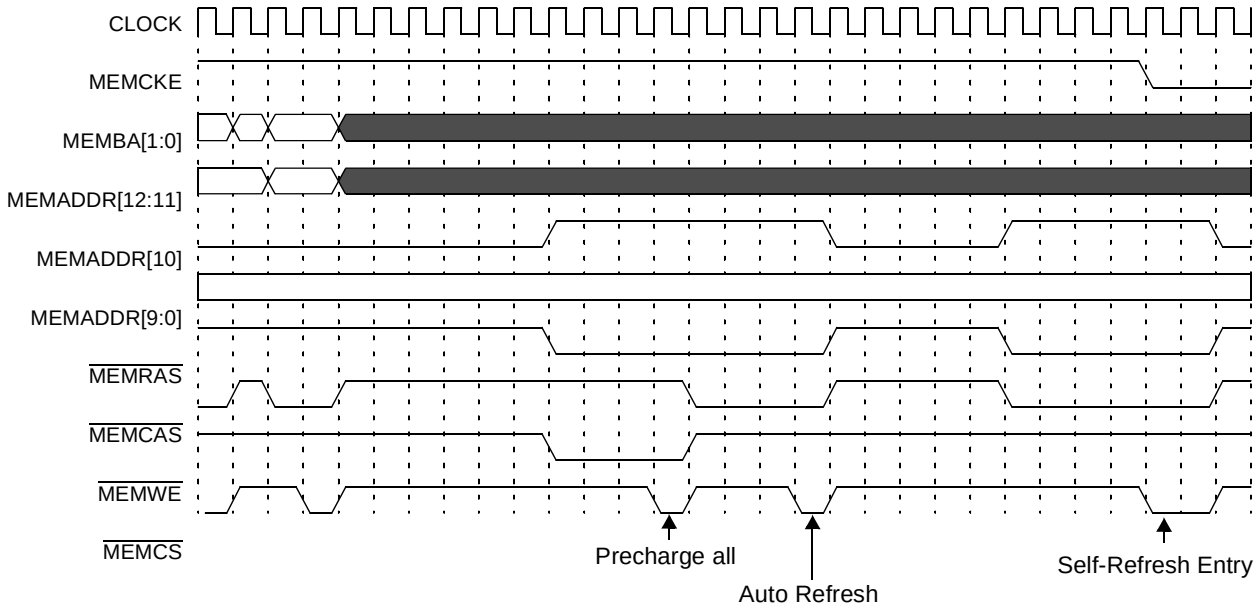


Figure 17-36. Self-Refresh Entry

Exit

Once SDRAM0_CFG1[SRE] is cleared, the SDRAM controller will perform the following:

1. Exit self-refresh mode.
2. Clear SDRAM0_MCSTS[SRMS].
3. Ready to service any memory request (after a minimum of 200 cycles following CKE assertion).

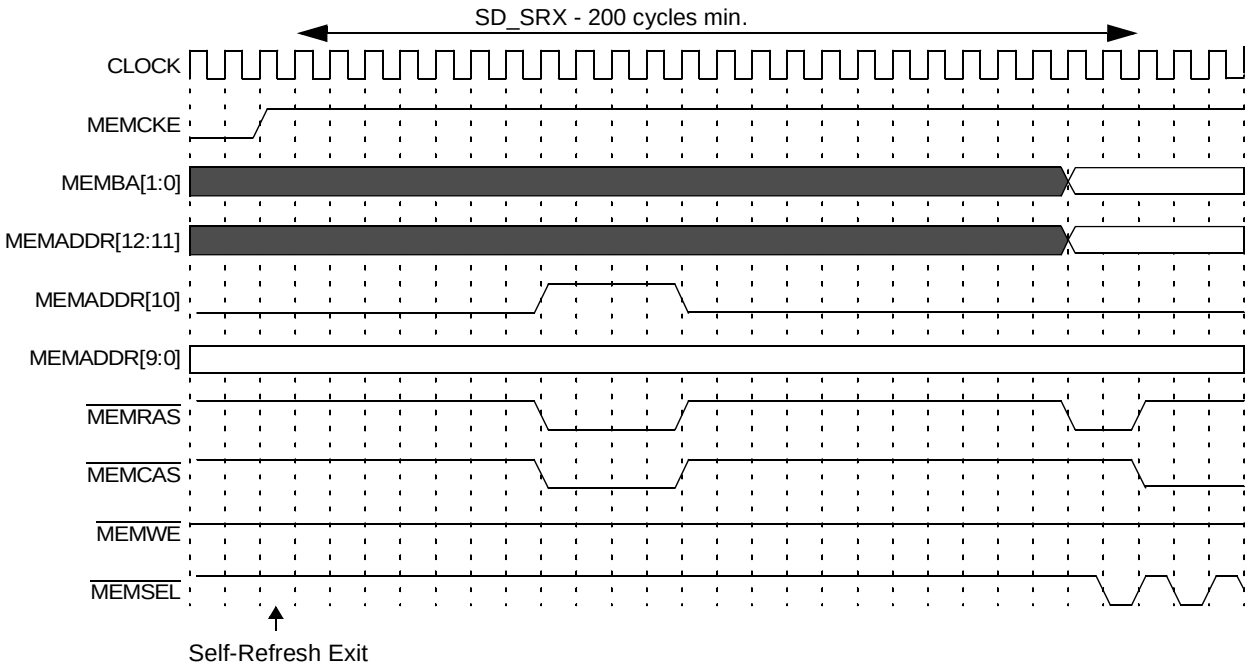


Figure 17-37. Self-Refresh Exit

17.8.5 Mode Register Write Commands

The mode register write commands are issued during initialization to configure the DDR SDRAM operating mode. A total of three mode register write commands are issued. The first command (EMRS) targets the extend mode register and is used to configure the supported DDR SDRAM device-specific options. The supported DDR SDRAM device-specific options are selected using configuration the DDR SDRAM Device Options (SDRAM0_DEVOPT) register. SDRAM0_DEVOPT[0] configures the DLL option and SDRAM0_DEVOPT[1] configures the device drive strength. The second command (MRS1) targets the base mode register and configures the device access parameters and resets the device DLL. The third command (MRS2) targets the base mode register and enables normal operation.

The base mode set command vector consists of the following fields:

- BA[1:0]: Select base mode register
- MA[12:7]: Operating mode
- MA[6:4]: CAS Latency - configured by SDRAM0_TR0[SDCL] register bits for a CAS latency of 2, 2.5, or 3.
- MA[3]: Burst type - hardcoded to 0 to select sequential wrap addressing.
- MA[2:0]: Burst length - hardcoded to 0b010 to configure mode set for a burst of 4.

Table 17-12 illustrates the various mode set command vectors.

Table 17-12. Mode Set Command Vectors

CMD	CL	BA1	BA0	MA11	MA10	MA9	MA8	MA7	MA6	MA5	MA4	MA3	MA2	MA1	MA0
EMRS		0	1	0	0	0	0	0	0	0	0	0	0	DS	DLL
MRS1	2	0	0	0	0	0	1	0	0	1	0	0	0	1	0
	2.5	0	0	0	0	0	1	0	1	1	0	0	0	1	0
	3	0	0	0	0	0	1	0	0	1	1	0	0	1	0
MRS2	2	0	0	0	0	0	0	0	0	1	0	0	0	1	0
	2.5	0	0	0	0	0	0	0	1	1	0	0	0	1	0
	3	0	0	0	0	0	0	0	0	1	1	0	0	1	0

After the SDRAM operating mode has been configured, normal memory accesses can proceed. The mode set command vector is placed on the external memory address bus in the corresponding mode register write command window.

17.9 Registered DIMM Support

Registered interface operation is supported and enabled using SDRAM0_CFG0[RDEN]. The CAS latency setting should be programmed based solely on the DDR SDRAM device CAS latency. Thus, a DDR SDRAM device CAS latency of 2 clocks (SDRAM0_TR0[SDCL] = 2'b01) corresponds to a DIMM CAS latency of 3 clocks, and a DDR SDRAM device CAS latency of 3 clocks (SDRAM0_TR0[SDCL] = 2'b11) corresponds to a DIMM CAS latency of 4 clocks.

17.10 Error Checking and Correction (ECC)

The DDR SDRAM controller supports ECC by enabling SDRAM0_CFG0[2:3]. The ECC block provides check bit generation on PLB-to-memory writes and checking/correction on PLB-to-memory reads. The ECC uses a dual 32-bit or 64-bit single-bit-error-correction/double-bit-error-detection (SEC/DED) code depending on the memory interface width. In 64-bit mode, the DDR SDRAM controller implements a 64-bit data/8-bit check interface on the memory. In 32-bit mode, the DDR SDRAM controller implements a dual 32-bit data/8-bit check interface on the memory. In dual 32-bit mode, the ECC (generation, check and correction) is performed on each 32 bit access to the memory allowing for SEC/DED coverage within each 32-bit word.

Read-Modify-Writes

Any single beat PLB write cycle to memory with byte enables that are not divisible by an ECC word (4 Byte and aligned for 32-bit mode and 8 Byte and aligned for 64-bit mode) will trigger the ECC block to generate a read cycle to fetch the appropriate quad word, modify that quad word, generate the ECC bits and write the quad word back out to memory.

Specifically, in 32-bit mode, PLB single beat write cycles with the following byte enables will not trigger a read-modify-write:

16'b0000_0000_0000_1111, 16'b0000_0000_1111_0000, 16'b0000_0000_1111_1111,
16'b0000_1111_0000_0000, 16'b0000_1111_0000_1111, 16'b0000_1111_1111_0000,
16'b0000_1111_1111_1111, 16'b1111_0000_0000_0000, 16'b1111_0000_0000_1111,
16'b1111_0000_1111_0000, 16'b1111_0000_1111_1111, 16'b1111_1111_0000_0000,
16'b1111_1111_0000_1111, 16'b1111_1111_1111_0000, 16'b1111_1111_1111_1111

Specifically, in 64-bit mode, PLB single beat writes cycles with the following byte enables will not trigger a read-modify-write:

16'b0000_0000_1111_1111, 16'b1111_1111_0000_0000, 16'b1111_1111_1111_1111

The ECC features are described in Table 17-13.

Table 17-13. ECC Features

Feature	Explanation
Standard SEC/DED coverage	The ECC module corrects all single bit errors and detects all double bit errors when reading from memory.
Aligned nibble error detect	The ECC module detects any and all errors which may exist in an aligned four bit nibble.
Checking and Correction Disable	ECC error correction may be disabled. Mixed use of ECC and non-ECC banks is not supported.

ECC Timing

The effect of ECC on read and write accesses is shown in Table 17-14:

Table 17-14. Effect of ECC on Timing

PLB Transaction	Added Latency	Comment
ECC enabled: Read	0/1 Clock	As defined at 0, Read Data is registered prior to going through ECC tree. Actual added latency is programmable and depends on SDRAM0_TR1T settings, operating frequency, and chip/system level timings. As it is defined, it equals 0.
ECC enabled: Burst or full single beat write	None	
ECC enabled: Partial Writes requiring read-modify-write	Read Latency + 2 clocks	On partial writes, a read-modify-write sequence, including bus turn around, is required to correctly generate the write check bits and store the resultant data.

ECC Configuration Register

The ECC module uses only one configuration register to control its operation. The bits are described in the *Memory Controller Options 0 (SDRAM0_CFG0)* section of this document.

ECC Error Register

After an ECC error has occurred, the ECC Error register must be reset in order to enable it to record future errors. **Reset** this register with a write of 0xFFFFFFFF.

Error Detection and Error Handling

The DDR SDRAM controller detects, reports, and logs errors and error status for PLB-to-memory accesses. enables "locking" of the SDRAM0_BESR and SDRAM0_BEAR of the master for errors reported on the PLB. When locked, the error registers remain locked until software clears the associated master lock error status bits in SDRAM0_BESR0/SDRAM0_BESR1.

Memory Read and Write Errors

ECC-related errors are detected and logged for DDR SDRAM PLB-to-memory accesses. ECC must be globally enabled, using SDRAM0_CFG0, and ECC error checking and correction must be enabled using SDRAM0_CFG0[3]. When ECC is disabled, or if ECC is enabled and ECC error checking and correction is disabled, no memory access error checking occurs. Specific error types, information logged, and error responses are detailed in the following sections.

Uncorrectable ECC Error on Memory Read

An Uncorrectable Error detected during a PLB-to-memory read causes the data being returned from system memory (unchanged) to be transferred on the PLB with the associated ERROR signal. The address associated with the error is logged in the SDRAM0_BEAR. The error status for the associated master is logged in either the SDRAM0_BESR0 or SDRAM0_BESR1, depending on the PLB master ID. The ECC Error Status Register is updated to indicate that an Uncorrectable Error occurred with the chip select SDRAM0_ECCESR[BKnE] associated with the logged error. The UNCORRECTABLE ERROR INTERRUPT signal will be asserted and remain asserted until the Uncorrectable Error bit is cleared in the SDRAM0_ECCESR Register. This level-sensed interrupt may be used to signal the Uncorrectable Error event to the system.

Uncorrectable ECC Error on Memory Partial Write

An Uncorrectable Error detected on the read portion of a read-modify-write sequence for a PLB-to-memory partial write results in the data returned from system memory (unchanged) being combined with the "write" data and written back to memory with new ECC check bits. The associated INTERRUPT signal is asserted to indicate the error. This signal will remain asserted until the corresponding bit is cleared in the SDRAM0_MIRQ register. The address associated with the error is logged in the SDRAM0_BEAR. The error status for the associated master is logged in either SDRAM0_BESR0 or SDRAM0_BESR1, depending on the PLB master ID. The ECC Error Status Register is updated to indicate that an Uncorrectable Error occurred with the chip select SDRAM0_ECCESR[BKnE] associated with the logged error. The UNCORRECTABLE ERROR INTERRUPT signal will be asserted and remain asserted until the Uncorrectable Error bit is cleared in the SDRAM0_ECCESR Register. This level-sensed interrupt may be used to signal the Uncorrectable Error event to the system.

Correctable ECC Error on Memory Read

A Correctable Error detected on a DDR SDRAM PLB-to-memory read results in the corrected data returned from system memory being transferred to the requesting master device on the PLB. The ECC Error Status Register (SDRAM0_ECCESR) is updated to indicate that a Correctable Error occurred with the byte lane corrected SDRAM0_ECCESR[BnCE], check bit status (if applicable), and the chip select SDRAM0_ECCESR[BKnE] of the logged error. The CORRECTABLE ERROR INTERRUPT signal will be asserted and remain asserted until the Correctable Error bit is cleared in the SDRAM0_ECCESR. This level-sensed interrupt may be used to signal the Correctable Error event to the system.

Correctable ECC Error on PLB Partial SDRAM Memory Write

A Correctable Error detected on the read portion of a read-modify-write sequence during a PLB-to-memory partial write (less than a double word) results in the corrected data returned from system memory being combined with the "write" data and written back to memory with new ECC check bits. The ECC Error Status Register is updated to indicate that a Correctable Error occurred with the byte lane corrected SDRAM0_ECCESR[BnCE], check bit status (if applicable), and the chip select SDRAM0_ECCESR[BKnE] of the logged error. The CORRECTABLE ERROR INTERRUPT signal will be asserted and remain asserted until the Correctable Error bit is cleared in the SDRAM0_ECCESR. This level-sensed interrupt may be used to signal the Correctable Error event to the system.

ECC Diagnostics

The occurrence of a correctable single bit error or an uncorrectable multi-bit error can simulated and the operation of the ECC verified through a series of memory accesses while varying the Memory Data error checking setting (SDRAM0_CFG0[MCHK]). The general sequence is as follows:

1.) Initialize all DDR SDRAM related configuration registers.

2.) Enable ECC generation only.

3.) Enable the SDRAM controller.

4.) Initialize (Write) the entire installed memory space to initialize the data and check bits.

5.) Allow the DDR SDRAM to become idle.

6.) Set the Memory Data error checking to "None."

7.) Modify one bit (in the case of a CE) or two bits (in the case of a UE) at the address to be tested.

8.) Allow the DDR SDRAM to become idle.

9.) Set the Memory Data error checking to "ECC Checking and Correction."

10.) Read the corresponding address.

In the case of simulating a Correctable Error, the corrected data should be returned to the system. In the case of simulating an Uncorrectable Error, the original data (with the two changed bits) should be returned to the system. In either case, the described error response corresponding to the error simulated should also occur.

17.10.1 ECC Code Matrix

The 32-bit mode ECC code matrix for word 0 is shown in Figure 17-38.

02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20
X	X	X	X	X	X	X	X	X	X	X	X	X	X				X	
		X				X				X				X	X	X	X	X
			X				X				X			X				X
X				X				X				X			X			
	X				X				X				X			X	X	
		X	X	X	X					X	X	X	X					X
						X	X	X	X	X	X	X	X					
X	X									X	X	X	X	X	X	X		

Figure 17-38. 32-Bit Mode ECC Code Matrix for Word 0

The 32-bit mode ECC code matrix for word 1 is shown in Figure 17-39.

27	28	29	30	31	UC0	UC1	UC2	UC3	UC4	UC5	UC6	UC7
X				X	X							
X	X	X	X	X		X						
	X						X					
		X						X				
X			X	X					X			
	X	X	X	X						X		
X	X	X	X	X							X	
X	X	X	X									X

Figure 17-38. 32-Bit Mode ECC Code Matrix for Word 0 (Continued)

34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51
X				X				X				X					X
	X				X				X				X	X			
X	X	X	X	X	X	X	X	X	X	X	X	X	X		X		
		X				X				X						X	
X	X		X	X	X		X	X	X		X	X	X				
		X	X	X	X					X	X	X	X	X	X	X	X
						X	X	X	X	X	X	X	X	X	X	X	X
				X	X			X	X	X	X						

Figure 17-39. 32-Bit Mode ECC Code Matrix for Word 1

58	59	60	61	62	63	LC0	LC1	LC2	LC3	LC4	LC5	LC6	LC7
	X		X			X							
				X			X						
					X			X					
X		X	X	X	X				X				
		X	X	X	X					X			
											X		
X	X	X	X	X	X							X	
X	X		X	X	X								X

Figure 17-39. 32-Bit Mode ECC Code Matrix for Word 1 (Continued)

The 64-bit mode ECC code matrix is shown in Figure 17-40. It is derived from the upper (UC 0:7) and lower (LC 0:7) checkbits calculated for 32-bit mode.

4	UC5	UC6	UC7	LC0	LC1	LC2	LC3	LC4	LC5	LC6	LC7	C0	C1	C2	C3	C4	C5	C6	C7
				X								X							
					X								X						
						X								X					
							X								X				
								X								X			
	X								X								X		
		X								X								X	
			X								X								X

Figure 17-40. 64-Bit Mode ECC Code Matrix

17.11 Power Management

The DDR SDRAM controller implements Class II (Macro Paced Sleep) power management as defined by the *Clock and Power Management Interface Specification for core + ASICs*. As such, the DDR SDRAM controller provides a single sleep request output which indicates that the controller is idle and may be put to sleep.

17.11.1 Sleep Mode Entry

The DDR SDRAM controller power management behavior is configurable using DCR registers SDRAM0_CFG1[PMEN], located at offset 0x21, and SDRAM0_PMIT[PM_C] located at offset 0x34. Setting the PMEN=1 bit enables power management logic to send a request to the Clock Power Management (CPM) unit during sleep once the controller has been idle for the programmed (PM_C) number of controller clock cycles. With PMEN=0, DDR SDRAM controller power management is disabled.

17.11.2 Sleep Mode

In sleep mode, all DDR SDRAM controller clocking is disabled, with the exception of SDRAM refresh logic and power management WAKE logic. SDRAM refresh preserves the contents of the memory and maintains the refresh interval. Power management WAKE logic monitors master device requests on the PLB and DCR bus interfaces and initiates the WAKE-UP sequence when a valid master request PLB or DCR bus cycle is received.

17.11.3 Sleep Mode Exit

Once a valid master request, PLB request, or DCR request targeting the DDR SDRAM is registered, the controller deasserts its sleep request in the following clock cycle. Once sleep mode is exited, the DDR SDRAM begins the requested access in the following clock cycle.

17.12 System Memory Clock Concerns

The DDR memory clock signal found in the PPC440GP has the ability to drive a limited number of loads, two to three. In many board level designs, the DDR memory clock signal must be buffered before sending the signal out to the memory devices. An external differential clock buffer is used to generate individual copies of the differential memory clock. This external differential clock buffer must be a zero-delay (PLL-based) differential clock buffer wire delay and clock buffer-to-memory wire delay.

When using the external clock buffer in general, it is expected that the DDR memory clock is advanced by 90 degrees to help compensate for the on-chip differential clock driver delay. It is also expected that the on-chip generated PLB clock is used to clock the read data pathe sample stage 2.

18. PLB-PCIX Bridge Controller

18.1 Overview

The PLB-PCIX Bridge provide an interface between the PCI bus and the Processor Local Bus (PLB). It enables PCI initiators to access PLB slaves and PLB masters to access PCI targets. PCI initiators and PCI targets may be in PCI conventional or PCI-X mode.

The PLB-PCIX Bridge also contains an internal register set that can be accessed by both PLB masters and PCI initiators.

The PLB-PCIX Bridge has many optional and programmable features that enable it to be configured for different applications. These options and features are set using ~~strapping pins or using~~ the internal registers.

The PLB-PCIX Bridge may be used in "host-bridge" mode or "adapter-bridge" mode.

The PLB-PCIX Bridge is a "bridge" device, in that it only generates transfers as a master on the PLB or the PCI bus in response to decoding a transfer as a slave on the other bus.

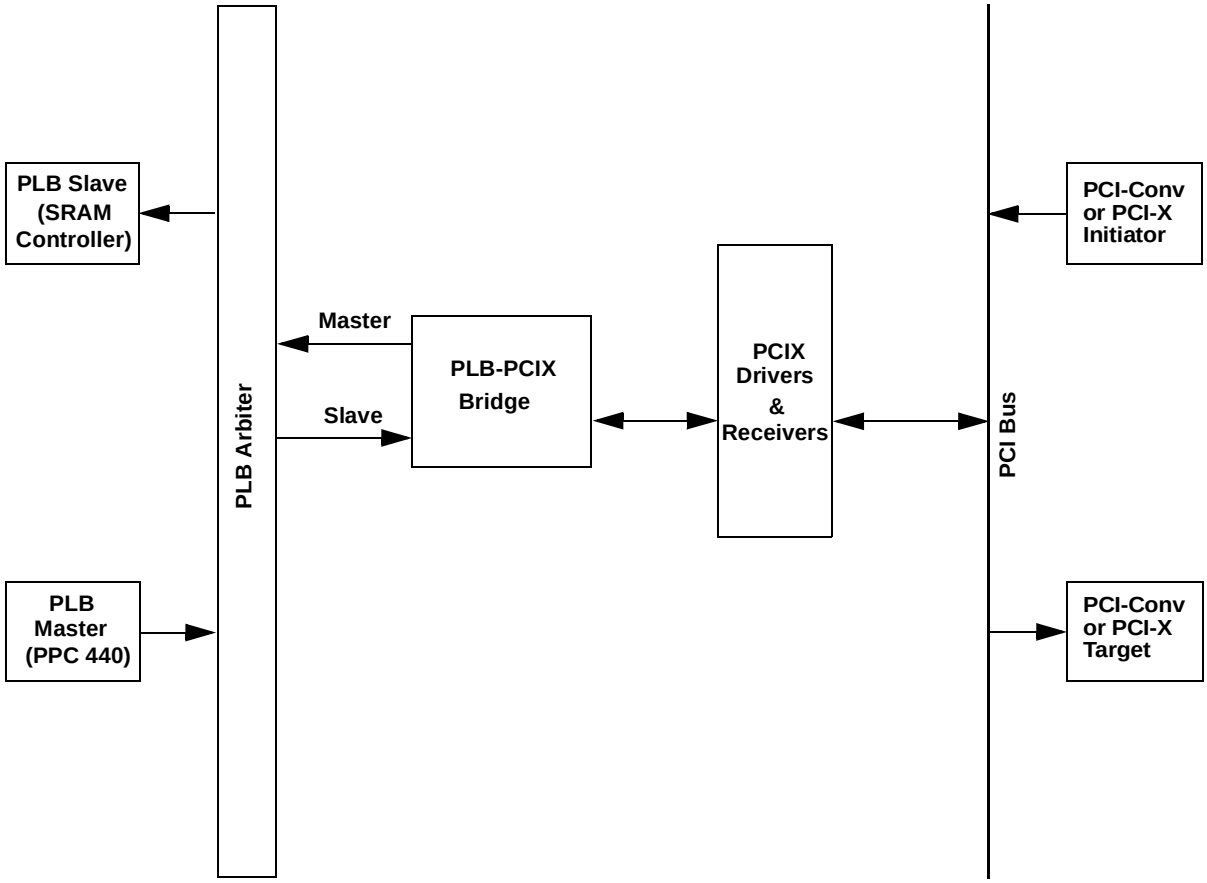
18.2 Features List

- Bridge between PLB and PCI bus
- Host bus bridge or adapter interface to PCI bus
- Designed to *Processor Local Bus Specification,* Version 4~~.0, compliant~~
- PLB frequency up to 133 MHz
- 64-bit address and 128-bit data when a PLB Master
- 64-bit address and 128-bit data when a PLB Slave
- Designed to *PCI,* Version 2.2~~, compliant~~
- Designed to *PCI-X Addendum to the PCI Local Bus Specification,* Version 1.~~0, compliant~~0a
- PCI-Conventional bus frequency up to 66 MHz
- PCI-X bus frequency up to 133 MHz
- 64-bit or 32-bit ~~PCI~~ PCI-X address/data bus
- Designed to *PCI Bus Power Management Interface Specification,* Version 1.1~~, compliant~~
- Buffering
  - PCI target: two 256 byte write post buffers
  - PCI target: two 256 byte read prefetch buffers (only one available in conventional PCI mode)
  - PLB slave: two 256 byte write post buffers
  - PLB slave: one 256 byte read prefetch buffer and one dedicated 512-byte read prefetch buffer
- Error tracking/status
- PCI arbitration function (optional)
- Asynchronous clocking between PLB and PCI busses
- Internal register set (PCI configuration register set) is accessible from both PLB and PCI sides
- Supports initiation (mastering) of transfer to the following PCI address spaces:

- Single-beat I/O reads and writes
- Single-beat and burst memory reads and writes
- Single-beat configuration reads and writes (Type 0 and Type 1)
- Single-beat special cycles

- Programmable address mapping and address translation
- Message Signaled Interrupts (MSI) support for inbound and outbound interrupts
- Simple message passing capability
- Ability to boot PLB processor from PCI bus memory
- PCI-X initialization sequence support (frequency & mode determination)

18.3 Typical Application



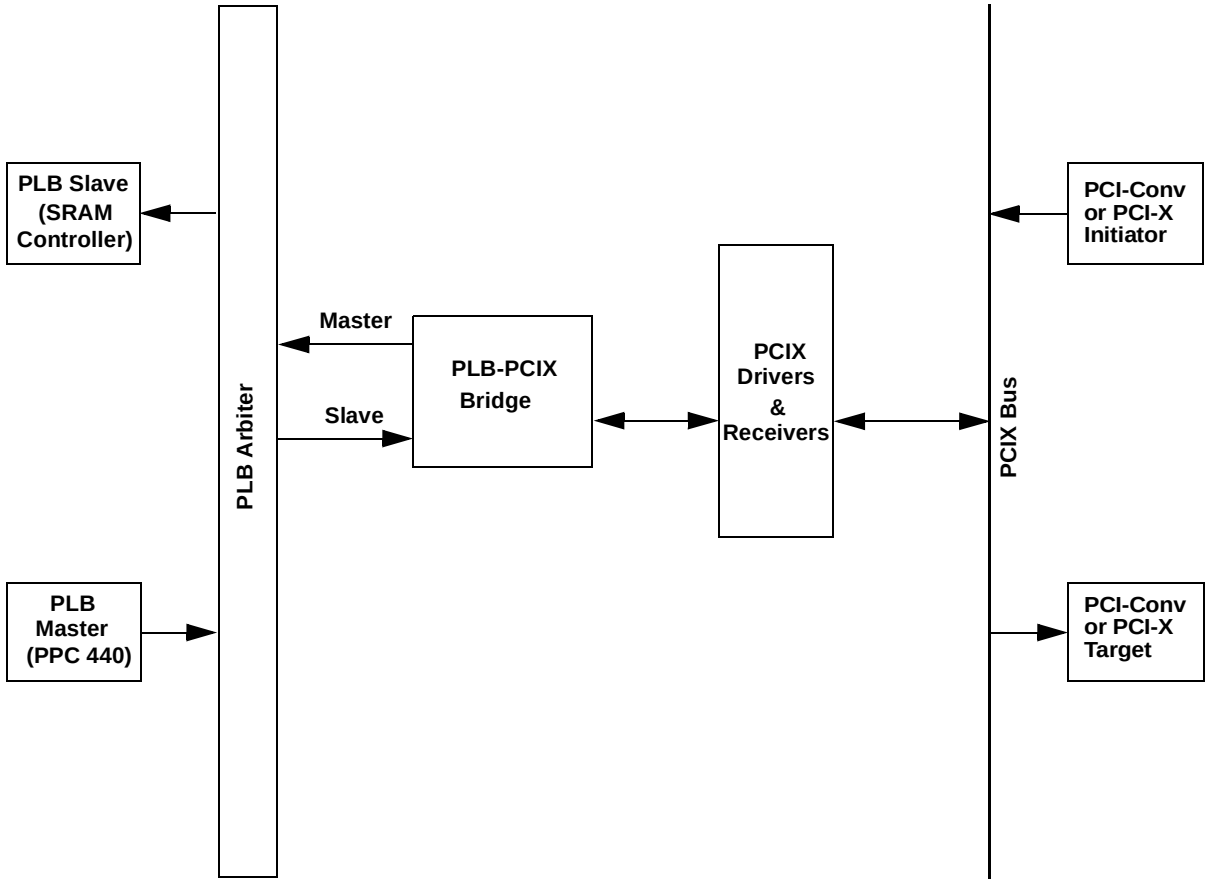


Figure 18-1. Typical Application using the PLB-PCIX Bridge The PLB-PCIX Bridge enables PCI initiators to access PLB slaves and PLB Masters to access PCI targets. PCI initiators and PCI targets may be in PCI Conventional (PCI-Conv) or PCI-X mode.

18.4 Block Diagram

The block diagram in ~~Figure 18-2~~*Figure 18-2* shows the internal structure of the PLB-PCIX Bridge.

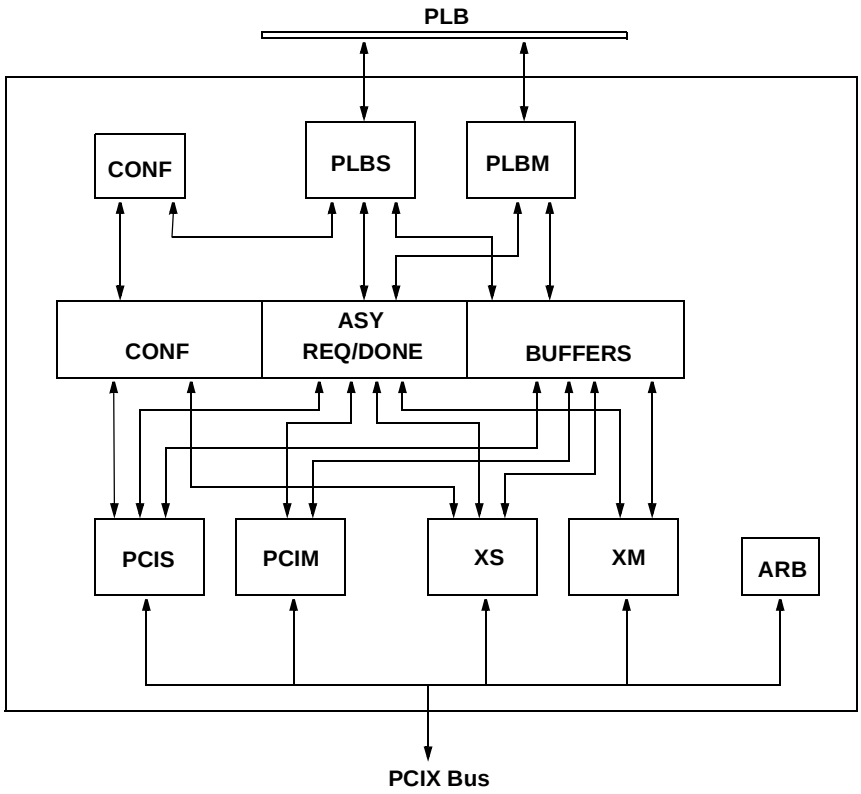


Figure 18-2. PLB-PCIX Bridge Internal Structure

18.5 References

- *Processor Local Bus Specification,* Version 4
- *PCI,* Version 2.2
- *PCI-X Addendum to the PCI Local Bus Specification,* Version 1.0
- *PCI Bus Power Management Interface Specification*, Version 1.1

18.6 Inbound Transaction Handling

The PLB-PCIX Bridge can claim a variety of inbound transaction types on the PCI bus and subsequently master a transfer on the PLB bus or access an internal register.

Note: This chapter defines the terms "outbound" and "inbound" to describe the direction of traffic through the PLB-PCI Bridge. The PLB bus is considered "in" and the PCI bus is considered "out". Thus, an "outbound" transfer is a read or write in which the PLB-PCI Bridge is the slave on the PLB bus and the initiator on the PCI bus. An "inbound" transfer is a read or write in which the PLB-PCI Bridge is the target on the PCI bus and the master on the PLB bus.

18.6.1 PCI Conventional

The PLB-PCIX Bridge can claim inbound transactions on the PCI bus for the command types listed below:

- Memory Read
- Memory Read Line
- Memory Read Multiple
- Memory Write
- Memory Write and Invalidate
- I/O Read
- I/O Write
- Configuration Type 0 Read
- Configuration Type 0 Write

Memory Reads are handled as prefetchable or non-prefetchable, as determined by the address (see section ~~"Interrupts and MSI" on page 19~~*Interrupts and MSI* on page 553 for details). For a region marked non-prefetchable, a corresponding single-beat PLB read is dispatched, which accesses the exact bytes as indicated by the byte enables of the PCI read (discontiguous byte enables are not supported).

For a region marked prefetchable, a corresponding burst PLB read is dispatched with the address truncated to a 128-bit boundary. When the command type is memory read, 32 bytes are read from the PLB. When the command type is memory read line, 64 bytes are read from the PLB, and when the command type is memory read multiple, 256 bytes are read from the PLB.

Note: The PLB-PCIX Bridge can be programmed to handle memory read as a memory read line or a memory read multiple (see Section ~~"Bridge Options Registers" on page 62~~*Bridge Options Registers* on page 594).

Memory Write and Memory Write and Invalidate are handled exactly the same. They are posted and then written to the PLB. The PLB transfer may include both single-beat and quad-word bursts, depending on the size and alignment of the write.

I/O Read and I/O Write are not handled as connected tenures. Instead, I/O read and I/O write are handled exactly like non-prefetchable memory reads/writes. Thus, I/O reads are delayed reads and I/O writes are posted.

Configuration Reads and Configuration Writes are disconnected after one beat. They access the internal register set directly (no delayed read or write posting). One exception is configuration writes to PCI power management power state that may be handled as a delayed write.

In response to the inbound commands of type Memory or I/O listed above, the PLB-PCIX Bridge masters on the PLB the following command types:

- Single Beat Read or Write
- Quadword Fixed Length Burst Read or Write

18.6.2 PCI-X

The PLB-PCIX Bridge can claim inbound transactions on the PCI bus for the command types listed below:

- Memory Read Block

- Memory Write Block
- Memory Write
- Alias to Memory Read Block
- Alias to Memory Write Block
- Split Completion
- I/O Read
- I/O Write
- Configuration Read
- Configuration Write
- Memory Read Doubleword

Memory Read Block is handled as a split read. The requested length, or length that is slightly more than the requested length, is read from the PLB. If the read does not cross a 64-bit boundary (single beat on the PCI bus), then the exact bytes requested are read from the PLB (discontiguous byte enables are not supported). If the read does cross a 64-bit boundary, then prefetching is assumed, and the starting address is rounded down to a 16-byte boundary and the ending address is rounded up to a 16-byte boundary for the PLB read. If a prefetching Memory Read Block occurs (crosses a 64-bit boundary) to a region of the PLB that is non-prefetchable, the results are undefined, because the accessed PLB slave may not be able to handle a PLB burst transaction or the rounding up and down of the address.

Memory Write Block is posted and the exact indicated length is written to the PLB.

Memory Write is handled exactly like Memory Write Block - byte enables are ignored. Thus, discontiguous byte enables are not supported.

Alias to Memory Read Block and Alias to Memory Write Block are handled exactly like Memory Read Block and Memory Write Block.

Split Completions with matching bus numbers and device IDs are always accepted and assumed (not checked) to have been expected. Unexpected split completions will result in undefined behavior.

I/O Read and I/O Write are not handled as connected tenures (PCI-PCI Bridges normally handle I/O transactions as connected tenures). Instead, I/O read and I/O write are handled exactly like Memory Read Doubleword and Memory Write.

Configuration read or write are handled with immediate completion. They access the internal register set directly (no delayed read or write posting). One exception is configuration writes to PCI power management power state that may be handled as a delayed write. See Section ~~"PCI Power Management Interface" on page 20~~*PCI Power Management Interface* on page 554 for details.

Memory Read Doubleword causes a split read. The exact requested length is read from the PLB (discontiguous byte enables are not supported).

In response to the inbound commands of type Memory or I/O listed above, the PLB-PCIX Bridge masters on the PLB the following command types:

- Single Beat Read or Write
- Quadword Fixed Length Burst Read or Write

18.7 Outbound Transaction Handing

The PLB-PCIX Bridge can claim outbound transactions on the PLB bus and subsequently initiate a transfer on the PCI bus or access an internal register.

The PLB-PCIX Bridge can claim PLB transactions of the following types:

- Single Beat Read or Write
- 4-Word Line Read or Write
- 8-Word Line Read or Write
- Doubleword Fixed Length Burst Read or Write
- Quadword Fixed Length Burst Read or Write

How these transfers are handled is determined by the address (see section ~~"Address Mapping and Address Translation" on page 9~~*Address Mapping and Address Translation* on page 542 for details). Based on the address, these transaction either access the internal register set or initiate a cycle to the PCI bus.

18.7.1 PCI Conventional

Based on the PLB address, the PLB-PCIX Bridge can initiate the following transactions on the PCI bus:

- Memory Read
- Memory Read Line
- Memory Read Multiple
- Memory Write
- Memory Write and Invalidate
- I/O Read
- I/O Write
- Configuration Read
- Configuration Write
- Special Cycle

Memory Read, Memory Read Line, or Memory Read Multiple is generated for an address to PCI memory space. The Memory Read command is used for transfers of 1 to 4 byte reads. Memory Read Line is used for reads of greater than 4 bytes and less than or equal to 32 bytes (16, 24, or 32 bytes). Memory Read Multiple is used for reads of greater than 32 bytes.

Note: 5 to 8 byte reads are run as a 32-bit mastered, 2-beat burst, rather than a single 64-bit transfer (this is recommended by the *PCI Specification, Version 2.2*).

Memory Write or Memory Write and Invalidate is generated for an address to PCI memory space. The Memory Write and Invalidate is used for transfers beginning and ending on a 64-byte boundary and whose length is a multiple of 64 bytes; otherwise, a Memory Write is used.

~~I/O Read and I/O Write are generated for an address to PCI I/O space.~~

I/O Read, I/O Write, and Special Cycle can be generated. The PLB transfer must be single beat that does not cross a 32-bit boundary.

Configuration Read and Configuration Write, of both Type 0 and Type 1 can be generated. See the section on ~~"External PCI Configuration Cycles" on page 8~~*External PCI Configuration Cycles* on page 541 for details.

~~Special Cycle can be generated.~~

The generation of Interrupt Acknowledge is not supported.

Dual address cycles are generated whenever the PCI address is greater than 4 GB.

18.7.2 PCI-X

Based on the PLB address, the PLB-PCIX Bridge can initiate the following transactions on the PCI bus:

- Memory Read Block
- Memory Write Block
- Split Completion
- I/O Read
- I/O Write
- Configuration Read
- Configuration Write
- Special Cycle

Memory Read Block is generated for reads that are non-prefetching or prefetching, as indicated by the PLB transfer. The length (byte count) may range from 1 to 512 bytes. Memory Read Doubleword is never generated.

Memory Write Block is generated for memory writes. The length (byte count) may range from 1 to 256 bytes. Memory Write is never generated.

Configuration Read and Configuration Write, of both Type 0 and Type 1 can be generated. See the section on ~~"External PCI Configuration Cycles" on page 8~~*External PCI Configuration Cycles* on page 541 for details.

I/O Read, I/O Write, and Special Cycle can be generated. The PLB transfer must be single beat that does not cross a 32-bit boundary.

18.8 External PCI Configuration Cycles

PLB masters generate configuration cycles to the PCI bus by accessing the CONFIG_ADDRESS and CONFIG_DATA registers that are located in PLB space. CONFIG_ADDRESS and CONFIG_DATA are implemented as specified in *PCI*, Version 2.2, Section 3.2.2.3 (Configuration Mechanism #1), except that they are located at addresses 0000_0002_0EC0_0000 and 0000_0002_0EC0_0004 rather than CF8h and CFCh, and 0 and 31 are used differently.

CONFIG_ADDRESS and CONFIG_DATA must be accessed with single-beat PLB transfers that don't cross a 32-bit boundary.

18.8.1 Configuration Mechanism

The general mechanism for accessing PCI configuration space is to write a value into CONFIG_ADDRESS that specifies the PCI bus number, the device number on that bus, and the configuration register in that device being accessed. Then, a read or write to CONFIG_DATA causes the bridge to generate the PCI configuration cycle, with the address and type translated from the CONFIG_ADDRESS value.

18.8.1.1 CONFIG_ADDRESS Register

This register controls what type of cycle is generated when CONFIG_DATA is accessed. Its fields are shown in ~~Figure 18-3~~*Figure 18-3*.

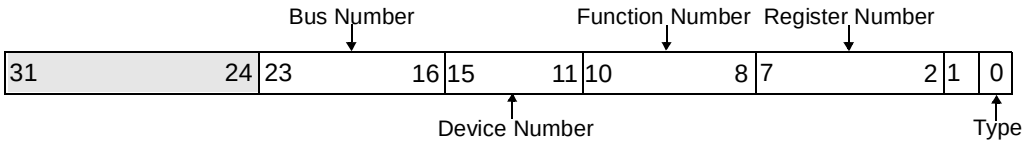


Figure 18-3. Format of CONFIG_ADDRESS Registers See *PCI Specification,* Version 2.2 for details about how the fields are used.

Bits 31 to 24 are reserved and always return a 0 when read.

Bit 0 determines whether the configuration cycle is Type 0 or Type 1.

18.8.1.2 CONFIG_DATA Register

Accesses to CONFIG_DATA cause one of two things to occur depending on the value of CONFIG_ADDRESS:

- Generate a Type 0 configuration cycle on the PCI bus. This happens when the Type bit (bit 0) is low. During the address phase of the configuration cycle, pci_ad[16] is asserted if the Device Number is 0, pci_ad[17] is asserted if the Device Number is 1, pci_ad[18] is asserted if the Device Number is 2, and so forth. pci_ad[16] through pci_ad[31] are all low if the device number is 16 through 31.
- Generate a Type 1 Configuration Cycle on the PCI bus. This happens when the Type bit (bit 1) is high.

Note: The PLB-PCIX Bridge will not respond as a PLB slave to accesses to the CONFIG_DATA register if the PCI Master Enable bit in the PCI command register is zero because accesses to the CONFIG_DATA register require the PLB-PCIX Bridge to become a PCI master.

18.9 Address Mapping and Address Translation

The PLB-PCIX Bridge register set has several ranges of PLB address space and several ranges of PCI address space that it can claim.

18.9.1 Outbound Address Map

The PLB-PCIX Bridge responds as a slave on the PLB in several address ranges. These ranges allow a PLB master to access the internal register set, and to cause the PLB-PCIX Bridge to generate memory, I/O, configuration and special cycles to the PCI bus.

PLB masters can access the PLB-PCIX Bridge internal register set or PCI slaves in Memory, I/O or Configuration address space by mastering a PLB cycle to an address that indicates the intended destination. There is one or more PLB address ranges for each destination, as shown in ~~Table 18-1, "Outbound Address Map," on p. 9~~*Table 18-1* on page 543.

If the destination is the internal register set, then the PLB address ranges are fixed, as shown in ~~Table 18-1, "Outbound Address Map," on p. 9~~*Table 18-1* on page 543.

If the destination is PCI Memory space, there are up to three PLB address ranges that PLB-PCIX Bridge can claim. The number of ranges and their location in PLB address space is programmable and determined by the "POM" registers. The PCI address may be a translation of the PLB address. The translation is programmable and also, determined by the "POM" registers. See section ~~"PCI Outbound Map (POM) Configuration" on page 11~~*PCI Outbound Map (POM) Configuration* on page 544 for more details.

If the destination is PCI I/O space, PCI Configuration space or PCI special cycle, then the PLB address ranges and the resulting PCI address ranged (translations) are fixed as shown in Table 18-1.

~~Table 18-1, "Outbound Address Map," on p. 9~~Table 18-1 shows the outbound address map from the view of the PLB, that is, as decoded by the PLB-PCIX Bridge as a PLB slave.

Table 18-1. Outbound Address Map

PLB Address Range	Description	PCI Address
0000_0002_0000_0000h 0000_0002_07FF_FFFFh	Reserved PLB-PCIX Bridge does not respond	
0000_0002_0800_0000h 0000_0002_0800_FFFFh	PCI I/O Accesses to this range are translated to an I/O access on PCI in the range 0 to 64K-1	0000_0000h 0000_FFFFh
0000_0002_0801_0000h 0000_0002_087F_FFFFh	Reserved: PLB-PCIX Bridge does not respond (Other bridges use this space for discontiguous I/O).	
0000_0002_0880_0000h 0000_0002_0BFF_FFFFh	PCI Extra I/O Accesses to this range are translated to an I/O access on PCI in the range 8 MB to 64 MB	0080_0000h 03FF_FFFF
0000_0002_0C00_0000h 0000_0002_0EBF_FFFFh	Reserved PLB-PCIX Bridge does not respond	
0000_0002_0EC0_0000h 0000_0002_0EC7_FFFFh	PCI CONFIG_ADDRESS and CONFIG_DATA nEC0_0000h: CONFIG_ADDRESS nEC0_0004h - nEC0_0007h: CONFIG_DATA nEC0_0008h - nEC7_FFFFh: Reserved (may contain mirrors of CONFIG_ADDRESS and CONFIG_DATA)	
0000_0002_0EC8_0000h 0000_0002_0ECF_FFFFh	PLB-PCIX Bridge Registers nEC8_0000h - nEC8_01FF: internal registers nEC8_0200h - nECF_FFFF: Reserved (may contain mirrors of internal registers.)	

Table 18-1. Outbound Address Map (continued)

PLB Address Range	Description	PCI Address
0000_0002_0ED0_0000h 0000_0002_0EDF_FFFFh	Special Cycle nED0_0000h four-byte write only: Special Cycle nED0_0000h read only: Reserved nED0_0004h - FEDF_FFFFh: Reserved (may contain mirrors of Special Cycle)	
0000_0000_0000_0000h FFFF_FFFF_FFFF_FFFFh[1]	PCI Memory - Range 0 POM 0 registers map a region in PLB space to a region in PCI memory space. The address ranges are fully programmable.	0000_0000_0000_0000h FFFF_FFFF_FFFF_FFFFh
0000_0000_0000_0000h FFFF_FFFF_FFFF_FFFFh[1]	PCI Memory - Range 1 POM 1 registers map a region in PLB space to a region in PCI memory space. The address ranges are fully programmable.	0000_0000_0000_0000h FFFF_FFFF_FFFF_FFFFh
8000_0000_0000_0000h FFFF_FFFF_FFFF_FFFFh	PCI Memory - Range 2 POM 2 maps a region in PLB space to a region in PCI memory space. The starting address for Range 2 is fixed to 8000_0000_0000_0000. The PCI address always equals the PLB address minus 8000_0000_0000_0000 in this range.	0000_0000_0000_0000h 7FFF_FFFF_FFFF_FFFFh
[1]These PLB address ranges are software programmable to any value. However, they must be set to values that do not overlap address ranges used by other devices. The 440GP address map requires that the POM0 and POM1 PLB address be in the range from 0000_0002_0EE0_0000h to FFFF_FFFF_FFFF_FFFFh.		

~~Figure 18-4~~*Figure 18-4* shows the detail of the POM register sets used to map PLB memory regions to PCI address space.

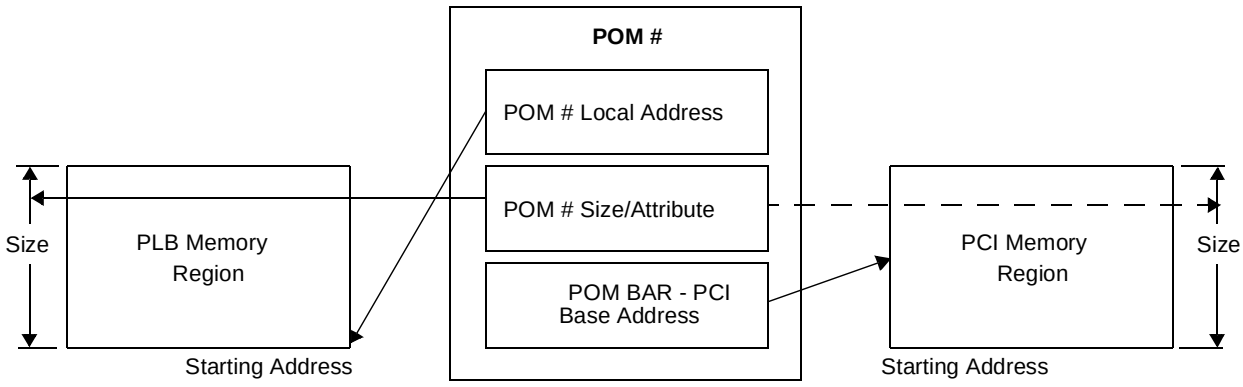


Figure 18-4. POM Register Sets Map PLB Address Space to PCI Address Space

18.9.2 PCI Outbound Map (POM) Configuration

The PLB-PCIX Bridge has three ranges in PLB space that are mapped to PCI Memory space. These ranges are referred to as POM 0, POM 1, POM 2. The characteristics of each POM are defined by a set of registers in the PLB-PCIX Bridge PCI configuration registers.

POM 0 is controlled by the following registers:

- POM 0 Local Low Address
- POM 0 Local High Address
- POM 0 Size/Attribute
- POM 0 PCI Low Address
- POM 0 PCI High Address

POM 1 is controlled by the following registers:

- POM 1 Local Low Address
- POM 1 Local High Address
- POM 1 Size/Attribute
- POM 1 PCI Low Address
- POM 1 PCI High Address

POM 2 is controlled by the following register:

- POM 2 Size/Attribute (enable bit only)

The location in PLB space of POM0 and POM1 is programmable using the Local Low/High Address registers. The location of POM 2 is fixed as defined above. The PLB address range assigned to each POM should not overlap any other range of PLB space in use. Overlapping will result in undefined behavior.

The range of PCI memory address space that is accessed through POM 0 and POM 1 is also programmable, using the PCI Low Address or PCI High Address registers. This allows address translation between the two busses. The range of PCI memory address space that is accessed through POM 2 is fixed and is a 63-bit address (lower half of the 64-bit memory address space). If the PCI high address is zero, then single-address cycles are generated to the PCI bus. If the PCI high address is greater than zero, then PCI dual address cycles are generated to the PCI bus.

The size of POM 0 and POM 1 is programmable, using the mask portion of the Size/Attribute registers. The size is a power of two, and is a minimum of 1 MB and a maximum of 4 GB. The size of POM 2 is fixed to 2^{63} bytes. The PLB and PCI address spaces for each POM are aligned to this size.

The POMs can be enabled/disabled by the Attribute bits in the Size/Attribute registers.

The address ranges should all be initialized before the POM is enabled.

See ~~"POM Registers" on page 73~~*POM Registers* on page 605 for details on PCI POM registers.

18.9.3 Inbound Address Map

The PLB-PCIX Bridge responds as a PCI target for memory accesses to BAR 0 and either BAR2 or Option ROM BAR, and I/O accesses to BAR 1. BAR1 and Option ROM BAR are 32-bit BARs and BAR0 and BAR2 are 64-bit BARs. (The PLB-PCIX Bridge also responds as a PCI target for Configuration Type 0 accesses).PCI initiators can access the PLB-PCIX Bridge internal register set or PLB slaves by initiating a PCI cycle to an address that indicates the intended destination.

If the destination is the internal register set, then the PCI initiator must run a type 0 configuration cycle, with the appropriate device select, such that the PLB-PCIX Bridge's IDSEL is active.

If the destination is PLB space, there are up to two PCI memory address ranges and up to one PCI I/O address range that PLB-PCIX Bridge can claim. The number of ranges and their locations in PCI address space is programmable, and determined by the "PIM" and "BAR" registers. Also, the PLB address may be a translation of the PCI address. The translation is programmable and is also determined by the "PIM" registers. See section ~~"PCI Inbound Map (PIM) Configuration" on page 13~~*PCI Inbound Map (PIM) Configuration on page 547* for more details.

~~Table 18-2~~*Table 18-2* shows the PCI address maps from the view of PCI, that is, as decoded by the PLB-PCIX Bridge as a PCI target.

Table 18-2. Inbound Address Map

PCI Memory Address	PCI I/O Address	Description	PLB Address
0000_0000_0000_0000h FFFF_FFFF_FFFF_FFFFh		PCI memory addresses that fall within BAR 0 space are normally translated through PIM 0.	0000_0000_0000_0000h FFFF_FFFF_FFFF_FFFFh
	0000_0000h FFFF_FFFFh	PCI I/O addresses that fall within BAR 1 space are translated through PIM 1 to generate the PLB address. The size of PIM 1 is hard-coded to 256 bytes.	0000_0000_0000_0000h FFFF_FFFF_FFFF_FFFFh
0000_0000_0000_0000h FFFF_FFFF_FFFF_FFFFh or 0000_0000h FFFF_FFFFh		PCI memory addresses that fall within BAR 2 space or Option ROM BAR Space are translated through PIM 2 to produce a memory address mappable anywhere from 0 to 2^{64} bytes in PLB space.	0000_0000_0000_0000h FFFF_FFFF_FFFF_FFFFh

~~Figure 18-5 on page 13~~Figure 18-5 shows the detail of the PIM/BAR register sets used to map PCI memory regions to PLB address space.

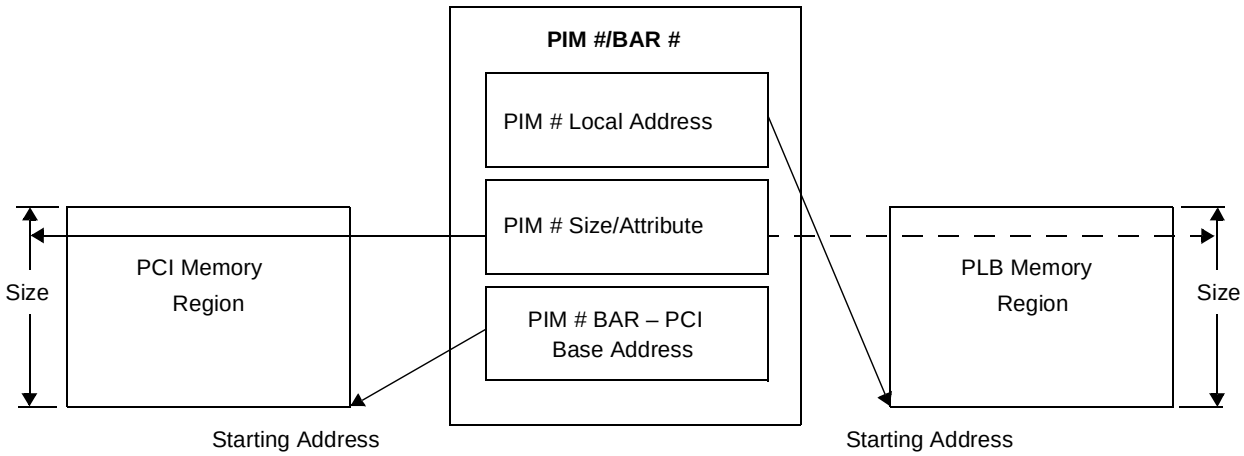


Figure 18-5. PIM Register Sets Map PCI Address Space to PLB Address Space

18.9.4 PCI Inbound Map (PIM) Configuration

The PLB-PCIX Bridge has three ranges in PCI space that are mapped to PLB space. These three ranges are referred to as PIM 0, PIM 1 and PIM 2. PIM 0 and PIM 2 map PCI memory space to PLB memory space. PIM 1 maps PCI I/O space to PLB memory space. The characteristics of each PIM are defined by a set of registers in the PLB-PCIX Bridge configuration registers.

PIM 0 is controlled by the following PCI configuration registers:

- PIM 0 ~~Memory Size~~ Size/Attribute Register PIM 0 Local Low Address Register
- PIM 0 Local High Address Register
- BAR 0 Low Register
- BAR 0 High Register

PIM 1 is controlled by the following PCI configuration registers:

- PIM 1 ~~Memory Size~~ Size/Attribute Register
- PIM 1 Local Low Address Register
- PIM 1 Local High Address Register
- BAR 1 Register

PIM 2 is controlled by the following PCI configuration registers:

- PIM 2 ~~Memory Size~~ Size/Attribute Register PIM 2 Local Low Address Register
- PIM 2 Local High Address Register
- BAR 2 Low Register
- BAR 2 High Register

The location in PCI space of each PIM is programmable using the BAR registers.

The range of PLB space that is accessed through each PIM is also programmable. This allows address translation between the two busses. The PLB address is defined in the Local Low/High Address registers.

The size of PIM 0 and PIM 2 is programmable using the Memory Size registers. The size is a power of two, and is a minimum of 4 KB and a maximum of ~~2^{64} bytes~~4 GB. The PLB and PCI address spaces for each PIM are aligned to this size. The size of PIM 1 is fixed to 256 bytes.

PIM 0 1 and 2 are enabled or disabled with the enable bit in their size_attribute register. PIM 0 and PIM 2 can also be enabled or disabled by the Memory Space bit of the PCI Command Register. PIM 1 can be enabled or disabled by the I/O Space bit of the PCI Command Register. The address ranges and sizes of the PIMs should be initialized before they are enabled.

See Section ~~"PIM Registers" on page 84~~*PIM Registers* on page 616 for details on PCI PIM registers.

18.10 Data Flow and Buffer Usage

The PLB-PCIX Bridge contains dedicated buffers for holding inbound write data (2x256 bytes), inbound read data (1 or 2x256 bytes), outbound write ~~data~~data (2x256 bytes), and outbound read data (1x256 bytes + 1x512 bytes). The buffers are implemented as compilable register arrays. The detailed description of their use is provided as follows.

18.10.1 Inbound Write Post Buffer

There are two 256-byte buffers to handle inbound writes. Each buffer has its own associated address; the second buffer can be filling with a second write while the first buffer is being drained by the PLB master of the PLB-PCIX Bridge. The PLB transaction starts when one of the buffers becomes full or the write has completed on the PCI side for that buffer.

In PCI-Conv mode, the PCI slave of the PLB-PCIX Bridge disconnects the PCI transaction when the current buffer is full on 32-byte boundaries. This means that the PCI transaction is disconnected when there is less than 32 bytes remaining in the buffer and a 32-byte address boundary is about to be crossed. This allows the master to continue the write with a memory write with invalidate if the system cache line size is 32 bytes. However, if the system cache line size is greater than 32 bytes, then the master may be forced to continue with a standard memory write transaction.

In PCI-X mode, if the total aligned (see note) byte count is less than or equal to the buffer size, then the transaction is never disconnected, and runs to completion. If the total aligned byte count is greater than buffer size, then the transaction fills the current buffer with the maximum whole number of Address Disconnected Boundaries (ADBs) that will fit, at which point the transaction is disconnected.

Note: The buffer space is organized as 128-bit words. Thus, the maximum length that one buffer can accommodate actually depends on the alignment of the starting address. For example, for 256 byte buffers, if the transfer is 128-bit aligned, then 256 bytes can fit. If the address is 128-bit + 1byte aligned, then only 255 bytes can fit, and so forth. Sizes of 241 bytes or less always fit.

18.10.2 Outbound Write Post Buffer

There are two buffers to handle outbound writes. Each buffer has its own associated address; the second buffer can be filling with a second write while the first buffer is being drained by the PCI master of the PLB-PCIX Bridge. The PCI transaction begins when one of the buffers becomes full or when the write has been completed on the PLB side for that buffer.

It is recommended for better performance that PLB masters align their transactions to 64-byte boundaries at the beginning of a transfer. This means that if a master starts a burst that is 64-byte unaligned, it should first run a short transfer that ends at the 64-byte boundary and then start a new transaction that is 64-byte aligned to finish the original transfer. The reason is that in PCI-Conv mode, cache line aligned transactions whose total byte length is a multiple of the cache line size are run as memory write with invalidate commands instead of standard memory write transactions. Memory write with invalidate avoids potential snoop push transactions that are unnecessary when entire cache lines are written. Though there is no memory write with invalidate command in PCI-X mode, the host bridge can similarly avoid the snoop pushes when memory write block commands are done cache line aligned and the total byte length is a multiple of the cache line size.

It is not necessary for PLB masters to end their transactions on 64-byte boundaries. The PLB slave of the PLB-PCIX Bridge disconnects when the current buffer is full on 64-byte boundaries. This means that a burst write is target disconnected on a 64-byte boundary if the burst write starts 64-byte aligned and the transfer length is not an integer multiple of 64 bytes.

For example, if a (64-byte aligned) 200-byte transfer is being requested, there will be a disconnect at 192 bytes forcing the remaining 8 bytes to be done as a separate transaction. This allows the 192-byte portion of the write to be run on the PCI bus as a memory write with invalidate transaction in conventional PCI mode. If the transfer length is less than or equal to 64-bytes or an integer multiple of 64 bytes, then the entire transfer is taken up to the size of the buffer.

If a PLB write starts unaligned, then the entire byte count is taken up to the size of the buffer, and the transfer is run as a standard PCI memory write. Since transactions that start unaligned cannot be run as memory write with invalidate transactions, they are not target disconnected at 64-byte boundaries. Aligned transactions whose byte length is less than 64-bytes are always run as standard memory write transactions.

18.10.2.1 Outbound Connected Writes

I/O writes, external configuration writes and special cycle writes are considered connected writes and are handled as delayed writes on the PLB. When a buffer is available, these writes are rearbitrated and dispatched to the PCI side. The PLB request is repeatedly rearbitrated until the transfer completes on the PCI side, at which point it is accepted. While the PLB request is repeatedly rearbitrated, all other outbound reads and writes are rearbitrated except outbound reads that hit in the prefetch buffer that are completed.

18.10.3 Inbound Read Buffer

PCI-Conv mode: There is one 256-byte buffer available to inbound reads in conventional PCI mode. PCI reads are handled as delayed reads. If the delayed read is to non-prefetchable memory, the PCI cycle is disconnected after one beat, regardless of command type (READ, READ LINE, READ MULT), and a single-beat read of the same size is run on the PLB.

If the delayed read is to prefetchable memory, a 32-byte, 64-byte or a 256-byte fixed length burst is run on the PLB to fill the inbound read buffer. A read line transaction fetches 64-bytes. A read multiple transaction fetches 256-bytes. A read transaction can be configured to do a 32-byte, a 64-byte or a 256-byte fetch, based on settings in the PCI target memory read command Interpretation bits in the Bridge Options 1 configuration register. The address for this prefetch read is truncated to ~~64~~128-bit alignment so that the transaction end is ~~64~~128-bit aligned.

Only one inbound read is permitted to be in progress at a time, so reads to other addresses are retried until the original read transaction has completed. The delayed read is accepted as soon as there is any data in the buffer. If the PLB side fills as fast or faster than the PCI side drains, then the burst can be sustained. If the PLB side fills slower than the PCI side drains, then the buffer may go empty. When the buffer goes empty, then one wait state is inserted on the PCI to wait for PLB data. If the buffer is still empty after one PCI wait state, then the PCI side is disconnected to allow time for the buffer to fill some more.

After a delayed read completion, data in the PCI inbound read buffer is considered prefetched data. This buffer is treated as a First-In First-Out (FIFO). Random access to the buffer by PCI masters is not supported. As the PCI master reads data from the buffer, the current top of FIFO pointer moves. Data is returned for new reads only if the address of the read transaction exactly matches the current top of FIFO pointer. Otherwise, the read is considered a new read, the FIFO data is discarded, and a new read is transacted to fill the FIFO. A transfer that is bursting out of the buffer is disconnected when it runs out of data in the buffer; there is no automatic refetch based on such a disconnect. The PCI inbound read buffer is marked invalid (flushed) by outbound writes or inbound writes to an address stored in the inbound read buffer as per PCI transaction ordering rules.

Note: Delayed read "hit" is determined by an address match and command of any memory read type (memory read, memory read line, and memory read multiple). The command does not need to exactly match. Also, the byte enables do not need to match, even if the region is non-prefetchable. This is acceptable assuming that no two (or more) different PLB slaves are mapped to the same word of PLB address space (but different bytes).

PCI-X mode: There are two 256-byte buffers for inbound reads in PCI-X mode. All inbound reads are handled as a split transaction. If the transfer length is less than an aligned (see note below) 256, then the entire transfer is read into the first buffer and returned as a split completion. If the transfer length is greater than an aligned 256, then the first buffer is filled to the second Address Disconnected Boundary (ADB) and that portion is returned as a partial split completion. The second buffer is then filled to the end of the buffer (or the end of the byte count) and returned as a partial split completion. As long as there is still more to do to complete the byte count, one buffer fills while the other buffer is being drained by a partial split completion transaction. Each partial split completion transaction in the sequence is 256-bytes except possibly for the first and last transaction.

While data for a first split transactions is being written out to the PCI bus, a second split transaction can be loading from the PLB, as long as the first transaction has completed reading from the PLB. Also, while the second split transactions is read from the PLB, a third split transaction can be accepted from the PCI bus. Thus, a maximum of three inbound read split transactions can be pending at one time, however, they are all handled serially.

Note: The 256-byte buffer space is organized as sixteen128-bit words. Thus, the maximum length that one buffer can accommodate actually depends on the alignment of the starting address. If the transfer is 128-bit aligned, then 256 bytes can fit. If the address is 128-bit+1byte aligned, then only 255 bytes can fit, and so forth. Sizes of 241 or less always fit.

18.10.4 Outbound Read Buffer

There is one general purpose buffer and one special purpose buffer used to hold outbound read data. The general purpose buffer and is used by all PLB masters except DMA. In PCI-Conv mode, one outstanding outbound read at a time is supported. In PCI-X mode, up to two outbound reads can be in progress at a time, provided the first read receives a split response and is from a different buffer.

The special purpose buffer is identified by the PLB master ID and is mapped using the Out Read Special Buffering Mapping register.

The general purpose buffer is a random access prefetch buffer, that is, any read to the general buffer with an address located anywhere within the buffer results in a buffer hit. On buffer hits, data is returned to the PLB master from the buffer and no PCI transaction is initiated.

The processor and other PLB masters have the ability to execute "guarded" reads. A guarded read has the PLB guarded attribute set.

Guarded reads to the general purpose buffer that are not buffer hits cause the exact amount of data requested (up to 256 bytes) to be fetched. Fixed length bursts of length greater than 256 bytes cause 256 bytes to be fetched. After the data is returned to the PLB master, the buffer is invalidated. Thus, any guarded data that was read from PCI can only be hit one time.

Non-guarded reads to the general purpose buffer that are not buffer hits cause 64 bytes to be fetched if the PLB transfer is for 64 bytes or less, or cause the exact amount (up to 256 bytes) if the PLB transfer is for more than 64 bytes. The data remains in the buffer until some event causes it to be invalidated.

The general purpose buffer is marked invalid (flushed) by writes in either direction or an outbound read that misses this buffer as per PCI transaction ordering rules.

Note: Outbound reads or writes by masters owning special purpose buffers do not invalidate the general purpose buffer.

One PLB master device can be assigned its own dedicated, special purpose read buffer. The PLB Master ID is used to determine which buffer to use for which transaction. In the 440GP, the firmware must map the master ID of the DMA controller to this buffer.

When a non-guarded outbound read from this special PLB master occurs, a 512-byte PCI transfer is issued to fill the buffer. Data is returned to the PLB master when the amount of data being requested exists in the buffer. In other words, fixed length PLB read requests are compared to the amount of data available in the buffer associated with the master running the transaction. If there is enough data in the buffer to satisfy the transaction, then data is returned; otherwise, the transaction is retried until the requested number of bytes exist in the buffer. The only exception is if the requested number of bytes extends beyond the length of the end of the buffer so that even when the PCI read completes, there will not be enough data in the buffer to satisfy the request. In this case, the transaction is accepted when the PCI transaction fills to the end of the buffer. Then, the amount of data in the filled-up buffer is returned and the PLB transaction is target disconnected when the end of the buffer is reached.

The special purpose read buffer is treated as a FIFO so that random access to it by PLB masters is not supported. As the PLB master reads data from the buffer, the current top of FIFO pointer moves. Data is returned for new reads from the same master only if the address of the read transaction exactly matches the current top of FIFO pointer. Otherwise, the read is considered a new read; the FIFO data is discarded and a new read is transacted to fill the FIFO.

The guarded attribute controls how data is fetched from the PCI bus. A non-guarded read, when the FIFO is empty, fetches 512-bytes into the buffer, starting with the address of the read. A guarded read, when the FIFO is empty, fetches the length indicated by the PLB master (fixed burst length) from the PCI, up to a maximum of 256 bytes. If the length is greater than 256 bytes, then 256 bytes are fetched. If the FIFO is not empty, a non-guarded read that has a byte count which will completely drain the buffer causes the next sequential 512-byte read to be issued automatically. This read transaction is issued immediately so it can start filling the FIFO as the remaining data is being read out by the PLB master. The PLB read is guaranteed to complete before the read data for the new transaction overwrites it, provided the PLB master does not underflow the read (disconnect prior to completing the requested byte count).

18.10.5 Outbound Connected Reads

I/O reads and external configuration reads are considered connected reads and are handled as delayed reads on the PLB. When a connected read is accepted, it is rearbitrated on the PLB side and dispatched to the PCI side. The PLB request is repeatedly rearbitrated until the transfer completes on the PCI side, at which point, it is accepted. All other outbound reads and writes are rearbitrated during this time.

-

18.11 Message Passing

18.11.1 Simple Message Passing Mechanism

The simple message passing mechanism enables messages to be exchanged between a PCI master device (the host) and a PLB master device (the local CPU).

The interface consists of a 32-bit Message In register and a 32-bit Message Out register. For 64-bit messages, the Message In High and Message Out High registers may be used.

Note: The message passing registers are 32-bit registers; thus, Message In/Out High cannot be written on the same cycle that Message In/Out (low) is written. Instead, two 32-bit writes are required.

18.11.1.1 Inbound Messages

The Message In register is written with "message" by the host, and may generate an interrupt to the local CPU. The local CPU can read the message. When processing of the message is completed, the local CPU clears the Message In register by writing to it. The host recognizes completion of the message by polling it (reading it) and waiting for it to be cleared, or by receiving a completion message (using Message Out).

The Message In register generates an interrupt to the local CPU whenever bit 0 of the Message In register is high. Thus, messages that cause an interrupt must have bit 0 set. There are no requirements on any of the other bits. Clearing of the message by the local CPU must include clearing bit 0 and may optionally include changes to other bits. If a 64-bit message is sent, then the host must write the Message In High register before writing the Message In register to ensure the interrupt does not reach the local CPU before Message In High is written.

18.11.1.2 Outbound Messages

The Message Out register is written with "message" by the local CPU, and may generate an interrupt to the host using the PCI bus. The interrupt may be an Outbound Message Signal Interrupts (MSI). The host can read the message. When processing of the message is completed, the host clears the Message Out register by writing to it. The local CPU recognizes completion of the message by polling it (reading it) and waiting for it to be cleared, or by receiving a completion message (using Message In).

The Message Out register generates an interrupt to the PCI bus whenever bit 0 of the Message Out register is high. Thus, messages that cause an interrupt must have bit 0 set. There are no requirements on any of the other bits. Clearing of the message by the host must include clearing bit 0 and may optionally include changes to other bits. If a 64-bit message is sent, then the local CPU must write the Message Out High register, before writing the Message Out register, to ensure the interrupt doesn't reach the host before Message Out High is written.

Messages sent out, when Outbound MSI is enabled, ensure that any write cycles posted prior to the Message Out are cleared out before the host receives the Message Out interrupt.

18.11.1.3 Access to Messaging Registers

The local CPU accesses the Message In and Message Out registers using a PLB transfer to the register set of the PLB-PCIX Bridge, starting at PLB address ~~0000_0002~~ 2_0EC8_0100h (see Section ~~"Register Descriptions" on page 32~~*Register Descriptions* on page 565 for more details).

In order for the host to access the Message In and Message Out registers, the "Split BAR0 Enable" bit of the PIM0 Size/Attribute register must be set. The host then accesses the Message In and Message Out registers using PCI memory transfers to BAR0 (see section ~~"Register Descriptions" on page 32~~*Register Descriptions* on page 565 for more details).

Note: This mode enables a split in the use of BAR0, as described in Section ~~"Inbound Interrupt Structure" on page 20~~*Inbound Interrupt Structure* on page 553, except that the first 1 KB of BAR0 is mapped to the Simple Message Passing and Inbound MSI registers, rather than to PLB space.

18.12 Interrupts and MSI

The PLB-PCIX Bridge can generate interrupts to the PCI (outbound interrupts) and receive interrupts from the PCI (inbound interrupts). These interrupts may be either standard interrupt signals or Message Signaled Interrupts (MSI). See *PCI*, Version 2.2, for details about MSI.

18.12.1 Outbound Interrupt Structure

The "Simple Message Passing" logic is the source of outbound interrupts and can be routed to the PCI using the PCI_INT(A)# pin, or using MSI. Outbound MSI is usually only used when the PLB-PCIX Bridge is in adapter mode.

18.12.2 MSI Capabilities Registers

The PLB-PCIX Bridge has a capabilities structure in the PCI configuration space that indicates that it is Outbound MSI capable. The capabilities structure includes the following registers:

- Cap_ID: read only value 05h: indicates the PLB-PCIX Bridge is MSI capable.
- Next_Item_Ptr: points to next item in the capabilities list.
- Message Control Register:
  - 64_Bit_Capable: read only value "1."
  - Multiple_Message_Enable: This field indicates how many interrupt messages are allocated to the PLB-PCIX Bridge by the system. It will be written by the system to a "0" or "1" indicating one or two messages respectively. The most significant two bits of this field are hardcoded to "0" to prevent errant system software from writing a value greater than "1" to this field.
  - Multiple_Message_Capable: read only value "1," indicating that the PLB-PCIX Bridge requests two messages from the system.
  - MSI_Enable: This read/write bit is used by the system to enable or disable MSI capability. A value of "1" means MSIs are enabled.
- Message Address: Contains address bits 0:31 of the address the PLB-PCIX Bridge should use to request MSI service. This field is configured by the system.
- Message Upper Address: Contains address bits 63:32 of the address the PLB-PCIX Bridge should use to request MSI service. This field is configured by the system.
- Message Data: This field is written by the system. When the PLB-PCIX Bridge issues an MSI message, it writes this value to the previously defined message address. When the PLB-PCIX Bridge is allocated two messages, the least significant bit of this field determines which message is being reported. If the PLB-PCIX Bridge is allocated only one message, then it cannot modify this data on an MSI write.
- Message End of Interrupt: This field is written following an MSI to allow a subsequent MSI to be sent.

18.12.3 Inbound Interrupt Structure

The PLB-PCIX Bridge can generate several interrupts to the local (PLB side) interrupt controller (inbound interrupts). The PLB-PCIX Bridge has several internal functions that can generate interrupts to the local interrupt controller. There is a unique local interrupt output for Simple Message Passing, PCI Command Register write, PCI Power Management and Error Handling. See Chapter 9 Universal Interrupt Controller for more information.

The PLB-PCIX Bridge also generates local interrupts based on interrupts received from the PCI bus. This feature is typically used only when the PLB-PCIX Bridge is in "host-bridge" mode. These interrupts arrive from Inbound MSI and are sent to the local interrupt controller.

When inbound MSI is enabled, one of 12 inbound interrupt signals is driven by the inbound MSI logic and is edge-sensitive (it is active for four PLB clocks). See Chapter 9 Universal Interrupt Controller for more information. Inbound MSI is enabled using the "Inbound MSI Enable" bit in the Bridge Options 1 register. Also, the "Split BAR0 Enable" bit of PIM0 Size/Attribute must be set. This mode enables a split in the use of BAR0, as described in the "Message Passing" section, where the first 1 KB of BAR0 is mapped to the Simple Message Passing and Inbound MSI registers, rather than to PLB space.

Inbound MSIs are Inbound PCI memory write cycles to address BAR0 + F8h. See section ~~"Address Mapping and Address Translation" on page 9~~*Address Mapping and Address Translation* on page 542 for more details. When a write to the PCI Inbound MSI register occurs, the inbound interrupt that corresponds to the data value is pulsed high.

There is no PCI capabilities structure associated with inbound MSIs.

18.13 PCI Power Management Interface

The PLB-PCIX Bridge is ~~compliant with~~ designed to the *PCI Bus Power Management Interface Specification,* Version 1.1. This section describes the implementation of this interface in the PLB-PCIX Bridge. This includes Capabilities and Power Management Status and Control Registers, Power State Control, and Changing of Power State.

18.13.1 Capabilities and Power Management Status and Control Registers

The PLB-PCIX Bridge has a capabilities structure in the PCI configuration space that indicates that the PLB-PCIX Bridge is PCI power management capable. The capabilities structure includes the following registers:

- Cap_ID, value 01h, indicates power management
- Next_Item_Ptr, points to next capabilities structure
- PMC, value 0202h, indicates no specific capabilities
- PMCSR, hold the current power state
- PMCSR_BSE, value 00h, unused, not PCI to PLB-PCIX Bridge
- Data, value 00h, not used

See section ~~"Register Descriptions" on page 32~~*Register Descriptions* on page 565 for details.

18.13.2 Power State Control

The current power management state is reported by reading the Power Management and Control Register (PMCSR). The PLB-PCIX Bridge supports states D0, D1, D3hot, and D3cold. State D2 is not supported. When the state is not D0, the PLB-PCIX Bridge is masked from being a master or a memory or I/O target on the PCI bus. It can still be a configuration target. Thus, accesses that are claimed by the PLB-PCIX Bridge when in state D0, are no longer claimed, resulting in master aborts on the PLB or PCI if such an access is attempted.

Note: This mask is independent of the state of the PCI Command Register.

18.13.3 Changing Power States

The PLB-PCIX Bridge has two registers that control changing of the power state. The host requests a change in the power state by writing to the Power Management and Control Register (PMCSR). The other register is the PM_SCRR (Power Management State Change Request Register) that provides a method of informing the local processor of a state change request and, thereby, preventing completion of the write to the PMCSR until the local processor indicates it is ready for the state change.

Power state changes are handled as follows:

- If host write to PMCSR requests a change from D3hot to D0, the write is accepted, then the PLB-PCIX Bridge causes the entire 440GP to be reset.
- All other change requests are handled in the following sequence:

1. The host requests a new power state by a PCI write to the PMCSR.
2. The host PCI write is retried unless the PM_SCRR "Delayed Write Enable" bit is off.
3. The host PCI write (retried or not) causes the PM_SCRR "State Change Request" bit to be set that drives an interrupt to the local processor.
4. The local processor recognizes the interrupt. The local processor can check the PM_SCRR "State Change Request" bit and the PM_SCRR "Requested State" bits to determine the nature of the request.
5. The local processor proceeds to power down those elements on the SOC and outside the SOC (on the adapter) that can be powered down if the requested state is valid.

Note: The PLB-PCIX Bridge cannot be powered down or put to sleep based on the PCI power management state.

6. When the subsystem has been powered down, and is ready to change state, the local processor clears the PM_SCRR "State Change Request" bit and sets the PM_SCRR "Accept PMCSR Write" bit.
7. When the host PCI write re-occurs, the following takes place:
    1. The host PCI write is accepted.
    2. The PMCSR is updated only if the transition is valid.
    3. PM_SCRR "Accept PMCSR Write" bit is automatically cleared unless the "Delayed Write Enable" bit is off, in which case the "Accept PMCSR Write" is always active.
    4. The PLB-PCIX Bridge enters the new power state.

Note: Delayed write enable off is a "safety" mode that can prevent bus hangs if the PM interrupt is not being serviced.

See section ~~"Register Descriptions" on page 32~~*Register Descriptions* on page 565 for more details.

18.14 PCI Arbiter

The PLB-PCIX Bridge includes an optional PCI arbiter that is typically used only in host-bridge mode. This "internal" arbiter can be used with up to six external PCI masters (six REQ#/GNT# pairs) or can be disabled. When disabled, the PLB-PCIX Bridge has one REQ#/GNT# pair (REQ0/GNT0) that attaches to an "external" arbiter. The PCI Arbiter Enable (PAE) serial ROM bit determines if the internal arbiter is enabled or not.

The same arbiter is used for both PCI-Conv and PCI-X modes, except that PCI-X inputs are registered before driven to the arbiter.

The arbiter's algorithm is set as follows:

> The arbiter keeps a set of "priorities" for each master. If a master loses an arbitration sequence, the losing master's priority increases and the winning master's priority decreases. Fairness is ensured in such a way that the master that keeps losing arbitration cycles eventually has the highest priority and is, therefore, guaranteed to win.

While the PCI bus is busy (not idle), the arbiter grants to the highest priority requester. The arbiter continually recalculates, such that if a new, higher-priority request arrives, the current grant is removed and the new requester receives a grant.

When the PCI bus is idle, the arbiter only recalculates grants if the granted REQ# is deasserted or the REQ#s go from none-active to one (or more) active. Thus, when the PCI bus is idle, the arbiter never removes GNT# from a device that has its REQ# active.

The arbiter does not check for a "broken requester." Thus, if a device continually asserts REQ# and does not run a cycle, the arbiter is hung asserting GNT# to it.

The arbiter has the following two parking modes:

- Park on PLB-PCIX Bridge
- Park on most-recently-active master

The mode is set by the "PCI Arbiter Park Mode" bit in the "Bridge Options 1" register.

Note: If the bus is non-idle, the arbiter always parks on the last granted master, regardless of the parking mode. This is done to avoid unnecessary master latency timeouts.

During reset, the PLB-PCIX Bridge ~~HIgh~~High-Zs all PCI outputs, including the GNT#s.

During reset (both System and PCI), ~~All~~ all grant lines are High-Z.

Internal Arbiter Disabled (~~default~~ default). CPC0_STRP1[PAE] ~~=~~= 0 –(bit11).

- PCIX_REQ0 -= ~~output~~ output, request from internal PLB-PCIX Bridge for bus ~~ownership~~ ownership.
- PCIX_REQ1 -= not ~~used~~ used, available for IRQ11 input ~~use~~ use.
- PCIX_REQ[2:5] -= not ~~used~~ used.

- _PCIX_GNT0 -= input, ~~acknoledge~~ acknowledge of bus grant to internal PLB-PCIX ~~Bridge~~ Bridge.
- PCIX_GNT1 -= not ~~used~~used - High-Z, available for IRQ12 input ~~use P~~use.
- ~~CIX_GNT~~PCIX_GNT[2:5] -= not used - High-~~Z~~ Z.

Internal Arbiter Enabled. [PAE] =1 set by IIC serial ROM bit or register ~~override~~ override.

- PCIX_REQ0 -= input request from external PCIX master[0~~]~~].
- PCIX_REQ1 -= input request from external PCIX master[1]. –NOT available for IRQ12 input use, -disable -IRQ11 use in UIC ~~register~~ register.
- PCIX_REQ[2:5] -= input request from external PCIX master[2:5].

- PCIX_GNT0 -= output, -grant signal to external PCIX master[0~~]~~].

- PCIX_GNT1 -= output, -grant signal to external PCIX master[0]1]. –NOT available for IRQ12 ~~input~~ input, use, -disable IRQ12 use in UIC ~~register~~ register.
- PCIX_GNT[2:5] -= output, -grant signal to external PCIX master[0]2:5].

18.15 Initialization Sequence for PCI-X

The PLB-PCIX Bridge is capable of providing the initialization sequence for PCI-X bus. This is typically only enabled when in host-bridge mode. This involves determining the mode (PCI-X versus PCI-Conv) and frequency range and then driving the PCI control signals (FRAME#, IRDY#, TRDY#, STOP# and DEVSEL#) during reset (PCI_RST_OUT#) to indicate this information. This function is enabled by strapping the bootstrap ROM PCI Initial Sequence Enable (PISE) strapping bit high.

When disabled (typical of "adapter-bridge mode"), the PLB-PCIX Bridge determines the mode (PCI-X versus PCI-Conv) and the frequency range by reading the PCI control signals when reset deasserts.

When enabled (typical of "Host-Bridge mode"), the PLB-PCIX Bridge determines the mode (PCI-X versus PCI-Conv) by the value on the PCIXCAP input. The frequency range is selected based on the PCI_M66EN input, PCIXCAP input, and the PCIX Frequency Selection (PXFS) ~~boostrap~~ bootstrap ROM strapping bits, as follows:

- If PCI-Conv, then the frequency range is determined by PCI_M66EN.
- If PCI-X and PCIXCAP indicate that they are not 133 MHz capable, then the frequency range is 50 MHz to 66 MHz.
- If PCI-X and PCIXCAP indicates 133 MHz capable, then the frequency range is determined by the PCIX Frequency Selection (PXFS) ~~boostrap~~ bootstrap ROM inputs. This allows the fastest (100-133 MHz) or slower (50-66 MHz or 66-100 MHz) ranges to selected based on the number of devices connected to the bus.

The PXFS serial ROM straps determine PCI-X frequency as follows:

- 00 = 100 - 133 MHz
- 01 = 66 - 100 MHz
- 10 = 50 - 66 MHz
- 11 = Reserved

The values determined for the mode and the frequency range are used internally, and can be read from the PLB-PCIX Bridge Options 2 register.

18.16 Error Handling

18.16.1 Introduction

The PLB-PCIX Bridge controller supports the detection and reporting of several types of errors. The errors are reported to the PLB or the PCI and status information is saved in the configuration register set, so that error type determination can be done.

There are four methods of reporting an error:

- Assert PCI_SERR#
- Assert PCI_PERR#
- Generate a PLB MERR
- Generate an error ~~interupt~~ interrupt (The PLB-PCIX Bridge error interrupt will be referred to as ERROR_INT from now on).

There are four categories of errors:

- PLB MERR or PLB MIRQ received while a PLB master. This error can be reported to the local CPU using ERROR_INT, which is connected to UIC1 IRQ[27], or to the PCI using PCI_SERR#.
- PCI error detected or received while a PCI initiator. This error can be reported to the PLB master via PLB MERR. It can also be reported to the local CPU using ERROR_INT or to the PCI using PCI_SERR#.
- PCI error detected while a PCI slave. This error can be reported to the PCI initiator using PCI_PERR#. It can also be reported to the local CPU using ERROR_INT or to the PCI using PCI_SERR#.
- Asynchronous error detected. This error can be reported to the local CPU using ERROR_INT or to the PCI using PCI_SERR#.

Each error that can be detected has a mask associated with it. If the mask is set, then the detection of that error condition is disabled. There are also separate masks for the PCI_SERR#, PCI_PERR#, PLB MERR, and ERROR_INT, that prevent reporting of these types of ~~errors~~errors. These masks do not prevent the detection of the errors.

18.16.2 Error Types

The following are the error types:

- While a PLB master, receive PLB Read MERR or PLB Write MERR or PLB MIRQ.
- While a PCI Initiator, receive master abort.
- While a PCI Initiator, receive target abort.
- While a PCI Initiator, receive/detect a data parity error.
- While a PCI-X Initiator, receive a split transaction error.
- While a PCI target, detect a data parity error.
- While a PCI target, detect an address or attribute parity error.
- Async to busses, detect PCI-Conv delayed read discard timer expired.
- Async to busses, receive PCI_SERR#.

18.16.3 Error Descriptions

The following subsections describe in detail how these errors are handled, what actions are taken for each error, and how to reset a given error.

18.16.3.1 While a PLB Master, Receive PLB Read MERR or PLB Write MERR or PLB MIRQ

This error occurs when the PLB-PCIX Bridge runs an inbound transfer to the PLB, but the PLB slave asserts PLB Read MERR or PLB Write MERR or PLB MIRQ (referred to collectively as PLB MERR). The PLB-PCIX Bridge does not attempt to associate the PLB MERR with a specific PLB cycle.

If this error is detected, the PLB-PCIX Bridge still completes the transfer on the PLB and PCI busses.

The following register bits are involved in this error:

- If the MErr Receive Enable bit of the Error Enable register is set, then detection of this error is enabled. If this error occurs, then the following registers are updated:
  - The MErr Received bit of the Error Status register is set.
  - The PLB Slave Error Address Low and High registers capture superfluous data.
  - The PLB Slave Error Attribute Register captures superfluous data.

  These register hold their values, regardless of the detection of other errors, until the Error Status register is cleared.
- This error can be reported locally or to the PCI, as determined by the Route Error Locally bit of the Error Enable register. If PLB MERR is detected as an error and routed locally, and the ERROR_INT Assertion Enable bit of the Error Enable register is set, then ERROR_INT is asserted, and the ERROR_INT Signaled bit of the Error Status register is set.
- If PLB MERR is detected as an error and routed to the PCI, and the SERR# Enable bit of the PCI Command register is set, then PCI_SERR# is asserted, and the Signaled System Error bit of the PCI Status register is asserted.

18.16.3.2 While a PCI Initiator, Receive Master Abort

This error is detected when the PLB-PCIX Bridge runs an outbound transfer to the PCI bus, but no target responds with PCI_DEVSEL# within the required time-out window and error detection is enabled.

Note: An error is never detected if the master abort occurs on a special cycle or a configuration cycle.

If this error is detected on a read, the PLB-PCIX Bridge still completes the transfer on the PLB, but drives all ones on the read data bus. On writes, the data is discarded.

The following register bits are involved in this error:

- If the Master Abort Error Enable bit of the Error Enable Register is set, then detection of this error is enabled. If this error occurs, the following registers are updated:
  - The Master-Abort bit of the PCI Status Register is set.
  - The PLB Slave Error Address Low and High registers capture the PLB address.
  - The PLB Slave Error Attribute Register captures information about the PLB address.

  These registers hold their values, regardless of the detection of other errors, until the PCI Status Register is cleared.
- On reads and connected-tenure writes, if the MERR Assertion Enable bit of the Error Enable Register is set, then PLB read or write MERR is asserted on the failed data beat, and all subsequent data beats, and the MERR Signaled bit of the Error Status Register is set.

- This error can be reported locally or to the PCI, as determined by the Route Error Locally bit of the Error Enable Register. If master abort is detected as an error and routed locally and the ERROR_INT Assertion Enable bit of the Error Enable Register is set, then ERROR_INT is asserted, and the ERROR_INT Signaled bit of the Error Status Register is set.
- If master abort is detected as an error and routed to the PCI, and the SERR# Enable bit of the PCI Command Register is set, then PCI_SERR# is asserted, and the Signaled System Error bit of the PCI Status Register is asserted.

18.16.3.3 While a PCI Initiator, Receive Target Abort

This error is detected when the PLB-PCIX Bridge runs an outbound transfer to the PCI bus and the PCI target signals a target abort.

If this error is detected on a read, the PLB-PCIX Bridge still completes the transfer on the PLB, but the read data bus is all ones. On a write, the data is discarded.

Note: If target abort is received on a read, any data received before the target abort is put in the read buffer and may be passed to the PLB master.

The following register bits are involved in this error:

- If the Target Abort Error Enable bit of the Error Enable Register is set, then detection of this error is enabled. If this error occurs, the following registers are updated:
  - The Target-Abort bit of the PCI Status Register is set.
  - The PLB Slave Error Address Low and High registers capture the PLB address.
  - The PLB Slave Error Attribute Register captures information about the PLB address.

  These registers hold their values, regardless of the detection of other errors, until the Error Status Register is cleared.
- On reads and connected-tenure writes, if the MERR Assertion Enable bit of the Error Enable Register is set, then PLB read or write MERR is asserted on the failed data beat, and all subsequent data beats and the MERR Signaled bit of the Error Status Register is set.
- This error can be reported locally or to the PCI, as determined by the Route Error Locally bit of the Error Enable Register. If target abort is detected as an error and routed locally, and the ERROR_INT Assertion Enable bit of the Error Enable Register is set, then ERROR_INT is asserted, and the ERROR_INT Signaled bit of the Error Status Register is set.
- If target abort is detected as an error and routed to the PCI, and the SERR# Enable bit of the PCI Command Register is set, then PCI_SERR# is asserted, and the Signaled System Error bit of the PCI Status Register is asserted.

18.16.3.4 While a PCI Initiator, Receive/Detect Data Parity Error

This error is detected when the PLB-PCIX Bridge runs an outbound transfer to the PCI bus (including receiving a PCI-X split completion) and a parity error is detected on read data or PCI_PERR# is received during a write.

If this error is detected on a read, the PLB-PCIX Bridge still completes the transfer on the PLB and PCI busses.

The following register bits are involved in this error:

- If a data parity error is detected on a read, the Detected Parity Error bit of the PCI Status Register is set. Setting of this bit is non-maskable. It can be reset by writing a "1" to it.
- If the Parity Error Response bit of the PCI Command Register is set, then detection of this error is enabled. If this error occurs, the following registers are updated:
  - The Master Data Parity Error bit of the PCI Status Register is set.
  - The PLB Slave Error Address Low and High registers capture the PLB address.
  - The PLB Slave Error Attribute Register captures info about the PLB address.

  These registers hold their values, regardless of the detection of other errors, until the PCI Status Register is cleared.
- On reads, PCI_PERR# is asserted.
- On reads and connected-tenure writes, if the MERR Assertion Enable bit of the Error Enable Register is set, then PLB read or write MERR is asserted on the failed data beat, and all subsequent data beats, and the MERR Signaled bit of the Error Status Register is set. If the parity error is detected on the last one or two beats of a read and the PCI frequency is low, it may fail to be reported using MErr.
- This error can be reported locally or to the PCI, as determined by the Route Error Locally bit of the Error Enable Register. If a master data parity error is detected as an error and routed locally, and the ERROR_INT Assertion Enable bit of the Error Enable Register is set, then ERROR_INT is asserted, and the ERROR_INT Signaled bit of the Error Status Register is set.
- If this error is detected and routed to the PCI, and the SERR# Enable bit of the PCI Command Register is set, then PCI_SERR# is asserted, and the Signaled System Error bit of the PCI Status Register is asserted.

18.16.3.5 While a PCI-X Initiator, Receive Split Transaction Error

This error is detected when the PLB-PCIX Bridge runs an outbound transfer to the PCI-X bus and the PCI target responds with a split transaction error.

If this error is detected on a read, the PLB-PCIX Bridge still completes the transfer on the PLB, but the read data bus is undefined. On a write, the data is discarded.

Note: If this error is received on a read, any data received before the split transaction error is put in the read buffer and may be passed to the PLB master.

The following register bits are involved in this error:

- If the Split Transaction Error Enable bit of the Error Enable Register is set, then detection of this error is enabled. If this error occurs, the following registers are updated:
  - The Received Split Completion Error Message bit of the PCI-X Status Register is set.
  - The PLB Slave Error Address Low and High registers capture the PLB address.
  - The PLB Slave Error Attribute Register captures information about the PLB address.

  These registers hold their values, regardless of the detection of other errors, until the PCI-X Status Register bit is cleared.

- On reads and connected-tenure writes, if the MERR Assertion Enable bit of the Error Enable Register is set, then PLB read or write MERR is asserted on the failed data beat and all subsequent data beats, and the MERR Signaled bit of the Error Status Register is set.
- This error can be reported locally or to the PCI, as determined by the Route Error Locally bit of the Error Enable Register. If a split transaction error is detected as an error and routed locally, and the ERROR_INT Assertion Enable bit of the Error Enable Register is set, then ERROR_INT is asserted, and the ERROR_INT Signaled bit of the Error Status Register is set.
- If a split transaction error is detected as an error and routed to the PCI, and the SERR# Enable bit of the PCI Command Register is set, then PCI_SERR# is asserted, and the Signaled System Error bit of the PCI Status Register is asserted.

The PLB-PCIX Bridge never generates split transaction errors.

18.16.3.6 While a PCI Target, Detect Data Parity Error

This error is detected when the PLB-PCIX Bridge receives an inbound write from the PCI bus (not including receiving a PCI-X split completion), and a parity error is detected on the write data.

If this error is detected, the PLB-PCIX Bridge still completes the transfer on the PLB and PCI busses.

The following register bits are involved in this error:

- If a data parity error is detected, the Detected Parity Error bit of the PCI Status Register is set. Setting of this bit is non-maskable. It can be reset by writing a "1" to it.
- If the Parity Error Response bit of the PCI Command Register is set, PCI_PERR# is asserted.
- If both the Parity Error Response bits of the PCI Command Register, the Inbound Write Data Parity Enable bit, and the Error Enable Register are set, then detection of this error is enabled. If this error occurs, the following registers are updated:
  - The Inbound Write Data Parity Error Detected bit of the Error Status Register is set.
  - The PLB Slave Error Address Low and High registers capture superfluous data.
  - The PLB Slave Error Attribute Register captures superfluous data.

  These registers hold their values, regardless of the detection of other errors, until the PCI Status Register is cleared.
- This error can be reported locally or to the PCI, as determined by the Route Error Locally bit of the Error Enable Register. If a master data parity error is detected as an error and routed locally, and the ERROR_INT Assertion Enable bit of the Error Enable Register is set, then ERROR_INT is asserted, and the ERROR_INT Signaled bit of the Error Status Register is set.
- If this error is detected and routed to the PCI, and the SERR# Enable bit of the PCI Command Register is set, then PCI_SERR# is asserted, and the Signaled System Error bit of the PCI Status Register is asserted.

18.16.3.7 While a PCI Target, Detect Address or Attribute Parity Error

This error is detected when the PLB-PCIX Bridge receives an inbound transfer or a split completion (PCI-X only) from the PCI bus and a parity error is detected on the address or the attribute (PCI-X only).

The PCI transfer is completed normally (as if no error). The PLB transfer is completed normally (as if no error).

The following register bits are involved in this error:

- If an address or attribute parity error is detected, the Detected Parity Error bit of the PCI Status Register is set. Setting of this bit is non-maskable. It can be reset by writing a "1" to it.
- If the Parity Error Response bit of the PCI Command Register and the Address/Attribute Parity Error Enable bit of the Error Enable Register are set, then detection of this error is enabled. If this error occurs, the following registers are updated:
  - The Address/Attribute Parity Error Detected bit of the Error Status Register is set.
  - The PLB Slave Error Address Low and High registers capture superfluous data.
  - The PLB Slave Error Attribute Register captures superfluous data.

  These registers hold their values, regardless of the detection of other errors, until the Error Status Register is cleared.
- This error can be reported locally or to the PCI, as determined by the Route Error Locally bit of the Error Enable Register. If a master data parity error is detected as an error and routed locally, and the ERROR_INT Assertion Enable bit of the Error Enable Register is set, then ERROR_INT is asserted, and the ERROR_INT Signaled bit of the Error Status Register is set.
- If this error is detected and routed to the PCI, and the SERR# Enable bit of the PCI Command Register is set, then PCI_SERR# is asserted, and the Signaled System Error bit of the PCI Status Register is asserted.

18.16.3.8 Detect PCI-Conv Delayed Read Discard Timer Expired

This error is detected when the PLB-PCIX Bridge runs an inbound read as a delayed read, but the PCI Initiator never returns for the data, causing the Discard Timer to expire. This error only occurs in PCI-Conv mode.

This error is detected asynchronous to any bus activity.

The following register bits are involved in this error:

- If the Delayed Read Discard Timer Expired Error Enable bit of the Error Enable Register is set, then detection of this error is enabled. If this error occurs, the following registers are updated:
  - The Delayed Read Discard Timer Expired bit of the Error Status Register is set.
  - The PLB Slave Error Address Low and High registers capture superfluous data.
  - The PLB Slave Error Attribute Register captures superfluous data.

  These registers hold their values, regardless of the detection of other errors, until the Error Status Register is cleared.
- This error can be reported locally or to the PCI, as determined by the Route Error Locally bit of the Error Enable Register. If this error is detected and routed locally, and the ERROR_INT Assertion Enable bit of the Error Enable Register is set, then ERROR_INT is asserted, and the ERROR_INT Signaled bit of the Error Status Register is set.
- If this error is detected and routed to the PCI, and the SERR# Enable bit of the PCI Command Register is set, then PCI_SERR# is asserted, and the Signaled System Error bit of the PCI Status Register is asserted.

18.16.3.9 Received PCI_SERR#

This error is detected whenever PCI_SERR# is asserted. This allows the local CPU to be the handler for PCI_SERR#.

The following register bits are involved in this error:

- If the PCI_SERR# Received as Error Enable bit of the Error Enable Register is set, then detection of this error is enabled. If this error occurs, the following registers are updated:
  - The PCI_SERR# Received bit of the Error Status Register is set.
  - The PLB Slave Error Address Low and High registers capture superfluous data.
  - The PLB Slave Error Attribute Register captures superfluous data.

  These registers hold their values, regardless of the detection of other errors, until the Error Status Register is cleared.
- This error is always reported locally regardless of the state of the Route Error Locally bit of the Error Enable Register. If this error is detected, and the ERROR_INT Assertion Enable bit of the Error Enable Register is set, then ERROR_INT is asserted, and the ERROR_INT Signaled bit of the Error Status Register is set.

18.16.4 Read Prefetch Boundary Crossing

In general, it is the responsibility of PLB masters and PCI Initiators not to cross any physical boundaries of the slave devices they are accessing. However, on read cycles, the PLB-PCIX Bridge is capable of prefetching. Since the PLB-PCIX Bridge has no knowledge of the slave devices it is accessing, it is possible for the PLB-PCIX Bridge to generate a prefetching read that does cross a physical boundary of the slave device.

The PLB-PCIX Bridge and the PLB and PCI busses are designed such that read prefetch boundary crossing does not cause errors.

18.16.4.1 Inbound Read Prefetch Boundary Crossing Handling

In PCI-Conv mode, the PLB-PCIX Bridge disconnects inbound reads on the PCI bus at the boundary of its address space, as defined by the BAR registers. In response to inbound read requests, the PLB-PCIX Bridge may request a PLB prefetching read that may cross any boundary. It is the responsibility of the PLB slave to disconnect if the burst reaches one of its boundaries. In PCI-Conv mode, the PLB-PCIX Bridge does not attempt to complete a read that is disconnect by the PLB slave. This ensures that the PLB-PCIX Bridge never prefetches into an invalid PLB address when it starts reading from a valid PLB address.

In PCI-X mode, the PLB-PCIX Bridge reads the PCI-X specified byte count from the PLB. The PLB-PCIX Bridge resumes the read, if the PLB slave disconnects, until the full amount has been transferred. According to *PCI-X Addendum to the PCI Local Bus Specification,* Version 1.0, it is the responsibility of the initiator to ensure that no read request crosses a device boundary.

18.16.4.2 Outbound Address Space Boundary Handling

In PCI-Conv mode, the PLB-PCIX Bridge selects an amount to read from the PCI bus that may include some prefetching. If the PCI bus is disconnected, the PLB-PCIX Bridge resumes the read until completion only if the disconnect is not on a 4 KB boundary. Since PCI memory devices are aligned to 4 KB boundaries, this prevents prefetching into another devices memory space. It is the responsibility of the PCI Slave to disconnect, if a burst reaches its boundary.

In PCI-X mode, the PLB-PCIX Bridge selects an amount to read from the PCI bus that may include some prefetching. The PLB-PCIX Bridge resumes the read until completion if both of the following conditions are met:

- PCI bus is disconnected during an immediate response.
- The disconnect is not on a 4-KB boundary.

If the PCI target responds with a split completion, it is the responsibility of the PLB-PCIX Bridge to ensure that the request doesn't cross the boundaries of the slave. This cannot be done. Instead, the PCI target issues a split transaction error, if the read attempts to cross a boundary. If PLB-PCIX Bridge receives a split transaction error, it determines whether it occurred on prefetch data. If so, it discards the error without logging it.

18.17 Oddities

This section describes items of the PLB-PCIX Bridge which may be different from expected. This section should be read carefully.

18.17.1 Discontiguous and Irregular Byte Enable Handling

Outbound transactions cannot generate discontiguous byte enables because they are unsupported on the PLB.

The PLB-PCIX Bridge does not support discontiguous byte enables for inbound transactions, that is, byte enables that are inactive in between active byte enables. An inbound transaction with discontiguous byte enables completes normally on the PCI bus, and no errors are detected. However, the results are undefined (which bytes are actually accessed is not determined).

Inbound reads to a non-prefetchable region (always single beat) with no active byte enables result in a 1-byte read on the PLB.

18.17.2 PLB Arbiter Must Be Fair

In order to ensure that a PLB master that has a delayed read pending has a chance to receive its data, the PLB arbiter must be in FAIR mode and all PLB masters that access the PCI must drive the same PLB priority level as the PLB-PCIX Bridge.

18.18 Register Descriptions

The PLB-PCIX Bridge register set are the internal registers used for controlling the PLB-PCIX Bridge. These registers can be accessed from both the PLB and the PCI (if enabled). Most are accessed from the PCI by Configuration Type 0 cycles when the IDSEL input of PLB-PCIX Bridge is active. The Message Passing and Inbound MSI registers are accessed from the PCI by memory cycles to the address pointed to by the PCI BAR0 Register.

PLB masters access these registers directly in the 2_0EC8_0000 to 2_0EC8_01FF address range.

These registers must be accessed with single-beat, 1 to 4 byte transfers. Software must use appropriate masks to extract the desired bits when the register includes reserved bits. Writes must preserve the values of reserved bit positions by first reading the register, merging the new value, and writing the result.

18.18.1 Byte Ordering

The register set of the PLB-PCIX Bridge is described using little endian bit ordering. Thus, the least significant bit is bit "0" and is always shown on the right.

Furthermore, the PLB-PCIX Bridge hardware implements the registers in little endian byte ordering. Thus, software that runs in big endian mode must take this into account when accessing the registers.

For example, as shown in ~~Figure 18-6~~*Figure 18-6*, it is desirable to put a value in the 32-bit register.

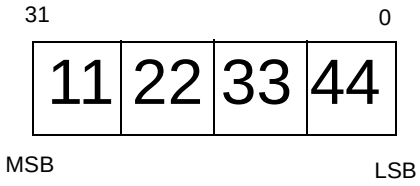


Figure 18-6. Little Endian Then, big endian software should either perform a store (STW) of the value 44332211 or do a byte-reversed store (STWBR) of the value 11223344. If the microprocessor supports little endian memory pages, then software can perform a store (STW) of the value 11223344 to a little endian page that maps to the configuration registers. In all these cases, the following will appear on the appropriate word of the PLB data bus (see ~~Figure 18-7~~*Figure 18-7*).

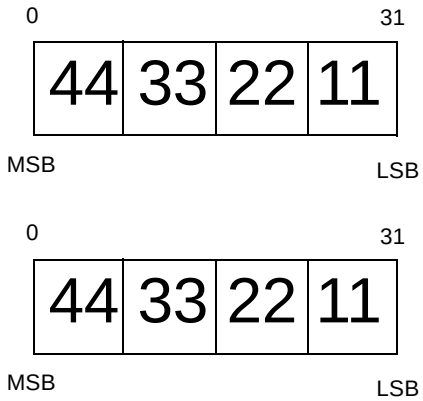


Figure 18-7. PLB Data Bus Value

18.18.2 Register Set

The two registers listed in Table 18-3 are used to access the configuration registers of external PCI-X devices only.

Table 18-3. External PCI-X Device Configuration Register Access

Register	Address	Access	Description
PCIX0_CFGADDR	0x2 0EC00000	R/W	PCI-X Configuration Address Register
PCIX0_CFGDATA	0x2 0EC00004	R/W	PCI-X Configuration Data Register

Registers listed in Table 18-4 can be accessed in two ways:

1. By the processor using the PLB address
2. By an external PCI-X device using configuration cycles to the proper offset.

Table 0-1. Internal PCI-X Configuration Registers

Register	PLB		PCI-X Configuration		Description
	Address	Access	Offset	Access	
PCIX0_VENDID	0x2 0EC80000	R/W	0x01 - 0x00	R	PCI-X Vendor ID
PCIX0_DEVID	0x2 0EC80002	R/W	0x03 - 0x02	R	PCI-X Device ID
PCIX0_CMD	0x2 0EC80004	R/W	0x05 - 0x04	R/W	PCI-X Command Register
PCIX0_STATUS	0x2 0EC80006	R/W	0x07 - 0x06	R/W	PCI-X Status Register
PCIX0_REVID	0x2 0EC80008	R/W	0x08	R	PCI-X Revision ID
PCIX0_CLS	0x2 0EC80009	R/W	0x0B - 0x09	R	PCI-X Class Register
PCIX0_CACHELS	0x2 0EC8000C	R/W	0x0C	R/W	PCI-X Cache Line Size
PCIX0_LATTIM	0x2 0EC8000D	R/W	0x0D	R/W	PCI-X Latency Timer
PCIX0_HDTYPE	0x2 0EC8000E	R	0x0E	R	PCI-X Header Type
PCIX0_BIST	0x2 0EC8000F	R	0x0F	R	PCI-X Built In Self Test Control
PCIX0_ BAR0L	0x2 0EC80010	R/W	0x13 - 0x10	R/W	PCI-X BAR 0 Low
PCIX0_ BAR0H	0x2 0EC80014	R/W	0x17 - 0x14	R/W	PCI-X BAR 0 High
PCIX0_BAR1	0x2 0EC80018	R/W	0x1B - 0x18	R/W	PCI-X BAR 1
PCIX0_ BAR2L	0x2 0EC8001C	R/W	0x1F - 0x1C	R/W	PCI-X BAR 2 Low
PCIX0_ BAR2H	0x2 0EC80020	R/W	0x23 - 0x20	R/W	PCI-X BAR 2 High
PCIX0_ BAR3	0x2 0EC80024	R	0x27 - 0x24	R	Unused BAR 3
PCIX0_CISPTR	0x2 0EC80028	R	0x2B - 0x28	R	Unused Cardbus CIS Pointer
PCIX0_SBSYSVID	0x2 0EC8002C	R/W	0x2D - 0x2C	R	PCI-X Subsystem Vendor ID
PCIX0_SBSYSID	0x2 0EC8002E	R/W	0x2F - 0x2E	R	PCI-X Subsystem ID
PCIX0_EROMBA	0x2 0EC80030	R/W	0x33 - 0x30	R/W	PCI-X Expansion ROM Base Address
PCIX0_CAP	0x2 0EC80034	R	0x34	R	PCI-X Capabilities Pointer
PCIX0_RES0	0x2 0EC80035	R	0x35	R	Reserved
PCIX0_RES1	0x2 0EC80036	R	0x37 - 0x6	R	Reserved
PCIX0_RES2	0x2 0EC80038	R	0x3B - 0x8	R	Reserved

Table 0-1. Internal PCI-X Configuration Registers (continued)

Register	PLB		PCI-X Configuration		Description
	Address	Access	Offset	Access	
PCIX0_INTLN	0x2 0EC8003C	R/W	0x3C	R/W	PCI-X Interrupt Line
PCIX0_INTPN	0x2 0EC8003D	R	0x3D	R	PCI-X Interrupt Pin
PCIX0_MINGNT	0x2 0EC8003E	R	0x3E	R	PCI-X Minimum Grant
PCIX0_MAXLTNCY	0x2 0EC8003F	R	0x3F	R	PCI-X Maximum Latency
PCIX0_BRDGOPT1	0x2 0EC80040	R/W	0x43 - 0x40	R/W	PCI-X Bridge Options 1
PCIX0_BRDGOPT2	0x2 0EC80044	R/W	0x47 - 0x44	R/W	PCI-X Bridge Options 2
PCIX0_ERREN	0x2 0EC80050	R/W	0x53 - 0x50	R/W	PCI-X Error Enable
PCIX0_ERRSTS	0x2 0EC80054	R/W	0x57 - 0x54	R/W	PCI-X Error Status
PCIX0_PLBBESR	0x2 0EC80058	R	0x5B - 0x58	R	PCI-X PLB Slave Error Syndrome Register
PCIX0_PLBBEARL	0x2 0EC8005C	R	0x5F - 0x5C	R	PCI-X PLB Slave Error Address Register Low
PCIX0_PLBBEARH	0x2 0EC80060	R	0x63 - 0x60	R	PCI-X PLB Slave Error Address Register High
PCIX0_POM0LAL	0x2 0EC80068	R/W	0x6B - 0x68	R/W	PCI-X POM0 Local Address Low
PCIX0_POM0LAH	0x2 0EC8006C	R/W	0x6F - 0x6C	R/W	PCI-X POM0 Local Address High
PCIX0_POM0SA	0x2 0EC80070	R/W	0x73 - 0x70	R/W	PCI-X POM0 Size Attribute
PCIX0_POM0PCIAL	0x2 0EC80074	R/W	0x77 - 0x74	R/W	PCI-X POM0 PCI Address Low
PCIX0_POM0PCIAH	0x2 0EC80078	R/W	0x7B - 0x78	R/W	PCI-X POM0 PCI Address High
PCIX0_POM1LAL	0x2 0EC8007C	R/W	0x7F - 0x7C	R/W	PCI-X POM1 Local Address Low
PCIX0_POM1LAH	0x2 0EC80080	R/W	0x83 - 0x80	R/W	PCI-X POM1 Local Address High
PCIX0_POM1SA	0x2 0EC80084	R/W	0x87 - 0x84	R/W	PCI-X POM1 Size Attribute
PCIX0_POM1PCIAL	0x2 0EC80088	R/W	0x8B - 0x88	R/W	PCI-X POM1 PCI Address Low
PCIX0_POM1PCIAH	0x2 0EC8008C	R/W	0x8F - 0x8C	R/W	PCI-X POM1 PCI Address High
PCIX0_POM2SA	0x2 0EC80090	R/W	0x93 - 0x90	R/W	PCI-X POM2 Size/Attribute
PCIX0_PIM0SA	0x2 0EC80098	R/W	0x9B - 0x98	R/W	PCI-X PIM0 Size/Attribute
PCIX0_PIM0LAL	0x2 0EC8009C	R/W	0x9F - 0x9C	R/W	PCI-X PIM0 Local Address Low
PCIX0_PIM0LAH	0x2 0EC800A0	R/W	0xA3 - 0xA0	R/W	PCI-X PIM0 Local Address High
PCIX0_PIM1SA	0x2 0EC800A4	R/W	0xA7 - 0xA4	R/W	PCI-X PIM1 Size/Attribute
PCIX0_PIM1LAL	0x2 0EC800A8	R/W	0xAB - 0xA8	R/W	PCI-X PIM1 Local Address Low
PCIX0_PIM1LAH	0x2 0EC800AC	R/W	0xAF - 0xAC	R/W	PCI-X PIM1 Local Address High
PCIX0_PIM2SA	0x2 0EC800B0	R/W	0xB3 - 0xB0	R/W	PCI-X PIM2 Size/Attribute
PCIX0_PIM2LAL	0x2 0EC800B4	R/W	0xB7 - 0xB4	R/W	PCI-X PIM2 Local Address Low
PCIX0_PIM2LAH	0x2 0EC800B8	R/W	0xBB - 0xB8	R/W	PCI-X PIM2 Local Address High
PCIX0_OMCAPID	0x2 0EC800C0	R	0xC0	R	PCI-X Outbound MSI Capability Identifier

Table 0-1. Internal PCI-X Configuration Registers (continued)

Register	PLB		PCI-X Configuration		Description
	Address	Access	Offset	Access	
PCIX0_OMNIPTR	0x2 0EC800C1	R/W	0xC1	R	PCI-X Outbound MSI Next Item Pointer
PCIX0_OMMC	0x2 0EC800C2	R/W	0xC3 - 0xC2	R/W	PCI-X Outbound MSI Message Control
PCIX0_OMMA	0x2 0EC800C4	R/W	0xC7 - 0xC4	R/W	PCI-X Outbound MSI Message Address
PCIX0_OMMUA	0x2 0EC800C8	R/W	0xCB - 0xC8	R/W	PCI-X Outbound MSI Message Upper Address
PCIX0_OMMDATA	0x2 0EC800CC	R/W	0xCD - 0xCC	R/W	PCI-X Outbound MSI Message Data
PCIX0_OMMEOI	0x2 0EC800CE	R/W	0xCE	R/W	PCI-X Outbound MSI Message End Of Interrupt
PCIX0_PMCAPID	0x2 0EC800D0	R/W	0xD0	R	PCI-X PMC Capability Identifier
PCIX0_PMNIPTR	0x2 0EC800D1	R/W	0xD1	R	PCI-X PMC Next Item Pointer
PCIX0_PMC	0x2 0EC800D2	R	0xD3 - 0xD2	R	PCI-X Power Management Capabilities
PCIX0_PMCSR	0x2 0EC800D4	R/W	0xD5 - 0xD4	R/W	PCI-X Power Management Control Status
PCIX0_PMCSRBSE	0x2 0EC800D6	R	0xD6	R	PCI-X PMCSR PCI to PCI Bridge Support Extensions
PCIX0_PMDATA	0x2 0EC800D7	R	0xD7	R	PCI-X PMC Data Register
PCIX0_PMSCRR	0x2 0EC800D8	R/W	0xD8	R/W	PCI-X Power Management State Change Request Register
PCIX0_CAPID	0x2 0EC800DC	R	0xDC	R	PCI-X Capability Identifier
PCIX0_NIPTR	0x2 0EC800DD	R	0xDD	R	PCI-X Next Item Pointer
PCIX0_PCIXCMD	0x2 0EC800DE	R/W	0xDF - 0xDE	R/W	PCI-X Command
PCIX0_STS	0x2 0EC800E0	R/W	0xE3 - 0xE0	R/W	PCI-X Status
PCIX0_IDR	0x2 0EC800E4	R/W	0xE7 - 0xE4	R/W	PCI-X Internal Debug Register
PCIX0_CID	0x2 0EC800E8	R	0xEB - 0xE8	R	PCI-X Internal Core Device ID
PCIX0_RID	0x2 0EC800EC	R	0xEF - 0xEC	R	PCI-X Internal Core Revision ID

Table 18-4. Internal PCI-X Configuration Registers

Register	PLB		PCI-X Configuration		Description
	Address	Access	Offset	Access	
PCIX0_VENDID	0x2 0EC80000	R/W	0x01 - 0x00	R	PCI-X Vendor ID
PCIX0_DEVID	0x2 0EC80002	R/W	0x03 - 0x02	R	PCI-X Device ID
PCIX0_CMD	0x2 0EC80004	R/W	0x05 - 0x04	R/W	PCI-X Command Register
PCIX0_STATUS	0x2 0EC80006	R/W	0x07 - 0x06	R/W	PCI-X Status Register
PCIX0_REVID	0x2 0EC80008	R/W	0x08	R	PCI-X Revision ID

Table 18-4. Internal PCI-X Configuration Registers (continued)

Register	PLB		PCI-X Configuration		Description
	Address	Access	Offset	Access	
PCIX0_CLS	0x2 0EC80009	R/W	0x0B - 0x09	R	PCI-X Class Register
PCIX0_CACHELS	0x2 0EC8000C	R/W	0x0C	R/W	PCI-X Cache Line Size
PCIX0_LATTIM	0x2 0EC8000D	R/W	0x0D	R/W	PCI-X Latency Timer
PCIX0_HDTYPE	0x2 0EC8000E	R	0x0E	R	PCI-X Header Type
PCIX0_BIST	0x2 0EC8000F	R	0x0F	R	PCI-X Built In Self Test Control
PCIX0_ BAR0L	0x2 0EC80010	R/W	0x13 - 0x10	R/W	PCI-X BAR 0 Low
PCIX0_ BAR0H	0x2 0EC80014	R/W	0x17 - 0x14	R/W	PCI-X BAR 0 High
PCIX0_BAR1	0x2 0EC80018	R/W	0x1B - 0x18	R/W	PCI-X BAR 1
PCIX0_ BAR2L	0x2 0EC8001C	R/W	0x1F - 0x1C	R/W	PCI-X BAR 2 Low
PCIX0_ BAR2H	0x2 0EC80020	R/W	0x23 - 0x20	R/W	PCI-X BAR 2 High
PCIX0_ BAR3	0x2 0EC80024	R	0x27 - 0x24	R	Unused BAR 3
PCIX0_CISPTR	0x2 0EC80028	R	0x2B - 0x28	R	Unused Cardbus CIS Pointer
PCIX0_SBSYSVID	0x2 0EC8002C	R/W	0x2D - 0x2C	R	PCI-X Subsystem Vendor ID
PCIX0_SBSYSID	0x2 0EC8002E	R/W	0x2F - 0x2E	R	PCI-X Subsystem ID
PCIX0_EROMBA	0x2 0EC80030	R/W	0x33 - 0x30	R/W	PCI-X Expansion ROM Base Address
PCIX0_CAP	0x2 0EC80034	R	0x34	R	PCI-X Capabilities Pointer
PCIX0_RES0	0x2 0EC80035	R	0x35	R	Reserved
PCIX0_RES1	0x2 0EC80036	R	0x37 - 0x6	R	Reserved
PCIX0_RES2	0x2 0EC80038	R	0x3B - 0x8	R	Reserved
PCIX0_INTLN	0x2 0EC8003C	R/W	0x3C	R/W	PCI-X Interrupt Line
PCIX0_INTPN	0x2 0EC8003D	R	0x3D	R	PCI-X Interrupt Pin
PCIX0_MINGNT	0x2 0EC8003E	R	0x3E	R	PCI-X Minimum Grant
PCIX0_MAXLTNCY	0x2 0EC8003F	R	0x3F	R	PCI-X Maximum Latency
PCIX0_BRDGOPT1	0x2 0EC80040	R/W	0x43 - 0x40	R/W	PCI-X Bridge Options 1
PCIX0_BRDGOPT2	0x2 0EC80044	R/W	0x47 - 0x44	R/W	PCI-X Bridge Options 2
PCIX0_ERREN	0x2 0EC80050	R/W	0x53 - 0x50	R/W	PCI-X Error Enable
PCIX0_ERRSTS	0x2 0EC80054	R/W	0x57 - 0x54	R/W	PCI-X Error Status
PCIX0_PLBBESR	0x2 0EC80058	R	0x5B - 0x58	R	PCI-X PLB Slave Error Syndrome Register
PCIX0_PLBBEARL	0x2 0EC8005C	R	0x5F - 0x5C	R	PCI-X PLB Slave Error Address Register Low
PCIX0_PLBBEARH	0x2 0EC80060	R	0x63 - 0x60	R	PCI-X PLB Slave Error Address Register High
PCIX0_POM0LAL	0x2 0EC80068	R/W	0x6B - 0x68	R/W	PCI-X POM0 Local Address Low
PCIX0_POM0LAH	0x2 0EC8006C	R/W	0x6F - 0x6C	R/W	PCI-X POM0 Local Address High
PCIX0_POM0SA	0x2 0EC80070	R/W	0x73 - 0x70	R/W	PCI-X POM0 Size Attribute
PCIX0_POM0PCIAL	0x2 0EC80074	R/W	0x77 - 0x74	R/W	PCI-X POM0 PCI Address Low

Table 18-4. Internal PCI-X Configuration Registers (continued)

Register	PLB		PCI-X Configuration		Description
	Address	Access	Offset	Access	
PCIX0_POM0PCIAH	0x2 0EC80078	R/W	0x7B - 0x78	R/W	PCI-X POM0 PCI Address High
PCIX0_POM1LAL	0x2 0EC8007C	R/W	0x7F - 0x7C	R/W	PCI-X POM1 Local Address Low
PCIX0_POM1LAH	0x2 0EC80080	R/W	0x83 - 0x80	R/W	PCI-X POM1 Local Address High
PCIX0_POM1SA	0x2 0EC80084	R/W	0x87 - 0x84	R/W	PCI-X POM1 Size Attribute
PCIX0_POM1PCIAL	0x2 0EC80088	R/W	0x8B - 0x88	R/W	PCI-X POM1 PCI Address Low
PCIX0_POM1PCIAH	0x2 0EC8008C	R/W	0x8F - 0x8C	R/W	PCI-X POM1 PCI Address High
PCIX0_POM2SA	0x2 0EC80090	R/W	0x93 - 0x90	R/W	PCI-X POM2 Size/Attribute
PCIX0_PIM0SA	0x2 0EC80098	R/W	0x9B - 0x98	R/W	PCI-X PIM0 Size/Attribute
PCIX0_PIM0LAL	0x2 0EC8009C	R/W	0x9F - 0x9C	R/W	PCI-X PIM0 Local Address Low
PCIX0_PIM0LAH	0x2 0EC800A0	R/W	0xA3 - 0xA0	R/W	PCI-X PIM0 Local Address High
PCIX0_PIM1SA	0x2 0EC800A4	R/W	0xA7 - 0xA4	R/W	PCI-X PIM1 Size/Attribute
PCIX0_PIM1LAL	0x2 0EC800A8	R/W	0xAB - 0xA8	R/W	PCI-X PIM1 Local Address Low
PCIX0_PIM1LAH	0x2 0EC800AC	R/W	0xAF - 0xAC	R/W	PCI-X PIM1 Local Address High
PCIX0_PIM2SA	0x2 0EC800B0	R/W	0xB3 - 0xB0	R/W	PCI-X PIM2 Size/Attribute
PCIX0_PIM2LAL	0x2 0EC800B4	R/W	0xB7 - 0xB4	R/W	PCI-X PIM2 Local Address Low
PCIX0_PIM2LAH	0x2 0EC800B8	R/W	0xBB - 0xB8	R/W	PCI-X PIM2 Local Address High
PCIX0_OMCAPID	0x2 0EC800C0	R	0xC0	R	PCI-X Outbound MSI Capability Identifier
PCIX0_OMNIPTR	0x2 0EC800C1	R/W	0xC1	R	PCI-X Outbound MSI Next Item Pointer
PCIX0_OMMC	0x2 0EC800C2	R/W	0xC3 - 0xC2	R/W	PCI-X Outbound MSI Message Control
PCIX0_OMMA	0x2 0EC800C4	R/W	0xC7 - 0xC4	R/W	PCI-X Outbound MSI Message Address
PCIX0_OMMUA	0x2 0EC800C8	R/W	0xCB - 0xC8	R/W	PCI-X Outbound MSI Message Upper Address
PCIX0_OMMDATA	0x2 0EC800CC	R/W	0xCD - 0xCC	R/W	PCI-X Outbound MSI Message Data
PCIX0_OMMEOI	0x2 0EC800CE	R/W	0xCE	R/W	PCI-X Outbound MSI Message End Of Interrupt
PCIX0_PMCAPID	0x2 0EC800D0	R/W	0xD0	R	PCI-X PMC Capability Identifier
PCIX0_PMNIPTR	0x2 0EC800D1	R/W	0xD1	R	PCI-X PMC Next Item Pointer
PCIX0_PMC	0x2 0EC800D2	R	0xD3 - 0xD2	R	PCI-X Power Management Capabilities
PCIX0_PMCSR	0x2 0EC800D4	R/W	0xD5 - 0xD4	R/W	PCI-X Power Management Control Status
PCIX0_PMCSRBSE	0x2 0EC800D6	R	0xD6	R	PCI-X PMCSR PCI to PCI Bridge Support Extensions
PCIX0_PMDATA	0x2 0EC800D7	R	0xD7	R	PCI-X PMC Data Register
PCIX0_PMSCRR	0x2 0EC800D8	R/W	0xD8	R/W	PCI-X Power Management State Change Request Register
PCIX0_CAPID	0x2 0EC800DC	R	0xDC	R	PCI-X Capability Identifier
PCIX0_NIPTR	0x2 0EC800DD	R	0xDD	R	PCI-X Next Item Pointer

Table 18-4. Internal PCI-X Configuration Registers (continued)

Register	PLB		PCI-X Configuration		Description
	Address	Access	Offset	Access	
PCIX0_PCIXCMD	0x2 0EC800DE	R/W	0xDF - 0xDE	R/W	PCI-X Command
PCIX0_STS	0x2 0EC800E0	R/W	0xE3 - 0xE0	R/W	PCI-X Status
PCIX0_IDR	0x2 0EC800E4	R/W	0xE7 - 0xE4	R/W	PCI-X Internal Debug Register
PCIX0_CID	0x2 0EC800E8	R	0xEB - 0xE8	R	PCI-X Internal Core Device ID
PCIX0_RID	0x2 0EC800EC	R	0xEF - 0xEC	R	PCI-X Internal Core Revision ID

Registers listed in ~~Table 18-5~~Table 18-4 can be accessed in two ways:

1. By the processor using the PLB address
2. By an external PCI-X device using memory cycles to the proper address.

Table 18-5. PCI-X Simple Message Passing and Inbound MSI Registers

Register	PLB		PCI-X Memory		Description
	Address	Access	Address	Access	
PCIX0_MSGIL	0x2 0EC80100	R/W	(PCIX0_BAR0H || PCIX0_BAR0L) + (0x03 - 0x00)	R/W	PCI-X Message In Low
PCIX0_MSGIH	0x2 0EC80104	R/W	(PCIX0_BAR0H || PCIX0_BAR0L) + (0x07 - 0x04)	R/W	PCI-X Message In High
PCIX0_MSGOL	0x2 0EC80108	R/W	(PCIX0_BAR0H || PCIX0_BAR0L) + (0x0B - 0x08)	R/W	PCI-X Message Out Low
PCIX0_MSGOH	0x2 0EC8010C	R/W	(PCIX0_BAR0H || PCIX0_BAR0L) + (0x0F - 0x0C)	R/W	PCI-X Message Out High
PCIX0_IM	0x2 0EC801F8	W	(PCIX0_BAR0H || PCIX0_BAR0L) + (0xFB - 0xF8)	W	PCI-X Inbound MSI

18.18.3 PCI-X Standard Header Registers

The following registers (offset 00h - 3Fh) are part of the standard, PCI-defined, configuration header.

18.18.3.1 PCI-X Vendor ID Register (PCIX0_VENDID)

The vendor ID register is a 16-bit register used to identify the manufacturer of the PCI device. The local CPU (PLB master) can write a different value to this register. ~~Figure 18-8~~Figure 18-8 describes PCIX0_VENDID register bits.

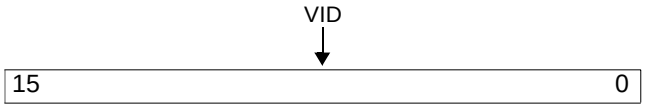


Figure 0-1. PCI-X Vendor ID Register (PCIX0_VENDID)

15:0	VID	Vendor ID

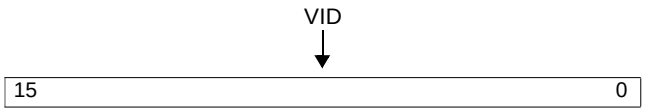


Figure 18-8. PCI-X Vendor ID Register (PCIX0_VENDID)

15:0	VID	Vendor ID

18.18.3.2 PCI-X Device ID Register (PCIX0_DEVID)

The device ID register is a 16-bit register used to identify the PCI device. The local CPU (PLB master) can write a different value to this register. ~~Figure 18-9~~Figure 18-9 describes PCIX0_DEVID register bits.

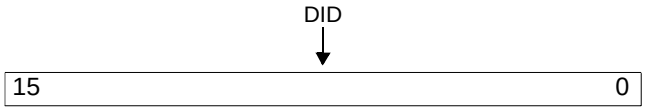


Figure 0-2. PCI-X Device ID Register (PCIX0_DEVID)

15:0	DID	Device ID

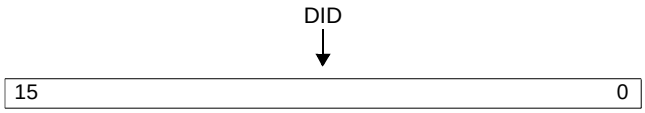


Figure 18-9. PCI-X Device ID Register (PCIX0_DEVID)

15:0	DID	Device ID

18.18.3.3 PCI-X Command Register (PCIX0_CMD)

The PCI command register is a 16-bit, read/write register used to control the operation of the PLB-PCIX Bridge on the PCI bus. ~~Figure 18-10~~Figure 18-10 describes PCIX0_CMD register bits.

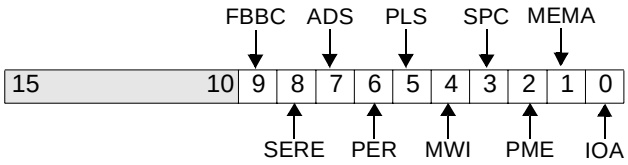


Figure 0-3. PCI-X Command Register (PCIX0_CMD)

15:10		Reserved	These bits are reserved, and return zeros when read.
9	FBBC	Fast Back-to-Back Capable Read-only; returns 0 when read.	Fast back-to-back write enable. This bit enables PCI masters to perform fast back-to-back transactions to different devices. The PLB-PCIX Bridge does not perform fast back-to-back transactions, therefore, this bit is read-only and returns zero when read.
8	SERE	SERR# Enable	Enable PCI_SERR#. Enables driving PCI_SERR# to report the detection of an error out to the PCI bus. If disabled, PCI_SERR# is never asserted regardless of the detection of any error.
7	ADS	Address Stepping	Address stepping wait states. PLB-PCIX bridge does not address step (except when generating a Configuration Type 0 cycle), therefore, this bit is read-only and returns zero when read.
6	PER	Parity Error Response	Parity error response. This bit enables the following types of PCI bus parity errors: PCI data bus parity errors while PCI is master PCI data bus parity errors while PCI is target PCI address bus parity errors When parity error response is disabled (set to 0), detection of these errors is masked and PERR# is not asserted, although parity is still generated.
5	PLS	Palette Snooping	Enable special palette snooping. The PLB-PCIX Bridge macro is not a VGA device, therefore, this bit is read-only and returns zero when read.
4	MWI	Memory Write and Invalidate	Enable memory write and invalidate command support. When high, PLB-PCIX Bridge is allowed to generate MWI transactions. Only used in PCI-Conv mode. This bit is 0 at reset.
3	SPC	Special Cycle	Enable special cycle operations. PLB-PCIX Bridge never monitors special cycles, therefore, this bit is read-only and returns zero when read.

2	PME	PCI Master Enable	Enables PLB-PCIX Bridge to master cycles on the PCI bus. When this bit is 0, PLB-PCI Bridge only responds as a PLB slave to accesses to the internal register set. However, the PLB-PCIX Bridge can still master transfers for split completions and to complete outbound writes that were posted prior to setting this bit to 0.
1	MEMA	Memory Access	Controls response of PLB-PCIX Bridge as a PCI memory target. A value of "1" enables PLB-PCIX Bridge to respond as a target. This bit is 0 (disabled) at reset.
0	IOA	I/O Access	Controls response of PLB-PCIX Bridge as a PCI I/O target. A value of 1 enables PLB-PCIX Bridge to respond as a target. This bit is 0 (disabled) at reset.

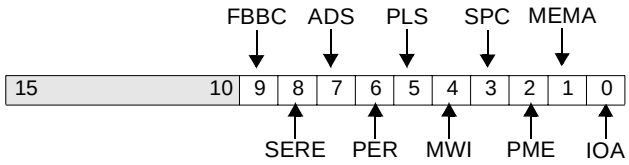


Figure 18-10. PCI-X Command Register (PCIX0_CMD)

15:10		Reserved	These bits are reserved, and return zeros when read.
9	FBBC	Fast Back-to-Back Capable Read-only; returns 0 when read.	Fast back-to-back write enable. This bit enables PCI masters to perform fast back-to-back transactions to different devices. The PLB-PCIX Bridge does not perform fast back-to-back transactions, therefore, this bit is read-only and returns zero when read.
8	SERE	SERR# Enable	Enable PCI_SERR#. Enables driving PCI_SERR# to report the detection of an error out to the PCI bus. If disabled, PCI_SERR# is never asserted regardless of the detection of any error.
7	ADS	Address Stepping	Address stepping wait states. PLB-PCIX bridge does not address step (except when generating a Configuration Type 0 cycle), therefore, this bit is read-only and returns zero when read.
6	PER	Parity Error Response	Parity error response. This bit enables the following types of PCI bus parity errors: PCI data bus parity errors while PCI is master PCI data bus parity errors while PCI is target PCI address bus parity errors When parity error response is disabled (set to 0), detection of these errors is masked and PERR# is not asserted, although parity is still generated.
5	PLS	Palette Snooping	Enable special palette snooping. The PLB-PCIX Bridge macro is not a VGA device, therefore, this bit is read-only and returns zero when read.
4	MWI	Memory Write and Invalidate	Enable memory write and invalidate command support. When high, PLB-PCIX Bridge is allowed to generate MWI transactions. Only used in PCI-Conv mode. This bit is 0 at reset.
3	SPC	Special Cycle	Enable special cycle operations. PLB-PCIX Bridge never monitors special cycles, therefore, this bit is read-only and returns zero when read.

2	PME	PCI Master Enable	Enables PLB-PCIX Bridge to master cycles on the PCI bus. When this bit is 0, PLB-PCI Bridge only responds as a PLB slave to accesses to the internal register set. However, the PLB-PCIX Bridge can still master transfers for split completions and to complete out-bound writes that were posted prior to setting this bit to 0.
1	MEMA	Memory Access	Controls response of PLB-PCIX Bridge as a PCI memory target. A value of "1" enables PLB-PCIX Bridge to respond as a target. This bit is 0 (disabled) at reset.
0	IOA	I/O Access	Controls response of PLB-PCIX Bridge as a PCI I/O target. A value of 1 enables PLB-PCIX Bridge to respond as a target. This bit is 0 (disabled) at reset.

18.18.3.4 PCI-X Status Register (PCIX0_STATUS)

The PCI Status Register is a 16-bit, read/bit-reset register used to record status information for PCI bus related events. Bits in this register are only set as a result of specific events occurring on the PCI bus. They are reset by writing a "1" to the desired bit location. Writing a "0" to a bit location leaves that bit unchanged. ~~Figure 18-11~~Figure 18-11 describes PCIX0_STATUS register bits.

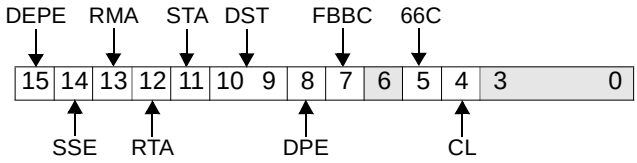


Figure 0-4. PCI-X Status Register (PCIX0_STATUS)

15	DPE	Detected Parity Error Write 1 to clear.	PLB-PCIX Bridge sets this bit whenever it detects a PCI bus parity error, regardless of the setting of any enable bits, that is, this bit is non-maskable. This bit is set when the following occurs: PCI address bus parity error detected when PLB-PCIX Bridge is a target. PCI data bus parity error detected when a PCI master writes to PLB memory (PLB-PCIX Bridge is the target). PCI data bus parity error detected when PLB-PCIX Bridge masters a PCI read cycle. Writing a 1 to this bit resets it to 0.
14	SSE	Signaled System Error Write 1 to clear.	Signaled system error. PLB-PCIX Bridge sets this bit if it asserts PCI_SERR#. Writing a 1 to this bit resets it to 0.
13	RMA	Received Master Abort Write 1 to clear.	This bit is set whenever PLB-PCIX Bridge terminates a PCI cycle for which it is the master with master abort (except configuration and special cycles) and the Master Abort Error Enable bit of the Error Enable Register is set. Writing a 1 to this bit resets it to 0.
12	RTA	Received Target Abort Write 1 to clear.	PLB-PCIX Bridge sets this bit whenever a PCI cycle for which it is the master is terminated with target abort and the Target Abort Error Enable bit of the Error Enable Register is set. Writing a 1 to this bit resets it to 0.
11	STA	Signaled Target Abort Write 1 to clear.	PLB-PCIX Bridge never signals a target abort. This bit is always 0.
10:9	DST	$\overline{\text{PCIDevSel}}$ Response Timing Read-only.	PCI_DEVSEL# response timing. PLB-PCIX Bridge asserts PCI_DEVSEL# on the second clock (also known as medium response time) after PCI_FRAME# is asserted by a PCI master attempting to access memory on the PLB side of the bridge. These bits are read-only and always return 01b when read.

8	DPE	Data Parity Error Detected Write 1 to clear.	This bit is set when the following two conditions are met: PLB-PCIX Bridge detects a data parity error (PCI_PERR# asserted) when it is the master on a PCI read cycle, or it is the master when it samples PCI_PERR# asserted on a PCI write cycle. The Parity Error Response bit (bit 6 of the PCI Command Register) is set. Writing a 1 to this bit resets it to 0.
7	FBBC	Fast Back-to-Back Capable Read-only; returns zero when read.	Indicates that the PCI target is capable of accepting fast back-to-back transactions when the transactions are not to the same agent. PLB-PCIX Bridge target does accept this type of fast back-to-back transaction, therefore, this bit is read-only and is 1 when in PCI-Conv mode. When in PCI-X mode, it is always 0.
6		Reserved	This bit is hardcoded to zero. This bit was the UDF support bit in PCI 2.1.
5	66C	66 MHz Capable	Indicates that the device is able to run at 66 MHz. Hardcoded to 1.
4	CL	Capabilities List	Indicates whether or not this device implements the pointer for a new capabilities linked list at offset 34h. This bit is read-only and returns 1 when read, indicating that the value read at offset 34h is a pointer in configuration space to a linked list of new capabilities.
3:0		Reserved	These bits are reserved, and return zeros when read.

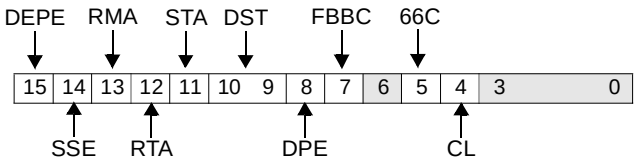


Figure 18-11. PCI-X Status Register (PCIX0_STATUS)

15	DPE	Detected Parity Error Write 1 to clear.	PLB-PCIX Bridge sets this bit whenever it detects a PCI bus parity error, regardless of the setting of any enable bits, that is, this bit is non-maskable. This bit is set when the following occurs: PCI address bus parity error detected when PLB-PCIX Bridge is a target. PCI data bus parity error detected when a PCI master writes to PLB memory (PLB-PCIX Bridge is the target). PCI data bus parity error detected when PLB-PCIX Bridge masters a PCI read cycle. Writing a 1 to this bit resets it to 0.
14	SSE	Signaled System Error Write 1 to clear.	Signaled system error. PLB-PCIX Bridge sets this bit if it asserts PCI_SERR#. Writing a 1 to this bit resets it to 0.
13	RMA	Received Master Abort Write 1 to clear.	This bit is set whenever PLB-PCIX Bridge terminates a PCI cycle for which it is the master with master abort (except configuration and special cycles) and the Master Abort Error Enable bit of the Error Enable Register is set. Writing a 1 to this bit resets it to 0.

12	RTA	Received Target Abort Write 1 to clear.	PLB-PCIX Bridge sets this bit whenever a PCI cycle for which it is the master is terminated with target abort and the Target Abort Error Enable bit of the Error Enable Register is set. Writing a 1 to this bit resets it to 0.
11	STA	Signaled Target Abort Write 1 to clear.	PLB-PCIX Bridge never signals a target abort. This bit is always 0.
10:9	DST	PCIDevSel Response Timing Read-only.	PCI_DEVSEL# response timing. PLB-PCIX Bridge asserts PCI_DEVSEL# on the second clock (also known as medium response time) after PCI_FRAME# is asserted by a PCI master attempting to access memory on the PLB side of the bridge. These bits are read-only and always return 01b when read.
8	DPE	Data Parity Error Detected Write 1 to clear.	This bit is set when the following two conditions are met: PLB-PCIX Bridge detects a data parity error (PCI_PERR# asserted) when it is the master on a PCI read cycle, or it is the master when it samples PCI_PERR# asserted on a PCI write cycle. The Parity Error Response bit (bit 6 of the PCI Command Register) is set. Writing a 1 to this bit resets it to 0.
7	FBBC	Fast Back-to-Back Capable Read-only; returns zero when read.	Indicates that the PCI target is capable of accepting fast back-to-back transactions when the transactions are not to the same agent. PLB-PCIX Bridge target does accept this type of fast back-to-back transaction, therefore, this bit is read-only and is 1 when in PCI-Conv mode. When in PCI-X mode, it is always 0.
6		Reserved	This bit is hardcoded to zero. This bit was the UDF support bit in PCI 2.1.
5	66C	66 MHz Capable	Indicates that the device is able to run at 66 MHz. Hardcoded to 1.
4	CL	Capabilities List	Indicates whether or not this device implements the pointer for a new capabilities linked list at offset 34h. This bit is read-only and returns 1 when read, indicating that the value read at offset 34h is a pointer in configuration space to a linked list of new capabilities.
3:0		Reserved	These bits are reserved, and return zeros when read.

18.18.3.5 PCI-X Revision ID Register (~~PCIC0_REVID~~PCIX0_REVID)

The Revision ID register is an 8-bit register used to hold the current incremental revision number. The local CPU (PLB master) can write a value to this register. ~~Figure 18-1~~Figure 18-12 describes PCIX0_REVID register bits.

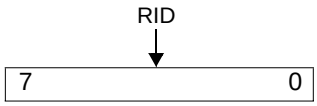


Figure 0-5. PCI-X Revision ID Register (PCIX0_REVID)

7:0	RID	Revision ID	Revision level of device.

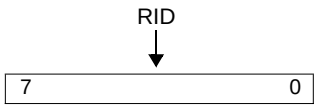


Figure 18-12. PCI-X Revision ID Register (PCIX0_REVID)

7:0	RID	Revision ID	Revision level of device.

18.18.3.6 PCI-X Class Register (PCIX0_CLS)

This register holds the class code. The local CPU (PLB master) can write a value to this register. ~~Figure 18-13~~Figure 18-13 describes PCIX0_CLS register bits.

Class information is defined in the *PCI*, Version 2.2, Appendix D.

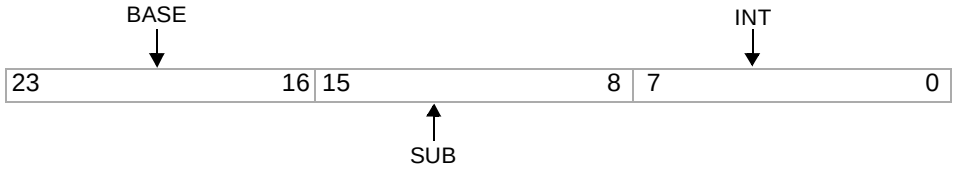


Figure 0-6. PCI-X Class Register (PCIX0_CLS)

23:16	BASE	Base Class	Reset to 0x06, which indicates bridge device.
15:8	SUB	Subclass	Reset to 00, which indicates host bridge.
7:0	INT	Interface Class	Reset to 00.

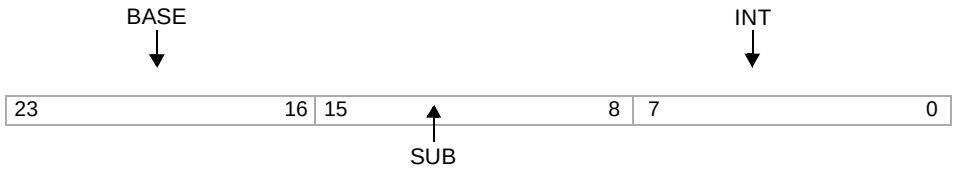


Figure 18-13. PCI-X Class Register (PCIX0_CLS)

23:16	BASE	Base Class	Reset to 0x06, which indicates bridge device.
15:8	SUB	Subclass	Reset to 00, which indicates host bridge.
7:0	INT	Interface Class	Reset to 00.

18.18.3.7 PCI-X Cache Line Size Register (PCIX0_CACHELS)

The PCI Cache Line Size Register determines the size of a PCI cache line. PLB-PCIX Bridge does not use this register for anything. MWI and MRL transactions issued by the PLB-PCIX Bridge are hardcoded to use a value of 64 bytes and 32 bytes respectively for the cache line size. ~~Figure 18-14~~Figure 18-14 describes PCIX0_CACHELS register bits.

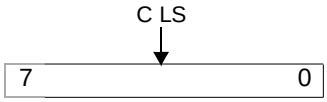


Figure 0-7. PCI-X Cache Line Size Register (PCIX0_CACHELS)

7:0	CLS	PCI Cache Line Size

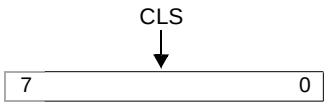


Figure 18-14. PCI-X Cache Line Size Register (PCIX0_CACHELS)

7:0	CLS	PCI Cache Line Size

18.18.3.8 PCI-X Latency Timer Register (PCIX0_LATTIM)

The PCI Latency Timer Register is an 8-bit read/write register used to hold the value of the PCI latency timer. The granularity of the latency timer is eight PCI cycles; therefore, the three low-order bits of this register are read-only and are hardwired to logic 0.

~~Figure 18-15~~Figure 18-15 describes PCIX0_LATTIM register bits.

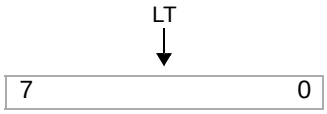


Figure 0-8. PCI-X Latency Timer Register (PCIX0_LATTIM)

7:0	LT	PCI Latency Timer

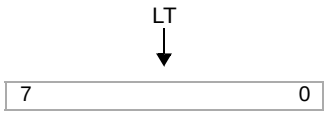


Figure 18-15. PCI-X Latency Timer Register (PCIX0_LATTIM)

7:0	LT	PCI Latency Timer

18.18.3.9 PCI-X Header Type Register (PCIX0_HDTYPE)

The PCI-X Header Type Register (bits 0-6) is used to identify the second part of the PCI header (beginning at offset 10h). It also determines whether a device contains multiple functions (bit 7). PLB-PCIX Bridge implements the standard header and is not a multi-function device, therefore, this register is read-only and returns "00h" when read. ~~Figure 18-16~~Figure 18-16 describes ~~PCIX0_HDTYPE~~ PCIX_HDTYPE register bits.

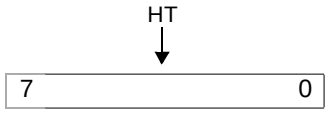


Figure 0-9. PCI-X Header Type Register (PCIX0_HDTYPE)

7:0	HT	PCI Header Type

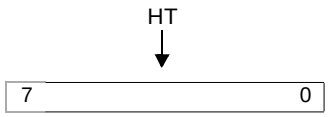


Figure 18-16. PCI-X Header Type Register (PCIX0_HDTYPE)

7:0	HT	PCI Header Type

18.18.3.10 PCI-X Built-in Self Test (BIST) Control Register (PCIX0_BIST)

The PCI-X BIST Register is used for control and status of BIST. PLB-PCIX Bridge does not implement BIST, therefore, this register is read-only and returns "00h" when read. ~~Figure 18-17~~Figure 18-17 describes PCIX0_BIST register bits.

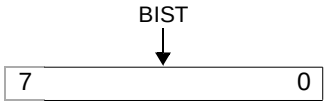


Figure 0-10. PCI-X BIST Control Register (PCIX0_BIST)

7:0	BIST	PCI Built-in-Self Test (BIST) Control

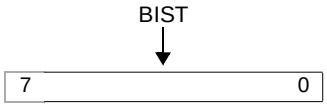


Figure 18-17. PCI-X BIST Control Register (PCIX0_BIST)

7:0	BIST	PCI Built-in-Self Test (BIST) Control

18.18.3.11 PCI-X BAR0 Low Register (PCIX0_BAR0L)

This register is enabled using the Enable bit of the PIM0 Size/Attribute Register. When disabled, this register is always zero (cannot be written). When enabled, this register is used as follows:

This register defines the PCI characteristics of a region of PCI memory space that is mapped to PLB space. ~~Figure 18-18~~Figure 18-18 describes PCIX0_BAR0L register bits.

Note: The PIM0 Size/Attribute Register must be initialized by the local CPU before any PCI device is allowed to configure this register.

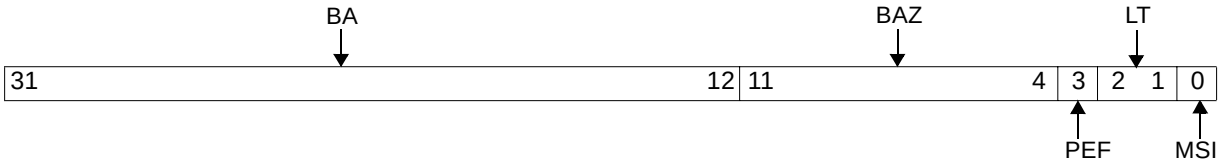


Figure 0-11. PCI-X BAR0 Low Register (PCIX0_BAR0L)

31:12	BA	Base Address	These bits determine the lower 32 bits of the PCI Memory address space where this region is located. Only corresponding bits that are 1 in the size field of the PIM0 Size/Attribute Register are writable. Bits that are 0 in the size field of the PIM0 Size/Attribute Register cause the corresponding Base Address bits to be always 0.
11:4	BAZ	Base Address - always zero	These bits are always 0 since the minimum size of this range is 4 KB.
3	PEF	Prefetchable	This bit determines if the region is prefetchable. Its value is determined by the PIM0 Size/Attribute, Prefetch Enable bit.
2:1	LT	Location Type	These bits are always 10 to indicate that the memory space can be located anywhere in the 64-bit address space.
0	MSI	Memory Space Indicator	This bit is always 0 to indicate memory space (rather than I/O).

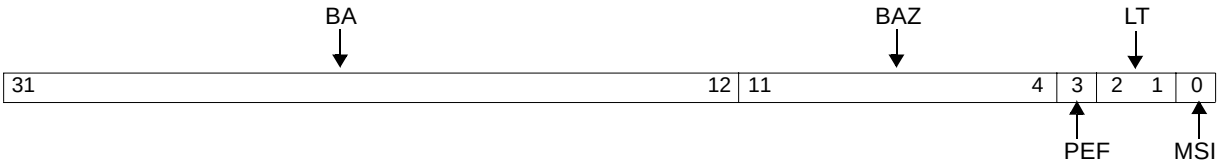


Figure 18-18. PCI-X BAR0 Low Register (PCIX0_BAR0L)

31:12	BA	Base Address	These bits determine the lower 32 bits of the PCI Memory address space where this region is located. Only corresponding bits that are 1 in the size field of the PIM0 Size/Attribute Register are writable. Bits that are 0 in the size field of the PIM0 Size/Attribute Register cause the corresponding Base Address bits to be always 0.
11:4	BAZ	Base Address - always zero	These bits are always 0 since the minimum size of this range is 4 KB.
3	PEF	Prefetchable	This bit determines if the region is prefetchable. Its value is determined by the PIM0 Size/Attribute , Prefetch Enable bit.
2:1	LT	Location Type	These bits are always 10 to indicate that the memory space can be located anywhere in the 64-bit address space.
0	MSI	Memory Space Indicator	This bit is always 0 to indicate memory space (rather than I/O).

18.18.3.12 PCI-X BAR0 High Register (PCIX0_BAR0H)

This register is enabled using the Enable bit of the PIM0 Size/Attribute register. When disabled, this register is always zero (cannot be written). When enabled, this register is used as follows:

This register defines the PCI characteristics of a region of PCI memory space that is mapped to PLB space. This register defines the higher 32 bits of the PCI memory address space where this region is located. This register, along with the PCI BAR0 Low register, defines the 64-bit starting address of the PCI memory space that is mapped to the PLB space. ~~Figure 18-19~~Figure 18-19 describes PCIX0_BAR0H register bits.

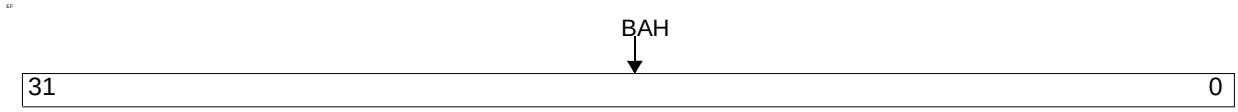


Figure 0-12. PCI-X BAR0 High Register (PCIX0_BAR0H)

31:0	BAH	Base Address High	Defines the upper 32 bits of PCI memory space that is mapped to PLB.

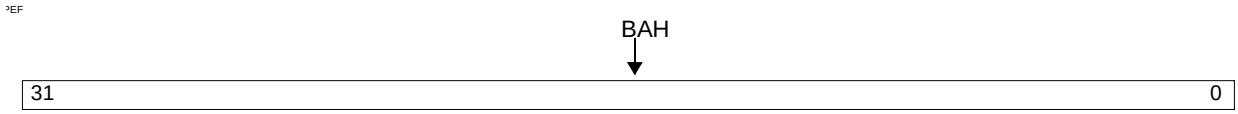


Figure 18-19. PCI-X BAR0 High Register (PCIX0_BAR0H)

31:0	BAH	Base Address High	Defines the upper 32 bits of PCI memory space that is mapped to PLB.

18.18.3.13 PCI-X BAR1 Register (PCIX0_BAR1)

This register is enabled using the Enable bit of the PIM1 Size/Attribute Register. When disabled, this register is always zero (cannot be written). When enabled, this register is used as follows:

This register defines the PCI characteristics of a region of PCI I/O space that is mapped to PLB space.

Note: The PIM1 Size/Attribute Register must be initialized by the local CPU before any PCI device is allowed to configure this register.

Figure 0-13. PCI-X BAR1 Register (PCIX0_BAR1)

31:8	BA	Base Address	These bits determine the PCI I/O address space where this region is located.
7:2	BAZ	Base Address always zero	These bits are always 0 since the minimum size of this range is 256 bytes.
1		Reserved	Returns zero when read.
0	MSI	Memory Space Indicator	This bit is always 1 to indicate I/O space (rather than memory).

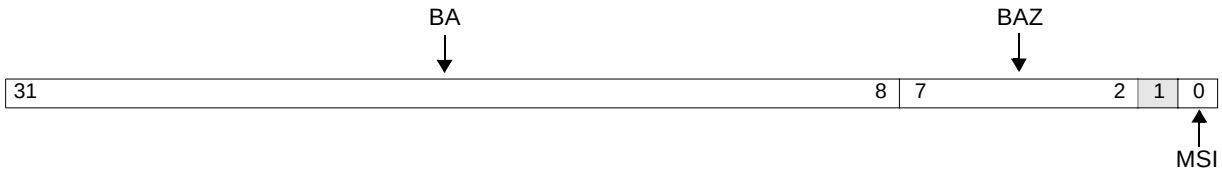


Figure 18-20. PCI-X BAR1 Register (PCIX0_BAR1)

31:8	BA	Base Address	These bits determine the PCI I/O address space where this region is located.
7:2	BAZ	Base Address always zero	These bits are always 0 since the minimum size of this range is 256 bytes.
1		Reserved	Returns zero when read.
0	MSI	Memory Space Indicator	This bit is always 1 to indicate I/O space (rather than memory).

18.18.3.14 PCI-X BAR2 Low Register (PCIX0_BAR2L)

This register is enabled using the Enable bit of the PIM2 Size/Attribute Register. When disabled, this register is always zero (cannot be written). When enabled, this register is used as follows:

This register defines the PCI characteristics of a region of PCI memory space that is mapped to PLB space. The Enable bit of the Expansion ROM BAR Register must be low for the PIM2 (BAR2) address space to be enabled.

Note: The PIM2 Size/Attribute Register must be initialized by the local CPU before any PCI device is allowed to configure this register.

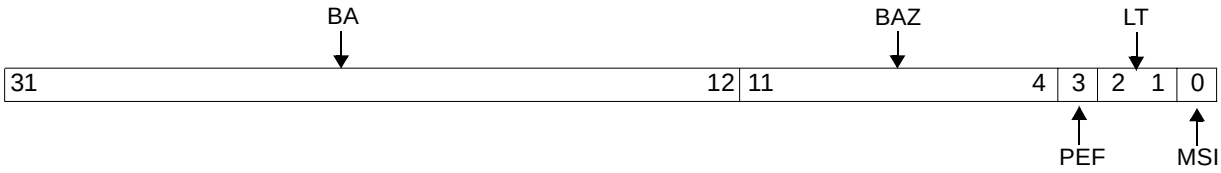


Figure 0-14. PCI-X BAR2 Low Register (PCIX0_BAR2L)

31:12	BA	Base Address	These bits determine the lower 32-bits of the PCI Memory address space where this region is located. Only corresponding bits that are 1 in the size field of the PIM0 Size/Attribute Register are writable. Bits that are 0 in the size field of the PIM2 Size/Attribute Register cause the corresponding Base Address bits to be always 0.
11:4	BAZ	Base Address - always zero	These bits are always 0 since the minimum size of this range is 4 KB.
3	PEF	Prefetchable	This bit determines if the region is prefetchable. Its value is determine by the PIM2 Size/Attribute, Prefetch Enable bit.
2:1	LT	Location Type	These bits are always 10 to indicate that the memory space can be located anywhere in the 64-bit address space.
0	MSI	Memory Space Indicator	This bit is always 0 to indicate memory space (rather than I/O).

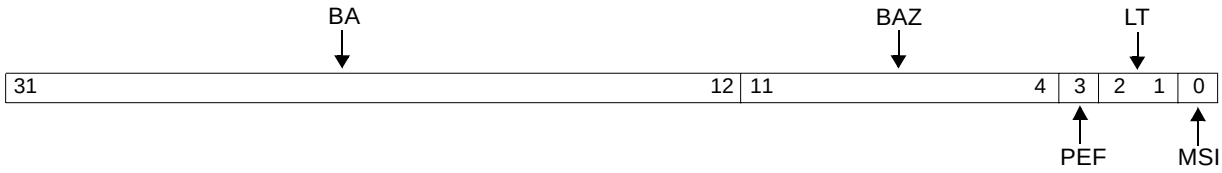


Figure 18-21. PCI-X BAR2 Low Register (PCIX0_BAR2L)

31:12	BA	Base Address	These bits determine the lower 32-bits of the PCI Memory address space where this region is located. Only corresponding bits that are 1 in the size field of the PIM2 Size/Attribute Register are writable. Bits that are 0 in the size field of the PIM2 Size/Attribute Register cause the corresponding Base Address bits to be always 0.
11:4	BAZ	Base Address - always zero	These bits are always 0 since the minimum size of this range is 4 KB.
3	PEF	Prefetchable	This bit determines if the region is prefetchable. Its value is determine by the PIM2 Size/Attribute , Prefetch Enable bit.
2:1	LT	Location Type	These bits are always 10 to indicate that the memory space can be located anywhere in the 64-bit address space.
0	MSI	Memory Space Indicator	This bit is always 0 to indicate memory space (rather than I/O).

18.18.3.15 PCI-X BAR2 High Register (PCIX0_BAR2H)

This register is enabled using the Enable bit of the PIM2 Size/Attribute Register. When disabled, this register is always zero (cannot be written). When enabled, this register is used as follows:

This register defines the PCI characteristics of a region of PCI memory space that is mapped to PLB space. This register defines the higher 32 bits of the PCI memory address space where this region is located. This register, along with the PCI PIM2 BAR Low Register, defines the 64-bit starting address of the PCI memory space that is mapped to the PLB space.

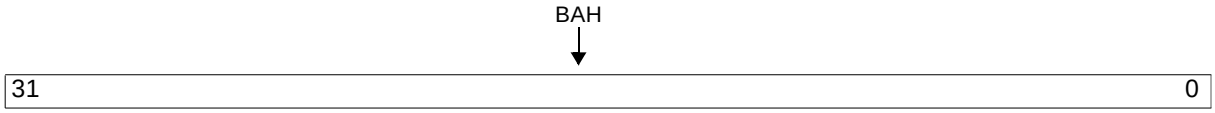


Figure 0-15. PCI-X BAR2 High Register (PCIX0_BAR2H)

31:0	BAH	Base Address High	Defines the upper 32 bits of PCI memory space that is mapped to PLB.

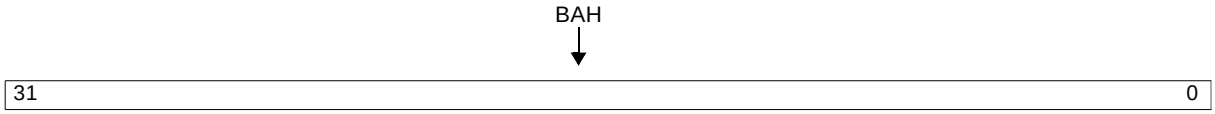


Figure 18-22. PCI-X BAR2 High Register (PCIX0_BAR2H)

31:0	BAH	Base Address High	Defines the upper 32 bits of PCI memory space that is mapped to PLB.

18.18.3.16 PCI-X Subsystem Vendor ID Register (PCIX0_SBSYSVID)

The subsystem Vendor ID Register is a 16-bit register used to hold the Vendor ID for the subsystem or add-in board.

Note: The Vendor ID Register holds the Vendor ID for PLB-PCIX Bridge that is used in many different designs.

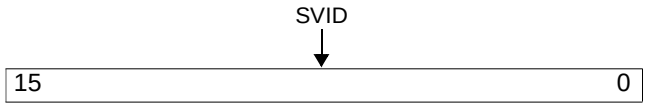


Figure 0-16. PCI-X Subsystem Vendor ID Register (PCIX0_SBSYSVID)

15:0	SVID	PCI Subsystem Vendor ID

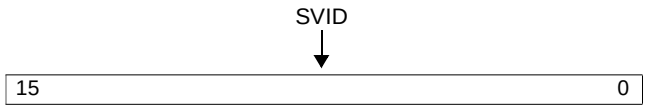


Figure 18-23. PCI-X Subsystem Vendor ID Register (PCIX0_SBSYSVID)

15:0	SVID	PCI Subsystem Vendor ID

18.18.3.17 PCI-X Subsystem ID Register (PCIX0_SBSYSID)

The Subsystem ID Register is a 16-bit register used to hold the Device ID of the subsystem or add-in board.

Note: The Device ID Register holds the Device ID for PLB-PCIX Bridge that can be used in many different designs.

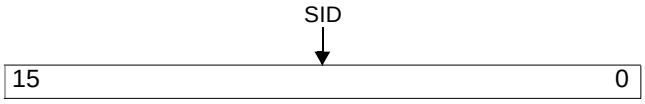


Figure 0-17. PCI-X Subsystem ID Register (PCIX0_SBSYSID)

15:0	SID	PCI Subsystem ID

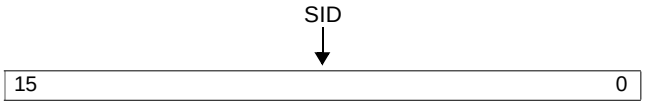


Figure 18-24. PCI-X Subsystem ID Register (PCIX0_SBSYSID)

15:0	SID	PCI Subsystem ID

18.18.3.18 PCI-X Expansion ROM Base Address Register (PCIX0_EROMBA)

This register is enabled using the Enable bit of the PIM2 Size/Attribute Register. When disabled, this register is always zero (cannot be written). When enabled, this register is used as follows:

If the Enable bit of this register is low, the PIM2 (BAR2) address space is enabled and the Expansion ROM address space is not. If the Enable bit of thsi register is high, the ~~PIM2 (BAR2)~~ BAR2 address space is disabled, and the Expansion ROM address space is enabled, translated through PIM2.

This register defines the PCI characteristics of a region of PCI memory space that is mapped to PLB space.

Note: The PIM2 Size/Attribute Register must be initialized by the local CPU before any PCI device is allowed to configure this register.

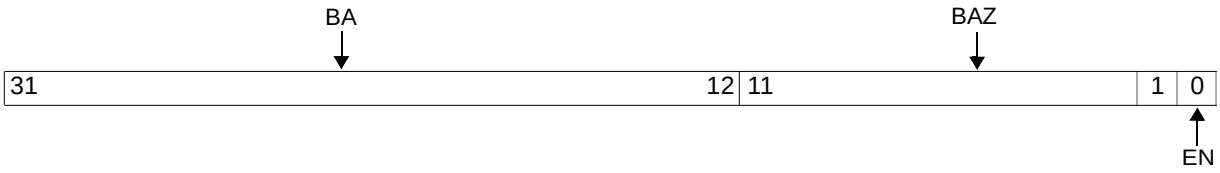


Figure 0-18. PCI-X Expansion ROM Base Address Register (PCIX0_EROMBA)

31:12	BA	Base Address	These bits determine the PCI I/O address space where this region is located.
11:1	BAZ	Base Address - always zero	These bits are always 0 since the minimum size of this range is 4 KB.
0	EN	Enable	When 1, the expansion ROM address space of the PLB_PCIX Bridge is enabled and the PIM2 (BAR2) address space is disabled. When 0, the expansion ROM address space is disabled and the PIM2 (BAR2) address space is enabled.

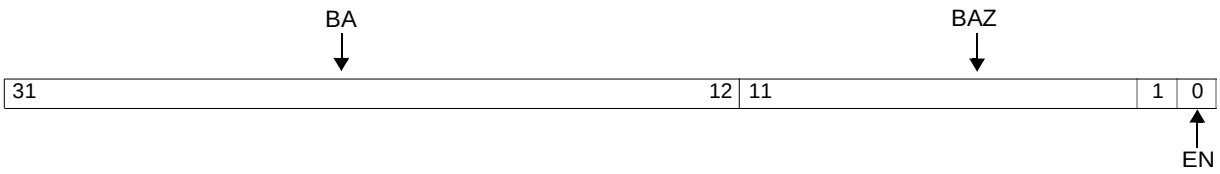


Figure 18-25. PCI-X Expansion ROM Base Address Register (PCIX0_EROMBA)

31:12	BA	Base Address	These bits determine the PCI I/O address space where this region is located.
11:1	BAZ	Base Address - always zero	These bits are always 0 since the minimum size of this range is 4 KB.
0	EN	Enable	When 1, the expansion ROM address space of the PLB-PCIX Bridge is enabled and the BAR2 address space is disabled. When 0, the expansion ROM address space is disabled and the BAR2 address space is enabled, translated through PIM2.

18.18.3.19 PCI-X Capabilities Pointer (PCIX0_CAP)

The PCI Capabilities Pointer is a 8-bit pointer to the next capability in configuration space. This data structure is indicated in the PCI Status Register ~~by setting the~~ capabilities list bit (bit 4). This register points to the first item in the list of capabilities which points to the MSI capability structure at address offset "C0h."

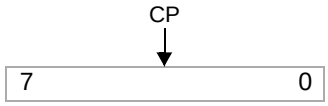


Figure 0-19. PCI-X Capabilities Pointer (PCIX0_CAP)

7:0	CP	PCI Capabilities Pointer

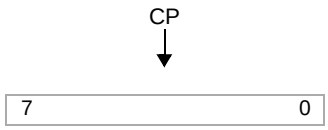


Figure 18-26. PCI-X Capabilities Pointer (PCIX0_CAP)

7:0	CP	PCI Capabilities Pointer

18.18.3.20 PCI-X Interrupt Line Register (PCIX0_INTLN)

The PCI Interrupt Line Register is used to communicate interrupt line routing information. It controls nothing in the PLB-PCIX Bridge.

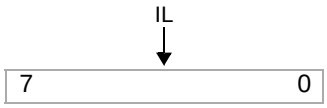


Figure 0-20. PCI-X Interrupt Line Register (PCIX0_INTLN)

7:0	IL	PCI Interrupt Line

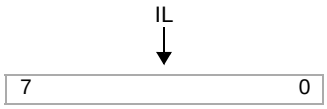


Figure 18-27. PCI-X Interrupt Line Register (PCIX0_INTLN)

7:0	IL	PCI Interrupt Line

18.18.3.21 PCI-X Interrupt Pin Register (PCIX0_INTPN)

The PCI Interrupt Pin Register tells which PCI interrupt line the device uses. This 8-bit register is read-only and the value "01h" indicates INTA#.

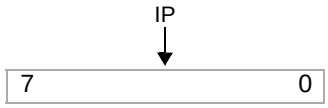


Figure 0-21. PCI-X Interrupt Pin Register (PCIX0_INTPN)

7:0	IP	PCI Interrupt Pin

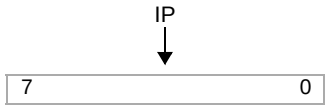


Figure 18-28. PCI-X Interrupt Pin Register (PCIX0_INTPN)

7:0	IP	PCI Interrupt Pin

18.18.3.22 PCI-X MIN_GNT Register (PCIX0_MINGNT)

The PCI MIN_GNT Register is used for specifying how long a burst period a PCI device needs. This 8-bit register is read-only and returns "00h" when read.

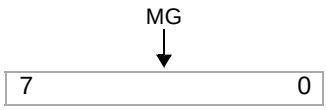


Figure 0-22. PCI-X Minimum Grant Register (PCIX0_MINGNT)

7:0	MG	PCI Minimum Grant

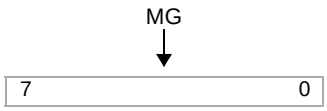


Figure 18-29. PCI-X Minimum Grant Register (PCIX0_MINGNT)

7:0	MG	PCI Minimum Grant

18.18.3.23 PCI-X Maximum Latency Register (PCIX0_MAXLTNCY)

The ~~PCI MAX_LAT~~ PCIX0_MAXLTNCY Register specifies how often a PCI device needs to gain access to the PCI bus. This 8-bit register is read-only and returns "00h" when read.

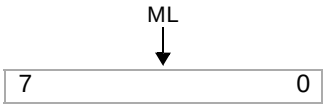


Figure 0-23. PCI-X Maximum Latency Register (PCIX0_MAXLTNCY)

7:0	ML	PCI Maximum Latency

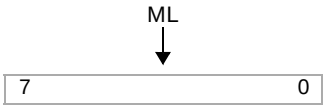


Figure 18-30. PCI-X Maximum Latency Register (PCIX0_MAXLTNCY)

7:0	ML	PCI Maximum Latency

18.18.4 Bridge Options Registers

The following registers control miscellaneous functions of the PLB-PCIX Bridge.

18.18.4.1 PCI-X Bridge Options 1 (PCIX0_BRDGOPT1)

The Bridge Options 1 Register controls various operating parameters of the PLB-PCIX Bridge. These must be set up before transactions through the bridge are allowed in.

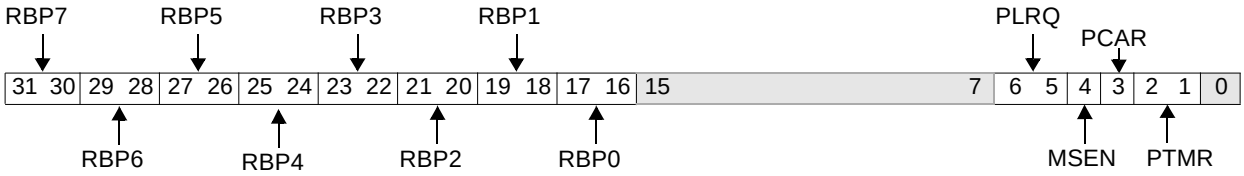


Figure 0-24. PCI-X Bridge Options 1 (PCIX0_BRDGOPT1)

31:30	RBP7		Must be set 00.
29:28	RBP6	Outbund Read Buffer Mapping for PLB master 6 DMA	Must be set to 01.
27:26	RBP5		Must be set to 00.
25:24	RBP4		Must be set to 00.
23:22	RBP3		Must be set to 00.
21:20	RBP2		Must be set to 00.
19:18	RBP1		Must be set to 00.
17:16	RBP0		Must be set to 00.
15:8		Reserved	These bits are always zero.
7		Reserved	This bit must be set to zero.
6:5	PLRQ	PLB Request Priority	This bit controls how the bridge PLB master controls the PLB arbitration priority for all PLB accesses. 11b: highest 10b: next highest 01b: next highest 00b: lowest
4	MSEN	Inbound MSI Enable	This bit enables Inbound MSI. This is typically only used in Host-Bridge Mode. When high, the Inbound MSI logic drives TBD to UIC.
3	PCAR	PCI Arbiter Park Mode	This bit defines how the internal PCI arbiter will handle bus parking. A value of 0 means that the arbiter will park on requester 0 (the bridge PCI master). A value of 1 means that the arbiter will park on the last master granted the bus. This bit must not be changed while PCI masters are active.

2:1	PTMR	PCI Target Memory Read Command interpretation	This field allows PLB-PCIX Bridge to be forced to treat a PCI memory read as a memory read multiple or memory read line with respect to the burst size implied by these read commands. This is for masters that use memory read for multiple beat bursts. 00 Memory read 01 Memory read line 10 Memory read multiple 11 Reserved
0		Reserved	Must be set to 0.

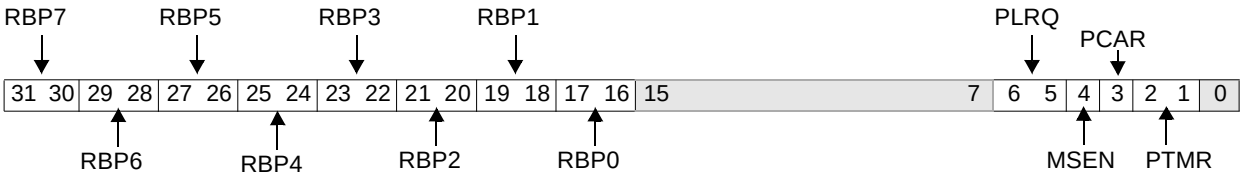


Figure 18-31. PCI-X Bridge Options 1 (PCIX0_BRDGOPT1)

31:30	RBP7		Must be set 00.
29:28	RBP6	Outbund Read Buffer Mapping for PLB master 6 DMA	Must be set to 01.
27:26	RBP5		Must be set to 00.
25:24	RBP4		Must be set to 00.
23:22	RBP3		Must be set to 00.
21:20	RBP2		Must be set to 00.
19:18	RBP1		Must be set to 00.
17:16	RBP0		Must be set to 00.
15:8		Reserved	These bits are always zero.
7		Reserved	This bit must be set to zero.
6:5	PLRQ	PLB Request Priority	This bit controls how the PLB-PCIX Bridge PLB master controls the PLB arbitration priority for all PLB accesses. 11b: highest 10b: next highest 01b: next highest 00b: lowest
4	MSEN	Inbound MSI Enable	This bit enables Inbound MSI. This is typically only used in Host-Bridge Mode. When high, the Inbound MSI logic drives TBD to UIC.
3	PCAR	PCI Arbiter Park Mode	This bit defines how the internal PCI arbiter will handle bus parking. A value of 0 means that the arbiter will park on requester 0 (the bridge PCI master). A value of 1 means that the arbiter will park on the last master granted the bus. This bit must not be changed while PCI masters are active.

2:1	PTMR	PCI Target Memory Read Command interpretation	This field allows PLB-PCIX Bridge to be forced to treat a PCI memory read as a memory read multiple or memory read line with respect to the burst size implied by these read commands. This is for masters that use memory read for multiple beat bursts. 00 Memory read 01 Memory read line 10 Memory read multiple 11 Reserved
0		Reserved	Must be set to 0.

18.18.4.2 PCI-~~XBridge~~ X Bridge Options 2 (PCIX0_BRDGOPT2)

The Bridge Options 2 Register controls various operating parameters of the PLB-PCIX Bridge.

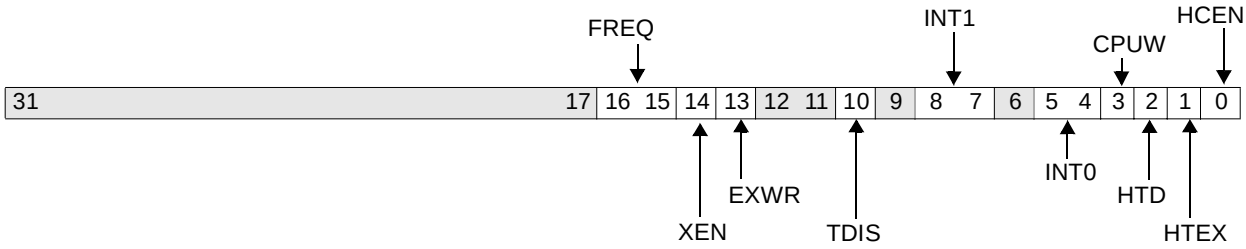


Figure 0-25. PCI-X Bridge Options 2 (PCIX0_BRDGOPT2)

31:17		Reserved	Always read as zero.
16:15	FREQ	PCI Frequency Range	Read only value that reports the PCI frequency range: For PCI-X: 00b = 100-133 MHz 01b = 66-100 MHz 10b = 50-66 MHz 11b = reserved. For PCI-Conv: 00b = reserved 01b = reserved 10b = 33-66 MHz 11b = 0-33 MHz.
14	XEN	PCI-X Mode Enabled	Read only value that reports if X mode was selected.
13	EXWR	External Write to PCI Command Interrupt	When the PCI Command Register is written from the PCI side (inbound config write), this bit is set. It causes the TBD interrupt, which goes to the local CPU, to assert. Software may also set or clear this bit normally.
12:11		Reserved	Must be set to 0.
10	TDIS	PCI Discard Timer Disable	When 1, PLB-PCIX Bridge never discards delayed read data.
9		Reserved	Must be set to 0.
8:7	INT1	PCI Interrupt Source 1	Reset to 01. Do not change.
6		Reserved	Must be set to 0.
5:4	INT0	PCI Interrupt Source 0	Reset to 00. Do not change.

3	CPUW	Local CPU wait	This bit controls the value of the PCI local CPU wait (PCWE) strapping bit, which prevents the local CPU from running when this bit is 1. By causing this bit to be 1 at reset, the host PCI master is given time to initialize the internal register set of the PLB-PCIX Bridge before the local CPU boots. This is typically used to set up boot code, when the local CPU is not booting from local ROM.
2	HTD	Host Configuration Enable Timer Disable	If this bit is set, the Host Configuration Enable timer never expires. See Host Configuration Enable bit below.
1	HTEX	Host Configuration Enable Timer Expired	This bit is set if the Host Configuration Enable timer expired. See Host Configuration Enable bit below.
0	HCEN	Host Configuration Enable	This bit controls host PCI access to the PCI configuration registers. If this bit is a 0, all host attempts to access PCI configuration registers (internal registers) of the PLB-PCIX Bridge are retried. By causing this bit to be 0 at reset, the local CPU is given time to initialize the internal register set before the host sees it. The reset value of this bit is determined by the PCI host configuration enable (PHCE) strapping bit. If PCI host configuration enable (PHCE) strapping bit is tied to a 1 (high), this bit has a value of 1 after reset. If PCI host configuration enable (PHCE) strapping bit is tied to a 0 (low), this bit has a value of 0 after reset. If this bit is cleared following reset, and the Host Configuration Enable Timer Disable bit is cleared, and if this bit does not get set by the local CPU within 2^{24} PLB clocks (126 ms at 133 MHz), then it is set automatically, and the Host Configuration Enable Timer Expired bit is set in this register. **Note:** As per the PCI Specification, Version 2.2, as little as 500 ms is allowed for configuration to be set up. Thus, if PLB is run below 34 MHz, this timer will not expire before this time period.

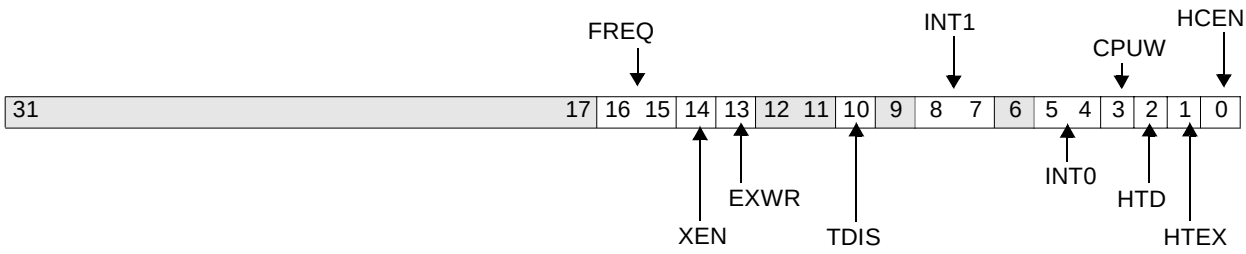


Figure 18-32. PCI-X Bridge Options 2 (PCIX0_BRDGOPT2)

31:17		Reserved	Always read as zero.
16:15	FREQ	PCI Frequency Range	Read only value that reports the PCI frequency range: For PCI-X: 00b = 100-133 MHz 01b = 66-100 MHz 10b = 50-66 MHz 11b = reserved. For PCI-Conv: 00b = reserved 01b = reserved 10b = 33-66 MHz 11b = 0-33 MHz.
14	XEN	PCI-X Mode Enabled	Read only value that reports if X mode was selected.
13	EXWR	External Write to PCI Command Interrupt	When the PCI Command Register is written from the PCI side (inbound config write), this bit is set. It causes the TBD interrupt, which goes to the local CPU, to assert. Software may also set or clear this bit normally.
12:11		Reserved	Must be set to 0.
10	TDIS	PCI Discard Timer Disable	When 1, PLB-PCIX Bridge never discards delayed read data.
9		Reserved	Must be set to 0.
8:7	INT1	PCI Interrupt Source 1	Reset to 01. Do not change.
6		Reserved	Must be set to 0.
5:4	INT0	PCI Interrupt Source 0	Resets to 00. Setting this field to 01 will mask the generation of outbound interrupts, both via INTA and MSI.
3	CPUW	Local CPU wait	This bit controls the value of the PCI local CPU wait (PCWE) strapping bit, which prevents the local CPU from running when this bit is 1. By causing this bit to be 1 at reset, the host PCI master is given time to initialize the internal register set of the PLB-PCIX Bridge before the local CPU boots. This is typically used to set up boot code, when the local CPU is not booting from local ROM.
2	HTD	Host Configuration Enable Timer Disable	If this bit is set, the Host Configuration Enable timer never expires. See Host Configuration Enable bit below.
1	HTEX	Host Configuration Enable Timer Expired	This bit is set if the Host Configuration Enable timer expired. See Host Configuration Enable bit below.

0	HCEN	Host Configuration Enable	This bit controls host PCI access to the PCI configuration registers. If this bit is a 0, all host attempts to access PCI configuration registers (internal registers) of the PLB-PCIX Bridge are retried. By causing this bit to be 0 at reset, the local CPU is given time to initialize the internal register set before the host sees it. The reset value of this bit is determined by the PCI host configuration enable (PHCE) strapping bit. If PCI host configuration enable (PHCE) strapping bit is tied to a 1 (high), this bit has a value of 1 after reset. If PCI host configuration enable (PHCE) strapping bit is tied to a 0 (low), this bit has a value of 0 after reset. If this bit is cleared following reset, and the Host Configuration Enable Timer Disable bit is cleared, and if this bit does not get set by the local CPU within 2^{24} PLB clocks (126 ms at 133 MHz), then it is set automatically, and the Host Configuration Enable Timer Expired bit is set in this register. **Note:** As per the PCI Specification, Version 2.2, as little as 500 ms is allowed for configuration to be set up. Thus, if PLB is run below 34 MHz, this timer will not expire before this time period.

18.18.5 Error Handling Registers

The following registers are associated with error handling and reporting.

18.18.5.1 PCI-X Error Enable (PCIX0_ERREN)

The Error Enable Register is a 32-bit read/write register used to enable detection and reporting of various errors for the PLB-PCIX Bridge.

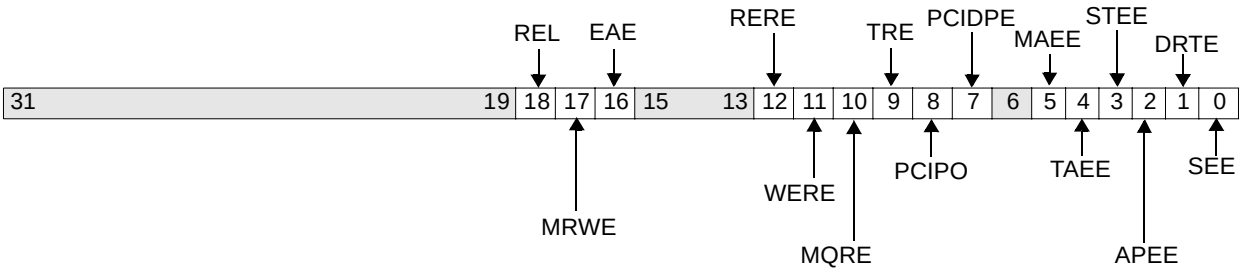


Figure 0-26. PCI-X Error Enable (PCIX0_ERREN)

31:19		Reserved	Always read as zero.
18	REL	Route Error Locally	When set, errors are routed locally using ERROR_INT. Assertion of PLB MERR is not affected by this bit. When cleared, errors are routed to the PCI using PCI_SERR#.
17	MRWE	MRdErr/MWrErr Assertion Enable	This bit enables the assertion of PLB Read MERR or PLB Write MERR when the PLB-PCIX Bridge detects an error.
16	EAE	ERROR_INT Assertion Enable	This bit enables the assertion of ERROR_INT when the PLB-PCIX Bridge detects an error.
13		Reserved	Always read as zero.
12	RERE	MRdErr Receive Enable	This bit enables the detection of PLB Read MERR when the PLB-PCIX Bridge is the PLB master. If this bit is set, the PLB-PCIX Bridge may drive ERROR_INT to the local CPU or PCI_SERR# to the PCI bus.
11	WERE	MWrErr Receive Enable	This bit enables the detection of PLB Write MERR when PLB-PCIX Bridge is the PLB master. If this bit is set, the PLB-PCIX Bridge may drive ERROR_INT to the local CPU or PCI_SERR# to the PCI bus.
10	MQRE	Mirq Receive Enable	This bit enables the detection of MIRQ when PLB-PCIX Bridge is the PLB master. If this bit is set, the PLB-PCIX Bridge may drive ERROR_INT to the local CPU or PCI_SERR# to the PCI bus.
9	TRE	Timeout Receive Enable	This bit enables the detection of Timeout when PLB-PCIX Bridge is the PLB master. If this bit is set, the PLB-PCIX Bridge may drive ERROR_INT to the local CPU or PCI_SERR# to the PCI bus.

8	PO	PCI Parity Option	This bit should normally be left 1. It may be set to 0 to mask all types of PCI parity error checking. This bit is 1 after reset.
7	IDPE	PCI Inbound Write Data Parity Error Enable	This bit enables the detection of PCI data parity errors on inbound writes. **Note:** The Detected Parity Error bit of the PCI Status Register and the masking of PCI PERR# is not affected by the above, but the assertion of ERROR_INT and PCI_SERR# is.
6		Reserved	Always read as zero.
5	MAEE	Master Abort Error Enable	This bit enables the detection of master aborts as an error condition, when PLB-PCIX Bridge is the PCI master. If this bit is set, PLB-PCIX Bridge may drive PLB Read MERR or PLB Write MERR on the PLB or PCI_SERR# on the PCI bus.
4	TAEE	Target Abort Error Enable	This bit enables the detection of a target abort received while PLB-PCIX Bridge is the PCI master to be detected as an error condition. If this bit is set, the PLB-PCIX Bridge may drive ERROR_INT to the local CPU or PCI_SERR# on the PCI bus.
3	STEE	Split Transaction Error Enable	This bit enables the detection of split transaction errors (PCI-X only), while the PLB-PCIX Bridge is the PCI master. If this bit is set, the PLB-PCIX Bridge may drive ERROR_INT to the local CPU or PCI_SERR# to the PCI bus.
2	APEE	Addr/Attrib Parity Error Enable	This bit enables the detection of Address or Attribute (PCI-X only) parity errors while the PLB-PCIX Bridge is the PCI target (including split response). If this bit is set, the PLB-PCIX Bridge may drive PCI_SERR# on the PCI bus.
1	DRTE	Delayed Read Discard Timer Expired Error Enable	This bit enables the detection of Delayed Read Discard Timer expiration as an error. If this bit is set, the PLB-PCIX Bridge may drive the ERROR_INT signal to the local CPU or PCI_SERR# on the PCI bus.
0	SEE	PCI_SERR# Received as Error Enable	This bit enables the detection of PCI_SERR# as an error. If this bit is set, the PLB-PCIX Bridge may drive the ERROR_INT signal to the local CPU. **Note:** The PLB-PCIX Bridge may be receiving this error, while at the same time, it is driving PCI_SERR# in response to some other error.

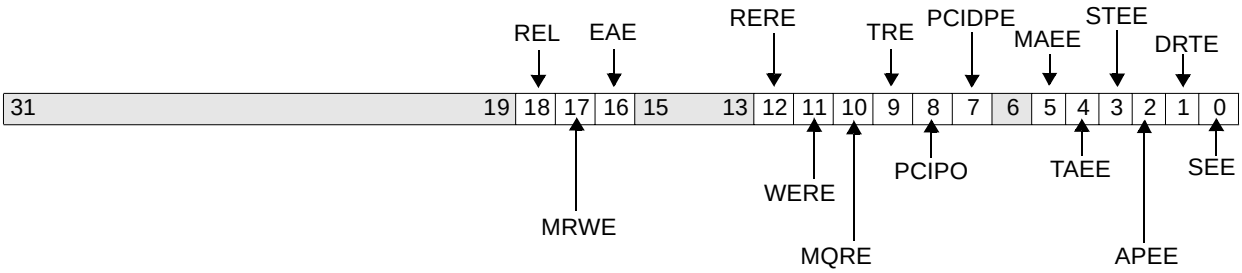


Figure 18-33. PCI-X Error Enable (PCIX0_ERREN)

31:19		Reserved	Always read as zero.
18	REL	Route Error Locally	When set, errors are routed locally using ERROR_INT. Assertion of PLB MERR is not affected by this bit. When cleared, errors are routed to the PCI using PCI_SERR#.
17	MRWE	MRdErr/MWrErr Assertion Enable	This bit enables the assertion of PLB Read MERR or PLB Write MERR when the PLB-PCIX Bridge detects an error.
16	EAE	ERROR_INT Assertion Enable	This bit enables the assertion of ERROR_INT when the PLB-PCIX Bridge detects an error.
13		Reserved	Always read as zero.
12	RERE	MRdErr Receive Enable	This bit enables the detection of PLB Read MERR when the PLB-PCIX Bridge is the PLB master. If this bit is set, the PLB-PCIX Bridge may drive ERROR_INT to the local CPU or PCI_SERR# to the PCI bus.
11	WERE	MWrErr Receive Enable	This bit enables the detection of PLB Write MERR when PLB-PCIX Bridge is the PLB master. If this bit is set, the PLB-PCIX Bridge may drive ERROR_INT to the local CPU or PCI_SERR# to the PCI bus.
10	MQRE	Mirq Receive Enable	This bit enables the detection of MIRQ when PLB-PCIX Bridge is the PLB master. If this bit is set, the PLB-PCIX Bridge may drive ERROR_INT to the local CPU or PCI_SERR# to the PCI bus.
9	TRE	Timeout Receive Enable	This bit enables the detection of Timeout when PLB-PCIX Bridge is the PLB master. If this bit is set, the PLB-PCIX Bridge may drive ERROR_INT to the local CPU or PCI_SERR# to the PCI bus.
8	PO	PCI Parity Option	This bit should normally be left 1. It may be set to 0 to mask all types of PCI parity error checking. This bit is 1 after reset.
7	IDPE	PCI Inbound Write Data Parity Error Enable	This bit enables the detection of PCI data parity errors on inbound writes. **Note:** The Detected Parity Error bit of the PCI Status Register and the masking of PCI PERR# is not affected by the above, but the assertion of ERROR_INT and PCI_SERR# is.
6		Reserved	Always read as zero.
5	MAEE	Master Abort Error Enable	This bit enables the detection of master aborts as an error condition, when PLB-PCIX Bridge is the PCI master. If this bit is set, PLB-PCIX Bridge may drive PLB Read MERR or PLB Write MERR on the PLB or PCI_SERR# on the PCI bus.
4	TAEE	Target Abort Error Enable	This bit enables the detection of a target abort received while PLB-PCIX Bridge is the PCI master to be detected as an error condition. If this bit is set, the PLB-PCIX Bridge may drive ERROR_INT to the local CPU or PCI_SERR# on the PCI bus.

3	STEE	Split Transaction Error Enable	This bit enables the detection of split transaction errors (PCI-X only), while the PLB-PCIX Bridge is the PCI master. If this bit is set, the PLB-PCIX Bridge may drive ERROR_INT to the local CPU or PCI_SERR# to the PCI bus.
2	APEE	Addr/Attrib Parity Error Enable	This bit enables the detection of Address or Attribute (PCI-X only) parity errors while the PLB-PCIX Bridge is the PCI target (including split response). If this bit is set, the PLB-PCIX Bridge may drive PCI_SERR# on the PCI bus.
1	DRTE	Delayed Read Discard Timer Expired Error Enable	This bit enables the detection of Delayed Read Discard Timer expiration as an error. If this bit is set, the PLB-PCIX Bridge may drive the ERROR_INT signal to the local CPU or PCI_SERR# on the PCI bus.
0	SEE	PCI_SERR# Received as Error Enable	This bit enables the detection of PCI_SERR# as an error. If this bit is set, the PLB-PCIX Bridge may drive the ERROR_INT signal to the local CPU. **Note:** The PLB-PCIX Bridge may be receiving this error, while at the same time, it is driving PCI_SERR# in response to some other error.

18.18.5.2 PCI-X Error Status (PCIX0_ERRSTS)

The Error Status Register is a 32-bit read/write register containing status on error conditions that have been detected. Bits in this register can only be set to "1" as a result of a system error occurring. These bits can be reset by writing a "1" to the desired bit location. Writing a "0" to any bit leaves that bit unchanged.

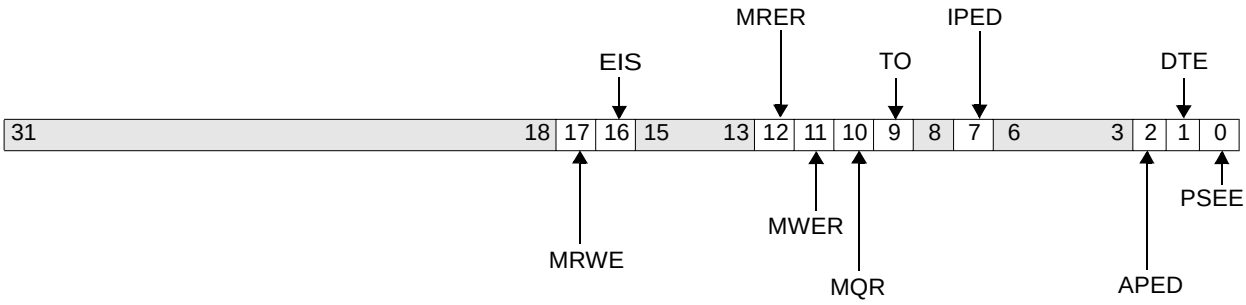


Figure 0-27. PCI-X Error Status (PCIX0_ERRSTS)

31:18		Reserved	Always read as zero.
17	MRWE	MRdErr or MWrErr Signaled	This bit is set whenever an error occurs that causes PLB-PCIX Bridge to assert PLB Read MERR or PLB Write MERR.
16	EIS	ERROR_INT Signaled	This bit is set whenever an error occurs that causes PLB-PCIX Bridge to assert ERROR_INT.
15:13		Reserved	Always read as zero.
12	MRER	MRdErr Received	This bit is set when PLB-PCIX Bridge is a PLB master and receives PLB Read MERR.
11	MWER	MWrErr Received	This bit is set when PLB-PCIX Bridge is a PLB master and receives PLB Write MERR.
10	MQR	Mirq Received	This bit is set when PLB-PCIX Bridge is a PLB master and receives MIRQ.
9	TO	Timeout	This bit is set when PLB-PCIX Bridge is a PLB master and receives Timeout.
8		Reserved	Always read as zero.
7	IPED	PCI Inbound Write Data Parity Error Detected	This bit is set when a PCI inbound write data parity error is detected and the PCI Inbound Write Data Parity Error Enable bit of the Error Enable Register is set.
6		Reserved	Always read as zero.
5		Reserved	Always read as zero. **Note:** The status bit for the detection of master aborts is in the PCI Status Register.
4		Reserved	Always read as zero. **Note:** The status bit for the detection of target aborts is in the PCI Status Register.

3		Reserved	Always read as zero. **Note:** The split completion error is indicated in the PCI-X Status Register.
2	APED	Addr/Attrib Parity Error Detected	This bit is set when the PLB-PCIX Bridge is the target on the PCI bus and detects an address or attribute parity error and the Address/Attribute Parity Error Enable bit of the Error Enable Register is set and the Parity Error Response bit of the PCI Command Register is set.
1	DTE	Delayed Read Discard Timer Expired	This bit is set when the PLB-PCIX Bridge detects that the Discard Timer Expires bit (PCI-Conv Only) and the Delayed Read Discard Timer Expired Error Enable bit of the Error Enable Register is set.
0	PSEE	PCI_SERR# Received	This bit is set when PCI_SERR# is asserted and the PCI_SERR# Receive as Error Enable bit of the Error Enable Register is set.

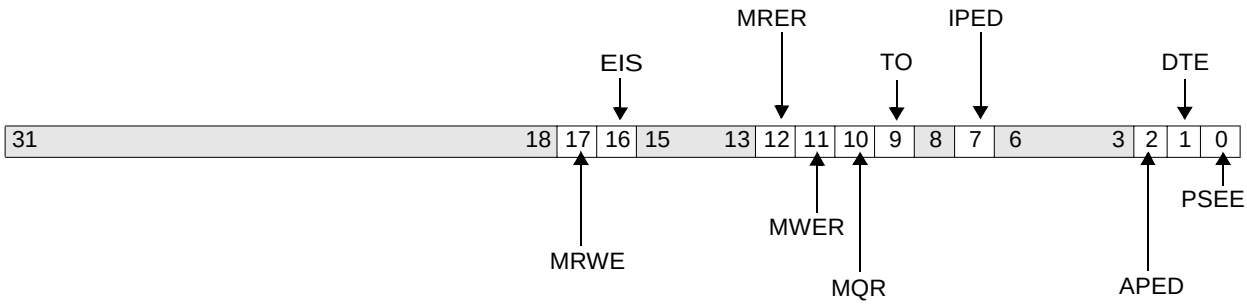


Figure 18-34. PCI-X Error Status (PCIX0_ERRSTS)

31:18		Reserved	Always read as zero.
17	MRWE	MRdErr or MWrErr Signaled	This bit is set whenever an error occurs that causes PLB-PCIX Bridge to assert PLB Read MERR or PLB Write MERR.
16	EIS	ERROR_INT Signaled	This bit is set whenever an error occurs that causes PLB-PCIX Bridge to assert ERROR_INT.
15:13		Reserved	Always read as zero.
12	MRER	MRdErr Received	This bit is set when PLB-PCIX Bridge is a PLB master and receives PLB Read MERR.
11	MWER	MWrErr Received	This bit is set when PLB-PCIX Bridge is a PLB master and receives PLB Write MERR.
10	MQR	Mirq Received	This bit is set when PLB-PCIX Bridge is a PLB master and receives MIRQ.
9	TO	Timeout	This bit is set when PLB-PCIX Bridge is a PLB master and receives Timeout.
8		Reserved	Always read as zero.
7	IPED	PCI Inbound Write Data Parity Error Detected	This bit is set when a PCI inbound write data parity error is detected and the PCI Inbound Write Data Parity Error Enable bit of the Error Enable Register is set.
6		Reserved	Always read as zero.

5		Reserved	Always read as zero. **Note:** The status bit for the detection of master aborts is in the PCI Status Register.
4		Reserved	Always read as zero. **Note:** The status bit for the detection of target aborts is in the PCI Status Register.
3		Reserved	Always read as zero. **Note:** The split completion error is indicated in the PCI-X Status Register.
2	APED	Addr/Attrib Parity Error Detected	This bit is set when the PLB-PCIX Bridge is the target on the PCI bus and detects an address or attribute parity error and the Address/Attribute Parity Error Enable bit of the Error Enable Register is set and the Parity Error Response bit of the PCI Command Register is set.
1	DTE	Delayed Read Discard Timer Expired	This bit is set when the PLB-PCIX Bridge detects that the Discard Timer Expires bit (PCI-Conv Only) and the Delayed Read Discard Timer Expired Error Enable bit of the Error Enable Register is set.
0	PSEE	PCI_SERR# Received	This bit is set when PCI_SERR# is asserted and the PCI_SERR# Receive as Error Enable bit of the Error Enable Register is set.

18.18.5.3 PCI-X PLB Slave Error Attribute Register (PCIX0_PLBBESR)

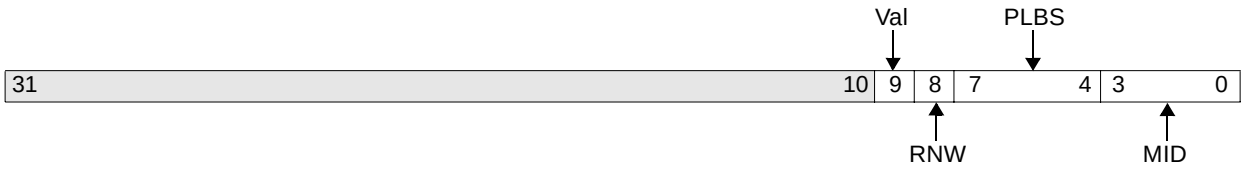


Figure 0-28. PCI-X PLB Slave Error Attribute Register (PCIX0_PLBBESR)

31:10		Reserved	Always read as zero.
9	Val	Valid	Indicates if bits 8:0 are valid (This bit is only meaningful when an outbound error is indicated in the PCIX0_ERRSTS register). **Note:** It is possible for a parity error to be detected too late for the PLB information to be saved.
8	RNW	RNW	RNW from the PLB master
7:4	PLBS	size	Size from the PLB master
3:0	MID	masterID	MasterID of the PLB master

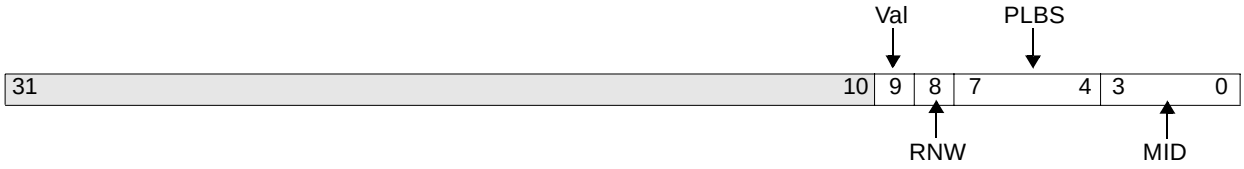


Figure 18-35. PCI-X PLB Slave Error Attribute Register (PCIX0_PLBBESR)

31:10		Reserved	Always read as zero.
9	Val	Valid	Indicates if bits 8:0 are valid (This bit is only meaningful when an outbound error is indicated in the PCIX0_ERRSTS register). **Note:** It is possible for a parity error to be detected too late for the PLB information to be saved.
8	RNW	RNW	RNW from the PLB master
7:4	PLBS	size	Size from the PLB master
3:0	MID	masterID	MasterID of the PLB master

18.18.5.4 PCI-X PLB Slave Error Address Low (PCIX0_PLBBEARL)

The PLB Slave Error Address Low Register contains the lower 32 bits of address associated with an error on the PLB when PLB-PCIX Bridge is a PLB slave. This register is read-only.

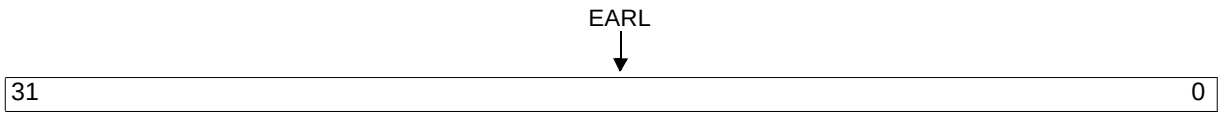


Figure 0-29. PCI-X PLB Slave Error Address Low (PCIX0_PLBEARL)

31:0	EARL	PLB Slave Error Address Low

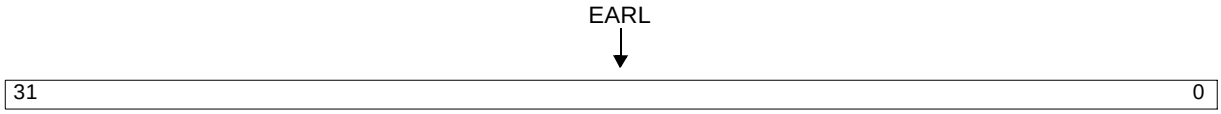


Figure 18-36. PCI-X PLB Slave Error Address Low (PCIX0_PLBEARL)

31:0	EARL	PLB Slave Error Address Low

18.18.5.5 PCI-X PLB Slave Error Address High (PCIX0_PLBEARH)

The PLB Slave Error Address High Register contains the higher 32 bits of address associated with an error on the PLB when PLB-PCIX Bridge is a PLB slave. This register is read-only.

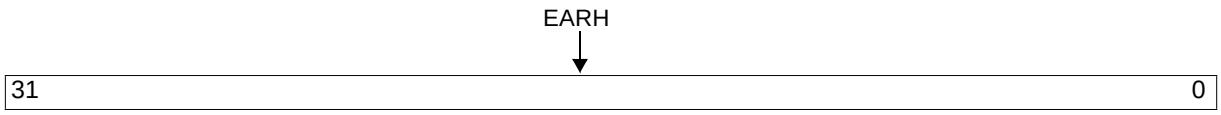


Figure 0-30. PCI-X PLB Slave Error Address High (PCIX0_PLBEARH)

31:0	EARH	PLB Slave Error Address High

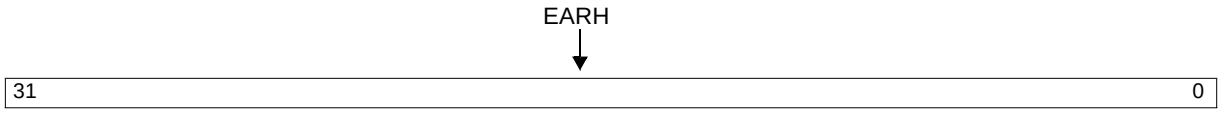


Figure 18-37. PCI-X PLB Slave Error Address High (PCIX0_PLBEARH)

31:0	EARH	PLB Slave Error Address High

18.18.6 POM Registers

The following registers control the mapping of PLB address space to PCI memory space.

18.18.6.1 PCI-X POM 0 Local Low Address (PCIX0_POM0LAL)

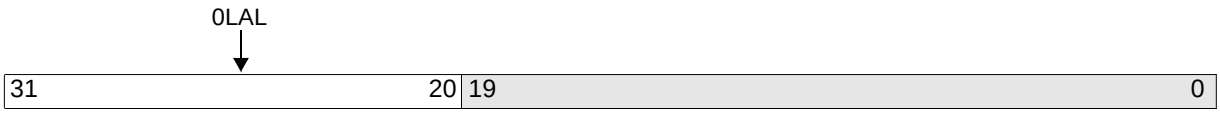


Figure 0-31. PCI-X POM 0 Local Low Address (PCIX0_POM0LAL)

31:20	0LAL	Local Address Low	Defines the starting address of range 0 in PLB space that is mapped to PCI memory. Only the bits that are 1 in the POM 0 size are actually used to determine the starting address, all other bits are don't cares.
19:0		Reserved	Returns zero when read.

Figure 18-38. PCI-X POM 0 Local Low Address (PCIX0_POM0LAL)

31:20	0LAL	Local Address Low	Defines the starting address of range 0 in PLB space that is mapped to PCI memory. Only the bits that are 1 in the POM 0 size are actually used to determine the starting address, all other bits are don't cares.
19:0		Reserved	Returns zero when read.

18.18.6.2 PCI-X POM 0 Local High Address (PCIX0_POM0LAH)

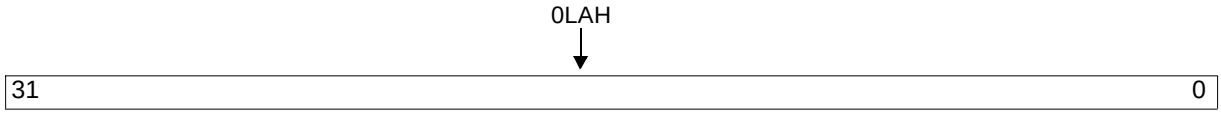


Figure 0-32. PCI-X POM 0 Local High Address (PCIX0_POM0LAH)

31:0	0LAH	Local Address High	Defines the upper 32 bits of the starting address of range 0 in PLB space that is mapped to PCI memory. All bits are writable.

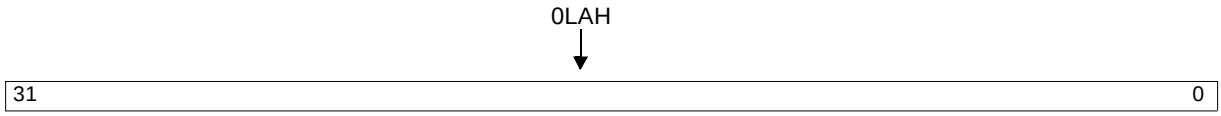


Figure 18-39. PCI-X POM 0 Local High Address (PCIX0_POM0LAH)

31:0	0LAH	Local Address High	Defines the upper 32 bits of the starting address of range 0 in PLB space that is mapped to PCI memory. All bits are writable.

18.18.6.3 PCI-X POM 0 Size/Attribute Register (PCIX0_POM0SA)

This register controls the size and attributes of the PLB space mapped to PCI memory for range 0.

Figure 0-33. PCI-X POM 0 Size/Attribute Register (PCIX0_POM0SA)

31:20	SIZE	Size of POM0.	The size of the POM0 can range from 1M to 4G and must be a power of 2. To determine the value to program in this field, use the following process: 1. Represent the desired size with a 32-bit number. Use only one bit set in that number since it is a power of two. Use all zeros to represent 4 GB. 2 Set all the bits to the left of the set bit. 3. Store bits [31:20] of the resulting number in this field. For example, if the desired POM0 size is 16M, start with 16x1024x1024 = 0100_0000h. Set all the bits to the left of the set bit (FF00_0000h). Bits [31:20] of that number are FF0h; therefore, FF0h should be stored in this field.
19:1		Reserved	Always read as zero.
0	En	Enable Mapping 0 Disable 1 Enable	This bit determines if range 0 is enabled to map PLB space to PCI memory space. **Note:** The POM 0 Local Address, POM 0 PCI Low Address, and POM 0 PCI High Address must be initialized before enabling. This field has a value of 0 after reset.

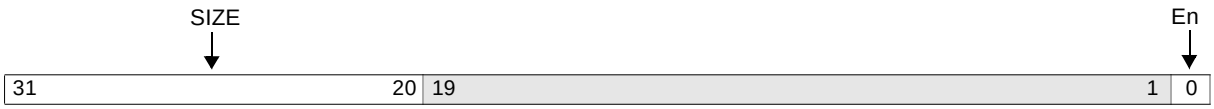


Figure 18-40. PCI-X POM 0 Size/Attribute Register (PCIX0_POM0SA)

31:20	SIZE	Size of POM0.	The size of the POM0 can range from 1M to 4G and must be a power of 2. To determine the value to program in this field, use the following process: 1. Represent the desired size with a 32-bit number. Use only one bit set in that number since it is a power of two. Use all zeros to represent 4 GB. 2 Set all the bits to the left of the set bit. 3. Store bits [31:20] of the resulting number in this field. For example, if the desired POM0 size is 16M, start with 16x1024x1024 = 0100_0000h. Set all the bits to the left of the set bit (FF00_0000h). Bits [31:20] of that number are FF0h; therefore, FF0h should be stored in this field.
19:1		Reserved	Always read as zero.
0	En	Enable Mapping 0 Disable 1 Enable	This bit determines if range 0 is enabled to map PLB space to PCI memory space. **Note:** The POM 0 Local Address, POM 0 PCI Low Address, and POM 0 PCI High Address must be initialized before enabling. This field has a value of 0 after reset.

18.18.6.4 PCI-X POM 0 PCI Address Low (PCIX0_POM0PCIAL)

This register along with POM 0 PCI High Address defines the PCI address that is generated in response to PLB access to range 0. The register defines the lower 32 bits of the POM 0 PCI address. Only the bits that are "1" in the POM 0 Mask are actually passed to the PCI address. The other (least significant) bits of the PCI address are passed through from the PLB address.

Figure 0-34. PCI-X POM 0 PCI Address Low (PCIX0_POM0PCIAL)

31:20	0PAL	PCI Low Address	
19:0		Reserved	Always read as zero.

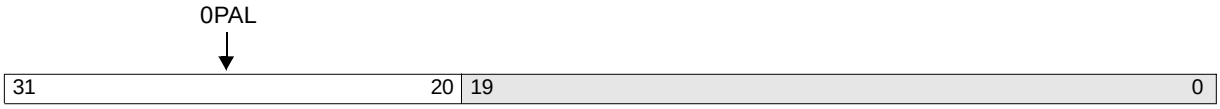


Figure 18-41. PCI-X POM 0 PCI Address Low (PCIX0_POM0PCIAL)

31:20	0PAL	PCI Low Address	
19:0		Reserved	Always read as zero.

18.18.6.5 PCI-X POM 0 PCI Address High (PCIX0_POM0PCIAH)

This register along with POM 0 PCI Low Address defines the PCI address that is generated in response to PLB access to range 0. The register defines the higher 32 bits of the POM 0 PCI Address.

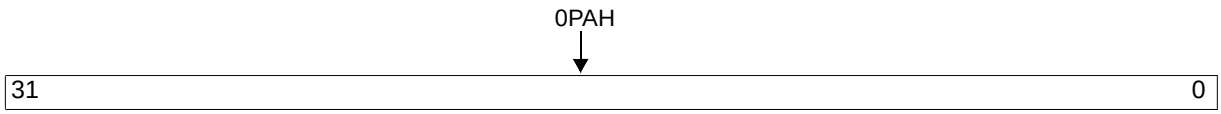


Figure 0-35. PCI-X POM 0 PCI Address High (PCIX0_POM0PCIAH)

31:0	0PAH	PCI High Address

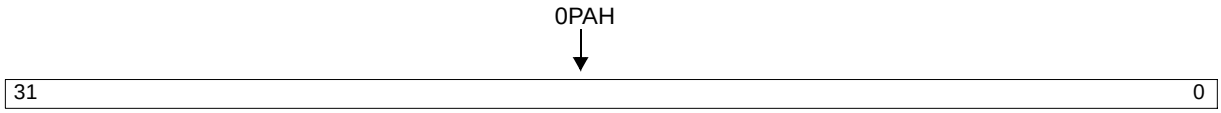


Figure 18-42. PCI-X POM 0 PCI Address High (PCIX0_POM0PCIAH)

31:0	0PAH	PCI High Address

18.18.6.6 PCI-X POM 1 Local Address Low (PCIX0_POM1LAL)

This register defines the starting address of range 1 in PLB space that is mapped to PCI memory. See ~~"PCI-X POM 0 Local Low Address (PCIX0_POM0LAL)" on page 73~~*PCI-X POM 0 Local Low Address (PCIX0_POM0LAL)* on page 605 for details.

Figure 0-36. PCI-X POM 1 Local Address Low (PCIX0_POM1LAL)

31:20	1LAL	Local Address Low	Defines the starting address of range 1 in PLB space that is mapped to PCI memory.
19:0		Reserved	Returns zero when read.

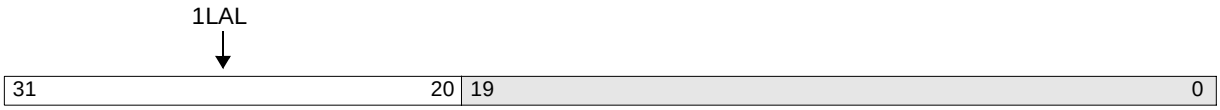


Figure 18-43. PCI-X POM 1 Local Address Low (PCIX0_POM1LAL)

31:20	1LAL	Local Address Low	Defines the starting address of range 1 in PLB space that is mapped to PCI memory.
19:0		Reserved	Returns zero when read.

18.18.6.7 PCI-X POM 1 Local Address High (PCIX0_POM1LAH)

This register defines the upper 32 bits of the starting address of range 1 in PLB space that is mapped to PCI memory. See ~~"PCI-X POM 0 Local High Address (PCIX0_POM0LAH)" on page 74~~*PCI-X POM 0 Local High Address (PCIX0_POM0LAH)* on page 606.

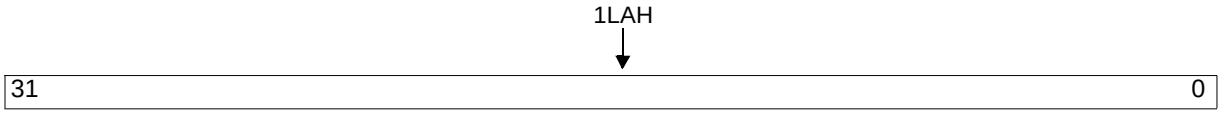


Figure 0-37. PCI-X POM 1 Local Address High (PCIX0_POM1LAH)

31:0	1LAH	Local Address High	Defines the starting address of range 1 in PLB space that is mapped to PCI memory. Only the bits that are 1 in the POM 1 Size are actually used to determine the starting address. All other bits are don't cares.

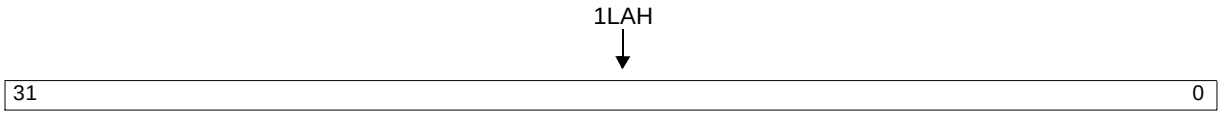


Figure 18-44. PCI-X POM 1 Local Address High (PCIX0_POM1LAH)

31:0	1LAH	Local Address High	Defines the starting address of range 1 in PLB space that is mapped to PCI memory. Only the bits that are 1 in the POM 1 Size are actually used to determine the starting address. All other bits are don't cares.

18.18.6.8 PCI-X POM 1 Size/Attribute Register (PCIX0_POM1SA)

This register controls the size and attributes of the PLB space mapped to PCI memory for range 1.

Figure 0-38. PCI-X POM 1 Size/Attribute Register (PCIX0_POM1SA)

31:20	SIZE	Size of POM 1	The size of the POM 1 can range from 1M to 4G and must be a power of 2. To determine the value to program in this field, use the following process: 1. Represent the desired size with a 32-bit number. Use only one bit set in that number since it is a power of two. 2 Set all the bits to the left of the set bit. 3. Store bits [31:20] of the resulting number in this field. For example, if the desired POM 1 size is 16M, start with 16x1024x1024 = 0100_0000h. Set all the bits to the left of the set bit (FF00_0000h). Bits [31:20] of that number are FF0h; therefore, FF0h should be stored in this field.
19:1		Reserved	Always read as zero.
0	En	Enable mapping 1 Enable 0 Disable	This bit determines if range 1 is enabled to map PLB space to PCI memory space. **Note:** The POM 1 Local Address, POM 1 PCI Low Address, and POM 1 PCI High Address must be initialized before enabling. This field has a value of 0 after reset.

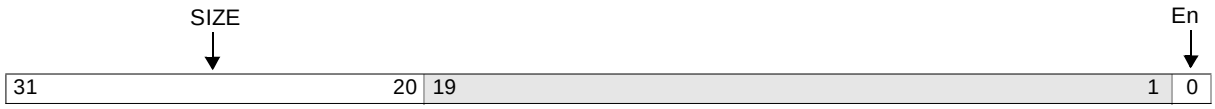


Figure 18-45. PCI-X POM 1 Size/Attribute Register (PCIX0_POM1SA)

31:20	SIZE	Size of POM 1	The size of the POM 1 can range from 1M to 4G and must be a power of 2. To determine the value to program in this field, use the following process: 1. Represent the desired size with a 32-bit number. Use only one bit set in that number since it is a power of two. Use all zeros to represent 4 GB. 2 Set all the bits to the left of the set bit. 3. Store bits [31:20] of the resulting number in this field. For example, if the desired POM 1 size is 16M, start with 16x1024x1024 = 0100_0000h. Set all the bits to the left of the set bit (FF00_0000h). Bits [31:20] of that number are FF0h; therefore, FF0h should be stored in this field.
19:1		Reserved	Always read as zero.
0	En	Enable mapping 1 Enable 0 Disable	This bit determines if range 1 is enabled to map PLB space to PCI memory space. **Note:** The POM 1 Local Address, POM 1 PCI Low Address, and POM 1 PCI High Address must be initialized before enabling. This field has a value of 0 after reset.

18.18.6.9 PCI-X POM 1 PCI Address Low (PCIX0_POM1PCIAL)

This register along with POM 1 PCI High Address defines the PCI address that is generated in response to PLB access to range 1. See section ~~"PCI-X POM 0 PCI Address Low (PCIX0_POM0PCIAL)" on page 76~~*PCI-X POM 0 PCI Address Low (PCIX0_POM0PCIAL)* on page 608 for details.

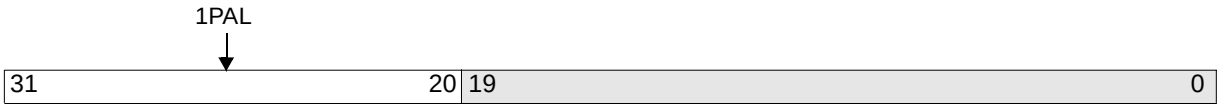


Figure 0-39. PCI-X POM 1 PCI Address Low (PCIX0_POM1PCIAL)

31:20	1PAL	PCI Address Low	
19:0		Reserved	Returns zero when read.

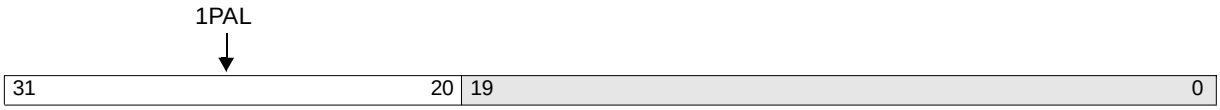


Figure 18-46. PCI-X POM 1 PCI Address Low (PCIX0_POM1PCIAL)

31:20	1PAL	PCI Address Low	
19:0		Reserved	Returns zero when read.

18.18.6.10 PCI-X POM 1 PCI Address High (PCIX0_POM1PCIAH)

This register along with POM 1 PCI Low Address defines the PCI address that is generated in response to PLB access to range 0. See ~~"PCI-X POM 0 PCI Address High (PCIX0_POM0PCIAH)" on page 77~~*PCI-X POM 0 PCI Address High (PCIX0_POM0PCIAH)* on page 609 for details.

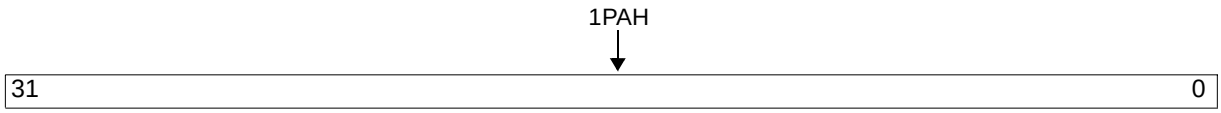


Figure 0-40. PCI-X POM 1 PCI Address High (PCIX0_POM1PCIAH)

31:0	1PAH	PCI Address High

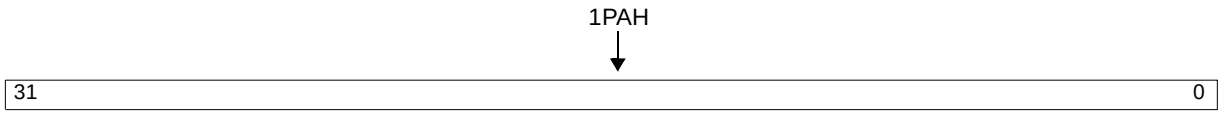


Figure 18-47. PCI-X POM 1 PCI Address High (PCIX0_POM1PCIAH)

31:0	1PAH	PCI Address High

18.18.6.11 PCI-X POM 2 Size/Attribute Register (PCIX0_POM2SA)

This register defines the attributes of the PLB space mapped to PCI memory for range 2.

Figure 0-41. PCI-X POM 2 Size/Attribute Register (PCIX0_POM2SA)

31:1		Reserved	Always read as zero.
0	En	Enables mapping 1 Enable 0 Disable	This bit determines if range 2 is enabled to map PLB space to PCI memory space. A value of 1 enables the mapping.

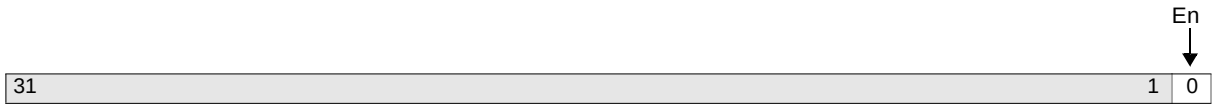


Figure 18-48. PCI-X POM 2 Size/Attribute Register (PCIX0_POM2SA)

31:1		Reserved	Always read as zero.
0	En	Enables mapping 1 Enable 0 Disable	This bit determines if range 2 is enabled to map PLB space to PCI memory space. A value of 1 enables the mapping.

There are no POM 2 Local Address and POM 2 PCI registers. The values for these registers, translation map, and the size are hardcoded for POM 2. The following table (~~Table 18-6~~*Table 18-6*) summarizes the hard-coded address map defined by POM 2.

Table 18-6. POM 2 Hardcoded Address Map

PLB Address Range	PCI Address Range
8000_0000_0000_0000 thru FFFF_FFFF_FFFF_FFFF	0000_0000_0000_0000 thru 7FFF_FFFF_FFFF_FFFF

18.18.7 PIM Registers

The following registers control how PCI memory space is mapped to PLB address space.

18.18.7.1 PCI-X PIM0 Size/Attribute Register (PCIX0_PIM0SA)

This register defines the size and attributes of the region of PCI memory space that is mapped to local (PLB) space through PIM0. It affects the way PCI configuration register BAR0 works.

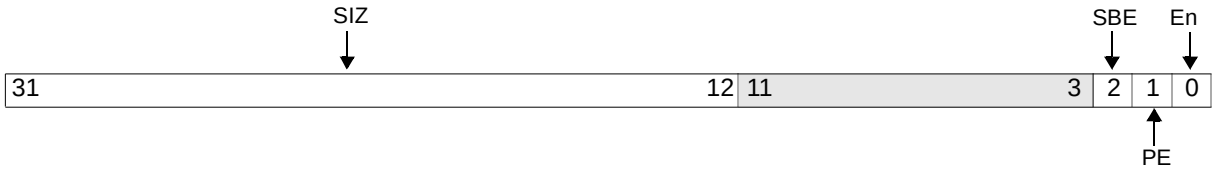


Figure 0-42. PCI-X PIM 0 Size/Attribute Register (PCIX0_PIM0SA)

31:12	SIZ	Size of PIM0	The size of the PIM0 can range from 4K bytes to 4GB and must be a power of 2. To determine the value to program in this field, use the following process: 1. Represent the desired size with a 32-bit number. Use only one bit set in that number since it is a power of two. 2. Set all the bits to the left of the set bit. 3. Store bits [31:12] of the resulting number in this field. For example, if the desired PIM0 size is 16M, start with 16x1024x1024 =0100_0000h. Set all the bits to the left of the set bit (FF00_0000h). Bits [31:12] of that number are FF00_0h; therefore, FF00_0h should be stored in this field.
11:3		Reserved	Always read as zero.
2	SBE	Split BAR0 Enable	This bit enables splitting the BAR0-specified region into two ranges. When this bit is low, the entire BAR0 region is mapped to one region of PLB space, as determined by PIM0 Local Address registers. When this bit is high, the BAR0 range is divided into two regions: the first 1 KB of the BAR0 range is mapped to either a PLB region as determined by the PIM1 Local Address registers, or to internal registers for using Simple Message Passing and/or Inbound MSI. The second region is the remainder of the BAR0 range and is mapped to the PLB address space as determined by the PIM0 Local Address Register. If this bit is low, the entire BAR0 address space is mapped to PLB address space as determined by the PIM0 Local Address Register. This bit is 0 after reset.
1	PE	Prefetch Enable	This bit determines whether this region is prefetchable. The value of this bit is also readable in BAR0 low, bit 3.
0	En	Enable	If set, enables the PCI BAR0 registers and decoders.

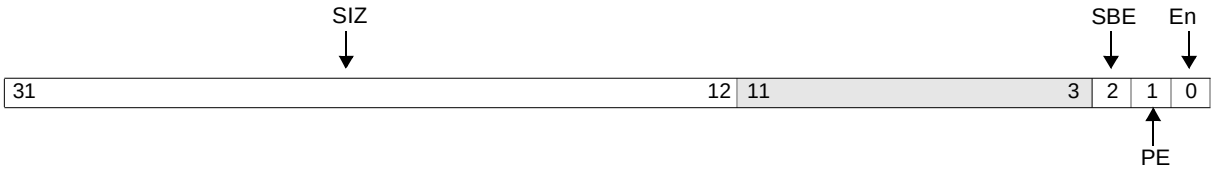


Figure 18-49. PCI-X PIM 0 Size/Attribute Register (PCIX0_PIM0SA)

31:12	SIZ	Size of PIM0	The size of the PIM0 can range from 4K bytes to 4GB and must be a power of 2. To determine the value to program in this field, use the following process: 1. Represent the desired size with a 32-bit number. Use only one bit set in that number since it is a power of two. Use all zeros to represent 4GB. 2. Set all the bits to the left of the set bit. 3. Store bits [31:12] of the resulting number in this field. For example, if the desired PIM0 size is 16M, start with 16x1024x1024 =0100_0000h. Set all the bits to the left of the set bit (FF00_0000h). Bits [31:12] of that number are FF00_0h; therefore, FF00_0h should be stored in this field.
11:3		Reserved	Always read as zero.
2	SBE	Split BAR0 Enable	This bit enables splitting the BAR0-specified region into two ranges. When this bit is low, the entire BAR0 region is mapped to one region of PLB space, as determined by PIM0 Local Address registers. When this bit is high, the BAR0 range is divided into two regions: the first 1 KB of the BAR0 range is mapped to the internal registers for using Simple Message Passing and/or Inbound MSI. The second region is the remainder of the BAR0 range and is mapped to the PLB address space as determined by the PIM0 Local Address Register. If this bit is low, the entire BAR0 address space is mapped to PLB address space as determined by the PIM0 Local Address Register. This bit is 0 after reset.
1	PE	Prefetch Enable	This bit determines whether this region is prefetchable. The value of this bit is also readable in BAR0 low, bit 3.
0	En	Enable	If set, enables the PCI BAR0 registers and decoders.

18.18.7.2 PCI-X PIM0 Local Low Address (PCIX0_PIM0LAL)

This register defines the local (PLB) address that is generated in response to a PCI access to local (PLB) space through PIM0. Only the bits that are 1 in the size field of the PIM0 Size/Attribute Register are actually passed to the PLB address. The other (least significant) bits of the PLB address are passed through from the PCI address. Only bits 31:12 are writable, bits 11:0 are always zero.

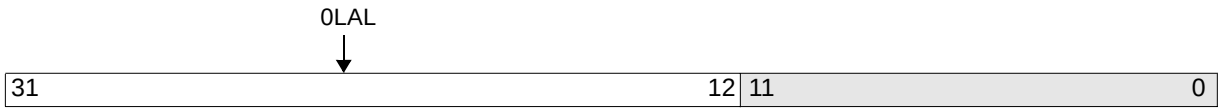


Figure 0-43. PIM 0 Local Low Address (PCIL0_PIM0LAL)

31:12	0LAL	Local Address Low	Defines the starting address of range 0 in PLB space that is mapped to PCI memory. Only the bits that are 1 in the PIM 0 Size/Attribute Register are actually passed to the PLB address.
11:0		Reserved	Returns zero when read.

Figure 18-50. PCI-X PIM 0 Local Low Address (PCIX0_PIM0LAL)

31:12	0LAL	Local Address Low	Defines the starting address of range 0 in PLB space that is mapped to PCI memory. Only the bits that are 1 in the PIM 0 Size/Attribute Register are actually passed to the PLB address.
11:0		Reserved	Returns zero when read.

18.18.7.3 PCI-X PIM0 Local High Address (PCIX0_PIM0LAH)

This register defines the upper 32 bits of the local (PLB) address that is generated in response to a PCI access to local (PLB) space through PIM0. All of these bits are passed to the upper PLB address.

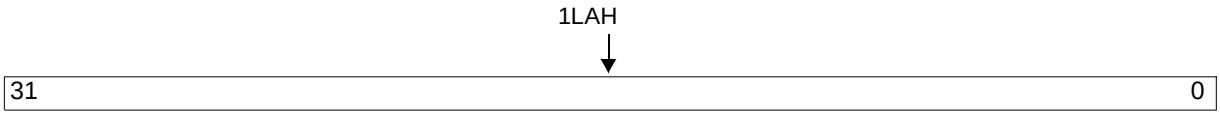


Figure 0-44. PCI-X PIM 1 Local High Address (PCIX0_PIM1LAH)

31:0	1LAH	Local Address High	Defines the upper 32 bits of the starting address of range 1 in PLB space that is mapped to PCI memory. All bits are writable.

All of these bits are passed to the upper PLB address.

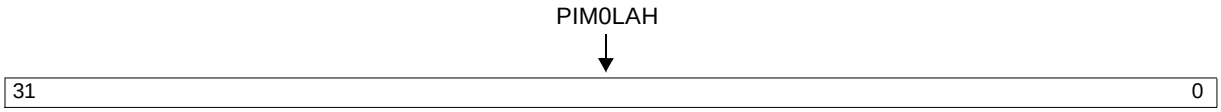


Figure 18-51. PCI-X PIM 0 Local High Address (PCIX0_PIM0LAH)

31:0	PIM0LAH	Local Address High	Defines the upper 32-bits of the starting address of range 0 in PLB space that is mapped to PCI memory.

18.18.7.4 PCI-X PIM1 Size/Attribute Register (PCIX0_PIM1SA)

This register defines the size and attributes of the region of PCI I/O space that is mapped to local (PLB) space through PIM 1. It affects the way PCI configuration register PIM1 BAR works.

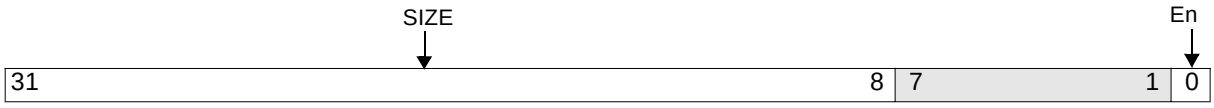


Figure 0-45. PCI-X PIM 1 Size/Attribute Register (PCIX0_PIM1SA)

31:8	SIZE	Size	SIZE defines the size of the region of PCI I/O space that is mapped to local (PLB) space through PIM 1. The size for the PIM1 is hardcoded to 256 bytes; therefore, read to these bits returns all ones. This size has no effect on the PIM0 1 KB region.
7:1		Reserved	Returns zero when read.
0	En	Enable	If set, enables the PCI BAR1 registers and decoder.

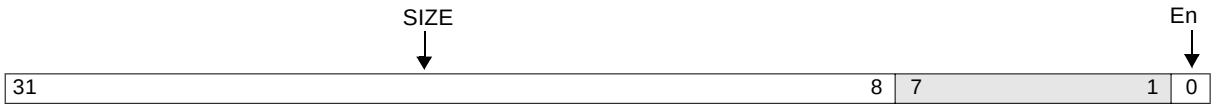


Figure 18-52. PCI-X PIM 1 Size/Attribute Register (PCIX0_PIM1SA)

31:8	SIZE	Size	SIZE defines the size of the region of PCI I/O space that is mapped to local (PLB) space through PIM 1. The size for the PIM1 is hardcoded to 256 bytes; therefore, read to these bits returns all ones. This size has no effect on the PIM0 1 KB region.
7:1		Reserved	Returns zero when read.
0	En	Enable	If set, enables the PCI BAR1 registers and decoder.

18.18.7.5 PCI-X PIM1 Local Low Address (PCIX0_PIM1LAL)

This register defines the local (PLB) address that is generated in response to a PCI I/O access to local (PLB) space through PIM1. Only the bits that are "1" in size field of the PIM1 Size/Attribute Register are actually passed to the PLB address. The other (least significant) bits of the PLB address are passed through from the PCI address. Only bits 31:12 are writable, bits 11:0 are always zero. If the "Split BAR0 Enable" bit in the PIM 0 Size/Attribute Register has a value of "1," see Section ~~18.18.7.1 "PCI-X PIM0 Size/Attribute Register (PCIX0_PIM0SA),"~~*PCI-X PIM0 Size/Attribute Register (PCIX0_PIM0SA)* on page 616 for description of the register behavior regarding bit 2- "Split BAR0 Enable.

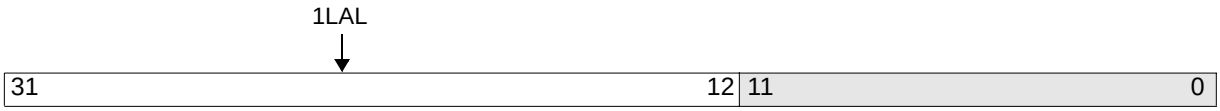


Figure 0-46. PCI-X PIM 1 Local Low Address (PCIX0_PIM1LAL)

31:12	1LAL	Local Address Low	Defines the starting address of range 1 in PLB space that is mapped to PCI memory. Only the bits that are "1" in the PIM 1 Size/Attribute Register are actually passed to the PLB address. Only bits 31:12 are writable.
11:0		Reserved	Returns zero when read.

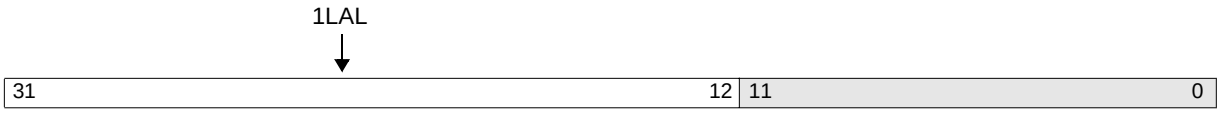


Figure 18-53. PCI-X PIM 1 Local Low Address (PCIX0_PIM1LAL)

31:12	1LAL	Local Address Low	Defines the starting address of range 1 in PLB space that is mapped to PCI memory. Only the bits that are "1" in the PIM 1 Size/Attribute Register are actually passed to the PLB address. Only bits 31:12 are writable.
11:0		Reserved	Returns zero when read.

18.18.7.6 PCI-X PIM1 Local High Address (~~PCIXO_PIM1LAH~~PCIX0_PIM1LAH)

This register defines the upper 32 bits of the local (PLB) address that is generated in response to a PCI I/O access to local (PLB) space through PIM1. The value of this register is passed on to the PLB upper address.

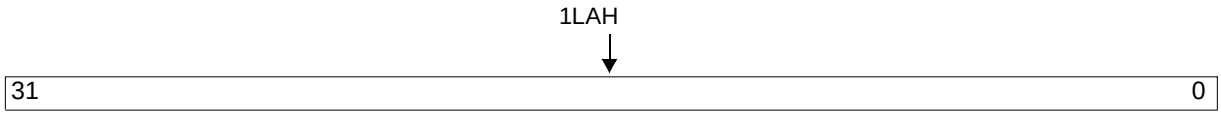


Figure 0-47. PCI-X PIM 1 Local High Address (PCIX0_PIM1LAH)

31:0	1LAH	Local Address High	Defines the upper 32 bits of the starting address of range 1 in PLB space that is mapped to PCI memory. All bits are writable.

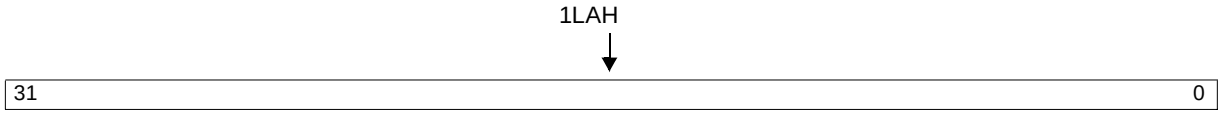


Figure 18-54. PCI-X PIM 1 Local High Address (PCIX0_PIM1LAH)

31:0	1LAH	Local Address High	Defines the upper 32 bits of the starting address of range 1 in PLB space that is mapped to PCI memory.

18.18.7.7 PCI-X PIM2 Size/Attribute Register (PCIX0_PIM2SA)

This register defines the size and attributes of the region of PCI memory space that is mapped to PLB space through PIM2. It affects the way PIM2 BAR Register and PCI Expansion ROM BAR Register work.

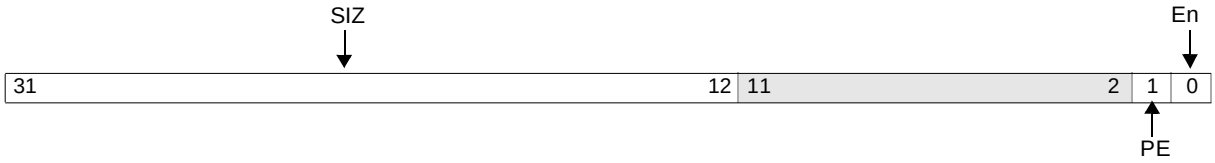


Figure 18-55. PCI-X PIM 2 Size/Attribute Register (PCIX0_PIM2SA)

31:12	SIZ	Size of PIM2	The size of the PIM2 can range from 4K bytes to 4GB and must be a power of 2. To determine the value to program in this field, use the following process: 1. Represent the desired size with a 32-bit number. Use only one bit set in that number since it is a power of two. Use all zeros to represent 4GB 2. Set all the bits to the left of the set bit. 3. Store bits [31:12] of the resulting number in this field. For example, if the desired PIM2 size is 16M, start with 16x1024x1024 =0100_0000h. Set all the bits to the left of the set bit (FF00_0000h). Bits [31:12] of that number are FF00_0h; therefore, FF00_0h should be stored in this field.
11:3		Reserved	Returns zero when read.
2		Reserved	Must be be set to zero.
1	PE	Prefetch Enable	This bit determines if this region is prefetchable. The value of this bit is also readable in BAR2 low, bit 3.
0	En	Enable	If set, enables the PCI PIM2 BAR Register and the Expansion ROM BAR Register and the decoder.

Figure 0-48. PCI-XPIM 2 Size/Attribute Register (PCIX0_PIM2SA)

31:12	SIZ	Size of PIM2	The size of the PIM2 can range from 4K bytes to 4GB and must be a power of 2. To determine the value to program in this field, use the following process: 1. Represent the desired size with a 32-bit number. Use only one bit set in that number since it is a power of two. 2. Set all the bits to the left of the set bit. 3. Store bits [31:12] of the resulting number in this field. For example, if the desired PIM2 size is 16M, start with 16x1024x1024 =0100_0000h. Set all the bits to the left of the set bit (FF00_0000h). Bits [31:12] of that number are FF00_0h; therefore, FF00_0h should be stored in this field.
11:3		Reserved	Returns zero when read.
2		Reserved	Must be be set to zero.
1	PE	Prefetch Enable	This bit determines if this region is prefetchable. The value of this bit is also readable in BAR2 low, bit 3.
0	En	Enable	If set, enables the PCI PIM2 BAR Register and the Expansion ROM BAR Register and the decoder.

18.18.7.8 PCI-X PIM2 Local Low Address (PCIX0_PIM2LAL)

This register defines the local (PLB) address that is generated in response to a PCI access to local (PLB) space through PIM2. Only the bits that are "1" in the size field of the PIM2 Size/Attribute Register are actually passed to the PLB address. The other (least significant) bits of the PLB address are passed through from the PCI address. Only bits 31:12 are writable, bits 11:0 are always zero.

If the Enable bit (bit 0) in the PCI Expansion ROM BAR Register at 30h is set, then PIM2 uses the PCI Expansion ROM Base Address Register instead of the BAR2 Register.

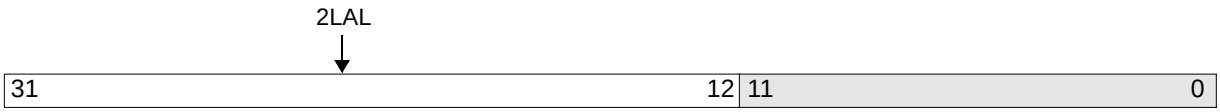


Figure 0-49. PCI-X PIM 2 Local Low Address (PCIX0_PIM2LAL)

31:12	2LAL	Local Address Low	Defines the starting address of range 2 in PLB space that is mapped to PCI memory. Only the bits that are 1 in the PIM 2 Size/Attribute Register are actually passed to the PLB address. Only bits 31:12 are writable.
11:0		Reserved	Returns zero when read.

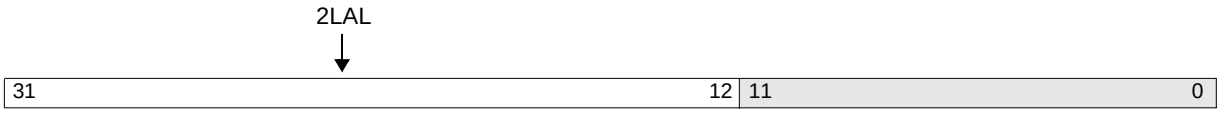


Figure 18-56. PCI-X PIM 2 Local Low Address (PCIX0_PIM2LAL)

31:12	2LAL	Local Address Low	Defines the starting address of range 2 in PLB space that is mapped to PCI memory. Only the bits that are 1 in the PIM 2 Size/Attribute Register are actually passed to the PLB address. Only bits 31:12 are writable.
11:0		Reserved	Returns zero when read.

18.18.7.9 PCI-X PIM2 Local Address High (PCIX0_PIM2LAH)

This register defines the upper 32 bits of the local (PLB) address that is generated in response to a PCI access to local (PLB) space through PIM2. The value of this register is passed onto the PLB upper address.

If the Enable bit (bit 0) in the PCI Expansion ROM BAR Register at 30h is set, then PIM2 uses the PCI Expansion ROM Base Address Register instead of the BAR2 Register.

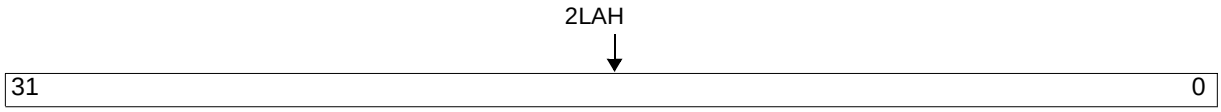


Figure 0-50. PCI-X PIM 2 Local Address High (PCIX0_PIM2LAH)

31:0	2LAH	Local Address High	Defines the upper 32 bits of the starting address of range 2 in PLB space that is mapped to PCI memory. All bits are writable.

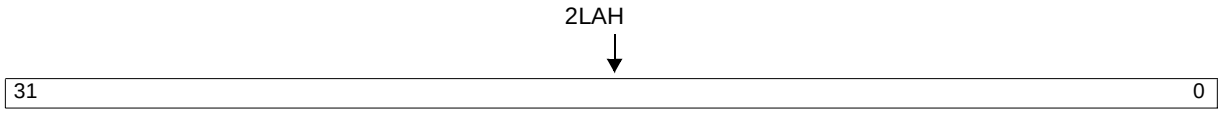


Figure 18-57. PCI-X PIM 2 Local Address High (PCIX0_PIM2LAH)

31:0	2LAH	Local Address High	Defines the upper 32 bits of the starting address of range 2 in PLB space that is mapped to PCI memory.

18.18.8 MSI Capability Block Definition Registers

This section describes the Outbound MSI (Message Signaled Interrupts) capability registers.

18.18.8.1 PCI-X MSI Capability Identifier (PCIX0_OMCAPID)

The Capability Identifier when read by system software as 05h indicates that the data structure currently being pointed to is the MSI capability structure.

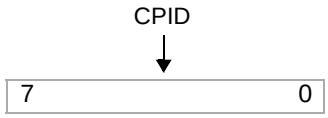


Figure 0-51. PCI-X MSI Capabilities Identifier (PCIX0_OMCAPID)

7:0	CPID	Message Signaled Interrupts (MSI) Capabilities Identifier

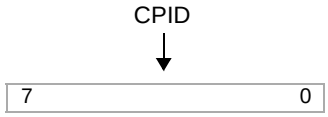


Figure 18-58. PCI-X MSI Capabilities Identifier (PCIX0_OMCAPID)

7:0	CPID	Message Signaled Interrupts (MSI) Capabilities Identifier

18.18.8.2 PCI-X Next Item Pointer (PCIX0_OMNIPTR)

This Next Item Pointer Register describes the location of the next item in the function's capability list. The Next Item Pointer default value points to the capability structure for PCI Power Management, located at D0h in the configuration space.

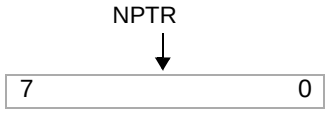


Figure 0-52. PCI-X MSI Next Item Pointer (PCIX0_OMNIPTR)

7:0	NPTR	Message Signaled Interrupts (MSI) Next Item Pointer

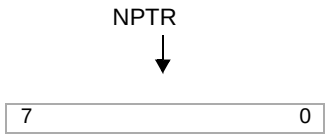


Figure 18-59. PCI-X MSI Next Item Pointer (PCIX0_OMNIPTR)

7:0	NPTR	Message Signaled Interrupts (MSI) Next Item Pointer

18.18.8.3 PCI-X Message Control Register (PCIX0_OMMC)

This 16-bit register provides system software control over MSI.

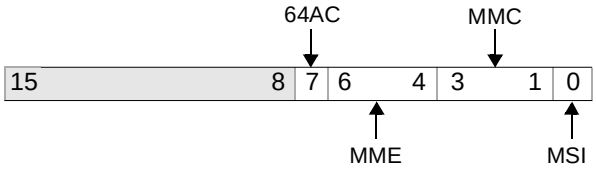


Figure 0-53. PCI-X Message Control Register (PCIX0_OMMC)

15:8		Reserved	Returns zero when read.
7	64AC	64-bit Address Capable	This bit is hardwired to a 1 and indicates that PLB-PCIX Bridge is capable of generating a 64-bit message address.
6:4	MME	Multiple Message Enable	System software writes to this field to indicate the number of allocated messages for PLB-PCIX Bridge. The number of allocated messages is aligned to a power of 2. Bit 6:5 of this field are hardwired to "0" and only bit 4 is writable. This field's state after reset is 000b.
3:1	MMC	Multiple Message Capable	System software reads this field to determine the number of messages requested by PLB-PCIX Bridge. The PLB-PCIX Bridge requests two messages and is indicated by hardwiring this field to a value of 001b.
0	MSI	MSI Enable 1 MSI enabled 0 MSI disabled	This read/write bit is used by the system to enable or disable MSI capability. This bit state after reset is 0 (MSI is disabled after reset).

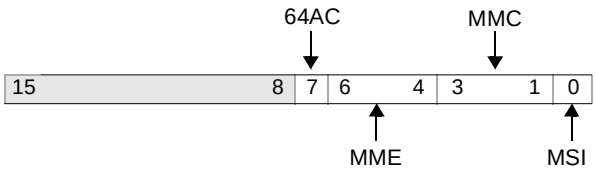


Figure 18-60. PCI-X Message Control Register (PCIX0_OMMC)

15:8		Reserved	Returns zero when read.
7	64AC	64-bit Address Capable	This bit is hardwired to a 1 and indicates that PLB-PCIX Bridge is capable of generating a 64-bit message address.

6:4	MME	Multiple Message Enable	System software writes to this field to indicate the number of allocated messages for PLB-PCIX Bridge. The number of allocated messages is aligned to a power of 2. Bit 6:5 of this field are hardwired to "0" and only bit 4 is writable. This field's state after reset is 000b.
3:1	MMC	Multiple Message Capable	System software reads this field to determine the number of messages requested by PLB-PCIX Bridge. The PLB-PCIX Bridge requests two messages and is indicated by hardwiring this field to a value of 001b.
0	MSI	MSI Enable 1 MSI enabled 0 MSI disabled	This read/write bit is used by the system to enable or disable MSI capability. This bit state after reset is 0 (MSI is disabled after reset).

18.18.8.4 PCI-X Message Address Register (PCIX0_OMMA)

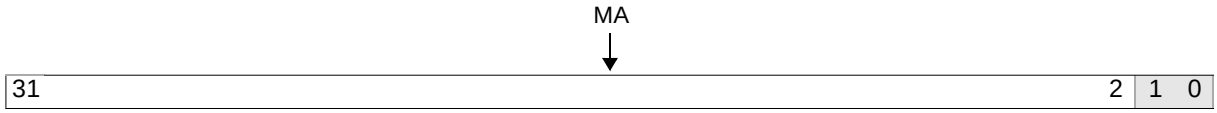


Figure 0-54. PCI-X Message Address Register (PCIX0_OMMA)

31:2	MA	Message Address	Indicates system specific message address. If the Message Enable bit (bit 0 of the Message Control Register) is set, the contents of this register specify the doubleword aligned address pci_ad[31:2] for the MSI memory write transaction. pci_ad[1:0] are driven to zero during the address phase. This field is read/write.
1:0		Reserved	Returns zero when read.

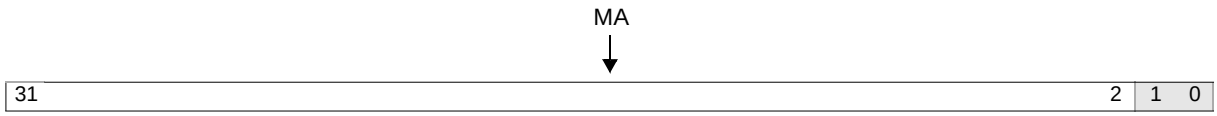


Figure 18-61. PCI-X Message Address Register (PCIX0_OMMA)

31:2	MA	Message Address	Indicates system specific message address. If the Message Enable bit (bit 0 of the Message Control Register) is set, the contents of this register specify the doubleword aligned address pci_ad[31:2] for the MSI memory write transaction. pci_ad[1:0] are driven to zero during the address phase. This field is read/write.
1:0		Reserved	Returns zero when read.

18.18.8.5 PCI-X Message Upper Address Register (PCIX0_OMMUA)

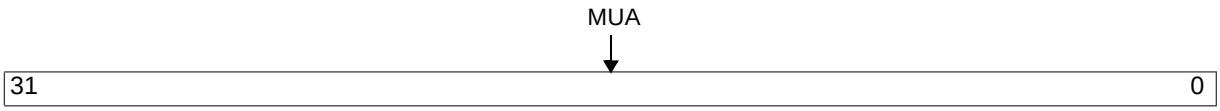


Figure 0-55. PCI-X Message Upper Address Register (PCIX0_OMMUA)

31:0	MUA	Message Upper Address	PLB-PCIX Bridge supports a 64-bit message address (bit 7 in Message Control Register is set to 1). The contents of this register specify the upper 32 bits of a 64-bit message address pci_ad[63:32]. If the contents of this register are zero, PLB-PCIX Bridge uses the 32-bit address specified by the Message Address Register. This field is read/write and is configured by the system.

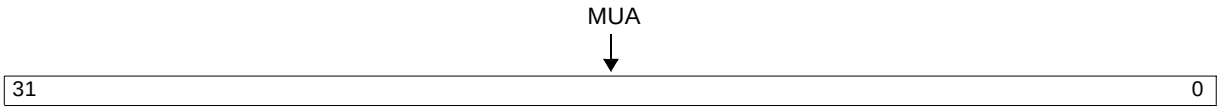


Figure 18-62. PCI-X Message Upper Address Register (PCIX0_OMMUA)

31:0	MUA	Message Upper Address	PLB-PCIX Bridge supports a 64-bit message address (bit 7 in Message Control Register is set to 1). The contents of this register specify the upper 32 bits of a 64-bit message address pci_ad[63:32]. If the contents of this register are zero, PLB-PCIX Bridge uses the 32-bit address specified by the Message Address Register. This field is read/write and is configured by the system.

18.18.8.6 PCI-X Message Data Register (PCIX0_OMMDATA)

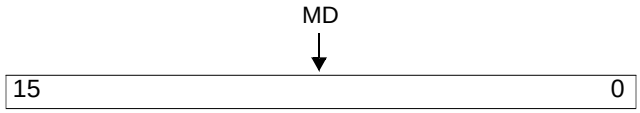


Figure 0-56. PCI-X Message Data Register (PCIX0_OMMDATA)

15:0	MD	Message Data	System-specified message. This field is written by the system. When PLB-PCIX Bridge issues an MSI message, it writes this value to the previously defined message address. If PLB-PCIX Bridge is allocated two messages by the system, the least significant bit of this field determines which message is being reported. If PLB-PCIX Bridge is allocated only one message by the system, then PLB-PCIX Bridge cannot modify this data on a MSI write.

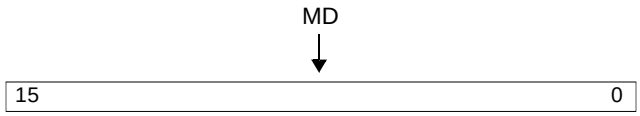


Figure 18-63. PCI-X Message Data Register (PCIX0_OMMDATA)

15:0	MD	Message Data	System-specified message. This field is written by the system. When PLB-PCIX Bridge issues an MSI message, it writes this value to the previously defined message address. If PLB-PCIX Bridge is allocated two messages by the system, the least significant bit of this field determines which message is being reported. If PLB-PCIX Bridge is allocated only one message by the system, then PLB-PCIX Bridge cannot modify this data on a MSI write.

18.18.8.7 PCI-X Message End of Interrupt Register (PCIX0_OMMEOI)

Figure 0-57. PCI-X Message End of Interrupt Register (PCIX0_OMMEOI)

7:1		Reserved	Returns zero when read.
0	MEOI	Message End of Interrupt	Following the generation of an outbound MSI, the outbound MSI unit must be rearmed before another outbound MSI can be generated. Writing a 1 to this bit rearms the MSI function. This bit automatically clears itself (always reads zero). EOI is required only when using MSI.

Figure 18-64. PCI-X Message End of Interrupt Register (PCIX0_OMMEOI)

7:1		Reserved	Returns zero when read.
0	MEOI	Message End of Interrupt	Following the generation of an outbound MSI, the outbound MSI unit must be rearmed before another outbound MSI can be generated. Writing a 1 to this bit rearms the MSI function. This bit automatically clears itself (always reads zero). EOI is required only when using MSI.

18.18.9 Power Management Register Block Definition Registers

This section describes the PCI Power Management Interface registers.

18.18.9.1 PCI-X Power Management Capability Identifier (PCIX0_PMCAPID)

The Capability Identifier when read by system software as "01h" indicates that PLB-PCIX Bridge supports power management and the data structure currently being pointed to is the PCI power management capability structure.

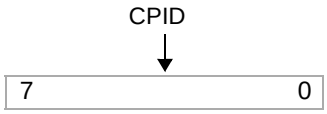


Figure 0-58. PCI-X Power Management Capabilities Identifier (PCIX0_PMCAPID)

7:0	CPID	Capabilities Identifier	When read by system software as 01h indicates that PLB-PCIX Bridge supports power management and the data structure currently being pointed to is the PCI power management capability structure.

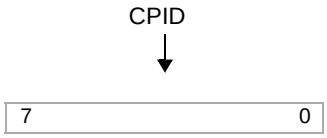


Figure 18-65. PCI-X Power Management Capabilities Identifier (PCIX0_PMCAPID)

7:0	CPID	Capabilities Identifier	When read by system software as 01h indicates that PLB-PCIX Bridge supports power management and the data structure currently being pointed to is the PCI power management capability structure.

18.18.9.2 PCI-X Power Management Next Item Pointer (PCIX0_PMNIPTR)

This Next Item Pointer Register describes the location of the next item in the function's capability list. The Next Item Pointer defaults to pointing to the capability structure for PCI-X located at DCh in the configuration space.

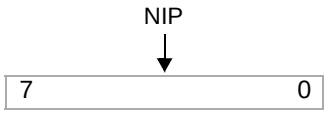


Figure 0-59. PCI-X Power Management Next Item Pointer (PCIX0_PMNIPTR)

7:0	NIP	Next Item Pointer	Describes the location of the next item in the function's capability list. The Next Item Pointer defaults to pointing to the capability structure for PCI-X located at DCh in the configuration space. This can be overwritten with 00h if support for PCI-X is not desired.

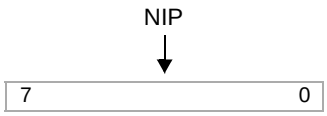


Figure 18-66. PCI-X Power Management Next Item Pointer (PCIX0_PMNIPTR)

7:0	NIP	Next Item Pointer	Describes the location of the next item in the function's capability list. The Next Item Pointer defaults to pointing to the capability structure for PCI-X located at DCh in the configuration space. This can be overwritten with 00h if support for PCI-X is not desired.

18.18.9.3 PCI-X Power Management Capabilities Register (PCIX0_PMC)

This register provides information on the capabilities of the function related to power management.

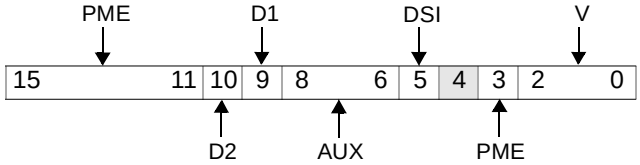


Figure 0-60. PCI-X Power Management Capabilities Register (PCIX0_PMC)

15:11	PME	PME_Support	The PLB-PCIXBridge does not support PME#; therefore, all the bits are hardwired to 0.
10	D2	D2_Support	This bit determines if the D2 Power Management State is supported. PLB-PCIX Bridge does not support the D2 Power Management State; therefore, this bit is hardwired to a 0.
9	D1	D1_Support	This bit determines if the D1 Power Management State is supported. PLB-PCIX Bridge supports the D1 Power Management State; therefore, this bit is hardwired to 1.
8:6	AUX	Aux_Current	PLB-PCIX Bridge does not support PME#; therefore, this field is unsupported. All the bits are hardwired to 0.
5	DSI	Device Specific Initialization (DSI)	The DSI bit indicates whether special initialization of this function is required (beyond the standard PCI configuration header) before the generic class device driver is able to use it. This bit is hardwired to 0.
4		Reserved	Returns zero when read.
3	PME	PME Clock	This bit is hardwired to 0 indicating that the function does not support PME# generation in any state.
2:0	V	Version	Returns a value of 010b on reads, indicating that this function complies with the *PCI Bus Power Management Interface Specification,* Version 1.1.

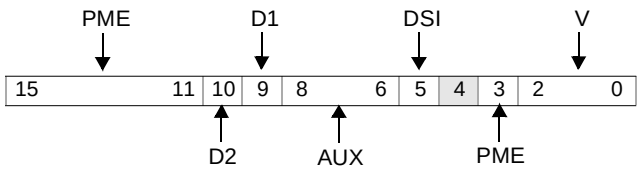


Figure 18-67. PCI-X Power Management Capabilities Register (PCIX0_PMC)

15:11	PME	PME_Support	The PLB-PCIX Bridge does not support PME#; therefore, all the bits are hardwired to 0.

10	D2	D2_Support	This bit determines if the D2 Power Management State is supported. PLB-PCIX Bridge does not support the D2 Power Management State; therefore, this bit is hardwired to a 0.
9	D1	D1_Support	This bit determines if the D1 Power Management State is supported. PLB-PCIX Bridge supports the D1 Power Management State; therefore, this bitis hardwired to 1.
8:6	AUX	Aux_Current	PLB-PCIX Bridge does not support PME#; therefore, this field is unsupported. All the bits are hardwired to 0.
5	DSI	Device Specific Initialization (DSI)	The DSI bit indicates whether special initialization of this function is required (beyond the standard PCI configuration header) before the generic class device driver is able to use it. This bit is hardwired to 0.
4		Reserved	Returns zero when read.
3	PME	PME Clock	This bit is hardwired to 0 indicating that the function does not support PME# generation in any state.
2:0	V	Version	Returns a value of 010b on reads, indicating that this function complies with the *PCI Bus Power Management Interface Specification,* Version 1.1.

18.18.9.4 PCI-X Power Management Control/Status Register (PCIX0_PMCSR)

Host writes may be handled as delayed writes.

This 16-bit register is used to manage the PCI function's power management state as well as to enable or monitor PMEs.

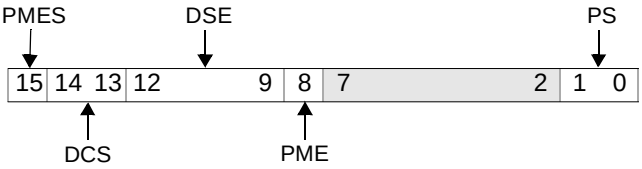


Figure 0-61. PCI-X Power Management Control/Status Register (PCIX0_PMCSR)

15	PMES	PME_Status	The PLB-PCIX Bridge does not support PME#; therefore, this bit is hardwired to a 0.
14:13	DSC	Data_Scale	The PLB-PCIX Bridge does not support the Data Register; therefore, these bits are hardwired to 00.
12:9	DSE	Data_Select	The PLB-PCIX Bridge does not support the Data Register; therefore, these bits are hardwired to 0000.
8	PME	PME_En	PLB-PCIX Bridge does not support PME# generation; therefore, this bit is hardwired to a 0.
7:2		Reserved	Retunrs zero when read.
1:0	PS	PowerState 00b - D0 01b - D1 10b - D2 11b - D3-Hot	This 2-bit R/W field is used both to determine the current power state of a function and to set the function into a new power state. If software attempts to write an unsupported, optional state to this field, the value of this field does not change. Writing this register may cause the PMSC_RR to change.

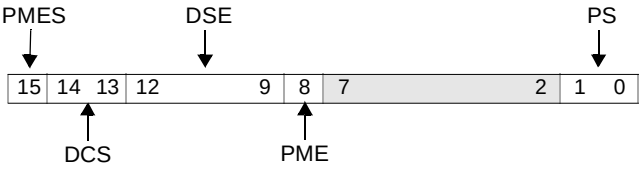


Figure 18-68. PCI-X Power Management Control/Status Register (PCIX0_PMCSR)

15	PMES	PME_Status	The PLB-PCIX Bridge does not support PME#; therefore, this bit is hardwired to a 0.
14:13	DSC	Data_Scale	The PLB-PCIX Bridge does not support the Data Register; therefore, these bits are hardwired to 00.

12:9	DSE	Data_Select	The PLB-PCIX Bridge does not support the Data Register; therefore, these bits are hardwired to 0000.
8	PME	PME_En	PLB-PCIX Bridge does not support PME# generation; therefore, this bit is hardwired to a 0.
7:2		Reserved	Retunrs zero when read.
1:0	PS	PowerState 00b - D0 01b - D1 10b - D2 11b - D3-Hot	This 2-bit R/W field is used both to determine the current power state of a function and to set the function into a new power state. If software attempts to write an unsupported, optional state to this field, the value of this field does not change. Writing this register may cause the PMSC_RR to change.

18.18.9.5 PCI-X PMCSR PCI-to-PLB-PCIX Bridge Support Extensions (PCIX0_PMCSRBSE)

This register is required for all PCI-to-PCI Bridges. The PLB-PCIX Bridge is not a PCI-to-PCI Bridge; therefore, it returns zero when this register is read.

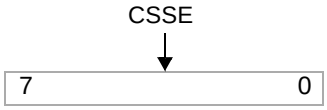


Figure 0-62. PCI-X PMCSR PCI-to-PLB-PCIX Bridge Support Extensions (PCIX0_PMCSRBSE)

7:0	CSSE	Power Management Control/Status PCI-to-PLB-PCIX Bridge Support Extensions.	Required for all PCI-to-PCI bridges. Always read as zero.

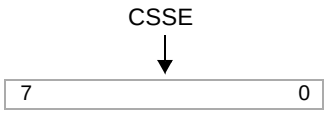


Figure 18-69. PCI-X PMCSR PCI-to-PLB-PCIX Bridge Support Extensions (PCIX0_PMCSRBSE)

7:0	CSSE	Power Management Control/Status PCI-to-PLB-PCIX Bridge Support Extensions.	Required for all PCI-to-PCI bridges. Always read as zero.

18.18.9.6 PCI-X Power Management Data (PCIX0_PMDATA)

The Data Register is an optional register that provides a mechanism for the function to report state dependent operating data such as power consumed or heat dissipation. PLB-PCIX Bridge does not implements this register; therefore, it returns zero when this register is read.

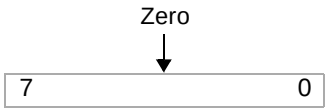


Figure 0-63. PCI-X Power Management Data (PCIX0_PMDATA)

7:0	Zero	Optional register	Returns zero when read.

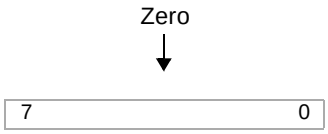


Figure 18-70. PCI-X Power Management Data (PCIX0_PMDATA)

7:0	Zero	Optional register	Returns zero when read.

18.18.9.7 PCI-X Power Management State Change Request Register (PCIX0_PMSCRR)

The PMSC_RR Register provides a method of informing the local processor of a power management state change request and prevents the completion of the write to the PMCSR Register until the local processor indicates it is ready for the state change. This register is used with the registers in the capability structure for the power management, but is not a part of the power management capability structure.

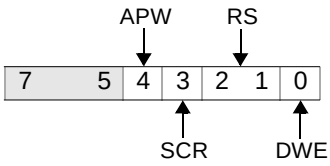


Figure 0-64. PCI-X Power Management State Change Request Register (PCIX0_PMSCRR)

7:5		Reserved	Returns zero when read.
4	APW	Accept PMCSR Write	The local processor sets this bit when it is ready to change the power management state. The bit is cleared when the host configuration write to the PMCSR is accepted. This bit is always a 1 if the Delayed Write Enable bit (bit 0) is a 0. (It is possible for the local processor to write a 0 to this bit).
3	SCR	State Change Request	The PLB-PCIX Bridge sets this bit when there is a host write to the PMCS for a power management state change request. This drives an interrupt to the local processor informing it of a state change request. The local processor must simultaneously clear this bit and set the Accept PMCSR Write bit when it is ready to change the state. After clearing this bit, new request will not be detected until the outstanding delayed write is accepted. (It is possible for the local processor to write a 1 to this bit.)
2:1	RS	Requested State	This field indicates the new power management state requested using the delayed host write to the PMCSR.
0	DWE	Delayed Write Enable 1 - Delayed write 0 - Immediate write	When 1, any configuration write to the PMCSR is completed as a delayed write. All writes to PMCSR are retried until the local processor sets the Accept PMCSR Write bit (bit 4). When 0, any configuration write to the PMCSR is completed immediately. This bit is a don't care if a host write to PMCSR requests a state change from D3hot to D0. The default state is 0 to prevent bus hang if the PM_Int service routine is not ready to go.

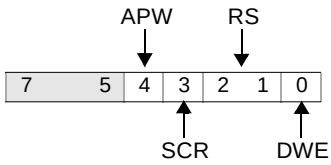


Figure 18-71. PCI-X Power Management State Change Request Register (PCIX0_PMSCRR)

7:5		Reserved	Returns zero when read.
4	APW	Accept PMCSR Write	The local processor sets this bit when it is ready to change the power management state. The bit is cleared when the host configuration write to the PMCSR is accepted. This bit is always a 1 if the Delayed Write Enable bit (bit 0) is a 0. (It is possible for the local processor to write a 0 to this bit).
3	SCR	State Change Request	The PLB-PCIX Bridge sets this bit when there is a host write to the PMCS for a power management state change request. This drives an interrupt to the local processor informing it of a state change request. The local processor must simultaneously clear this bit and set the Accept PMCSR Write bit when it is ready to change the state. After clearing this bit, new request will not be detected until the outstanding delayed write is accepted. (It is possible for the local processor to write a 1 to this bit.)
2:1	RS	Requested State	This field indicates the new power management state requested using the delayed host write to the PMCSR.
0	DWE	Delayed Write Enable 1 - Delayed write 0 - Immediate write	When 1, any configuration write to the PMCSR is completed as a delayed write. All writes to PMCSR are retried until the local processor sets the Accept PMCSR Write bit (bit 4). When 0, any configuration write to the PMCSR is completed immediately. This bit is a don't care if a host write to PMCSR requests a state change from D3hot to D0. The default state is 0 to prevent bus hang if the PM_Int service routine is not ready to go.

18.18.10 PCI-X Capability Block Definition Registers

This section describes the PCI-X Capability registers.

18.18.10.1 PCI-X Capability Identifier (PCIX0_PCIXCAPID)

The Capability Identifier when read by system software as 07h indicates that the data structure currently being pointed to is the PCI-X capability structure.

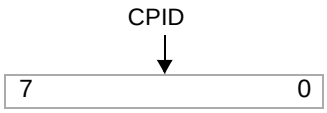


Figure 0-65. PCI-X Capability Identifier (PCIX0_PCIXCAPID)

7:0	CPID	Capability Identifier

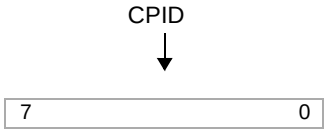


Figure 18-72. PCI-X Capability Identifier (PCIX0_PCIXCAPID)

7:0	CPID	Capability Identifier

18.18.10.2 PCI-X Next Item Pointer Register (PCIX0_PCIXNIPTR)

This Next Item Pointer Register describes the location of the next item in the capability list of the function. This is the last item in the capability list; therefore, this field is hardwired to a value of 00h.

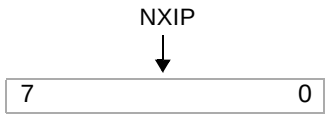


Figure 0-66. PCI-X Next Item Pointer Register (PCIX0_PCIXNIPTR)

7:0	NXIP	Next Item Pointer

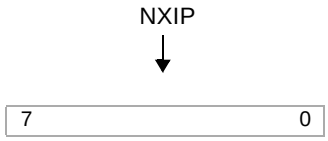


Figure 18-73. PCI-X Next Item Pointer Register (PCIX0_PCIXNIPTR)

7:0	NXIP	Next Item Pointer

18.18.10.3 PCI-X Command Register (PCIX0_PCIXCMD)

The reset value of bits 6:4 equal the reset value of the DMOS field of the PCIX0_PCIXSTS register, which in turn, depends on the number of special purpose outbound read buffers present.

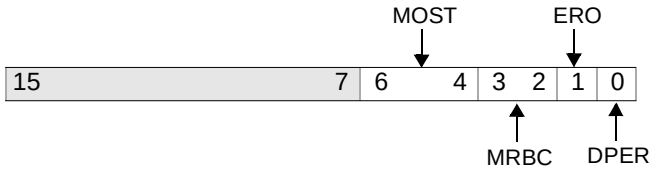


Figure 0-67. PCI-X Command Register (PCIX0_PCIXCMD)

15:7		Reserved	Returns zero when read.
6:4	MOST	Maximum Outstanding Split Transactions	This field indicates the number of outstanding split transactions allowed. If it is written to value less than the DMOS field of the PCIX0_PCIXSTS register, the PLB-PCIX Bridge limits the number of outstanding split transactions accordingly.
3:2	MRBC	Maximum Memory-Read Byte Count	This field defaults to 00 to indicate that the PLB-PCIX Bridge can request a maximum byte count of 512 bytes in a single memory read transaction. If the MRBC field of PCIX0_PCIXSTS register is not 0, this field can be programmed by the system to 01 or 10 to allow the PLB-PCIX Bridge to request 1K or 2K bytes, respectively. If it is programmed to 11, the PLB-PCIX Bridge will behave as if it is set to 10.
1	ERO	Enable Relaxed Ordering	This bit is hardwired to a 0, indicating that the PLB-PCIX Bridge never sets the Relaxed Ordering attribute bit.
0	DPER	Data Parity Error Recovery Enable	This bit is writable but has no effect.

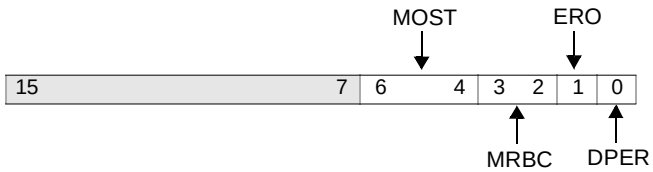


Figure 18-74. PCI-X Command Register (PCIX0_PCIXCMD)

15:7		Reserved	Returns zero when read.

6:4	MOST	Maximum Outstanding Split Transactions	This field indicates the number of outstanding split transactions allowed. If it is written to value less than the DMOS field of the PCIX0_PCIXSTS register, the PLB-PCIX Bridge limits the number of outstanding split transactions accordingly.
3:2	MRBC	Maximum Memory-Read Byte Count	This field defaults to 00 to indicate that the PLB-PCIX Bridge can request a maximum byte count of 512 bytes in a single memory read transaction. The system will not set this field higher than 00 because 512 bytes is the designed maximum memory read byte count.
1	ERO	Enable Relaxed Ordering	This bit is hardwired to a 0, indicating that the PLB-PCIX Bridge never sets the Relaxed Ordering attribute bit.
0	DPER	Data Parity Error Recovery Enable	This bit is writable but has no effect.

18.18.10.4 PCI-X Status Register (PCIX0_PCIXSTS)

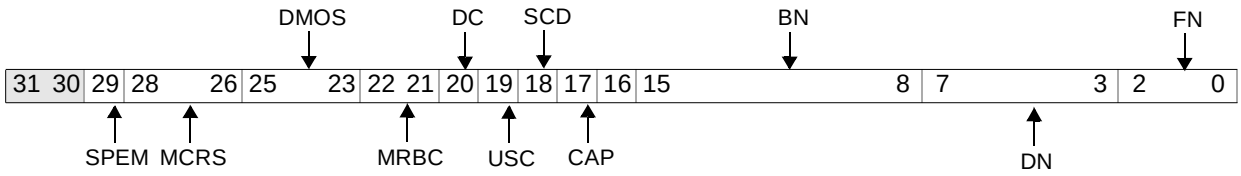


Figure 0-68. PCI-X Status Register (PCIX0_PCIXSTS)

31:30		Reserved	Returns zero when read.
29	SPEM	Received Split Completion Error Message	This bit is set if a split completion error message is received. Writing a value of 1 to this bit clears this bit.
28:26	MCRS	Designed Max Cumulative Read Size	This field is hardwired to 001b to indicate that the maximum cumulative outbound read size is less than or equal to 2 KB (it is 1.6 KB).
25:23	DMOS	Designed Max Outstanding Splits	This field indicates the maximum number of split transactions that the PLB-PCIX Bridge can have outstanding. Its reset value depends on the number of special purpose outbound read buffers present. The reset value is 010b, indicating that the PLB-PCIX Bridge can have three outstanding split completions.
22:21	MRBC	Designed Max Memory Read Byte Count	This field is hardwired to 00b to indicate that the maximum read byte count that the PLB-PCIX Bridge generates on outbound reads is 512 bytes.
20	DC	Device Complexity	This bit is hardwired to a 0 indicating that the PLB-PCIX Bridge is a simple device. (It does not handle LOCK# and never retries or disconnects split completions).
19	USC	Unexpected Split Completion	This bit is set if an unexpected split completion with Initiator Bus Number and Initiator Number of the PLB-PCIX Bridge is received. This bit is hardwired to 0, because this error is not checked.
18	SCD	Split Completion Discarded	This bit is set if the PLB-PCIX Bridge discards a split completion because the requestor would not accept it. This bit is hardwired to 0, because PLB-PCIX Bridge never discards a split completion.

17	CAP	133 MHz Capable	This bit indicates 133 MHz capability. This bit is hardwired to 1.
16	DEV	64-bit Device	This bit indicates the size of the data bus of the chip containing the PLB-PCIX Bridge. 1 means the bus is 64 bits. Its value is equal to the PCI initialize Req64 enable (PR64E) strapping bit.
15:8	BN	Bus Number	This field indicates the number of the bus segment for the PLB-PCIX Bridge containing this function. These bits are read/writable from the PLB side, but read only from the PCI side.
7:3	DN	Device Number	This field indicates the number of the device for the PLB-PCIX Bridge containing this function. These bits are read/writable from the PLB side, but read only from the PCI side.
2:0	FN	Function Number	This field indicates the number of the function for the PLB-PCIX Bridge containing this function. These bits are hardwired to 000b.

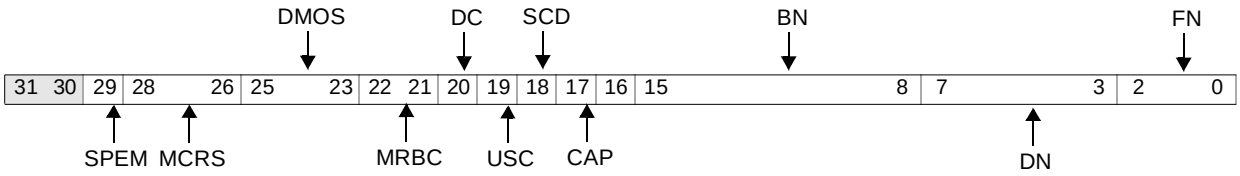


Figure 18-75. PCI-X Status Register (PCIX0_PCIXSTS)

31:30		Reserved	Returns zero when read.
29	SPEM	Received Split Completion Error Message	This bit is set if a split completion error message is received. Writing a value of 1 to this bit clears this bit.
28:26	MCRS	Designed Max Cumulative Read Size	This field is hardwired to 001b to indicate that the maximum cumulative outbound read size is less than or equal to 2 KB (it is 1.6 KB).
25:23	DMOS	Designed Max Outstanding Splits	This field indicates the maximum number of split transactions that the PLB-PCIX Bridge can have outstanding. Its reset value depends on the number of special purpose outbound read buffers present. The reset value is 010b, indicating that the PLB-PCIX Bridge can have three outstanding split completions.
22:21	MRBC	Designed Max Memory Read Byte Count	This field is hardwired to 00b to indicate that the maximum read byte count that the PLB-PCIX Bridge generates on outbound reads is 512 bytes.

20	DC	Device Complexity	This bit is hardwired to a 0 indicating that the PLB-PCIX Bridge is a simple device. (It does not handle LOCK# and never retries or disconnects split completions).
19	USC	Unexpected Split Completion	This bit is set if an unexpected split completion with Initiator Bus Number and Initiator Number of the PLB-PCIX Bridge is received. This bit is hardwired to 0, because this error is not checked.
18	SCD	Split Completion Discarded	This bit is set if the PLB-PCIX Bridge discards a split completion because the requestor would not accept it. This bit is hardwired to 0, because PLB-PCIX Bridge never discards a split completion.
17	CAP	133 MHz Capable	This bit indicates 133 MHz capability. This bit is hardwired to 1.
16	DEV	64-bit Device	This bit indicates the size of the data bus of the chip containing the PLB-PCIX Bridge. 1 means the bus is 64 bits. Its value is equal to the PCI initialize Req64 enable (PR64E) strapping bit.
15:8	BN	Bus Number	This field indicates the number of the bus segment for the PLB-PCIX Bridge containing this function. These bits are read/writable from the PLB side, but read only from the PCI side.
7:3	DN	Device Number	This field indicates the number of the device for the PLB-PCIX Bridge containing this function. These bits are read/writable from the PLB side, but read only from the PCI side.
2:0	FN	Function Number	This field indicates the number of the function for the PLB-PCIX Bridge containing this function. These bits are hardwired to 000b.

18.18.10.5 PCI-X Internal Debug Register (PCIX0_PCIXIDR)

Bits 31:0 are reserved.

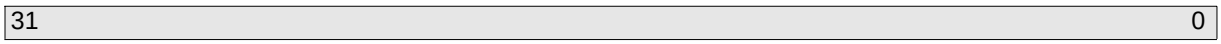


Figure 0-69. PCI-X Internal Debug Register (PCIX0_PCIXIDR)

31:0		Reserved

Figure 18-76. PCI-X Internal Debug Register (PCIX0_PCIXIDR)

31:0		Reserved

18.18.10.6 PCI-X Internal Core Device ID Register (PCIX0_PCIXCID)

This register is not used under normal operation. It is for debug purposes only. Its contents may be requested by IBM support.

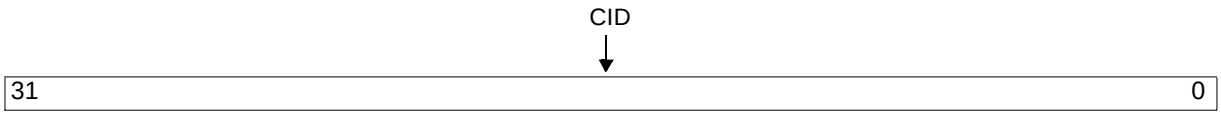


Figure 0-70. PCI-X Internal Core Device ID Register (PCIX0_PCIXCID)

31:0	CID	Internal Core Device ID	Read only.

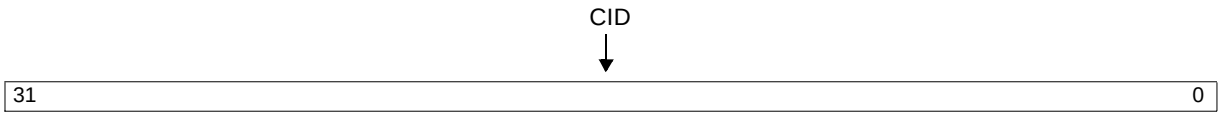


Figure 18-77. PCI-X Internal Core Device ID Register (PCIX0_PCIXCID)

31:0	CID	Internal Core Device ID	Read only.

18.18.10.7 PCI-X Internal Core Revision ID Register (PCIX0_PCIXRID)

This register is not used under normal operation. It is for debug purposes only. Its contents may be requested by IBM support.

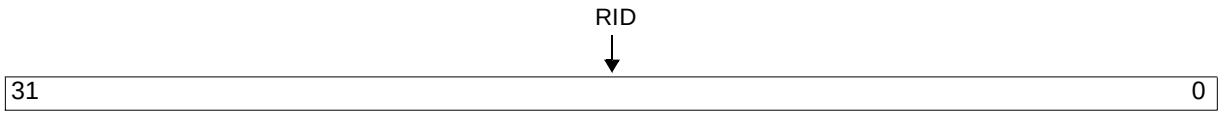


Figure 0-71. PCI-X Internal Core Revision ID Register (PCIX0_PCIXRID)

31:0	RID	Internal Core Revision ID	Read only.

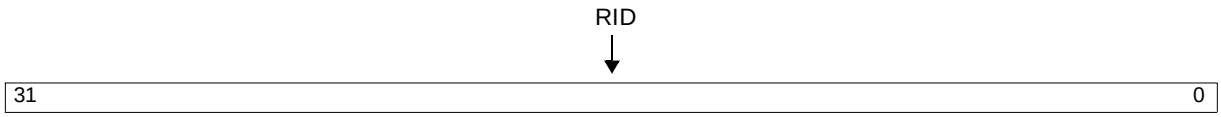


Figure 18-78. PCI-X Internal Core Revision ID Register (PCIX0_PCIXRID)

31:0	RID	Internal Core Revision ID	Read only.

18.18.11 Simple Message Passing and Inbound MSI Registers

The following registers are for the simple message passing mechanism and for inbound MSI.

Note 1: The registers are not accessible from the PCI side in configuration space, but rather in PCI memory space, using BAR0. They must be read or written with 32-bit transfers.

Note 2: The registers are only accessible from the PCI side when the "Split BAR0 Enable" bit of the PIM0 Size/Attribute register is set.

18.18.11.1 PCI-X Message In Low (PCIX0_MSGIL)

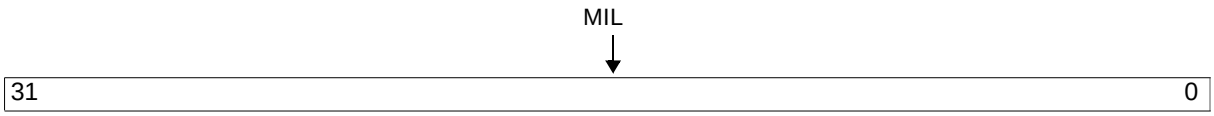


Figure 0-72. PCI-X Message In Low (PCIX0_MSGIL)

31:0	MIL	Message In Low	This holds an inbound message. If bit 0 is high, then an inbound interrupt is generated.

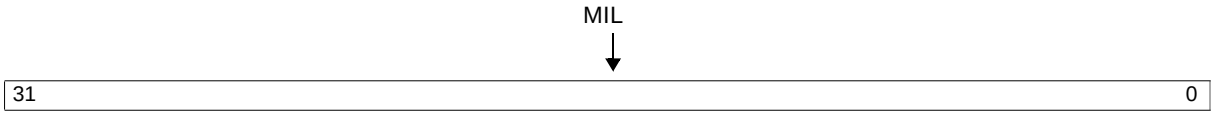


Figure 18-79. PCI-X Message In Low (PCIX0_MSGIL)

31:0	MIL	Message In Low	This holds an inbound message. If bit 0 is high, then an inbound interrupt is generated.

18.18.11.2 PCI-X Message In High (PCIX0_MSGIH)

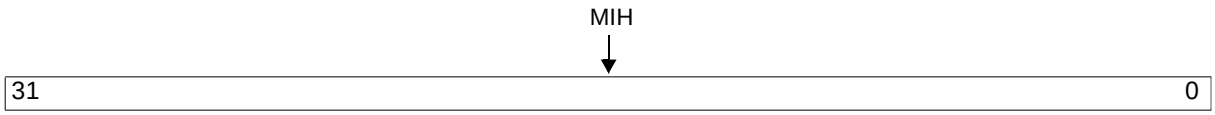


Figure 0-73. PCI-X Message In High (PCIX0_MSGIH)

31:0	MIH	Message In High	This holds the upper 32 bits of a 64-bit inbound message.

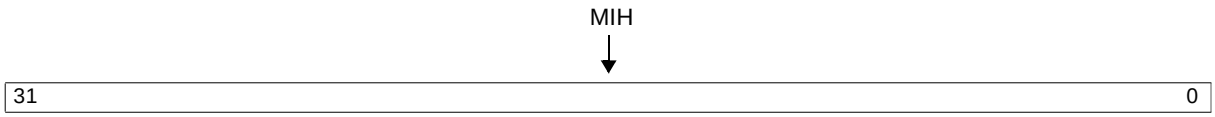


Figure 18-80. PCI-X Message In High (PCIX0_MSGIH)

31:0	MIH	Message In High	This holds the upper 32 bits of a 64-bit inbound message.

18.18.11.3 PCI-X Message Out Low (PCIX0_MSGOL)

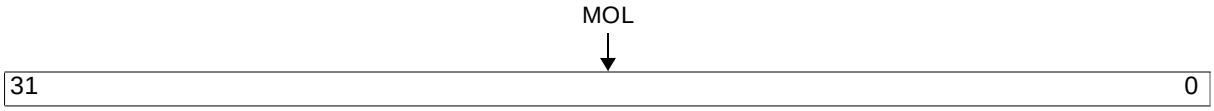


Figure 0-74. PCI-X Message Out Low (PCIX0_MSGOL)

31:0	MOL	Message Out Low	This holds an outbound message. If bit 0 is high, then an outbound interrupt is generated.

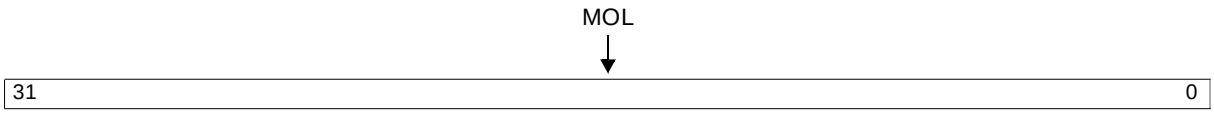


Figure 18-81. PCI-X Message Out Low (PCIX0_MSGOL)

31:0	MOL	Message Out Low	This holds an outbound message. If bit 0 is high, then an outbound interrupt is generated.

18.18.11.4 PCI-X Message Out High (PCIX0_MSGOH)

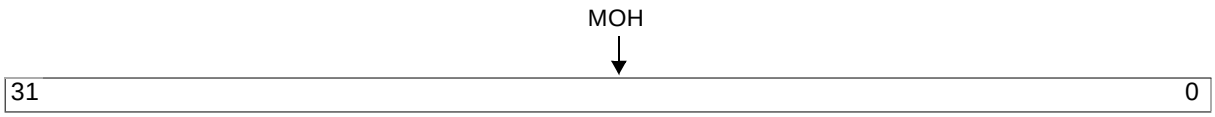


Figure 0-75. PCI-X Message Out High (PCIX0_MSGOH)

31:0	MOH	Message Out High	This holds the upper 32 bits of a 64-bit outbound message.

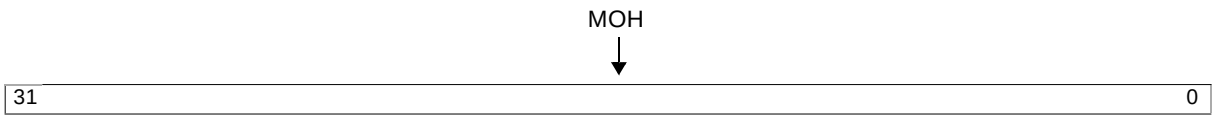


Figure 18-82. PCI-X Message Out High (PCIX0_MSGOH)

31:0	MOH	Message Out High	This holds the upper 32 bits of a 64-bit outbound message.

18.18.11.5 PCI-X Inbound Message MSI (PCIX0_IM)

Inbound MSIs are enabled by setting the Inbound MSI Enable bit of the Bridge Options 1 register. Address mapping must be also be setup appropriately.

Figure 0-76. PCI-X Inbound Message MSI (PCIX0_IM)

31:4		Reserved	Returns zero when read.
3:0	IMSI	MSI In	Inbound MSIs write this register. The data value written causes the corresponding pci_msi_int0:12 to go active to the interrupt controller.

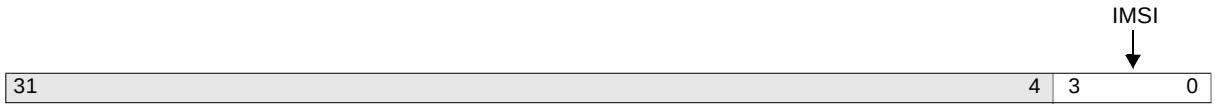


Figure 18-83. PCI-X Inbound Message MSI (PCIX0_IM)

31:4		Reserved	Returns zero when read.
3:0	IMSI	MSI In	Inbound MSIs write this register. The data value written causes the corresponding pci_msi_int0:12 to go active to the interrupt controller.

18.19 Boot Modes

There are three supported methods of booting the local (PLB) CPU. The method chosen depends upon where the boot code is located and how the boot code is initialized. The methods are:

- Boot from local ROM, when in host-bridge mode
- Boot from local ROM, when in adapter-bridge mode
- Boot from PCI, when in adapter-bridge mode

The boot sequence of most PCI systems includes a few steps that are relevant to this Boot Modes discussion. This first such step is the host CPU initializing its PCI host bridge. In the next step, the host CPU runs the PCI configuration sequence, during which all the adapter's configuration registers are read and initialized. This step requires many of the configuration registers to contain values that indicate things about the adapter. The final step is when the host CPU boots its operating system.

The register set of the PLB-PCIX Bridge includes the following registers:

- Vendor ID
- Device ID
- Revision ID
- Class
- Subsystem
- Vendor ID
- Subsystem Vendor ID
- Subsystem Device ID

The above register are read by the PCI configuration sequence to gather information about the adapter when in adapter-bridge mode. The register set of the PLB-PCIX Bridge also includes the BAR0, BAR1, BAR2 and Expansion ROM BAR registers that behave in a manner that indicates to the PCI configuration sequence memory space needs of the adapter. The behavior of these registers is governed by the values of the PIM0, PIM1, and PIM2 Size/Attribute registers. These registers are referred to as "adapter needs" registers.

The values of these "adapter's needs" registers can be ~~set at reset using the strapping pins, or can be~~ written from either the local CPU (PLB) or the host (PCI bus).

18.19.1 Booting from Local ROM, when in Host-Bridge Mode

When the PLB-PCIX Bridge is used as the host-bridge, the local CPU must boot from local ROM (the local CPU's access to the local ROM does not involve the PLB-PCIX Bridge). As part of the boot process, the local CPU initializes the registers of the PLB-PCIX Bridge before PCI activity begins. Generally, the configuration registers of the PLB-PCIX Bridge are only accessed by the local CPU, never from the PCI side.

In this mode, the "adapter's needs" registers are not read (since the PLB-PCIX Bridge is not an adapter). Instead, the boot code contains all necessary information needed for initializing the PLB-PCIX Bridge. Thus, it is not necessary to have any specific values strapped into the registers.

Once the registers of the PLB-PCIX Bridge are initialized, PCI activity may begin. At some point after this, the local CPU runs the PCI configuration sequence during which it determines what devices are attached to the PCI bus and initializes their configuration registers.

18.19.2 Booting from Local ROM, when in Adapter-Bridge Mode

When the PLB-PCIX Bridge is used in adapter-bridge mode, the most common method of booting is booting from local ROM or flash (the local CPU's access to the local ROM does not involve the PLB-PCIX Bridge). Typically, this boot process goes on in parallel with the host booting.

As part of the host's boot process, it typically runs the PCI configuration sequence. However, before the PCI configuration sequence can run, the configuration registers of the PLB-PCIX Bridge must be set to indicate the adapter's needs. This can be done ~~either~~ by ~~setting the appropriate strapping pins, or by~~ having the local CPU write to the registers.

If the values are set using local CPU writes, then it must be ensured that the host does not proceed to the PCI configuration sequence before the local CPU completes its initializations. This is guaranteed by strapping the PHCE strapping ~~bitl~~ bit low, causing the "Host Configuration Enable" bit of the Bridge Options 2 Register to be cleared at reset. When cleared, host accesses to the configuration registers are retried. This ensures that the local CPU has time to initialize the "adapter's needs" registers. When initialization is complete, the local CPU writes the "Host Configuration Enable" bit high, allowing the host CPU to complete the PCI configuration sequence.

18.19.3 Booting from PCI, when in Adapter-Bridge Mode

When the PLB-PCIX Bridge is used in adapter-bridge mode, it can be enabled to allow the local CPU to boot from memory on the PCI bus. The typical boot sequence for this mode is as follows:

1. The local CPU is initially prevented from booting while the host boots. This is done by strapping the PCWE strapping ~~bitl~~ bit high, which causes the "Local CPU Wait" bit of the Bridge Options 2 Register to be high at reset.
2. The host CPU boots and runs the PCI initialization sequence. Thus, the "adapter's needs" registers of the PLB-PCIX Bridge must be strapped appropriately.
3. Eventually (probably after the operating system boots), the host runs a program that puts boot code for the adapter in PCI memory space and initializes the POM0 registers of the PLB-PCIX Bridge to point to the boot code.
4. The host then enables the local CPU to boot. This is done by writing a "0" to the "Local CPU Wait" bit of the Bridge Options 2 Register.

Note 1: If the PLB contains another interface controller that can be a boot device (such as a flash controller), then it must be disabled.

Note 2: Booting from the PCI immediately after reset (ROM on PCI bus) is not supported.

Note 3: To ensure that there are no deadlocks, the PLB arbiter must be set to FAIR, and the PLB priority of all PLB masters that access the PLB-PCIX Bridge must be set to the same PLB priority as the PLB-PCIX Bridge.

18.19.4 Option ROM Support

The PLB-PCIX Bridge supports PCI option ROM by allowing a region of PCI memory space to be mapped to local (PLB) memory (SDRAM or ROM). The region of PCI memory space is determined by the "Expansion ROM BAR" at offset 30h in PCI configuration space. The PLB-PCIX Bridge allows the use of either PIM2 or Option ROM, not both. The "Expansion ROM Enable" bit in the PCI Expansion ROM Base Address register determines the choice. See section ~~"PCI-X Expansion ROM Base Address Register (PCIX0_EROMBA)" on page 56~~*PCI-X Expansion ROM Base Address Register (PCIX0_EROMBA)* on page 588.

18.20 Software Initialization

The PLB-PCIX Bridge has a set of registers that must be initialized before general operation begins.

18.20.1 Address Map Initialization

When the PLB-PCIX Bridge is the master on the PCI bus, it can generate memory, I/O, configuration and special cycles. The PLB address ranges used to generate these cycle are all fixed, except for memory cycles. PCI memory cycles range sizes and PLB address space is specified in the POM registers. Also, the address of the resulting PCI memory cycle may be an offset from the PLB address (address translation occurs). This translation is also specified in the POM registers. The POM registers do not default to usable values following reset; they must be initialized before attempting to generate PCI memory cycles.

When the PLB-PCIX Bridge is a target on the PCI bus, it can respond to memory cycles. The memory cycle address ranges that the PLB-PCIX Bridge responds to are specified in the BAR0, BAR1 and BAR2 registers. The BAR registers are typically initialized as part of the standard PCI initialization process; however, before the BAR registers are initialized, the PIM Size/Attribute registers must be set to control the BAR behavior. The PIM Size/Attribute registers may be strapped or written to specify a usable value.

Before inbound traffic begins, the PIM Local Address registers must be specified to control the resulting PLB address (translated address). The PIM Local Address registers do not default to usable values following reset; they must be initialized before responding as a PCI memory target is allowed.

18.20.2 Other Registers that must be Initialized

- Error handling is initially disabled (error detection masked). If error handling is to be enabled, then the Error Enable Register must be initialized appropriately.
- The Bridge Options 1 Register and the Bridge Options 2 Register contain a variety of options that may need to be changed.

18.20.3 Adapter-Bridge Mode Configuration Access

When the PLB-PCIX Bridge is used as a PCI adapter interface (in adapter-bridge mode), it can respond as a configuration target. The PCI_IDSEL pin must be hooked up (typically only done in adapter-bridge mode) and active for the PLB-PCIX Bridge to respond as a configuration target. Before the host runs the PCI Configuration sequence, the following registers must have the appropriate values.

- PCI Vendor ID
- PCI Device ID
- PCI Revision ID
- PCI Class
- PCI Subsystem ID
- PCI Subsystem Vendor ID

They may be ~~initialized using the strapping pins or they may be~~ written by the local CPU with values taken from a local ROM. If these values are written by the local ~~CPU rather than taken from strapping register~~ sCPU, the Host Configuration Enable bit in the Bridge Options 2 register must be 0 to prevent the host CPU from reading these values before the local CPU has written them.

18.21 Initialization/Reset Sequence

When the PCI initial sequence enable (PISE) strapping bit is set in the serial EPROM, the PLB-PCIX Bridge performs the PCIX initialization sequence using the method described in section 14.2 of the PCIX Addendum 1.0 to sample PCIXCAP. During the sampling of PCIXCAP, the PLB-PCIX Bridge drives low and high values onto the PCI_X133CAP 440GP output, which must be connected to the system PCIXCAP net through the resistor network shown in figure 14-3 of the PCIX Addendum 1.0. Based on the sampled values of the PCI_XCAP 440GP input connected to the system PCIXCAP net during this initialization sequence, the PLB-PCIX Bridge determines what mode (conventional or X) the devices on the bus are capable of and, if they are X-mode capable, what frequency (66 MHz or 133 MHz) they are capable of.

In conventional PCI mode, the value of the PCIFREQ field of the PCI Bridge Options 2 register depends entirely on the sampled value of PCI_M66EN. If PCI_M66EN is low, then the 0 to 33 MHz (PLL Bypassed) frequency mode is entered. If PCI_M66EN is high, then the 33 to 66 MHz mode is entered. In X mode, the value of the PCIFREQ field is a function of the PCIX Frequency Selection (PCXF) bootstrap controller straps and the actual capabilities of the cards determined by PCIXCAP sampling during the initialization sequence. If the cards are all determined to be 133 MHz X-capable, PCIFREQ in the PCI Bridge Options 2 register will equal the PCXF straps. If the bus capability is found to be 66MHz (due to at least one card being only 66MHz capable), PCIFREQ will equal 10b selecting the 50 to 66 MHz PCIX frequency range.

During reset, when the frequency of the PCI or PCIX bus has been determined, the PLB-PCIX Bridge will drive the PCIX initialization pattern onto the PCI bus. The initialization pattern driven by the PLB-PCIX Bridge is defined in table 6-2 of the PCIX Addendum 1.0. Note that the assertion or deassertion of PCI_REQ64_N based on the value of the PCI Initialize Req64 Enable (PR64E) bootstrap controller strap is also part of the initialization sequence driven here. If PR64E is high and the PISE (PCI Initial Sequence Enable) bootstrap controller strap is high, then PCI_REQ64_N is driven active during reset to indicate the PCI bus is 64 bits wide.

The motherboard has the job of generating the PCI_CLK for the 440GP when the PLB-PCIX Bridge is in host bridge mode and for all the other PCI devices (including PCI connectors). If all the PCI devices in the system are known, then the correct frequency is known, and that one frequency is generated for PCI_CLK. However, if the attached devices are unknown, PCI(X) requires support for a variety of frequencies, based on what is plugged in, and indicated via the M66EN and PCIXCAP bus signals. The PLB-PCIX Bridge expects the motherboard to sample M66EN and PCIXCAP, and set up and stabilize the motherboard PCI_CLK generation before deasserting SYS_RESET_N to the 440GP. The PCIXCAP pin is a three-state signal (pulled up, tie low, or pulled low). The PCIX Addendum offers two methods of determining the value of PCIXCAP (see PCIX Addendum 1.0 chapter 14), either of which can be used by the motherboard. However, once SYS_RESET_N is deasserted to the 440GP, the 440GP will then interrogate the PCIXCAP pin itself; thus, whatever method the motherboard used to sample PCIXCAP, it must leave the net in a state where the only motherboard effect is the resistor network shown in figure 14-3 of the PCIX Addendum 1.0. This means that if the motherboard uses a method of detecting PCIXCAP that includes pullup (or pulldown) resistors of other values (such as the method described in figure 14-1 of the PCIX Addendum 1.0), then it must electrically disconnect those resistors from the net after sampling PCIXCAP and before deasserting SYS_RESET_N. Or if the motherboard uses the method of detecting PCIXCAP described in section 14-2 of the PCIX Addendum 1.0, then the motherboard must ~~tristate~~ High-Z its Strong Pullup driver prior to deasserting SYS_RESET_N. The motherboard may share the resistor network in figure 14-3 used by the 440GP.

18.22 PCI - PLB Relative Clock Frequency Requirements

The relative frequency requirements between PLB clock and PCI clock are:

0.0MHz <= PCI clock frequency <= 1.5 x PLB clock frequency

In words, PCI clock can be as slow as 0 MHz, or as fast as 50% faster than the PLB clock frequency. For example, if PLB clock is 66 MHz, then PLB clock can be 0 to 100 MHz.

19. Direct Memory Access Controller

The Direct Memory Access (DMA) controller is a Processor Local Bus (PLB) and On-chip Peripheral Bus (OPB) master which supports the autonomous transfer of data between memory and peripherals and from memory-to-memory. The controller provides four DMA channels, each of which has an independent set of configuration registers. Each channel has its own control, count and control, source address, destination address, and scatter/gather address registers. Once these registers are programmed by ~~the PPC440~~ thePPC440GP processor, the DMA controller performs the requested data transfer without the need for processor intervention.

The four DMA channels also support scatter/gather transfers. During a scatter/gather transfer, the configuration registers for a particular DMA channel are automatically loaded from a data structure in memory instead of being individually programmed. Since the scatter/gather address register is updated in this process, the channel can optionally reconfigure itself for another transfer when the current one completes.

As master on both the PLB and OPB, the DMA controller can read and write any address accessible by ~~the PPC440~~ thePPC440GP processor. This includes memory and memory-mapped peripherals on the EBC interface and SDRAM memory and PCI addresses that have been mapped into PLB address space. The DMA controller can also service DMA peripherals attached to the EBC via the DMAReqn, DMAAckn and EOTn[TCn] I/Os, along with the OPB-attached UART0 and UART1.

19.1 External Interface Signals

~~Table 19-1~~*Table 19-1* illustrates the external I/Os associated with the DMA controller, and Figure 19-2 illustrates the EBC I/Os used during external peripheral transfers. External peripheral and EBC device-paced memory transfers request service from the DMA controller by driving a DMA request line (DMAReq0-3) active. For peripheral mode transfers the DMA controller acknowledges the request and transfers data by asserting a DMA acknowledge signal (DMAAck0-3). In contrast, an EBC device-paced memory-to-memory transfer occurs when the chip select ($\overline{\text{PerCSn}}$) associated with the memory location is driven active. The timing of $\overline{\text{PerCSn}}$ and the other EBC I/Os is determined by the Bank Access Parameter Register (EBC0_BnAP) for the particular memory location. See ~~Chapter 27, "External Bus Controller"~~*Section 27 External Bus Controller* on page 855 for details on EBC configuration and timings.

Table 19-1. DMA Controller External I/Os

Signal	Usage
DMAReqn	DMA Request, used to request either a peripheral mode transfer or an EBC device-paced memory-to-memory transfer.
DMAAckn	DMA Acknowledge, instructs an EBC-attached DMA peripheral to transfer data.
EOTn[TCn]	End of Transfer or Terminal Count. Used to stop the channel when programmed as EOT. When configured as TC, goes active when the channel transfer count register (DMA0_CTCn) reaches zero.

Note: The active level (polarity) of DMAReqn, DMAAckn, and EOTn[TCn] are individually programmable for external signals only. All other signals are active high.

.

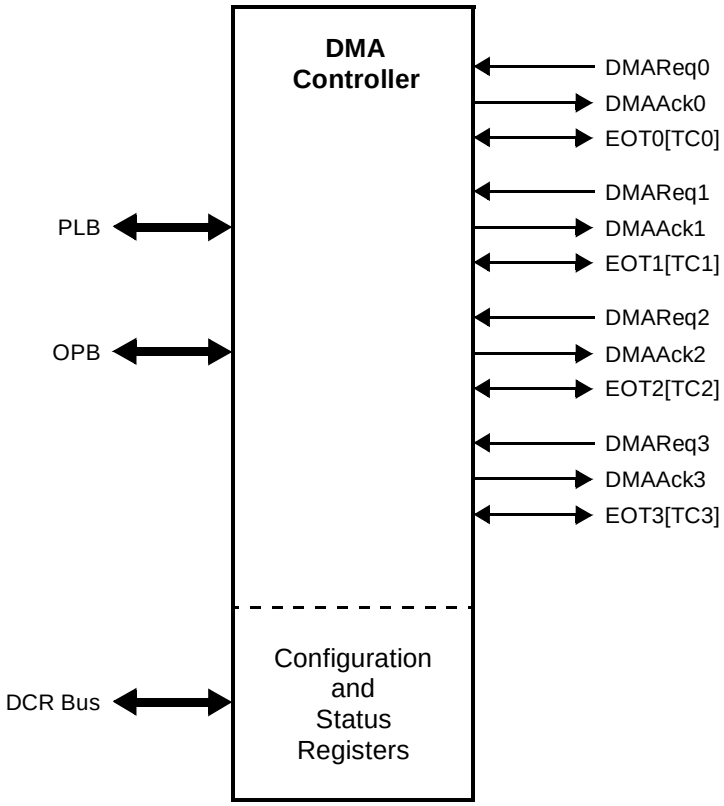


Figure 19-1. DMA Controller External Bus Control Signals

.

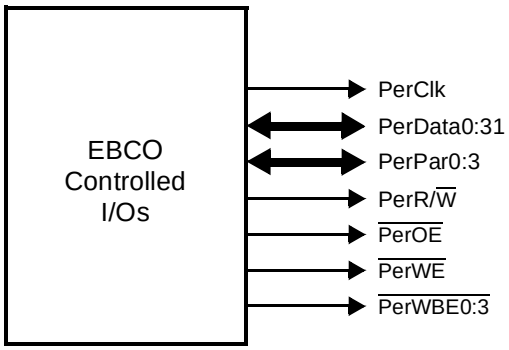


Figure 19-2. External Bus Control Signals

19.2 DMA Transfers

As a specialized controller, the DMA provides system designers and programmers with a highly efficient method of moving data. During any DMA transfer the controller always buffers data read from the source prior to writing the data to the destination. Since many buses, including the internal PLB, provide substantially better performance when bursting data, the DMA controller includes a 128-byte (8 quadword) buffer. This buffer is enabled on a per-channel basis by setting DMA0_CRn[BEN] and serves to minimize the number of discrete memory transactions. Each of the four DMA channels is configurable for either peripheral or memory-to-memory transfers.

19.2.1 Peripheral Mode Transfers

Peripherals are either devices attached to the EBC interface via the DMAReqn and DMAAckn lines, or the internal serial ports (UART0 and UART1). Memory is any address accessible from the PLB or OPB, including PLB-mapped PCI address space, SDRAM memory, and external memory. During a peripheral mode transfer the peripheral requests a DMA transfer by asserting a DMA request line. For UART0 and UART1, this signal is internal to the ~~PPC440GP~~ PPC440GP chip, while external peripherals use one of the DMAReqn lines. When the requesting channel has the highest priority of any active channel, the DMA executes a peripheral transfer, and the requesting device receives a DMA acknowledge signal. In the case of external peripherals, the appropriate DMAAckn line is driven to the active state, with the timing specified in the DMA Control Register for the channel (DMA0_CRn).

There are two types of peripheral mode transfers: peripheral-to-memory and memory-to-peripheral. A peripheral-to-memory transfer reads data from a DMA device, while a memory-to-peripheral transfer writes data. In both cases, the peripheral interface transfers data at the programmed width of the peripheral (DMA0_CRn[PW]). When the DMA buffer is disabled for the active channel (DMA0_CRn[BEN]=0), each peripheral transfer causes a corresponding memory operation.

When buffering is enabled during a peripheral-to-memory transfer, data is collected until the 128-byte buffer is full, the peripheral deasserts DMAReqn, or the channel completes. The buffer contents are then written to the target memory as efficiently as possible.

If the destination address is located on PLB space, the data is written to memory with one burst operation if the destination address is quadword aligned and the buffer is full; or, it is written with a combination of single beat and burst transfers, depending on the destination address alignment and the amount of data stored in the DMA buffer.

If the destination address is located on OPB memory space, the DMA writes out the content of the DMA buffer with a series of single beat transfers if burst mode is disabled; or, it is written with a series of burst transfers if bursting is enabled.

Memory-to-peripheral transfers differ since the amount of data that will be requested by the peripheral is unknown. If the DMA buffer is disabled (DMA0_CRn[BEN]=0) a discrete source memory read occurs for each element in the DMA transfer. Since this is inefficient, the buffer should only be disabled for low data rate transfers. When the 128-byte buffer is enabled the controller uses the setting in DMA0_CRn[PF] to prefetch 1, 2, 4, or 8 128-bit quadwords from the source memory. The DMA controller provides data from the buffer until the peripheral deasserts its request or the transfer completes. Whenever any of these conditions occurs, any unused data in the DMA buffer is discarded.

For both peripheral-to-memory and memory-to-peripheral transfers the DMA controller supports fixed length bursts of 2, 4, 8, or 16 data elements on the peripheral side of the transfer. When bursting is enabled (DMA0_CTCn[BTEN]=1) an active DMA request (DMAReqn) causes the DMA controller to transfer the programmed number of data items (DMA0_CTCn[BSIZ]) in one atomic operation.

19.2.2 Memory-to-Memory Transfers

The DMA controller can perform either device-paced (hardware-initiated) or software-initiated memory-to-memory transfers. Device-paced memory transfers function identically to peripheral mode transfers, except that a chip select ($\overline{\text{PerCSn}}$) serves as the DMA acknowledge instead of a DMAAckn output.

19.2.2.1 Device-Paced Memory Mode Transfers (Hardware Initiated)

Device-paced memory (DPM) is either devices attached to the EBC interface via the DMAReqn, External signals (including EXT_CS and EXT_ready), or the internal DPM connected internally to the chip. Memory is any address accessible from the PLB or OPB, including PLB-mapped PCI address space, SDRAM memory, and memory connected to the external bus.

During a DPM mode transfer, the DPM device requests a DMA transfer by asserting a DMA request line. For internal DPMs, this signal is internal to the chip, while external peripherals use one of the DMAReqn lines. When the requesting channel has the highest priority of any active channel, the DMA core executes a DPM transfer, and the requesting device, if located in the external bus, receives an EXT_CS signal. External DPMs use the EXT_ready signal to control the occurrence of the data Transfer.

There are two types of DPM transfers: DPM-to-memory and memory-to-DPM. A DPM-to-memory transfer reads data from a DMA DPM device, while a memory-to-DPM transfer writes data. In both cases, the DPM interface transfers data at the programmed width of the peripheral (DMA0_CRn[PW]). When the DMA buffer is disabled for the active channel (DMA0_CRn[BEN]=0), each DPM transfer causes a corresponding memory operation. When buffering is enabled during a DPM-to-memory transfer, data is collected until the 128-byte buffer is full, the DPM deasserts DMAReqn, or the channel completes. The buffer contents are then written to the target memory as efficiently as possible.

If the Destination address is located on PLB space, the data is written to memory with one burst operation if the destination address is quadword aligned and the buffer is full; or, it is written with a combination of single beat and burst transfers, depending on the destination address alignment and the amount of data stored in the DMA buffer. If the destination address is located on OPB memory space, the DMA controller writes out the content of the DMA buffer with a series of single beat or burst transfers, depending on whether bursting is enabled.

Memory-to-DPM transfers differ since the amount of data that will be requested by the DPM is unknown. If the DMA buffer is disabled (DMA0_CRn[BEN]=0), a discrete source memory read occurs for each element in the DMA transfer. Since this is inefficient, the buffer should only be disabled for low data rate transfers. When the 128-byte buffer is enabled, the controller uses the setting in DMA0_CRn[PF] to prefetch 1, 2, 4, or 8 128-bit quadwords from the source memory. The DMA controller provides data from the buffer until the DPM deasserts its request or the transfer completes. Whenever any of these conditions occurs, any unused data in the DMA buffer is discarded.

For both DPM-to-memory and memory-to-DPM transfers, the DMA controller supports fixed length bursts of 2, 4, 8, or 16 data elements on the peripheral side of the transfer. When bursting is enabled (DMA0_CTCn[BTEN]=1), an active DMA request (DMAReqn) causes the DMA controller to transfer the programmed number of data items (DMA0_CTCn[BSIZ]) in one atomic operation.

19.2.2.2 Software-Initiated Memory-to-Memory Transfers

Software-initiated memory-to-memory transfers between memories with fixed timings provide the best overall performance. During a software-initiated transfer, the DMA controller knows the exact amount of data to be transferred. As a result, when the 128-byte DMA buffer is enabled (DMA0_CRn[BEN]=1) the controller uses bursts as much as possible if the memory is located on PLB space. For memory located on OPB space, bursting has to be enabled (DMA0_CTCn[BTEN]=1) for the DMA controller to burst. To ensure the highest bandwidth with memory locate on PLB space, source and destination addresses should be aligned on 128-byte boundaries.

There are two special cases of software-initiated memory-to-memory transfers.

- Source address increment bit is reset (DMA0_CRn[SAI]=0).

  In this special case, the source memory device responds to a unique address. If the source memory device (FIFO-like device) is located on PLB space, bursting is not possible. If the source memory device is located on OPB space, bursting is possible if enabled (DMA0_CTCn[BTEN]=1).

- Destination address increment bit is reset (DMA0_CRn[DAI]=0).

  In this case, the destination memory device responds to a unique address. If the destination memory device (FIFO-like device) is located on PLB space, bursting is not possible. If the destination memory device is located on OPB space, bursting is possible if enabled (DMA0_CTCn[BTEN]=1).

19.2.3 Scatter/Gather Transfers

Each of the four DMA channels supports scatter/gather transfers. This scatter/gather capability allows the chaining of multiple DMA controller operations within a channel. During a normal DMA operation software must program the control, count and control, source address, and destination address registers for each transfer. Scatter/gather transfers differ in that these registers are automatically loaded from a linked list data structure in system memory. When a channel completes one transfer, the DMA controller loads the next set of configuration values into the channel registers and the channel continues with the new programming.

19.3 Configuration and Status Registers

Table 19-2 lists the DMA configuration and status registers, which are accessed using the **mtdcr** and **mfdcr** instructions. As example, the following PowerPC assembly code writes the control register for DMA channel 0 and then reads the DMA status register:

```
#define DMA0_CR0  0x100
#define DMA0_SR   0x120

    mtdcr    DMA0_CR0,r3              ! write r3 to channel 0 control register
    mfdcr    r4,DMA0_SR               ! read contents of status register into r4
```

The DMA configuration and status registers are readable at any time; however, since each register read requires a separate operation, it is not possible to guarantee that the values read from multiple registers correspond to a state that ever existed in the DMA controller. To illustrate, consider software that reads the destination address registers for channel 0 (DMA0_DAH0 and DMA0_DAL0) and the count for channel 0 (DMA0_CTC0). If the DMA controller updates the destination address or count between these operations, the values read differ from what is expected.

While reads can occur at any time, software must not write the configuration registers for any channel that is currently enabled (DMA0_CRn[CE]=1). The only exception is that a channel may be disabled by reading the channel control register, clearing the channel enable bit, and then writing the new value to the control register. Note that before enabling the channel enable bit, the status has to be queried until the channel busy status bit is cleared. Once a channel is disabled, all of its configuration registers may be reprogrammed as desired.

Table 19-2. DMA Controller Configuration and Status Registers

Mnemonic	Name	DCR Number	Access	Page
DMA0_CR0	DMA Channel Control Register 0	0x100	R/W	19-7661
DMA0_CTC0	DMA Count and Control Register 0	0x101	R/W	19-1063
DMA0_SAH0	DMA Source Address High Register 0	0x102	R/W	19-1164
DMA0_SAL0	DMA Source Address Low Register 0	0x103	R/W	19-1164
DMA0_DAH0	DMA Destination Address High Register 0	0x104	R/W	19-1265
DMA0_DAL0	DMA Destination Address Low Register 0	0x105	R/W	19-1265
DMA0_SGH0	DMA Scatter/Gather Descriptor Address High Register 0	0x106	R/W	19-1366
DMA0_SGL0	DMA Scatter/Gather Descriptor Address Low Register 0	0x107	R/W	19-1366
DMA0_CR1	DMA Channel Control Register 1	0x108	R/W	19-7661
DMA0_CTC1	DMA Count and Control Register 1	0x109	R/W	19-7661
DMA0_SAH1	DMA Source Address High Register 1	0x10A	R/W	19-1164
DMA0_SAL1	DMA Source Address Low Register 1	0x10B	R/W	19-1164
DMA0_DAH1	DMA Destination Address High Register 1	0x10C	R/W	19-1265
DMA0_DAL1	DMA Destination Address Low Register 1	0x10D	R/W	19-1265
DMA0_SGH1	DMA Scatter/Gather Descriptor Address High Register 1	0x10E	R/W	19-1366
DMA0_SGL1	DMA Scatter/Gather Descriptor Address Low Register 1	0x10F	R/W	19-1366
DMA0_CR2	DMA Channel Control Register 2	0x110	R/W	19-7661
DMA0_CTC2	DMA Count and Control Register 2	0x111	R/W	19-1063
DMA0_SAH2	DMA Source Address High Register 2	0x112	R/W	19-1164

Table 19-2. DMA Controller Configuration and Status Registers (continued)

Mnemonic	Name	DCR Number	Access	Page
DMA0_SAL2	DMA Source Address Low Register 2	0x113	R/W	19-11664
DMA0_DAH2	DMA Destination Address High Register 2	0x114	R/W	19-12665
DMA0_DAL2	DMA Destination Address Low Register 2	0x115	R/W	19-12665
DMA0_SGH2	DMA Scatter/Gather Descriptor Address High Register 2	0x116	R/W	19-13666
DMA0_SGL2	DMA Scatter/Gather Descriptor Address Low Register 2	0x117	R/W	19-13666
DMA0_CR3	DMA Channel Control Register 3	0x118	R/W	19-76661
DMA0_CTC3	DMA Count and Control Register 3	0x119	R/W	19-10663
DMA0_SAH3	DMA Source Address High Register 3	0x11A	R/W	19-11664
DMA0_SAL3	DMA Source Address Low Register 3	0x11B	R/W	19-11664
DMA0_DAH3	DMA Destination Address High Register 3	0x11C	R/W	19-12665
DMA0_DAL3	DMA Destination Address Low Register 3	0x11D	R/W	19-12665
DMA0_SGH3	DMA Scatter/Gather Descriptor Address High Register 3	0x11E	R/W	19-13666
DMA0_SGL3	DMA Scatter/Gather Descriptor Address Low Register 3	0x11F	R/W	19-13666
DMA0_SR	DMA Status Register	0x120	R/Clear	19-14667
DMA0_SGC	DMA Scatter/Gather Command Register	0x123	R/W	19-15668
DMA0_SLP	DMA Sleep Mode Register	0x125	R/W	19-16669
DMA0_POL	DMA Polarity Configuration Register	0x126	R/W	19-17670

19.3.1 DMA Channel Control Registers (DMA0_CRn)

The DMA Channel Control Registers (DMA0_CRn) are used to configure and enable their respective DMA channels. Before a DMA channel can transfer data, the channel control, count and control, source address and destination address registers must be programmed. If a DMA channel is setup for scatter/gather transfers (DMA0_SGC[SSGn]=1), the DMA channel control register is automatically loaded from memory. For additional details see "Scatter/Gather Transfers" on page 19-21.detailss see *Scatter/Gather Transfers* on page 674

–

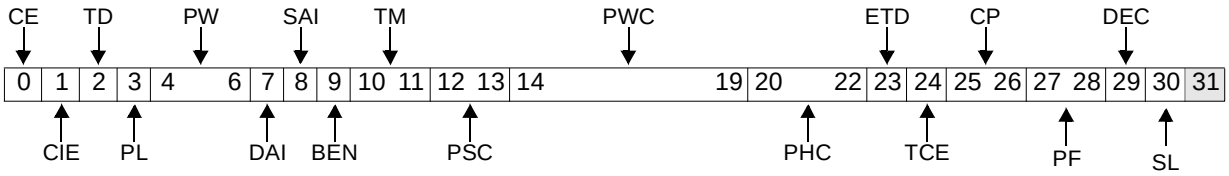


Figure 0-1. DMA Channel Control Registers (DMA0_CR0-DMA0_CR3)

0	CE	Channel Enable 0 Channel is disabled 1 Channel is enabled	This field is automatically cleared when the transfer completes or an error occurs.
1	CIE	Channel Interrupt Enable 0 Disable interrupts from this channel 1 Enable interrupts from this channel	When enabled, interrupts can be generated for terminal count, end of transfer, and errors conditions. Each of these interrupt types must be individually enabled in DMA0_CTn. See "Interrupts."
2	TD	In peripheral mode: 0 Transfers are from memory-to-peripheral 1 Transfers are from peripheral-to-memory In device-paced memory-to-memory mode: 0 Peripheral is at the destination address 1 Peripheral is at the source address	TD is not used (don't care) for software-initiated memory-to-memory transfers.
3	PL	Peripheral Location/Destination Memory Location For memory-to-memory transfers: 0 Destination address located in PLB memory space 1 Destination address located in OPB memory space For peripheral/DPM-to-memory or memory-to-DPM/peripheral transfers: 0 Device located on external bus 1 Device located on the OPB.	
4:6	PW	Peripheral Width/Transfer Width 000 Byte (8 bits) 001 Halfword (16 bits) 010 Word (32 bits) 011 Doubleword (64 bits) 100 Quadword (128 bits)	Transfers involving peripheral, device-paced-memory, and OPB FIFO devices can only be byte, halfword, or word. This limitation also applies to transfers involving EBC memory when SAI=0 or DAI=0. PLB FIFO devices can only be quadword size.

7	DAI	Destination Address Increment 0 Do not increment destination address 1 After each data transfer increment the destination address by: 1, for byte (8-bit) transfers 2, for halfword (16-bit) transfers 4, for word (32-bit) transfers 8, for doubleword (64-bit) transfers 16, for quadword (128-bit) transfers	Valid only when DEC=0. Is required that DAI=1 for peripheral-to-memory and device-paced memory-to-memory transfers since peripheral-to-FIFO type devices or DPM-to-FIFO type devices are not supported.
8	SAI	Source Address Increment 0 Do not increment source address 1 After each data transfer increment the source address by: 1, for byte (8-bit) transfers 2, for halfword (16-bit) transfers 4, for word (32-bit) transfers 8, for doubleword (64-bit) transfers 16, for quadword (128-bit) transfers	Valid only when DEC=0. Is required that SAI=1 for memory-to-peripheral and memory-to-device-paced memory transfers since FIFO-to-peripheral/DPM transfers are not supported.
9	BEN	Buffer Enable 0 Disable DMA 128-byte buffer 1 Enable DMA 128-byte buffer	If BEN=0 discrete read and write operations occur for each data transfer.
10:11	TM	Transfer mode 00 Peripheral 01 Reserved 10 Software-initiated memory-to-memory 11 Device-paced memory-to-memory	
12:13	PSC	Peripheral Setup Cycles 0-3	Number of PerClk cycles that the peripheral bus is idle from the last peripheral bus transaction to DMAAckn becoming active. Used only for the peripheral side of peripheral mode transfers.
14:19	PWC	Peripheral Wait Cycles 0-63	DMAAckn remains active for PWC+1 PerClk cycles. Used only for the peripheral side of peripheral mode transfers.
20:22	PHC	Peripheral Hold Cycles 0-7	The number of PerClk cycles between the time that DMAAckn becomes inactive until the peripheral bus is available for the next bus access. Used only during the peripheral side of peripheral mode transfers.
23	ETD	End-of-Transfer/Terminal Count (EOTn[TCn]) Pin Direction 0 EOTn[TCn] is an EOT input 1 EOTn[TCn] is a TC output	ETD must be set to 1 if the channel is configured for software-initiated memory-to-memory transfers.
24	TCE	Terminal Count (TC) Enable 0 Channel does not stop when TC is reached 1 Channel stops when TC is reached	If TCE=1, it is required that ETD=1.

25:26	CP	Channel Priority 00 Low priority 01 Medium low priority 10 Medium high priority 11 High priority	Actively requesting channels of the same priority are ranked in order by channel number, channel 0 having the highest priority. For more information, see "DMA Arbitration Transfer Priorities."
27:28	PF	Memory Read Prefetch Transfer 00 Prefetch 1 quadword (128-bits) 01 Prefetch 2 quadwords 10 Prefetch 4 quadwords 11 Prefetch 8 quadwords	Used only during memory-to-peripheral and memory to deviced-paced memory transfers. To enable prefetching it is required that BEN=1.
29	DEC	Address Decrement 0 SAI and DAI fields control memory address incrementing. 1 After each data transfer the memory address is decremented by the transfer width.	If DEC=1, it is required that BEN=0. This field is valid only for peripheral mode transfers (TM=00).
30	SL	Source Address Location/Memory Location 0 PLB 1 OPB	For memory-to-memory transfers (TM=1x) SL indicates whether the source memory controller is on the PLB or OPB. For peripheral mode transfers, this field denotes the location of the memory controller.
31		Reserved	

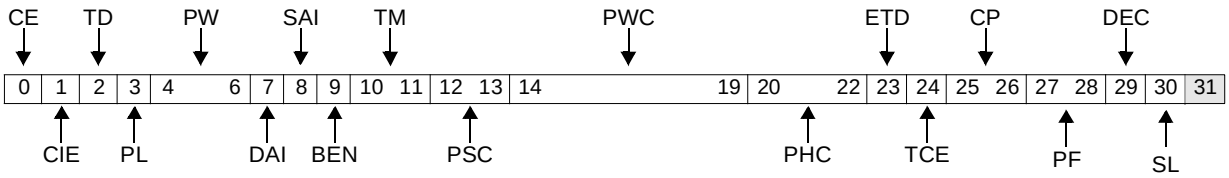


Figure 19-3. DMA Channel Control Registers (DMA0_CR0-DMA0_CR3)

0	CE	Channel Enable 0 Channel is disabled 1 Channel is enabled	This field is automatically cleared when the transfer completes or an error occurs.
1	CIE	Channel Interrupt Enable 0 Disable interrupts from this channel 1 Enable interrupts from this channel	When enabled, interrupts can be generated for terminal count, end of transfer, and errors conditions. Each of these interrupt types must be individually enabled in DMA0_CTn. See "Interrupts."
2	TD	In peripheral mode: 0 Transfers are from memory-to-peripheral 1 Transfers are from peripheral-to-memory In device-paced memory-to-memory mode: 0 Peripheral is at the destination address 1 Peripheral is at the source address	TD is not used (don't care) for software-initiated memory-to-memory transfers.

3	PL	Peripheral Location/Destination Memory Location For memory-to-memory transfers: 0 Destination address located in PLB memory space 1 Destination address located in OPB memory space For peripheral/DPM-to-memory or memory-to-DPM/peripheral transfers: 0 Device located on external bus 1 Device located on the OPB.	
4:6	PW	Peripheral Width/Transfer Width 000 Byte (8 bits) 001 Halfword (16 bits) 010 Word (32 bits) 011 Doubleword (64 bits) 100 Quadword (128 bits)	Transfers involving peripheral, device-paced-memory, and OPB FIFO devices can only be byte, halfword, or word. This limitation also applies to transfers involving EBC memory when SAI=0 or DAI=0. PLB FIFO devices can only be quadword size.
7	DAI	Destination Address Increment 0 Do not increment destination address 1 After each data transfer increment the destination address by: 1, for byte (8-bit) transfers 2, for halfword (16-bit) transfers 4, for word (32-bit) transfers 8, for doubleword (64-bit) transfers 16, for quadword (128-bit) transfers	Valid only when DEC=0. Is required that DAI=1 for peripheral-to-memory and device-paced memory-to-memory transfers since peripheral-to-FIFO type devices or DPM-to-FIFO type devices are not supported.
8	SAI	Source Address Increment 0 Do not increment source address 1 After each data transfer increment the source address by: 1, for byte (8-bit) transfers 2, for halfword (16-bit) transfers 4, for word (32-bit) transfers 8, for doubleword (64-bit) transfers 16, for quadword (128-bit) transfers	Valid only when DEC=0. Is required that SAI=1 for memory-to-peripheral and memory-to-device-paced memory transfers since FIFO-to-peripheral/DPM transfers are not supported.
9	BEN	Buffer Enable 0 Disable DMA 128-byte buffer 1 Enable DMA 128-byte buffer	If BEN=0 discrete read and write operations occur for each data transfer.
10:11	TM	Transfer mode 00 Peripheral 01 Reserved 10 Software-initiated memory-to-memory 11 Device-paced memory-to-memory	
12:13	PSC	Peripheral Setup Cycles 0-3	Number of PerClk cycles that the peripheral bus is idle from the last peripheral bus transaction to DMAAckn becoming active. Used only for the peripheral side of peripheral mode transfers.
14:19	PWC	Peripheral Wait Cycles 0-63	DMAAckn remains active for PWC+1 PerClk cycles. Used only for the peripheral side of peripheral mode transfers.
20:22	PHC	Peripheral Hold Cycles 0-7	The number of PerClk cycles between the time that DMAAckn becomes inactive until the peripheral bus is available for the next bus access. Used only during the peripheral side of peripheral mode transfers.
23	ETD	End-of-Transfer/Terminal Count (EOTn[TCn]) Pin Direction 0 EOTn[TCn] is an EOT input 1 EOTn[TCn] is a TC output	ETD must be set to 1 if the channel is configured for software-initiated memory-to-memory transfers.

24	TCE	Terminal Count (TC) Enable 0 Channel does not stop when TC is reached 1 Channel stops when TC is reached	If TCE=1, it is required that ETD=1.
25:26	CP	Channel Priority 00 Low priority 01 Medium low priority 10 Medium high priority 11 High priority	Actively requesting channels of the same priority are ranked in order by channel number, channel 0 having the highest priority. For more information, see "DMA Arbitration Transfer Priorities."
27:28	PF	Memory Read Prefetch Transfer 00 Prefetch 1 quadword (128-bits) 01 Prefetch 2 quadwords 10 Prefetch 4 quadwords 11 Prefetch 8 quadwords	Used only during memory-to-peripheral and memory to deviced-paced memory transfers. To enable prefetching it is required that BEN=1.
29	DEC	Address Decrement 0 SAI and DAI fields control memory address incrementing. 1 After each data transfer the memory address is decremented by the transfer width.	If DEC=1, it is required that BEN=0. This field is valid only for peripheral mode transfers (TM=00).
30	SL	Source Address Location/Memory Location 0 PLB 1 OPB	For memory-to-memory transfers (TM=1x) SL indicates whether the source memory controller is on the PLB or OPB. For peripheral mode transfers, this field denotes the location of the memory controller.
31		Reserved	

19.3.2 DMA Count and Control Registers (DMA0_CTCn)

The DMA Count and Control Registers (DMA0_CTCn) contain the number of transfers remaining in the DMA transaction for their respective channels when EOTn[TCn] is programmed as a terminal count output, along with additional channel control information. In addition to the transfer count, the Count and Control Registers contain fields used to enable interrupts for terminal count, end of transfer and error conditions. Other fields enable parity checking during the peripheral portion of peripheral type (DMA0_CRn[TM]=00) transfers, define subchannel ID information, and enable bursting during peripheral and device-paced memory transfers. For additional details on bursting, see ~~"Peripheral and Deviced-Paced Memory Bursts" on page 19~~*Peripheral and Deviced-Paced Memory Bursts* on page 672.

When a DMA channel is setup for scatter/gather transfers (DMA0_SGC[SSGn]=1) the count and control register is automatically loaded from memory. For additional details see ~~"Scatter/Gather Transfers" on page 21~~*Scatter/Gather Transfers* on page 674.

The value in the Transfer Count (TC) field of DMA count and control register is interpreted as the number of **transfers** of the width specified in DMA0_CRn[PW], **not** the total number of bytes. The maximum number of transfers is 1024K, and each transfer can be either 1, 2, 4, 8, or 16 bytes as programmed in DMA0_CRn[PW]. The maximum count of 1024K transfers is programmed by writing zero to DMA0_CTCn[TC].

The subchannel ID bits can be used to allow one DMA channel to support up to eight different peripherals. Bits 5:7 control the external logic that selects the current peripheral.

When EOTn[TCn] is programmed as an end-of-transfer input (DMA0_CRn[ETD]=0), the channel will not stop when the count reaches zero. Instead, DMA0_CTCn[TC] continues to count down past zero until EOTn is asserted. For DMA0_CRn[ETD]=0, DMA0_CTCn[TCE] must be set to zero.

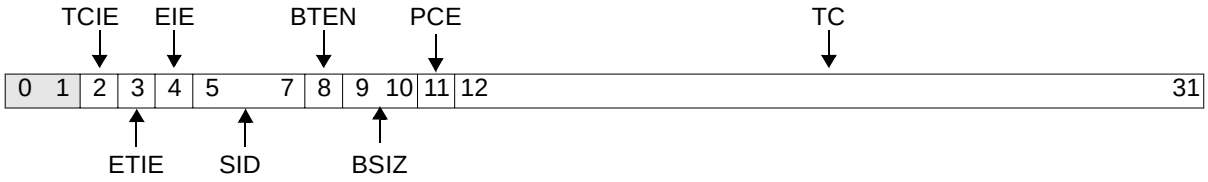


Figure 0-2. DMA Count and Control Registers (DMA0_CT0-DMA0_CT3)

0:1		Reserved	
2	TCIE	Terminal Count Interrupt Enable 0 Disable terminal count interrupts 1 Enable terminal count interrupts	Terminal Count interrupts are further qualified by DMA0_CR0-DMA0_CR3[CIE] and UIC0_ER[DMAn].
3	ETIE	EOT Interrupt Enable 0 Enable end of transfer interrupts 1 Disable end of transfer interrupts	EOT interrupts are further qualified by DMA0_CR0-DMA0_CR3[CIE] and UIC0_ER[DMAn].
4	EIE	Error Interrupt Enable 0 Enable error interrupts 1 Disable error interrupts	Error interrupts are further qualified by DMA0_CR0-DMA0_CR3[CIE] and UIC0_ER[DMAn].
5:7	SID	Subchannel ID	
8	BTEN	Burst Enable 0 Disable bursting 1 Enable bursting	Controls bursting to and from peripheral devices and EBC-attached device-paced memory. See "Peripheral and Deviced-Paced Memory Bursts."
9:10	BSIZ	Burst Size 00 Burst of 2 01 Burst of 4 10 Burst of 8 11 Burst of 16	Determines the burst length used when BEN=1. When BEN=1, TC must be programmed to a multiple of BRSTSIZ. For OPB memory burst transfers, these bits determine the maximum burst size.
11	PCE	Parity Check Enable 0 Disable parity checking 1 Enable parity checking	Enables parity for peripheral mode transfers. See "Data Parity During DMA Peripheral Transfers."
12:31	TC	Transfer Count 0 - 1024K-1	Programming TC=0 results in the maximum count of 1024K transfers.

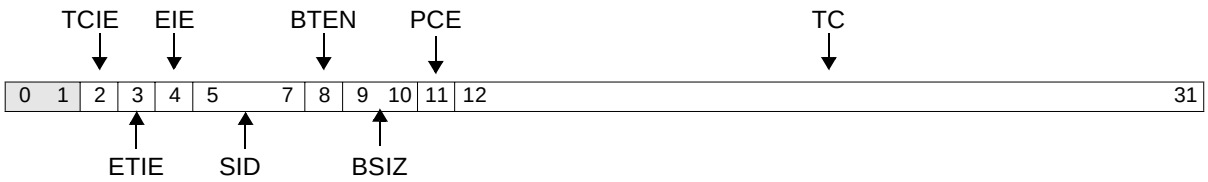


Figure 19-4. DMA Count and Control Registers (DMA0_CT0-DMA0_CT3)

0:1		Reserved

2	TCIE	Terminal Count Interrupt Enable 0 Disable terminal count interrupts 1 Enable terminal count interrupts	Terminal Count interrupts are further qualified by DMA0_CR0-DMA0_CR3[CIE] and UIC0_ER[DMAn].
3	ETIE	EOT Interrupt Enable 0 Enable end of transfer interrupts 1 Disable end of transfer interrupts	EOT interrupts are further qualified by DMA0_CR0-DMA0_CR3[CIE] and UIC0_ER[DMAn].
4	EIE	Error Interrupt Enable 0 Enable error interrupts 1 Disable error interrupts	Error interrupts are further qualified by DMA0_CR0-DMA0_CR3[CIE] and UIC0_ER[DMAn].
5:7	SID	Subchannel ID	
8	BTEN	Burst Enable 0 Disable bursting 1 Enable bursting	Controls bursting to and from peripheral devices and EBC-attached device-paced memory. See "Peripheral and Deviced-Paced Memory Bursts."
9:10	BSIZ	Burst Size 00 Burst of 2 01 Burst of 4 10 Burst of 8 11 Burst of 16	Determines the burst length used when BEN=1. When BEN=1, TC must be programmed to a multiple of BRSTSIZ. For OPB memory burst transfers, these bits determine the maximum burst size.
11	PCE	Parity Check Enable 0 Disable parity checking 1 Enable parity checking	Enables parity for peripheral mode transfers. See "Data Parity During DMA Peripheral Transfers."
12:31	TC	Transfer Count 0 - 1024K-1	Programming TC=0 results in the maximum count of 1024K transfers.

19.3.3 DMA Source Address Registers)

The DMA Source Address Registers (DMA0_SAHn and DMA0_SALn) contain the source address for memory-to-memory and memory-to-peripheral transfers. If a DMA channel is setup for scatter/gather transfers (DMA0_SGC[SSGn]=1) the source address registers are automatically loaded from memory. For additional details ~~see "Scatter/Gather Transfers" on page 21.~~ *Scatter/Gather Transfers* on page 674

The source address must be aligned at the transfer width programmed in DMA0_CRn[TW], otherwise the error bit (DMA0_SR[RIn]) is set for the channel and no transfer occurs. If the source address increment bit in the channel's control register is set (DMA0_CRn[SAI]) the address is incremented by the transfer width after each data transfer. In contrast, if the channel is performing a memory-to-peripheral transfer and the address decrement bit is set (DMA0_CRn[DEC]=1), the address is decremented by the transfer width after each transfer.

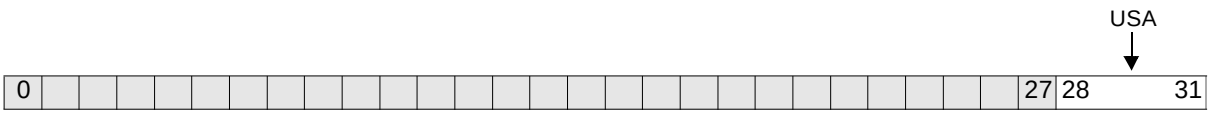


Figure 0-3. DMA Source Address High Registers (DMA0-SAH0-DMA0-SAH3)

0:27		Reserved

28:31	USA	Upper 4-bits of source address for memory-to-memory and memory-to-peripheral transfers.

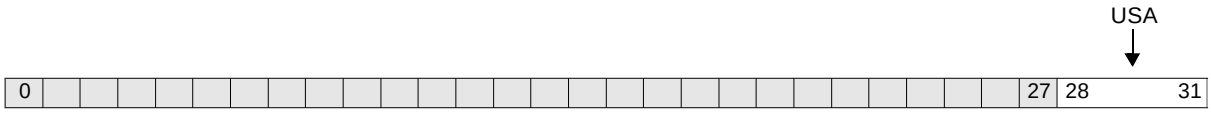


Figure 19-5. DMA Source Address High Registers (DMA0-SAH0-DMA0-SAH3)

0:27		Reserved
28:31	USA	Upper 4-bits of source address for memory-to-memory and memory-to-peripheral transfers.

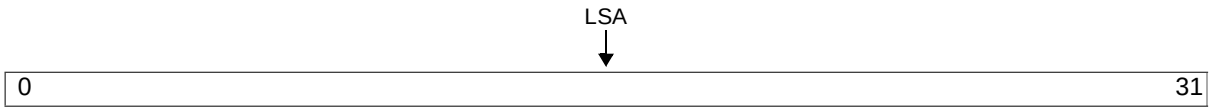


Figure 0-4. DMA Source Address Low Registers (DMA0-SAL0-DMA0-SAL3)

0:31	LSA	Lower 32-bits of source address for memory-to-memory and memory-to-peripheral transfers.

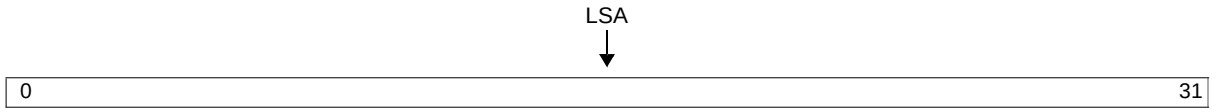


Figure 19-6. DMA Source Address Low Registers (DMA0-SAL0-DMA0-SAL3)

0:31	LSA	Lower 32-bits of source address for memory-to-memory and memory-to-peripheral transfers.

19.3.4 DMA Destination Address Registers

The DMA Destination Address Registers (DMA0_DAHn and DMA0_DALn) contain the destination address for memory-to-memory and peripheral-to-memory transfers. When a DMA channel is configured for scatter/gather transfers (DMA_SGC[SSGn]=1) the destination address registers are automatically loaded from memory. For additional details see ~~see "Scatter/Gather Transfers" on page 21.~~*Scatter/Gather Transfers* on page 674

The destination address must be aligned at the transfer width programmed in DMA0_CRn[TW], otherwise the error bit (DMA0_SR[RIn]) is set for the channel and no transfer occurs. If the destination address increment bit in the channel's control register is set (DMA0_CRn[DAI]) the address is incremented by the transfer width after each data transfer. However, if the channel is performing a peripheral-to-memory transfer and the address decrement bit is set (DMA0_CRn[DEC]=1), the destination address is decremented by the transfer width after each transfer.

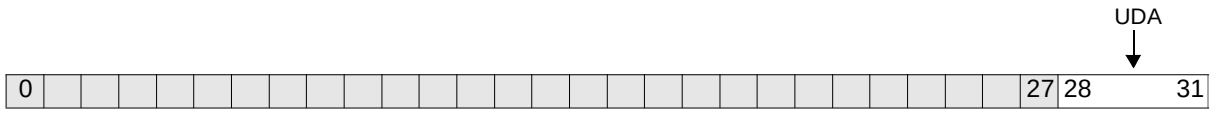


Figure 0-5. DMA Destination Address High Registers (DMA0_DAH0-DMA0_DAH3)

0:27		Reserved
28:31	UDA	Upper 4-bits of destination address for memory-to-memory and peripheral-to-memory transfers.

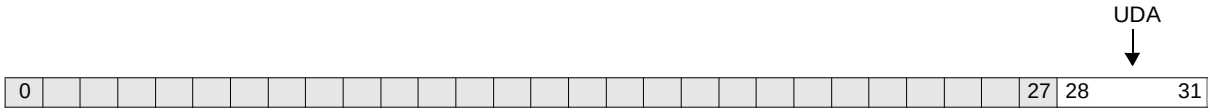


Figure 19-7. DMA Destination Address High Registers (DMA0_DAH0-DMA0_DAH3)

0:27		Reserved
28:31	UDA	Upper 4-bits of destination address for memory-to-memory and peripheral-to-memory transfers.

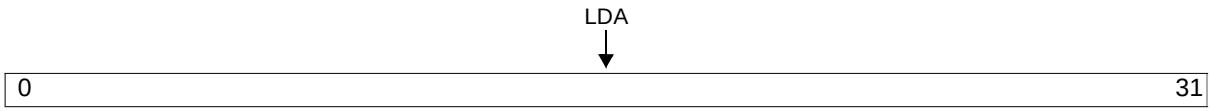


Figure 0-6. DMA Destination Address Low Registers ((DMA0_DAL0-DMA0_DAL3)

0:31	LDA	Lower 32-bits of destination address for memory-to-memory and peripheral-to-memory transfers.

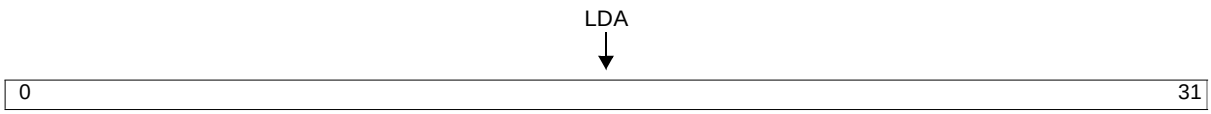


Figure 19-8. DMA Destination Address Low Registers ((DMA0_DAL0-DMA0_DAL3)

0:31	LDA	Lower 32-bits of destination address for memory-to-memory and peripheral-to-memory transfers.

19.3.5 DMA Scatter/Gather Descriptor Address Registers

When a DMA channel is setup for scatter/gather transfers (DMA0_SGC[SSGn]=1), the Scatter/Gather Descriptor Address Registers (DMA0_SGHn and DMA0_SGLn) contain the memory address of the next scatter/gather descriptor table. Prior to starting a scatter/gather transfer, software must write the address of the channel's descriptor table to DMA0_SGHn and DMA0_SGLn. Once the scatter/gather transfer starts, DMA0_SGHn and DMA0_SGLn are automatically updated from the descriptor table. For additional details see ~~see "Scatter/Gather Transfers" on page 21.~~*Scatter/Gather Transfers* on page 674

Figure 0-7. DMA Scatter/Gather Descriptor Address High Registers (DMA0_SGH0-DMA0_SGH3)

0:27		Reserved
28:31	USGD	Upper 4-bits of address of next scatter/gather descriptor table.

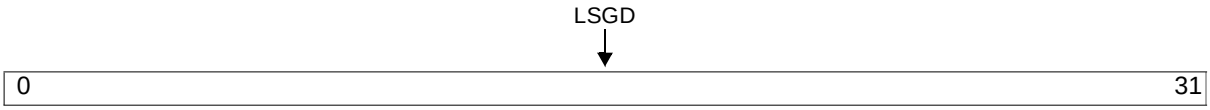


Figure 0-8. DMA Scatter/Gather Descriptor Address Low Registers (DMA0_SGL0-DMA0_SGL3)

0:31	LSGD	Lower 32-bits of address of next scatter/gather descriptor table.

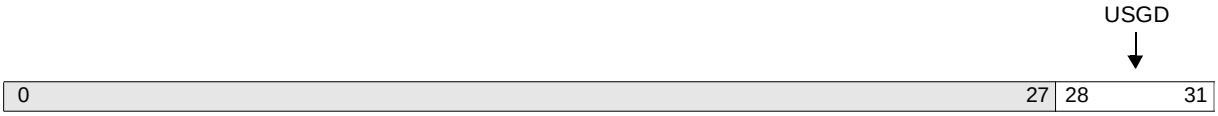


Figure 19-9. DMA Scatter/Gather Descriptor Address High Registers (DMA0_SGH0-DMA0_SGH3)

0:27		Reserved
28:31	USGD	Upper 4-bits of address of next scatter/gather descriptor table.

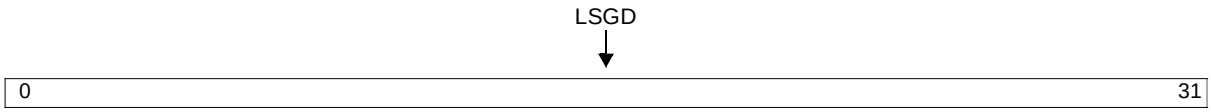


Figure 19-10. DMA Scatter/Gather Descriptor Address Low Registers (DMA0_SGL0-DMA0_SGL3)

0:31	LSGD	Lower 32-bits of address of next scatter/gather descriptor table.

19.3.6 DMA Status Register (DMA0_SR)

As shown in ~~Figure 19-3~~Figure 19-11, the DMA Status Register (DMA0_SR) provides status information for each of the DMA channels. Bits in DMA0_SR are set in hardware, and can be either read or cleared by software. Clearing is performed by writing a word to DMA0_SR containing a 1 in any bit position to be cleared and 0 in all other bit positions.

The terminal count status (DMA0_SR[CSn]), end of transfer status (DMA0_SR[TSn]) and error status (DMA0_SR[RIn]) must be cleared for a DMA channel to operate. If a scatter/gather operation generates an interrupt for any of the above conditions, the channel pauses until software clears the associated status field(s) in DMA0_SR.

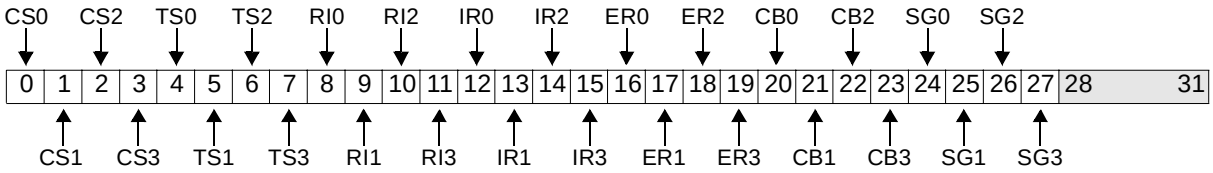


Figure 0-9. DMA Status Register (DMA0_SR)

0:3	CS[0:3]	Channel 0-3 Terminal Count Status 0 Terminal count has not occurred 1 Terminal count has been reached	Set when the transfer count reaches 0 and the DMA0_CRn(TCE) bit is set.
4:7	TS[0:3]	Channel 0-3 End of Transfer Status 0 End of transfer has not been requested 1 End of transfer has been requested	Only valid for channels with DMA0_CRn[ETD]=0 (no memory-to-memory transfers).
8:11	RI[0:3]	Channel 0-3 Error Status 0 No error 1 Error occurred	See Errors section for more information.
12:15	IR[0:3]	Internal DMA Request 0 No internal DMA request pending 1 Internal DMA request is pending	
16:19	ER[0:3]	External DMA Request 0 No external DMA request pending 1 External DMA request is pending	
20:23	CB[0:3]	Channel Busy 0 Channel is idle 1 Channel currently active	During catter/gather fetches, these bits are active.
24:27	SG[0:3]	Scatter/Gather Status 0 No scatter/gather operation in progress 1 Scatter/gather operation in progress	For multiple scatter/gather links, the scatter/gather status bit is set when the first link is loaded and is kept adctive until the last link is completed.
28:31		Reserved	

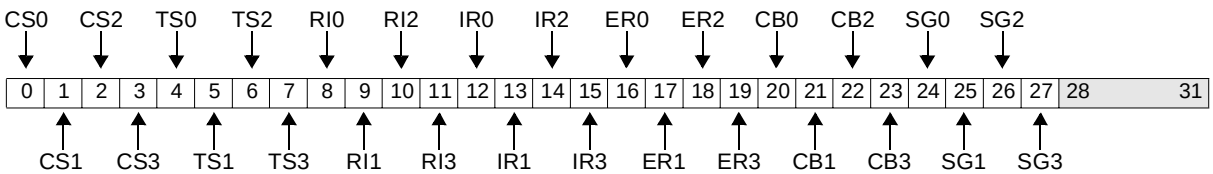


Figure 19-11. DMA Status Register (DMA0_SR)

0:3	CS[0:3]	Channel 0-3 Terminal Count Status 0 Terminal count has not occurred 1 Terminal count has been reached	Set when the transfer count reaches 0 and the DMA0_CRn(TCE) bit is set.

4:7	TS[0:3]	Channel 0-3 End of Transfer Status 0 End of transfer has not been requested 1 End of transfer has been requested	Only valid for channels with DMA0_CRn[ETD]=0 (no memory-to-memory transfers).
8:11	RI[0:3]	Channel 0-3 Error Status 0 No error 1 Error occurred	See Errors section for more information.
12:15	IR[0:3]	Internal DMA Request 0 No internal DMA request pending 1 Internal DMA request is pending	
16:19	ER[0:3]	External DMA Request 0 No external DMA request pending 1 External DMA request is pending	
20:23	CB[0:3]	Channel Busy 0 Channel is idle 1 Channel currently active	During catter/gather fetches, these bits are active.
24:27	SG[0:3]	Scatter/Gather Status 0 No scatter/gather operation in progress 1 Scatter/gather operation in progress	For multiple scatter/gather links, the scatter/gather status bit is set when the first link is loaded and is kept adctive until the last link is completed.
28:31		Reserved	

19.3.7 DMA Scatter/Gather Command Register (DMA0_SGC)

The DMA Scatter/Gather Command Register (DMA0_SGC) is a 32-bit register, of which 12-bits are implemented. Bits 0:3 are the Start Scatter/Gather Enable bits for channels 0 to 3, bits 4:7 determine whether a given channel's Scatter/Gather Descriptor Table resides in PLB or OPB address space, and bits 16:19 are the corresponding Enable Mask bits for the Start Scatter/Gather Enable bits. Setting a Start Scatter/Gather Enable bit causes the selected channel to begin a scatter/gather operation, while writing a 0 stops the Scatter/Gather operation. To start or stop a specific Scatter/Gather channel, the corresponding Enable Mask bit must be set to 1; otherwise, the register holds the previous value. Note that halting a scatter/gather transfer prevents the channel from continuing to the next descriptor, but does not stop the transfer currently in progress.

Upon completion of a scatter/gather sequence of transfers the DMA controller clears DMA0_SGC[SSGn].

If an error occurs when the DMA controller is reading the Scatter/Gather descriptor table, DMA0_SGC[SSGn] is cleared for the affected channel, and the channel's error status bit (DMA0_SR[RIn]) is set.

For additional details ~~see "Scatter/Gather Transfers" on page 21.~~see *Scatter/Gather Transfers* on page 674

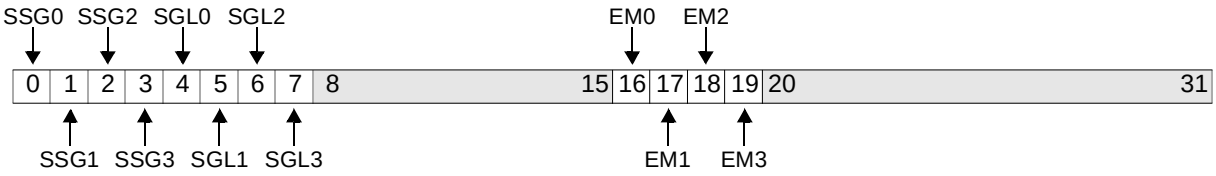


Figure 0-10. DMA Scatter/Gather Command Register (DMA0_SGC)

0:3	SSG[0:3]	Start Scatter/Gather for channels 0-3 0 Scatter/gather support is disabled 1 Scatter/gather support is enabled	To start a scatter/gather operation for channel n, EM[n] must also be set.
4:7	SGL[0:3]	Scatter/Gather Descriptor Table Location for channels 0-3 0 PLB memory space 1 OPB memory space	A channel's descriptor table may not cross between PLB and OPB address space. see "Scatter/Gather Transfers" on page 19-21.
8:15		Reserved	
16:19	EM[0:3]	Enable Mask for channels 0-3 0 Writes to SSG[n] and SGL[n] are ignored 1 Allow writing to SSG[n] and SGL[n]	To write SSG[n] or SGL[n], EM[n] must be set. Otherwise, writing SSG[n] or SGL[n] has no effect.
20:31		Reserved	

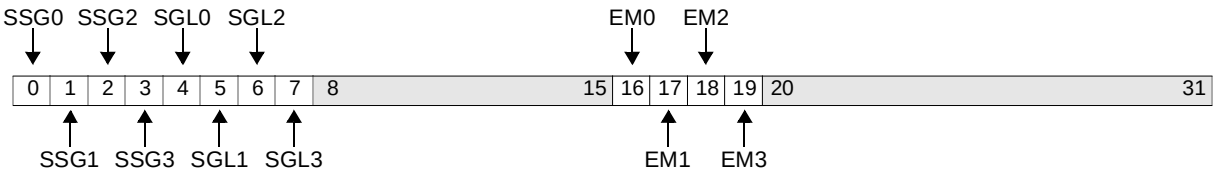


Figure 19-12. DMA Scatter/Gather Command Register (DMA0_SGC)

0:3	SSG[0:3]	Start Scatter/Gather for channels 0-3 0 Scatter/gather support is disabled 1 Scatter/gather support is enabled	To start a scatter/gather operation for channel n, EM[n] must also be set.
4:7	SGL[0:3]	Scatter/Gather Descriptor Table Location for channels 0-3 0 PLB memory space 1 OPB memory space	A channel's descriptor table may not cross between PLB and OPB address space. See *Scatter/Gather Transfers* on page 674.
8:15		Reserved	
16:19	EM[0:3]	Enable Mask for channels 0-3 0 Writes to SSG[n] and SGL[n] are ignored 1 Allow writing to SSG[n] and SGL[n]	To write SSG[n] or SGL[n], EM[n] must be set. Otherwise, writing SSG[n] or SGL[n] has no effect.
20:31		Reserved	

19.3.8 DMA Sleep Mode Register (DMA0_SLP)

The Sleep Mode Register (DMA0_SLP) enables the DMA controller to enter sleep (low-power) mode and programs the number of PLB clock cycles to wait when the controller is idle before going to sleep. The DMA controller only goes to sleep when no DMA channels are enabled, no scatter/gather operation is pending, and no configuration or status (DCR) register operations are in progress. Reading or writing any of the DMA DCRs awakens the DMA controller.

To enable sleep mode set DMA0_SLP[SME] and CPM0_ER[DMA]. When sleep mode is enabled and the DMA controller becomes idle the 10-bit idle timer begins counting down from the programmed value. Only the upper 5 bits of the idle counter are programmable; the lower 5 bits are hardcoded to 5'0b11111; therefore, the minimum granularity of the idle timer is 32 PLB clock cycles. When the counter reaches zero, the controller is placed in sleep mode.

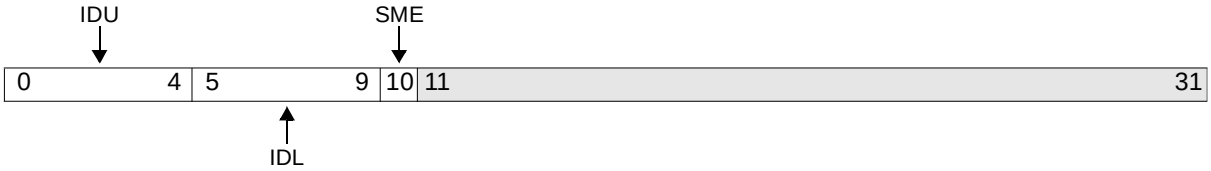


Figure 0-11. DMA Sleep Mode Register (DMA0_SLP)

0:4	IDU	Idle Timer Upper 0-31	Upper 5 bits of the idle timer.
5:9	IDL	Idle Timer Lower Hardcoded to 0b11111	Lower 5-bit portion of the idle timer. Writing this field has no effect.
10	SME	Sleep Mode Enable 0 Sleep disabled 1 Sleep enabled	If SME=1, also set CPM0_ER[DMA] to enable the Clock and Power Management macro to put the DMA controller to sleep.
11:31		Reserved	

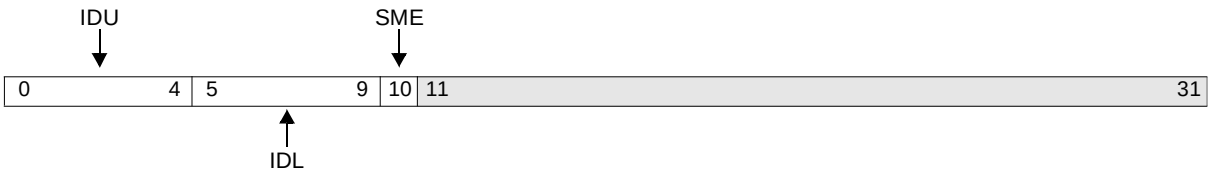


Figure 19-13. DMA Sleep Mode Register (DMA0_SLP)

0:4	IDU	Idle Timer Upper 0-31	Upper 5 bits of the idle timer.
5:9	IDL	Idle Timer Lower Hardcoded to 0b11111	Lower 5-bit portion of the idle timer. Writing this field has no effect.

10	SME	Sleep Mode Enable 0 Sleep disabled 1 Sleep enabled	If SME=1, also set CPM0_ER[DMA] to enable the Clock and Power Management macro to put the DMA controller to sleep.
11:31		Reserved	

19.3.9 DMA Polarity Configuration Register (DMA0_POL)

The Polarity Configuration Register (DMA0_POL) is used to set the polarity (active state) of the external DMA I/O signals: DMAReqn, DMAAckn, and EOTn[TCn]. As shown in ~~Figure 19-14~~Figure 19-14, if a bit in DMA0_POL is zero, the corresponding signal is active high, otherwise the signal is active low.

Whenever any of the EOT polarities are changed (DMA0_POL[EnP), software must subsequently clear the corresponding EOT status bits in the DMA Status Register (DMA0_SR[TS0:3]) prior to enabling the associated DMA channel. This is necessary to prevent a channel from being disabled because of an incorrect EOT status stored in the status bits.

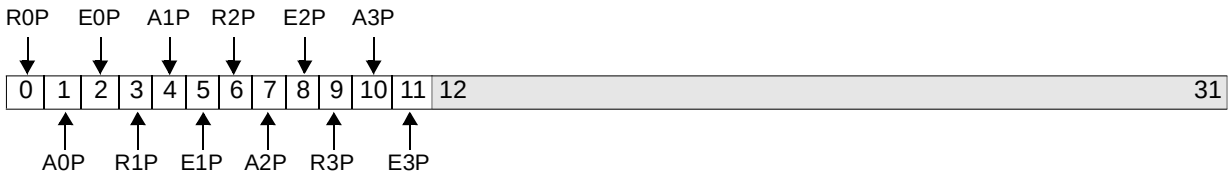


Figure 0-12. DMA Polarity Configuration Register (DMA0_POL)

0	R0P	DMAReq0 Polarity 0 Active high 1 Active low
1	A0P	DMAAck0 Polarity 0 Active high 1 Active low
2	E0P	EOT0[TC0] Polarity 0 Active high 1 Active low
3	R1P	DMAReq1 Polarity 0 Active high 1 Active low
4	A1P	DMAAck1 Polarity 0 Active high 1 Active low
5	E1P	EOT1[TC1] Polarity 0 Active high 1 Active low
6	R2P	DMAReq2 Polarity 0 Active high 1 Active low

7	A2P	DMAAck2 Polarity 0 Active high 1 Active low
8	E2P	EOT2[TC2] Polarity 0 Active high 1 Active low
9	R3P	DMAReq3 Polarity 0 Active high 1 Active low
10	A3P	DMAAck3 Polarity 0 Active high 1 Active low
11	E3P	EOT3[TC3] Polarity 0 Active high 1 Active low
12:31		Reserved

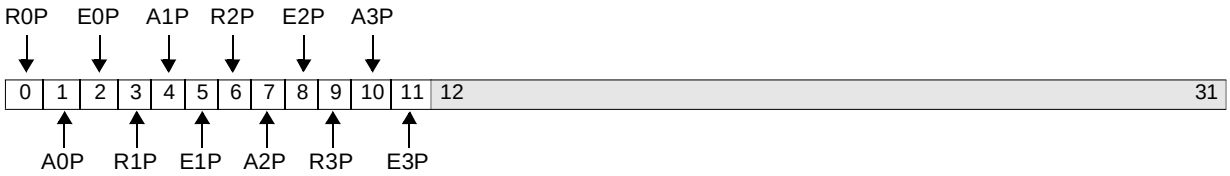


Figure 19-14. DMA Polarity Configuration Register (DMA0_POL)

0	R0P	DMAReq0 Polarity 0 Active high 1 Active low
1	A0P	DMAAck0 Polarity 0 Active high 1 Active low
2	E0P	EOT0[TC0] Polarity 0 Active high 1 Active low
3	R1P	DMAReq1 Polarity 0 Active high 1 Active low
4	A1P	DMAAck1 Polarity 0 Active high 1 Active low
5	E1P	EOT1[TC1] Polarity 0 Active high 1 Active low
6	R2P	DMAReq2 Polarity 0 Active high 1 Active low
7	A2P	DMAAck2 Polarity 0 Active high 1 Active low

8	E2P	EOT2[TC2] Polarity 0 Active high 1 Active low
9	R3P	DMAReq3 Polarity 0 Active high 1 Active low
10	A3P	DMAAck3 Polarity 0 Active high 1 Active low
11	E3P	EOT3[TC3] Polarity 0 Active high 1 Active low
12:31		Reserved

19.4 Channel Priorities

The priority of DMA transfers is controlled on a per-channel basis by the channel priority field in the channel control register. Table 19-3 shows the different priority settings for DMA0_CRn[PW].

Table 19-3. DMA Transfer Priorities

DMA0_CRx[CP]	Priority Level
0b00	Low
0b01	Medium Low
0b10	Medium High
0b11	High

These priorities serve two purposes. First, the DMA controller arbitrates among all actively requesting channels and selects the highest priority one for service. If multiple channels request at the same priority, the arbiter selects the lowest numbered channel for service. Secondly, DMA0_CRn[CP] determines the priority of the internal PLB transactions that the DMA controller uses to read and write data. During a Scatter/Gather transfer, the next descriptor is read using the priority current programmed in DMA0_CRn[CP].

19.5 Data Parity During DMA Peripheral Transfers

The DMA controller works in conjunction with the peripheral bus controller (EBC) to generate and check parity during peripheral mode DMA transfers. When DMA0_CTCn[PCE]=1 parity checking is enabled for peripheral mode transfers on channel n.

During memory-to-memory transfers any data error checking and/or correction is dependent on the configuration of the relevant memory controller. For example, if ECC is enabled on the SDRAM controller, only uncorrectable errors are reported to the DMA controller. Similarly, an EBC memory bank with parity enabled will report an error if a parity error is encountered during a memory read operation. See Peripheral Bank Access Parameters Register (EBCx_BnAP) in the EBC Chapter for more information on peripheral bus parity checking.

19.6 Peripheral and Deviced-Paced Memory Bursts

Normally, peripheral and device-paced (hardware initiated) DMA transfers move one data item at a time. The device requesting DMA service asserts DMAReqn and the DMA controller returns DMAAckn for a peripheral transfer or PerCSn for a device-paced transfer. This process continues for each item of the DMA transfer.

To improve performance, the DMA controller supports fixed-length bursts during the peripheral portion of peripheral mode DMA transfers and on the device-paced side of device-paced memory-to-memory transfers. When bursting is enabled (DMA0_CTCn[BTEN]=1), the DMA controller responds to an active DMAReqn by reading or writing DMA0_CTCn[BSIZ] data items in one atomic, non-interruptible operation. The timing on the external interface is dictated by DMA0_CRn for peripheral mode transfers and EBCx_BnAP for device-paced memory-to-memory transfers. The external device is required to keep DMAReqn asserted until at least the first transfer of the burst. During the final transfer of the fixed-length burst DMAReqn is sampled on the PerClk edge where DMAReqn or PerCSn goes inactive. To prevent another burst from starting, DMAReqn must be sampled inactive at this point. This requirement means that single-cycle DMA transfers require that DMAReqn be deasserted via combinational logic.

When a DMA channel has bursting enabled (DMA0_CTCn[BTEN]=1) it is required that the prefetch count (DMA0_CRn[PF]) be sufficient to complete at least one fixed-length burst transfer. In addition, the transfer count (DMA0_CTCn[TC]) must be a multiple of the burst size (DMA0_CTCn[BSIZ]). If either of these requirements is not met, an error results, no data transfer occurs, and the channel is disabled. For additional details see "Errors" on page 19-19.see *Errors* on page 672

19.7 Errors

The DMA controller detects and reports five types of errors: address alignment, burst count, burst prefetch, PLB/OPB timeout, and slave errors. The DMA controller reports errors through the channel error status bit in the DMA status register (DMA0_SR[RIn]). Whenever the error status bit for a channel is set, the channel enable bit (DMA0_CRn[CE]) is cleared, disabling the channel. An interrupt signal is also presented to the interrupt controller if DMA0_CRn[CIE] and DMA0_CTCn[EIE] are set. For more information on interrupt processing, see see "DMA Interrupts" on page 21.*DMA Interrupts* on page 674

When an error is detected during the execution of an external cycle, the DMA terminates the transfer and deasserts all external signals.

When the DMA controller has multiple channels active, an error may be reported on the current channel which was in actuality caused by a previously active channel. This causes the current channel to have its error status bit set; therefore, for deterministic error analysis with multiple DMA channels active, the slave bus controller's error status registers (the bus error address register in particular) must be queried to isolate the actual channel that encountered the error. In any case, the channel causing the errors will eventually cause all active channels, including itself, to be disabled.

19.7.1 Address Alignment Error

The source address (DMA0_SAHn || DMA0_SALn) and destination address (DMA0_DAHn || DMA0_DALn) registers must be aligned to the programmed transfer width (DMA0_CRn[PW]).The address alignment rules are outlined in Table 19-4. In addition, when a channel is configured for scatter/gather transfers, the scatter/gather table must be quadword-aligned. If the source, destination, and scatter/gather address registers are not appropriately aligned, an error occurs immediately after the channel is enabled.

Table 19-4. Address Alignment Requirements

DMA0_CRx[PW] Setting	Required Alignment for: Source Address (DMA0_SAHn || DMA0_SALn) Destination Address (DMA0_DAHn || DMA0_DALn)
0b000	Byte (8-bit)
0b001	Halfword (16-bit)
0b010	Word (32-bit)
0b011	Doubleword (64-bit)
0b100	Quadword (128-bit)

19.7.2 Burst Count Error

If peripheral/device-paced bursting is enabled for a particular DMA channel (DMA0_CTCn[BTEN]=1), the programmed transfer count (DMA0_CTCn[TC]) must be a multiple of the burst size (DMA0_CTCn[BSIZ]).

19.7.3 Burst Prefetch Error

If peripheral/device-paced bursting is enabled for a particular DMA channel (DMA0_CTCn[BTEN]=1), the programmed prefetch count (DMA0_CRn[PF]) must be sufficient to perform at least one fixed-length burst. More precisely, DMA0_CRn[PF] must prefetch at least DMA0_CTCn[BSIZ] * DMA0_CRn[PW] bytes.

19.7.4 PLB or OPB Timeout

The DMA controller uses PLB and OPB operations to read and write memory. A PLB or OPB timeout results if the DMA controller attempts to access a non-existent memory location. This will occur if the source, destination or scatter/gather address registers do not map to valid memory locations.

19.7.5 Slave Transfer Errors

If the DMA controller detects an error from a slave, it stops the execution of the current transfer, disables the channel, and then reports an error. If a signal is asserted by the PLB slave as a result of such error, the DMA finishes any active read/write pair transfer before disabling the channel. An SDRAM uncorrectable ECC error and an EBC bank protection error are examples of slave errors.

19.8 DMA Interrupts

Each DMA channel can generate interrupts for end of transfer, terminal count and error conditions. All interrupts from a particular DMA channel are enabled by setting the channel enable bit in the channel's control register (DMA0_CRn[CIE=1) and enabling the specific type of interrupt in the count and control register: end of transfer (DMA0_CTCn[ETIE]), terminal count (DMA0_CTCn[TCIE]), and error (DMA0_CTCn[EIE]). When

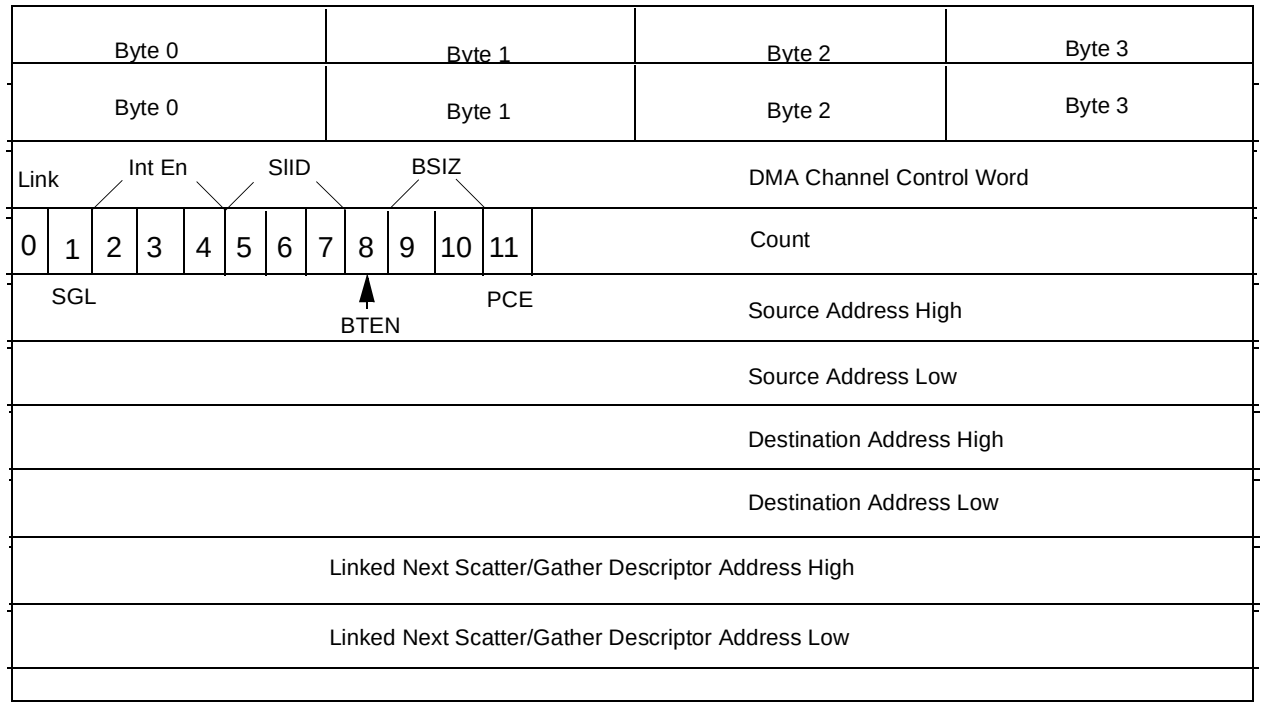
an interrupt occurs for a given channel, the DMA controller sends a signal to the Universal Interrupt Controller. For the PPC440GP's CPU to take an exception, interrupts from the particular DMA channel must be enabled in the interrupt controller's interrupt enable register (UIC0_ER[DMA0]=1). Also, the CPU's machine state register's interrupt enable bit must be enabled for the appropriate interrupt type (critical or non-critical), MSR[EE,CE]. See ~~Chapter 9, "Universal Interrupt Controller"~~*Universal Interrupt Controller* on page 299 for more information on interrupt controller processing.

When the DMA controller generates an interrupt, the interrupt remains active until the appropriate bits are cleared in the DMA Status Register (DMA0_SR). In addition, interrupts from a channel performing a scatter/gather transfer cause the channel to pause until the interrupt is cleared.

19.9 Scatter/Gather Transfers

With a normal DMA transfer it is necessary to program a channel's control, count and control, source, and destination registers for each transfer. The scatter/gather capability of the DMA controller provides a more efficient solution for applications that require multiple transactions on a single DMA channel. Instead of individually programming a channel's registers, software creates a set (linked list) of descriptor tables in system memory. Table 19-5 illustrates the required table format

Table 19-5. Scatter/Gather Descriptor Table



The most significant bit of word 2, the transfer count and configuration register value, is the link (LK) bit. Bit number 1 (SGL) of the same word is the scatter/gather locator bit. It determines the location of the next scatter/gather table (0=PLB space, 1=OPB space).Other than these bits, the contents of the descriptor table are identical to the associated register definitions.

To configure a channel for a scatter/gather transfer the DMA Scatter/Gather Descriptor Address Registers (DMA0_SGHn and DMA0_SGLn) for the channel are set to the address of the first descriptor table, which must be quadword (16 byte) aligned. The linked list of descriptor tables for a given programming of the DMA controller must not cross between PLB or OPB address space. The location of the linked list is programmed into DMA0_SGC[SGLn].

To begin a scatter/gather transfer, software writes DMA0_SGC with the enable mask (DMA0_SGC[EMn]) set, the start scatter/gather bit (DMA0_SGC[SSGn]) set and indicates the location (PLB or OPB) of the descriptor tables via DMA0_SGC[SGLn]. The DMA controller then reads the descriptor table at address (DMA0_SGHn || DMA0_SGLn) and updates the DMA controller registers as shown in Table 19-6. Upon receiving the data from the scatter/gather descriptor table, the channel's terminal count status bit (DMA0_SR[TCn]) and end of transfer status bit (DMA0_SR[EOTn]) are automatically cleared.

Table 19-6. DMA Registers Loaded from Scatter/Gather Descriptor Table

Descriptor Table Entry	Register Loaded
Channel Control Word	DMA0_CRn
Source Address	DMA0_SAHx and DMA0_SALn
Destination Address	DMA0_DAHn and DMA0_DALn
Count and Control	DMA0_CTCn
Next Descriptor Address	DMA0_SGHn and DMA0_SGLn

After loading the channel's registers from the descriptor table, the transfer functions as a normal non-scatter/gather operation.

If the LK (link) bit was not set the scatter/gather process stops when the current transfer completes. Otherwise, the DMA controller reads the descriptor table at address (DMA0_SGHn || DMA0_SGLn) and the process repeats.

19.10 Programming the DMA Controller

Before the DMA controller can transfer data it must be configured, both globally and on a per-channel basis. Global settings include the DMA Polarity Register (DMA0_POL) and DMA Sleep Mode Register (DMA0_SLP). For most applications, these registers should be configured when the DMA controller is first initialized. To prevent spurious activity resulting from changing the active level for DMAReqn, DMAAckn, or EOTn[TCn], a channel's configuration in the Polarity Register should not be altered when the channel is enabled (DMA0_CRn[CE]=1).

Each channel has Control (DMA0_CRn), Count and Control (DMA0_CTCn), Source Address (DMA0_SAHn and DMA0_SALn), Destination Address (DMA0_DAHn and DMA0_DALn), and Scatter/Gather Descriptor Address (DMA0_SGHn and DMA0_SGLn) registers. The type of DMA transfer determines which of these registers must be programmed and what causes the channel to start. In all cases, the terminal count (CSn), end of transfer (TSn) and error status (RIn) bits in the DMA Status Register (DMA0_SR) must be cleared or the channel will not start.

The programming information that follows assumes that the DMA controller is operating in non-scatter/gather mode. To use scatter/gather transfers the channel configuration data must be written into a set of descriptor tables in system memory. See ~~"Scatter/Gather Transfers" on page 21~~*Scatter/Gather Transfers* on page 674 for additional details.

19.10.1 Peripheral Mode Transfers

DMA peripherals are either devices attached to the EBC interface via the DMAReqn and DMAAckn lines, or the internal serial ports (UART0, UART1). During a peripheral mode transfer, a peripheral asserts DMAReqn to request a DMA transfer. For metastability protection DMAReqn is double latched in the DMA upon assertion, and sampled with a single latch on the deassertion.

Timings on the memory-access portion of peripheral mode DMA transfers are governed by the configuration of the associated memory controller. In contrast, timing during the peripheral portion of the transfer is controlled by the PSC, PWC and PHC fields in the DMA Channel Control Registers. The effect of these parameters on peripheral timings is illustrated in Figure 19-15 and Figure 19-16. Although shown as active high, the polarity (active state) of DMAReqn, DMAAckn, and EOT0[TCn] are programmable via the DMA Polarity Register (DMA0_POL).

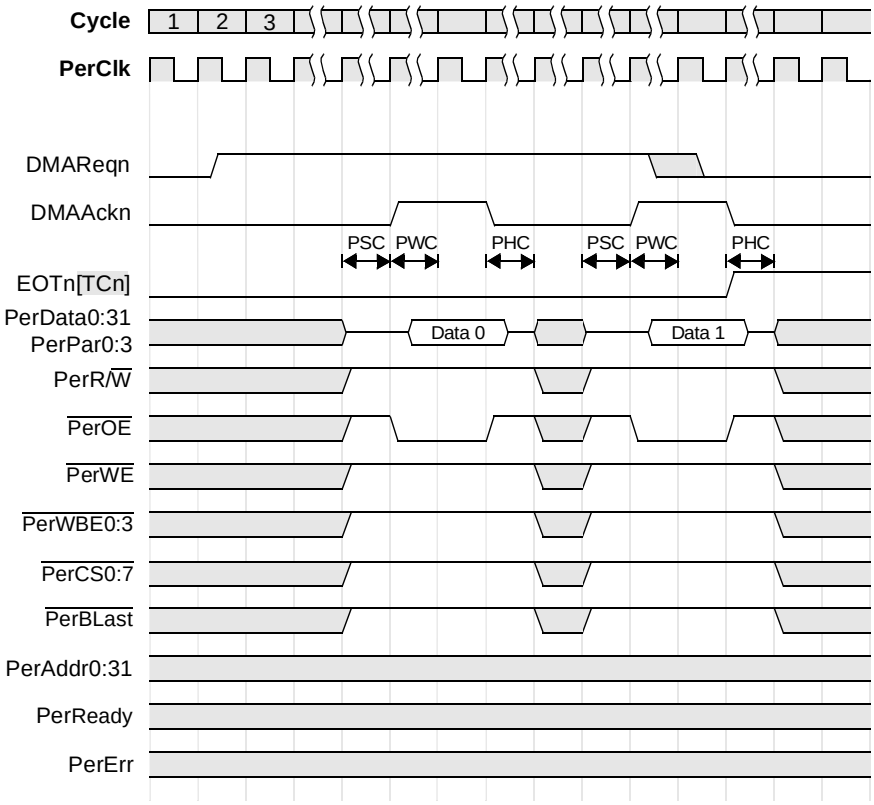


Figure 19-15. Peripheral-to-Memory DMA Transfer

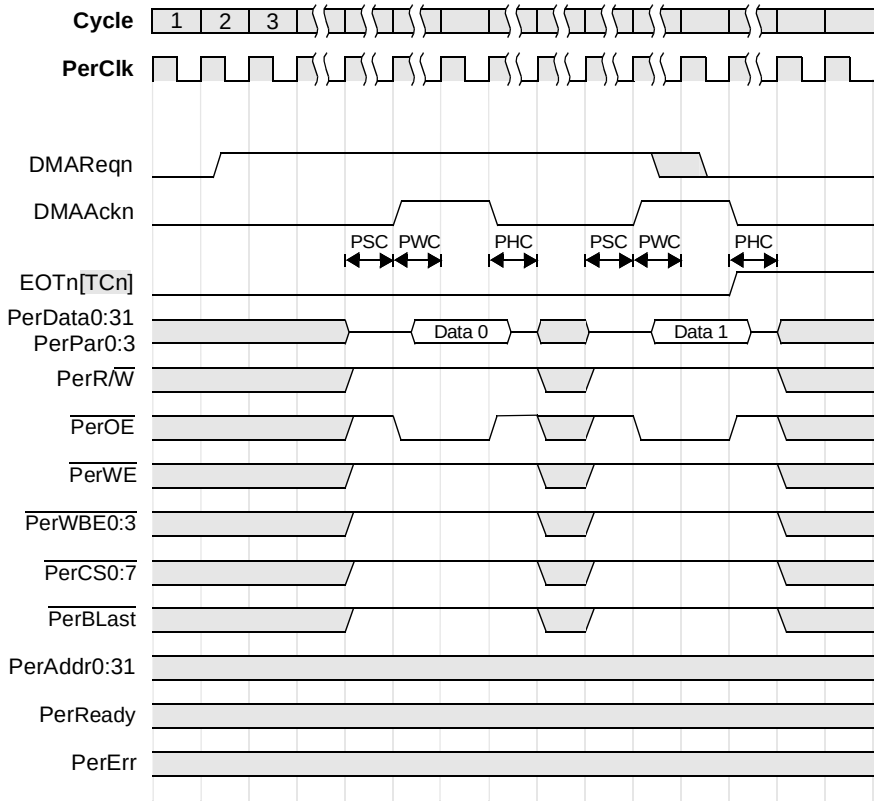


Figure 0-13. Peripheral-to-Memory DMA Transfer

Peripheral setup cycles (DMA0_CRn[PSC]) provide a delay between any previous operation on the peripheral bus and DMAAckn becoming active. Following the setup time, DMAAckn is driven active for (DMA0_CRn[PWC] + 1) PerClk cycles. During peripheral-to-memory transfers, read data from the peripheral is sampled on the last cycle where DMAAckn is active. After DMAAckn becomes inactive the peripheral bus is held idle for DMA0_CRn[PHC] PerClk cycles.

The second transfer in Figure 19-15 illustrates the required DMAReqn timing to prevent a subsequent DMA transfer. For all peripheral mode transfers, DMAReqn must be sampled inactive at the end of the last PerClk cycle where DMAAckn is active.

The EOTn[TCn] I/O can be configured either as an end of transfer input (DMA0_CRn[ETD]=0) or a terminal count output (DMA0_CRn[ETD]=1). When programmed as a terminal count output, EOTn[TCn] is asserted in the cycle after DMAAckn became inactive and the channel's count (DMA0_CTCn[TC]) reached zero. EOTn[TCn] remains active until the terminal count status bit is cleared in the DMA status register (DMA0_SR[CSn]).

If EOTn[TCn] is configured as an end of transfer input (DMA0_CRn[ETD]=0), EOTn[TCn] must be sampled active one external cycle before the last DMAAckn for external peripherals or during the last DMAAckn cycle for internal peripherals. If the channel is configured for scatter/gather transfers EOTn[TCn] should be immediately deasserted to prevent the subsequent transfer from ending prematurely. ~~Figure 19-16~~*Figure 19-16*

shows the required timing.

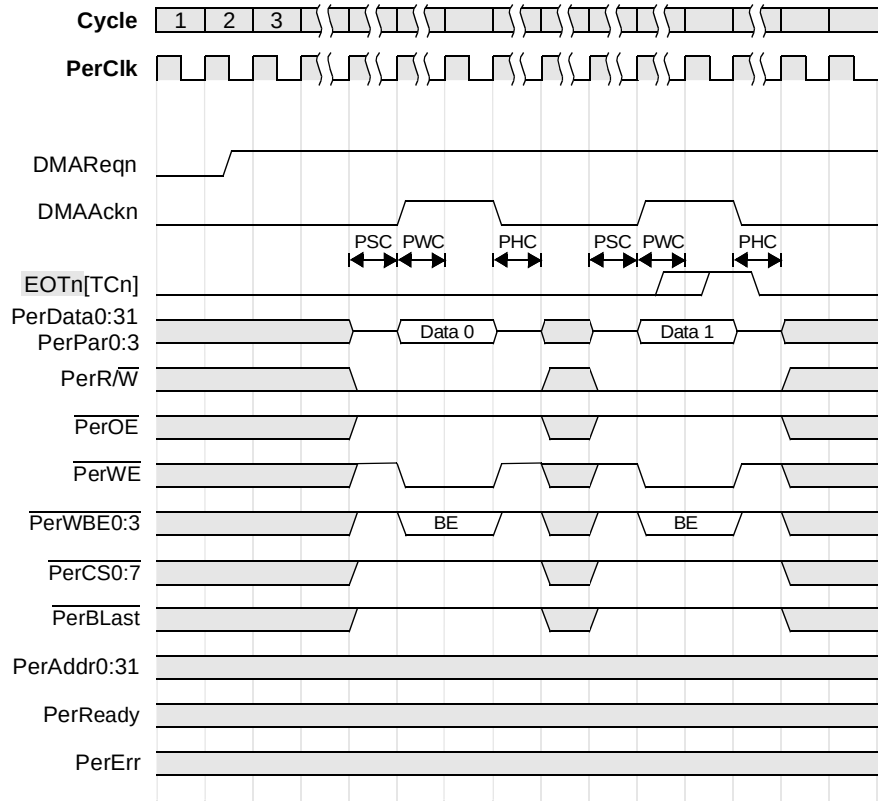


Figure 0-14. Memory to Peripheral DMA Transfer

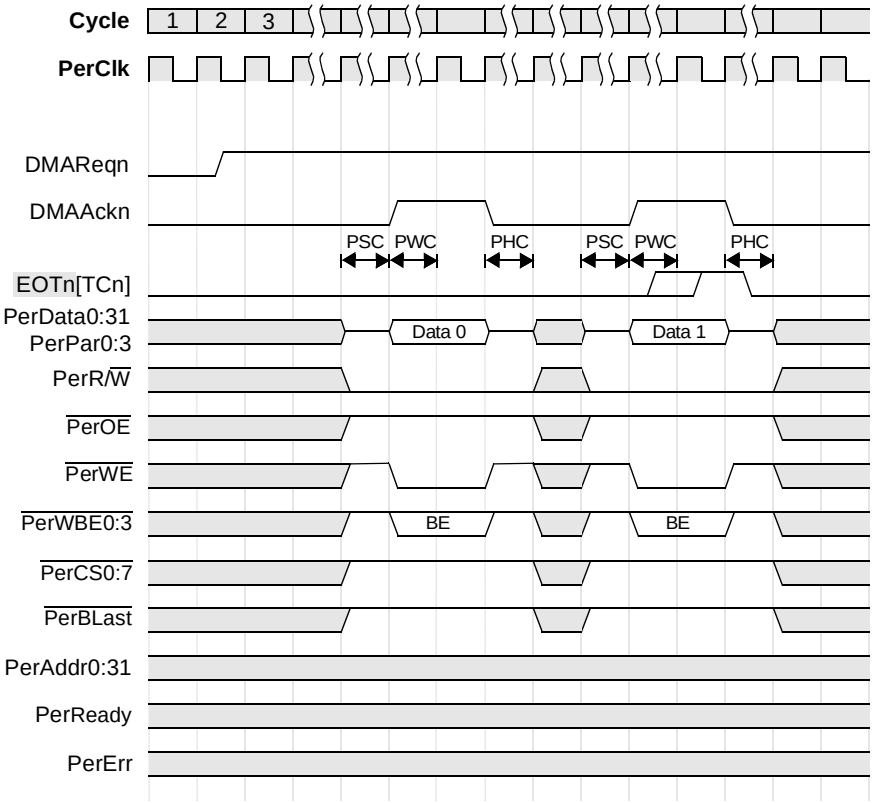


Figure 19-16. Memory to Peripheral DMA Transfer

For both peripheral-to-memory and memory-to-peripheral transfers the transfer width (DMA0_CRn[PW]) must be set to the data bus width of the peripheral. This is because the DMA controller does not pack or unpack data on the peripheral side of transaction.

19.10.1.1 Peripheral-to-Memory Transfer

To perform a peripheral-to-memory DMA transfer from an EBC-attached DMA peripheral:

1. Set destination address register (DMA0_DAHn and DMA0_DALn) to the desired memory location. The address must be aligned to the programmed transfer width (DMA0_CRn[PW]), otherwise an alignment error will occur.
2. In the count and control register:
    1. Program the desired count in the transfer count, TC, field.
    2. Optionally enable parity checking, PCE=1.
    3. Optionally enable bursting, BRSTEN=1, and set the burst size field, BRSTSIZ, as desired.
3. Clear the channel's status bits in the DMA status register (DMA0_SR).
4. In the channel control register (DMA0_CRn):
    1. Optionally enable the DMA buffer, BEN=1.
    2. Set the destination address increment, DAI=1.
    3. Set the transfer mode to peripheral, TM=0b00.

4. Set the peripheral location to external, PL=0.
5. Set the setup/wait/hold times for the transfer.
6. Set interrupts if required.
7. Set the transfer direction to peripheral-to-memory, TD=1.
8. Enable the channel CE=1.

Once the DMA channel is active, the peripheral initiates a transfer by activating the DMAReqn pin for the channel. The ~~PPC440GP~~ PPC440GP chip then activates the DMAAckn pin to read data from the peripheral. This continues until either a terminal count or end of transfer condition occurs.

19.10.1.2 Memory-to-Peripheral Transfer

To perform a memory-to-peripheral DMA transfer to an EBC-attached DMA peripheral:

1. Set source address register (DMA0_SAHn and DMA0_SALn) to the desired memory location. The address must be aligned to the programmed transfer width (DMA0_CRn[PW]), otherwise an alignment error will occur.
2. In the count and control register:
    1. Program the desired count in the transfer count, TC, field.
    2. Optionally enable parity checking, PCE=1.
    3. Optionally enable bursting, BRSTEN=1, and set the burst size field, BRSTSIZ, as desired.
3. Clear the channel's status bits in the DMA status register (DMA0_SR).
4. In the channel control register (DMA0_CRn):
    1. Optionally enable the DMA buffer, BEN=1, and set the desired prefetch count, PF.
    2. Set the source address increment, SAI=1.
    3. Set the transfer mode to peripheral, TM=0b00.
    4. Set the peripheral location to external, PL=0.
    5. Set the transfer direction to memory-to-peripheral TD=0.
    6. Enable the channel, CE=1.

Once the DMA channel is active, the peripheral initiates a transfer by activating the DMAReqn pin for the channel. The ~~PPC440GP~~ PPC440GP chip then reads the source memory and subsequently activates the DMAAckn pin to write data to the peripheral. This continues until either a terminal count or end of transfer condition occurs.

19.10.2 Memory-to-Memory Transfers

Memory-to-memory transfers can be initiated either by software or by an external device. If initiated via software, the transfer begins as soon as the channel is configured and enabled. When initiated by hardware (also known as a device-paced memory-to-memory transfer), software configures the channel for a memory-to-memory move and transfers begin when an external device places an active request on the channel request line, DMAReqn.

19.10.2.1 Hardware-Initiated (Device-Paced) Memory-to-Memory Transfers

To perform a device-paced memory-to-memory DMA transfer:

1. Set the transfer width (DMA0_CRn[PW]) to the width of the device-paced memory.
2. Set the source (DMA0_SAHn and DMA0_SALn) and destination (DMA0_DAHx and DMA0_DALn) address registers to the desired memory locations. These addresses must be aligned to the programmed transfer width (DMA0_CRn[PW]), otherwise an alignment error will occur.
3. In the count and control register:
    1. Program the desired count in the transfer count, TC, field.
    2. Optionally enable bursting, BRSTEN=1, and set the burst size field, BRSTSIZ, as desired.
4. Clear the channel's status bits in the DMA status register (DMA0_SR).
5. In the channel control register (DMA0_CRn):
    1. Optionally enable the DMA buffer, BEN=1, and set the desired prefetch count, PF.
    2. If the device-paced memory is at the source memory location set PL=1.
    3. Set the source address increment, SAI, and destination address increment, DAI, as desired.
    4. Set the transfer mode to device-paced memory-to-memory, TM=0b11.
    5. If the source memory controller is on the PLB set the source location bit, SL=~~1~~0. If the controller is on the OPB, clear SL.
    6. Enable the channel, CE=1.

Once the DMA channel is configured for this mode, the external device initiates a transfer by activating the DMAReqn input. The ~~PPC440GP~~ PPC440GP chip then reads the source memory, buffers the data in the DMA controller and then outputs the data to the destination memory address. Transfers continue as long as the controlling device maintains an active signal on DMAReqn and the channel count register (DMA0_CTCn) is non-zero. To pause a device paced memory-to-memory transfer, the controlling device must deassert DMAReqn one PerClk cycle before the last cycle in the device-paced memory access.

19.10.2.2 Software-Initiated Memory-to-Memory Transfers (Non-Deviced Paced)

To perform a software-initiated memory-to-memory DMA transfer:

1. Set the transfer width (DMA0_CRn[PW]) as desired.
2. Set the source (DMA0_SAHn and DMA0_SALn) and destination (DMA0_DAHn and DMA0_DALn) address registers to the desired memory locations. These addresses must be aligned to the programmed transfer width (DMA0_CRn[PW]), otherwise an alignment error will occur.
3. Program the transfer count field in the control and count register (DMA0_CTCn[TC]) for the number of transfers.
4. Clear the channel's status bits in the DMA status register (DMA0_SR).
5. In the channel control register (DMA0_CRn):
    1. Optionally enable the DMA buffer, BEN=1.
    2. Set the source address increment, SAI, and destination address increment, DAI, as desired.
    3. Set the transfer mode to software-initiated memory-to-memory, TM=0b10.
    4. If the source memory controller is on the PLB set the source location bit, SL=~~1~~0. If the controller is on the OPB, clear SL.
    5. Enable the channel, CE=1.

Once the channel is enable the DMA controller transfers data from source to destination until the channel count reaches zero. Note that memory-to-memory transfers initiated by software do not use DMAReqn or DMAAckn.

20. Memory Access Layer

The Memory Access Layer (MAL) is a hardware core that manages data transfers between packet-oriented communications cores, also known as COMMACs (communications media access controllers), and memory. On the PPC440GP the two Ethernet MACs are the COMMACs that are managed by MAL. Only one MAL is required to handle all the data channels of ~~both~~ the EMACs. COMMAC is a generic term that is used to refer to the EMACs throughout the chapter.

To communicate with software device drivers, MAL utilizes a buffer descriptor ring structure in memory. A software device driver uses the buffer descriptor structure to inform MAL about buffer locations and packet or buffer status. MAL uses the buffer descriptors to convey packet transfer status from the COMMAC core back to the software device driver. Each MAL channel requires its own buffer descriptor table ring structure in memory.

The PPC440GP MAL manages the transfer of packets between the two Ethernet Media Access Controllers (EMAC0 and EMAC1) and the memory attached to the PPC440GP (SDRAM or SRAM). The primary function of MAL is to move packets directly between memory and a COMMAC core with minimal involvement of the processor core.

20.1 MAL Features

- No restrictions on buffer alignment
- Aligned bus accesses to enable burst operation with external memories
- Configurable receive buffer size (configurable per channel)
- No minimum transmit buffer size
- Maximum buffer sizes of 4095 bytes (TX) and 4080 bytes (RX)
- Up to 256 descriptors in the buffer descriptor table per channel
- Configures COMMAC according to commands specified in the descriptor status/control field
- Updates the descriptor status/control field at the end of packet transfer according to the status received from COMMAC
- Buffer-based interrupt capabilities for each channel
- Concurrent operation of RX and TX channels
- Configuration using Device Control Registers (DCRs)
- Programmable PLB arbitration priority
- PLB/OPB error detection

Figure 20-1 illustrates a general system structure overview of an embedded PowerPC processor core integrated with a packet oriented communication core. For the PPC440GP, the sole COMMAC is EMAC.

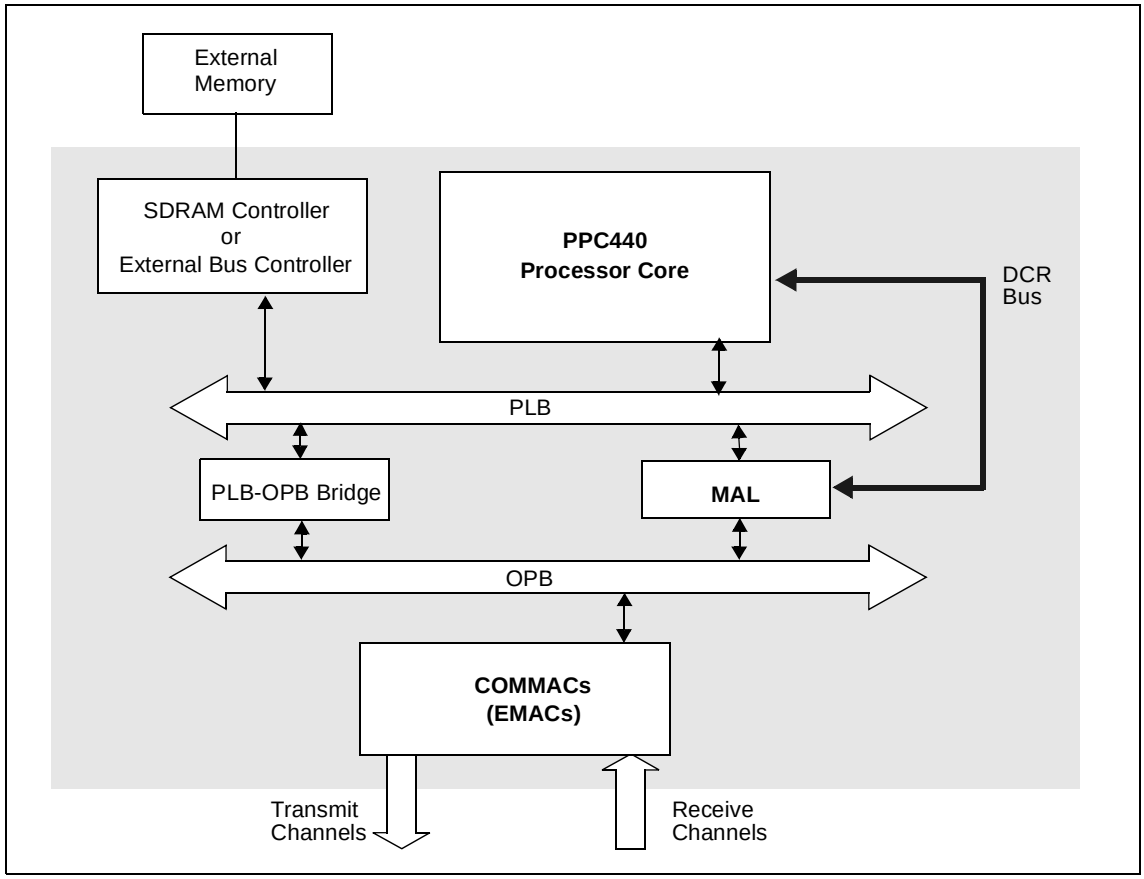


Figure 20-1. General PPC440GP Structure

Two COMMACs are configured and controlled by the processor core using the OPB without MAL intervention. Packet data to be transmitted and received are stored in buffers in external memory. The MAL processes buffer descriptors and provides all data access facilities to the COMMACs.

The MAL is not aware of COMMACs such as EMAC as an entity. It is only aware of the COMMAC's channels. In the PPC440GP, each EMAC contains two TX channels and one RX channel. Transmit and receive operations can be performed simultaneously by MAL (full duplex). When a channel wins arbitration, MAL transfers data between system memory and the COMMAC. MAL and the software driver maintain separate, dedicated buffer descriptor tables for each channel to maintain channel, packet, and buffer status. Packets can be constructed from one data buffer, or several data buffers (known as buffer chaining).

20.1.1 MAL Internal Structure

Figure 20-2 illustrates the MAL internal structure followed by a brief description of its contents.

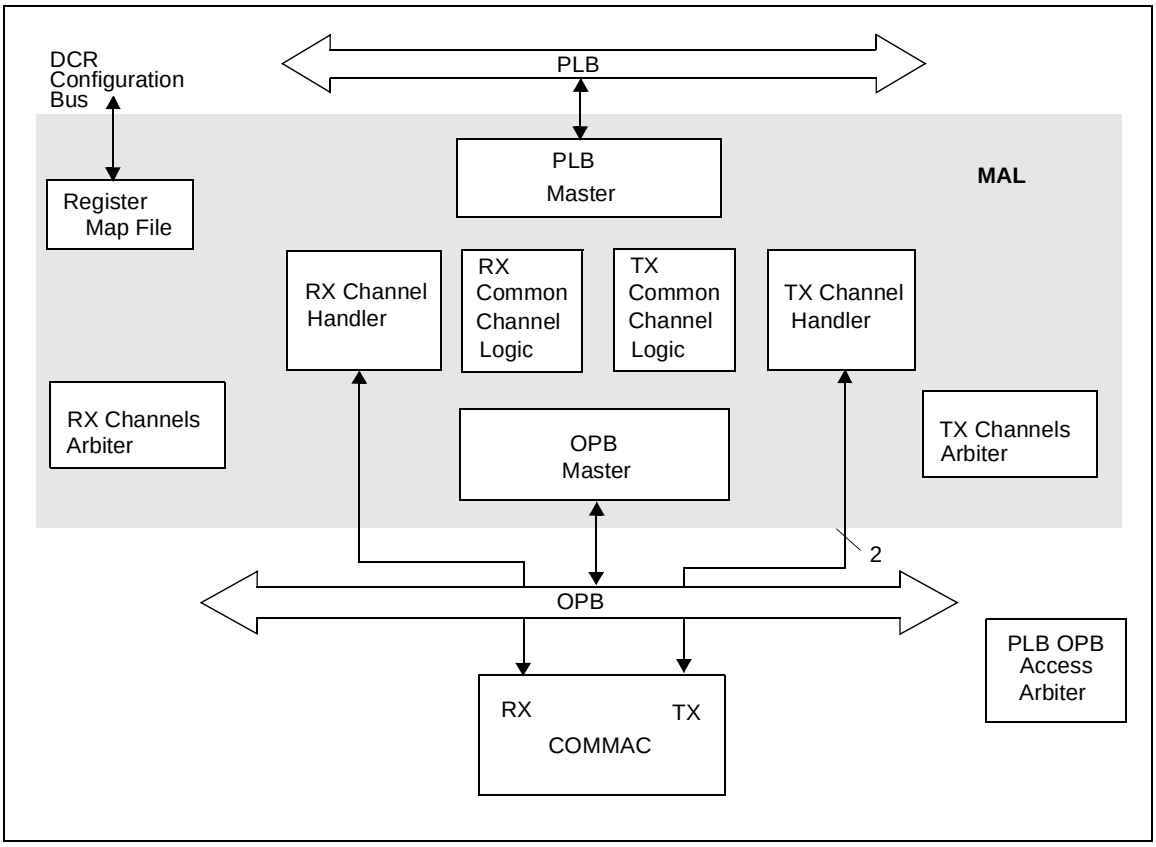


Figure 20-2. MAL Internal Structure

20.1.1.1 PLB Master

The PLB Master performs PLB transactions for MAL, and is used to transfer data between a COMMAC and memory, fetch buffer descriptors, and communicate status regarding data transfer.

20.1.1.2 OPB Master

The OPB Master performs OPB transactions for MAL, and is used to transfer data between a COMMAC and memory.

20.1.1.3 TX Channel Handler

The TX channel handler is a dedicated section for each TX channel. It keeps a record of the descriptor information and the current state of each channel.

20.1.1.4 RX Channel Handler

The RX channel handler is a dedicated section for each RX channel. It keeps a record of the descriptor information and the current state of each channel.

20.1.1.5 TX Channel Arbiter

The TX channel arbiter, connected to request lines from each TX channel, arbitrates between the TX channels and decides which channel gains access to the TX common channel logic.

20.1.1.6 RX Channel Arbiter

The RX channel arbiter, connected to request lines from each RX channel, arbitrates between the RX channels and decides which channel gains access to the RX common channel logic.

20.1.1.7 TX Common Channel Logic

The TX common channel logic is shared by all TX channels. It services a single TX channel at a time (selected by the TX arbiter). This logic activates the PLB and OPB masters for data and buffer descriptor transactions.

20.1.1.8 RX Common Channel Logic

The RX common channel logic is shared by all RX channels. It services a single RX channel at a time (selected by the RX arbiter). This logic activates the PLB and OPB masters for data and buffer descriptor transactions.

20.1.1.9 Register Map File

The register map file is used to configure MAL and read its status registers. Software accesses the MAL register file using the **mtdcr** and **mfdcr** instructions.

20.2 MAL0 Interfaces and Channel Assignments

MAL0 is comprised of 8 channels (4 transmit channels and 4 receive channels). Each channel is dedicated to one of two EMAC cores with the exception of 2 receive channels which are unused. See Table 20-1 for MAL0 channel assignments.

In the PPC440GP, MAL0 uses 1/1 clocking, that is, the interface between MAL0 and the EMAC cores is clocked at the same frequency as the interface between MAL0 and the PLB.

Table 20-1. MAL0 Channel Assignment

MAL0 Channel	EMAC Channel
RX Channel 0	EMAC0 RX Channel 0
RX Channel 1	EMAC1 RX Channel 1
RX Channel 2	Unused
RX Channel 3	Unused
TX Channel 0	EMAC0 TX Channel 0

Table 20-1. MAL0 Channel Assignment

MAL0 Channel	EMAC Channel
TX Channel 1	EMAC0 TX Channel 1
TX Channel 2	EMAC1 TX Channel 0
TX Channel 3	EMAC1 TX Channel 1

Sideband signals control slave selection and data transfer framing.

20.3 Transmit and Receive Operations

The device driver is responsible for configuring MAL before a COMMAC can begin requesting MAL to process packets of data. The device driver should ensure that channels are not enabled during reconfiguration; otherwise, fatal errors may occur.

For more information about the MAL software interface, see ~~"MAL Programming Notes" on page 20-18~~*MAL Programming Notes* on page 697.

20.3.1 Transmit Operations

Figure 20-3 describes the software and hardware operations involved in a typical transmit operation.

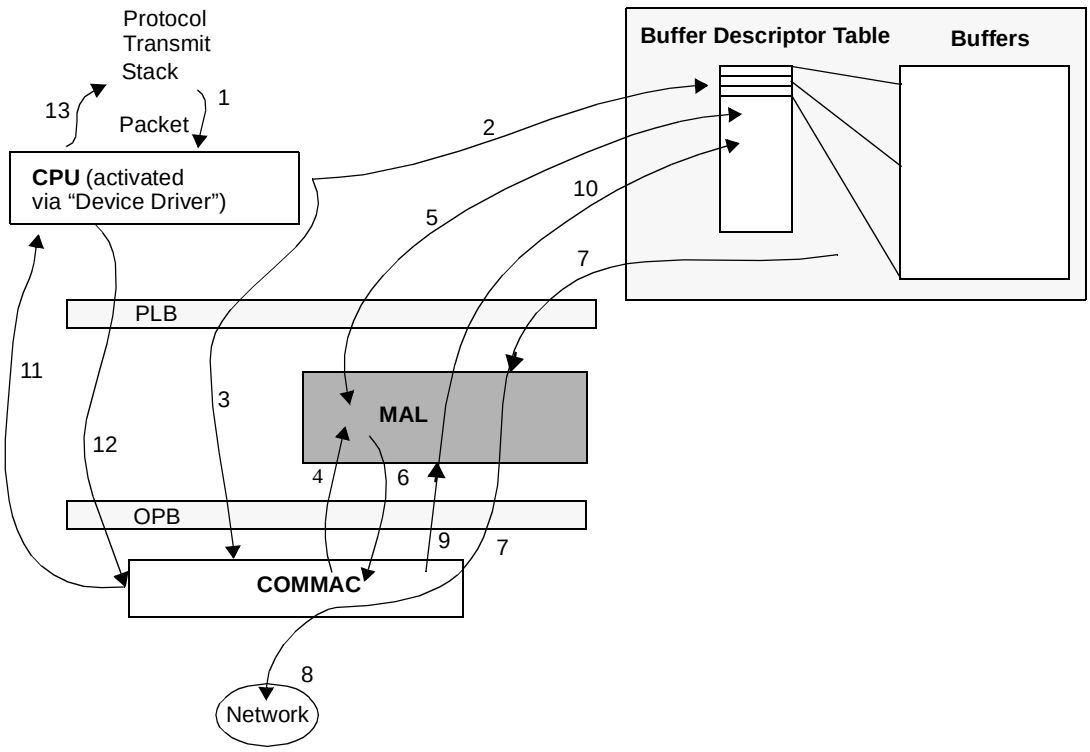


Figure 20-3. Transmit Operations

The numbered steps above are described as follows:

1. The protocol stack (high-level software layer) initiates a packet transmit.
2. Software device driver parses the protocol stack buffer into descriptor table entries and buffers.
3. Software device driver instructs the COMMAC to process a new transmit packet.
4. The COMMAC channel requests MAL to process a new packet.
5. MAL fetches descriptor information.
6. MAL writes control information into the COMMAC and initiates the data move.
7. Packet data is transferred from memory into the COMMAC (the COMMAC controls the pace of the data transfer).
8. The packet is transmitted on the media (steps 7 and 8 can overlap).
9. The COMMAC requests that MAL read the packet status.
10. The status read by MAL is written back into a buffer descriptor.
11. Software is interrupted (if interrupt conditions are met) by the COMMAC or by the MAL end-of-buffer interrupt.
12. Software clears the interrupt status bits in the COMMAC and in MAL.
13. Software informs the protocol stack that transmission is complete.

20.3.2 Receive Operations

~~Figure 20-4~~*Figure 20-4* describes the software and hardware operations when receiving a typical packet.

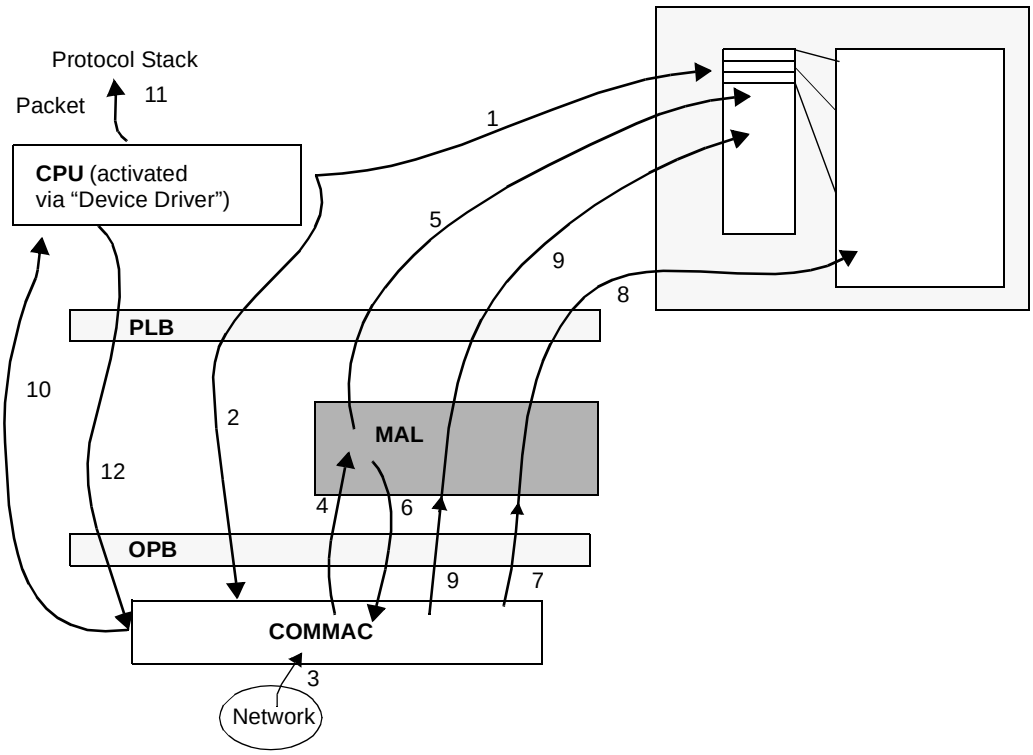


Figure 20-4. Receive Operations

The numbered steps above are described as follows:

1. Software device driver initializes the receive buffer descriptors.
2. Software device driver enables the COMMAC to process a new packet.
3. A packet is received from the network (steps 2 and 3 can change in order).
4. The COMMAC channel requests that MAL process a new packet.
5. MAL fetches receive buffer information from the descriptor table.
6. MAL writes the control word from the descriptor to the COMMAC and initiates the data transfer.
7. The COMMAC channel fills its FIFO storage.
8. MAL stores the packet in system memory buffers pointed to by the descriptors.
9. MAL reads status information from the COMMAC and writes it to the buffer descriptors.
10. Software device driver is interrupted (if interrupt conditions are met) by the COMMAC or by the MAL end-of-buffer interrupt.
11. The receive packet is passed to the protocol stack.
12. Software clears the receive buffer descriptor positions allowing them to be used again.

Note: The description in Figure 20-4 is the general scheme for MAL operation in the software environment. The device driver should follow recommendations from ~~"MAL Programming Notes" on page 20-18~~*MAL Programming Notes* on page 697.

20.4 Buffer Descriptor

The software interface for buffer descriptor (BD) processing consists of a set of registers within MAL and a set of circular queues in memory. Each transmit and receive COMMAC channel has a descriptor table that contains buffer location and status information allocated to the channel.

Note: Since MAL uses a flat addressing scheme on the PLB, the physical memory can be located anywhere on the PLB address space. However, each data structure MAL works with (data buffers and buffer description tables) will not cross a 4GB address border.

During its operation, MAL is able to modify the contents of memory directly without processor core knowledge. Data cache coherency is the responsibility of the software device driver. To simplify device driver software, the MAL buffer descriptor tables should be placed in non-cached memory when possible. If this is not possible, the software driver must maintain cache coherency of the buffer descriptor tables by performing data cache flushes or invalidates when appropriate. When descriptors are in cached memory, the driver software must be aware that multiple descriptors are present in a single cache line and that cache invalidate or flush operations will be performed on multiple descriptors at the same time. This is significant because a cache line flush done by the driver to force a descriptor from the data cache to physical memory could corrupt another descriptor that occupies the same data cache line and is simultaneously being updated in physical memory by MAL.

Data buffers, in contrast, should be placed in cachable memory if possible. The software driver can easily maintain cache coherency of data buffers if:

- All buffers are aligned on a data cache line boundary
- All buffers are a multiple of a data cache line in size

Note: The data cache line size and alignment in the PPC440GP is 32 bytes.

Before using a received packet, the software driver must invalidate the memory occupied by the buffer in the data cache for the length specified in the RX buffer descriptor data length field. Before transmitting a packet the software driver must flush the data buffer from the data cache before setting the Ready bit in the TX buffer descriptor.The software device driver fills the buffers pointed to by transmit buffer descriptors with packets to be transmitted, and/or provides empty buffers pointed to by receive buffer descriptors to be filled with received packets. Meanwhile, the hardware processes the descriptors, transfers the packet data to/from the COMMAC, and updates the status fields of the descriptors.

Each individual transmit or receive channel has its own buffer descriptor table. They are managed independently of each other. This section describes the individual transmit and receive interfaces.

Packet data associated with each transmit or receive channel is stored in buffers. Each buffer has an entry dedicated to it in one of the channel's buffer descriptor tables. MAL has a Channel Table Pointer Register for each of its channels. The COMMAC (EMAC in the PPC440GP) device driver sets the contents of these registers to point to the starting address of the buffer descriptor table for the associated channel.

The buffer descriptor table forms a circular queue with a programmable length. The last descriptor in the table is defined by setting the Wrap bit in the status/control field (see ~~"Status/Control Field Format" on page 20-14~~*Status/Control Field Format* on page 694). If there is no Wrap bit set in the table, then MAL automatically wraps after processing 256 descriptors (the maximum number of descriptors allow per channel).

The format of the buffer descriptor (BD) (see Figure 20-5) is the same for all COMMACs, and has the same structure for both transmit and receive. The most significant halfword in each buffer descriptor contains a status/control halfword. This field contains two parts: the first part (6 bits) is BD handling information used by the MAL for descriptor processing, the second field (10 bits) is content specific for each COMMAC. The second halfword determines the 4 MSbs of the data buffer pointer and the data length (in bytes) referenced in this buffer descriptor. The second word in the buffer descriptor contains the 32 LSbs of the data buffer pointer that points to the actual data buffer in memory. It is suggested that each data buffer start on a cache line boundary and be a multiple of a cache line in size if it resides in cachable memory. (The cache line size in the PPC440GP processor core is 32 bytes.)

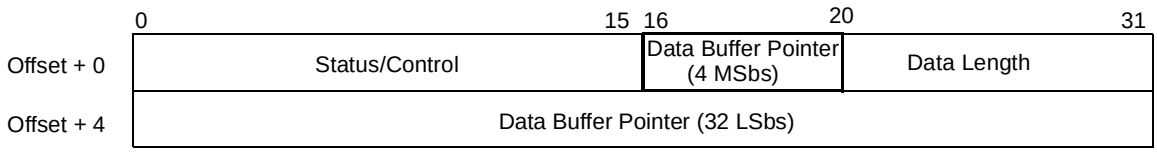
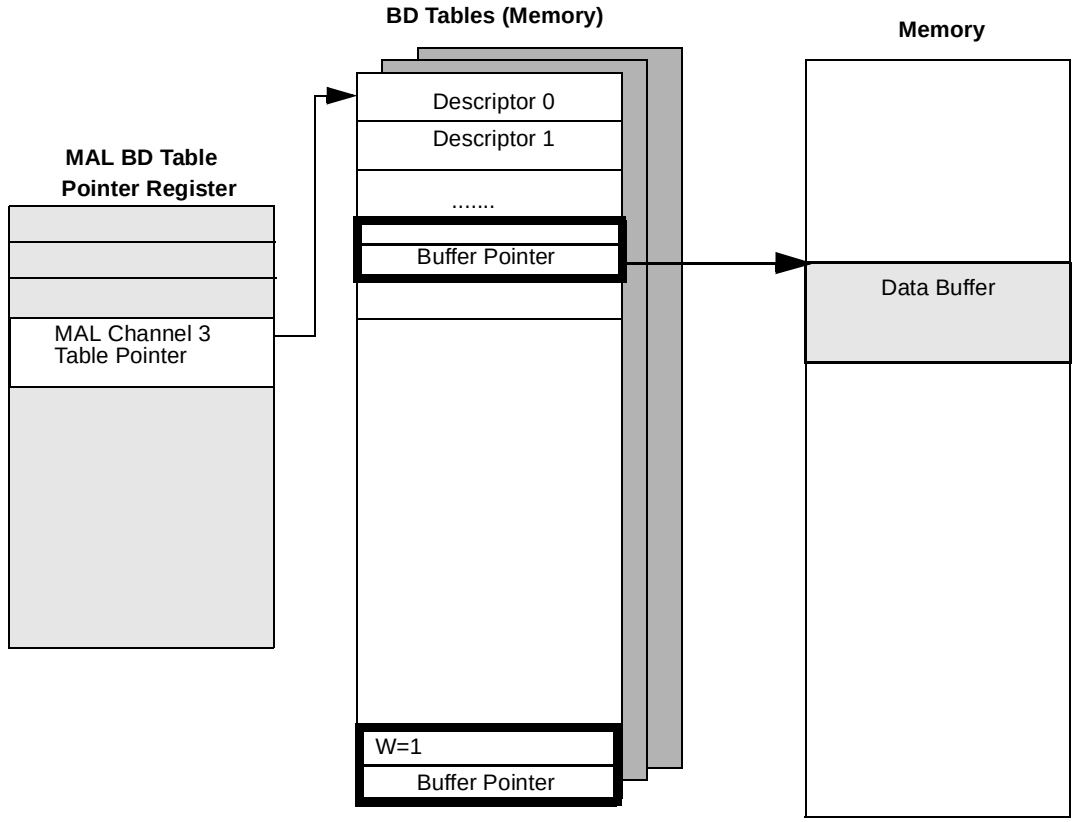


Figure 20-5. Buffer Descriptor Structure

A packet may reside in as many buffers as necessary (transmit or receive). Each buffer has a maximum length of (4KB – 16) bytes. In TX channels, the buffer descriptor length field is written by the device driver and defines the number of bytes in the data buffer that is identified by the data buffer pointer. In RX channels, the buffer descriptor length field is written by MAL and defines the number of bytes written by MAL to the buffer that is identified by the data buffer pointer (see ~~"Receive Software Interface" on page 20-12~~*Receive Software Interface* on page 692).

When processing a packet, MAL does not require that all buffers of the current packet are already valid. It requires the buffers to be ready in due time to be transmitted or received. Failure of the software to provide the descriptors in time may result in an error.

Figure 20-6*Figure 20-6* describes the structure of the packet in memory.



* W=1 means the wrap bit is set for this descriptor

Figure 20-6. Packet Memory Structure

20.5 Transmit Software Interface

Once a channel is enabled in MAL (this is done by setting the appropriate bit in the Channel Active Register), a channel may request service from MAL. When the first transmit request comes in from a COMMAC TX channel, MAL finds the starting address of the buffer descriptor table for the channel by looking in the corresponding Transmit Channel Table Pointer Register. If the first descriptor is marked as ready, MAL will start processing the associated buffer.

When MAL begins processing a packet, it writes the contents of the descriptor status/control field into the COMMAC. This information, (depending on communication core implementation), may be used by the communication core to configure each packet transfer.

Once all data from the current buffer has been transferred to the communication core on the channel, MAL moves on to the next buffer descriptor in the table.

If a given buffer descriptor indicates that it contains the last section of the current packet, MAL informs the channel that the last data transferred to the channel completed the transfer of a data packet. At this point, the COMMAC asks MAL to read the packet status. MAL then writes this information back into the status/control field of the last buffer descriptor of the packet.

The COMMAC channel may request that MAL process the next buffer descriptor and the same packet handling process will be initiated. The first descriptor in the next packet follows the descriptor marked "last" in the previous packet.

20.5.1 Wrapping the BD Table for Transmit

When MAL processes a buffer descriptor (while handling a packet for a COMMAC channel), it may encounter a Wrap indication within a buffer descriptor control field. This causes MAL to go back to the beginning of the buffer descriptor table for the next descriptor table entry. (This will also happen when MAL reaches the maximum number of descriptors.) The wrapping of the BD table, like all other BD table handling processes, is transparent to the COMMAC.

20.5.2 Continuous Mode for Transmit

After transmitting the data pointed to by a buffer descriptor, MAL clears the Ready bit in the buffer descriptor control/status field. In this way, MAL will not process the same buffer descriptor again until software has filled the buffer with valid data and set the Ready bit in the descriptor again. While the Continuous Mode (CM) bit is set in the status/control field, MAL will not clear the Ready bit. The Continuous Mode allows re-transmission of the current data buffer without software intervention. This mode is generally used by protocols in which frequent re-transmission is an integral part of the protocol itself. In such cases, re-transmission can be performed without software intervention.

20.5.3 Back Up a Packet for Transmit

MAL is capable of re-transmitting the last packet ("back up a packet") following a request from a COMMAC. If re-transmission is requested by the COMMAC, it must be assured that all the buffers of the re-transmitted packet are available and were not re-processed by the device driver. In regular operation, MAL resets the Ready bit of each buffer descriptor when finished processing the descriptor. When MAL is requested by the COMMAC to retransmit the last packet (the Back Up a Packet bit in the COMMAC TX channel Status Half-word is set), MAL doesn't reset the READY bit in the last processed buffer descriptor, activate the end of packet interrupt, or write the status back to the descriptor in the memory. MAL also doesn't consider this as an end of packet event.

On the next service request from the same channel, MAL will start transmitting the packet again, starting from the first descriptor.

Note: The last processed buffer descriptor can be either the last descriptor of the packet or, in case of early packet termination, the buffer descriptor that was being processed when the transmit channel initiated the early packet termination. MAL will retransmit the backed-up packet regardless of the Ready bit value.

During retransmission of a backed-up packet, MAL may use descriptors on which the Ready bit was already cleared. Therefore, the device driver should not reuse descriptors before the Ready bit of the last descriptor is cleared.

Note: In the case of descriptor not valid, which is the first one in TX channel, COMMAC is not allowed to return a status that contains a Back-Up a Packet request.

20.5.4 Descriptor Not Valid for Transmit

When MAL accesses a buffer descriptor, it checks whether or not the Ready bit is set. If the Ready bit is not set, two cases apply (special treatment of the READY bit is performed in the case of Back-up a packet):

For the case when the READY bit is not set:

- If the descriptor is the first descriptor of the packet MAL informs the channel that data is currently unavailable. Further handling of this scenario is COMMAC-specific. The channel might either instruct MAL to access the same buffer descriptor periodically (by keeping its service request to MAL active) until it becomes ready, or 'give up' on the descriptor, completing the end-of-packet protocol with MAL. The channel might also indicate the buffer descriptor status to the device driver via an interrupt. However, in this case the COMMAC should eventually complete the packet transfer protocol with MAL. Following a descriptor not valid indication, MAL's BD pointer continues pointing to the same location in the BD table. The next time a descriptor read is initiated by the COMMAC, MAL will search for the buffer in the same location.
- If the descriptor is not the first descriptor of the packet, it is considered a descriptor error. MAL deactivates the channel and from its point of view, the processing of the current packet has ended. Software may learn about this situation from one of two MAL interrupts (or from both). The first one is a non-maskable interrupt that indicates the number of the TX channel, in which the descriptor error had occurred (interrupt bit for each TX channel, see ~~"MAL Interrupt Enable Register (MAL0_IER)" on page 20-32~~*MAL Interrupt Enable Register (MAL0_IER)* on page 711). The second one is a maskable interrupt which indicates a descriptor error event, regardless the channel number (one interrupt bit for all the channels, see ~~"MAL Error Status Register (MAL0_ESR)" on page 20-29~~*MAL Error Status Register (MAL0_ESR)* on page 709). For more information about error handling, see ~~"Error Handling" on page 20-19~~*Error Handling* on page 699.

For the case of a back-up packet:

- When the current transmitted packet is a backed-up packet, all descriptors except the last, are valid even if the READY bit is not set. In this case, (not the last descriptor) MAL processes the packet descriptors regardless the READY bit value. If the READY bit of the last descriptor in the backed-up packet is not set, MAL treats it as a descriptor error. MAL handles the descriptor error as described above for the case when the packet isn't a backed-up packet.

20.5.5 Scroll Descriptors for Transmit

MAL may be configured by software, in the case of early packet termination, to scroll in the buffer descriptor table to the first descriptor of the next packet.

When a multiple-buffer packet is terminated early by the COMMAC, while MAL is processing a buffer which is not the last buffer in the packet, MAL can operate in one of the following ways:

The MAL Scroll Descriptor in the configuration register is set:

- In this case MAL will read the status word from the COMMAC channel. Then MAL will reset the READY bit in all the remaining buffer descriptors of the current packet. In addition, MAL will write the status to all the buffer descriptors. On the next service of this channel, MAL will fetch the first descriptor of the next packet.

The MAL Scroll Descriptor in the configuration register is clear:

- In this case MAL will read the Status word from the COMMAC channel. Then MAL will terminate the current channel service by resetting the READY bit of the last processed buffer descriptor (the one in which there was an early termination) and will write the status only to this descriptor. On the next service of this channel, MAL will fetch the next descriptor in the current packet. In this case, the software is responsible to monitor the MAL location in the buffer descriptor table.

In the case that the COMMAC requests a re-transmit of the early terminated packet (when the "backup" bit in the COMMAC status is set), MAL will re-transmit the packet regardless of the MAL Scroll Descriptor bit.

20.6 Receive Software Interface

MAL uses the RX channel buffer descriptors in a manner similar to that used for transmission. Once an RX COMMAC channel requests that a new packet be processed, MAL starts processing the channel's next buffer descriptor in the table. Once a channel is enabled in MAL, the channel may request MAL service. When it does, MAL accesses the first buffer descriptor (in that channel's buffer descriptor table) that is pointed to by the COMMAC channel table pointer register. If that descriptor is ready (empty for RX), MAL will start processing the buffer.

When it begins processing each packet, MAL writes the contents of the status/control field into the COMMAC. This information can be used by the COMMAC for a per-packet configuration.

Once data is received from the memory, MAL moves the data from the RX channel FIFO into the data buffer pointed to by the first buffer descriptor. The current buffer descriptor may be closed for two reasons: there is no more room left in the buffer, or the COMMAC channel indicated that the packet reception ended. If additional buffering space is needed for the current packet, MAL moves on to the next buffer descriptor. As each buffer descriptor is closed, MAL updates the length field with the actual amount of bytes written into the buffer. The maximal length of the buffers for each channel is defined by a configuration register (see ~~"RX Channel Buffer Size Register (MAL0_RCBSx" on page 20-37~~*RX Channel Buffer Size Register (MAL0_RCBSx* on page 717). The maximal receive buffer length is defined per channel.

Once the COMMAC channel indicates that the packet reception has ended, it is expected to request that MAL update the received packet status in the BD status/control field. MAL updates the packet status and notifies the COMMAC. At this point the packet is considered received and the COMMAC may request that MAL begin the process of receiving a new packet. The first buffer of the next packet is the buffer in the BD table that followed the last descriptor of the previous packet.

20.6.1 Wrapping the BD Table for Receive

When MAL processes the buffer descriptor, it may encounter a Wrap indication within a buffer descriptor control field. This causes MAL to go back to the head of the channel's buffer descriptor for the next buffer descriptor. This also happens when MAL reaches the maximal number of descriptors.

20.6.2 Continuous Mode for Receive

After using a buffer descriptor, MAL sets the buffer descriptor control to the Not-Empty state. In this way, MAL will not use the same buffer descriptor a second time until the software has processed the not-empty buffer descriptor and set it to Empty again. MAL will not clear the Empty bit while the Continuous Mode (CM) bit is

set in the status/control field. The Continuous Mode is generally used by protocols where frequent collisions are an integral part of the protocol itself (forcing the COMMAC to abort a reception process and restart). In such cases, re-reception can be performed without software intervention.

20.6.3 Descriptor Not Valid for Receive

When MAL accesses a buffer descriptor it may find that the Empty bit is not set. In the case of an RX channel descriptor, this situation is considered as a descriptor error. MAL deactivates the channel and from its point of view, the processing of the current packet has ended. Software may learn about this situation from one of two MAL interrupts (or from both):

- An RXDE interrupt with the MAL0_RXDEIR indicating which channel caused the interrupt
- An SERR interrupt (system error) with one interrupt bit for all channels in the MAL0_ESR

For more about error handling, see ~~"Error Handling" on page 20-19~~*Error Handling* on page 699.

20.6.4 Buffer Length for Receive

The maximum length of an RX buffer descriptor is predetermined for all RX descriptors in each channel. The data-length value is programmable through a set of MAL registers (see ~~"MAL Registers" on page 20-24~~*MAL Registers* on page 704). The actual data length field within the RX buffer descriptor is written by MAL. If the buffer is completely filled up, the value written will match the value programmed into the matching RX-Channel-Descriptor data-length register. If the buffer is only partially filled up (for example, when the RX packet ended before running out of buffer space), the actual amount of space filled is written into the length field.

20.7 Descriptor Buffer Status/Control Fields

The following sections detail the status/control field bits. The information fields within the status/control field can be divided as follows:

- Information from a software device driver directed to MAL and COMMAC
- Information from MAL and COMMAC directed to software
- Status/control field handling
- Status/control field format
- TX status/control field format
- RX status/control field format

20.7.1 Information from a Software Device Driver Directed To MAL and COMMAC

- MAL-related buffer descriptor processing information:
  - Buffer Ready/Not Ready (determines the buffer's validity).
  - Wrap to top of table or continue to next descriptor.
  - In a transmit buffer descriptor – Is the current buffer the last one in the packet?
  - Continuous or normal mode; that is, should MAL change the Ready/Not Ready value?

- COMMAC channel configuration information:
  - Should the channel generate an interrupt following the end of packet processing.
  - Protocol specific configuration.

20.7.2 Information from MAL and COMMAC Directed to Software

- MAL generated status information:
  - Buffer Ready/Not Ready (passes the buffer handling to software).
  - In receive buffer descriptor - Is the current buffer the first one in the packet?
  - In receive buffer descriptor - Is the current buffer the last one in the packet?
- COMMAC channel generated status information:
  - Protocol specific error and status information (transmit and receive).

20.7.3 Status/Control Field Handling

When MAL accesses a new buffer descriptor, the status/control word is written to the COMMAC channel. This allows the channel to configure itself for the current packet.

For all "intermediate" buffer descriptors (all descriptors that do not contain the packet's ending), the status/control field is written by MAL (rather than the COMMAC). In this case, the status/control field indicates that the current buffer is not the last one in the current packet.

As MAL finishes processing the last buffer descriptor in a given packet, it reads the channel's status (via an OPB transaction) and writes it into the buffer descriptor's status/control field.

In effect, since all of the various control and status fields do not overlap, the status/control halfword is read/written as a whole. Each agent (MAL, COMMAC channel, and software) reads the entire status/control halfword, relates to specific fields of interest, and updates another subset of fields within the same halfword. While an agent modifies its related fields, all other fields remain unchanged.

20.7.4 Status/Control Field Format

The status/control halfword is divided into COMMAC channel data and MAL related data. As explained above, the MAL related fields are either aimed at controlling MAL or written by MAL for use by the software. The MAL fields are of no interest to the COMMAC (except the Ready and Empty bits).

The same applies to the COMMAC channel fields. The COMMAC related fields are either aimed at controlling the COMMAC or written by COMMAC for use by the software. These fields are of no interest to MAL.

MAL will not manipulate the COMMAC related fields, and COMMAC is not allowed to manipulate the MAL related fields.

20.7.5 TX Status/Control Field Format

The bit numbering in ~~Figure 20-7~~Figure 20-7 relates to the Buffer Descriptor's fullword which contains both the status/control and the length fields.

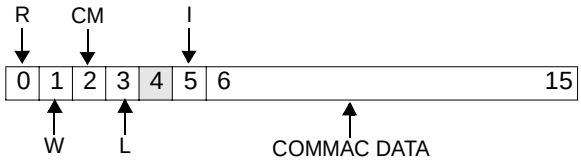


Figure 0-1. TX Status Control Field

0	R	Ready 0 Not ready 1 Ready	This bit is set by the device driver and is cleared by MAL. The device driver sets this bit after preparing the buffer for transmission. MAL clears this bit when finish processing the buffer descriptor. MAL doesn't clear the Ready bit in the case of backing-up a packet request and in case of continuous mode (see "Back Up a Packet for Transmit" on page 20-10 and"Continuous Mode for Transmit" on page 20-10).
1	W	Wrap 0 This is not the last data buffer descriptor in the buffer descriptor table. 1 This is the last data buffer descriptor in the buffer descriptor table.	After this buffer has been used, MAL will transmit data from the first descriptor buffer in the table. This bit is controlled by software only. It controls MAL activities, and does not affect the COMMAC channel.
2	CM	Continuous Mode 0 Normal Operation 1 Continuous Operation.	After this buffer descriptor is closed, the R-bit is not cleared by MAL. This ensures that the data buffer is ready for transmission when MAL next accesses this buffer descriptor. However, the R-bit is cleared if an error occurs during transmission. This bit is controlled by software only. It controls MAL activities, and does not affect the COMMAC channel.
3	L	Last 0 This is not the last buffer in the current packet. 1 This is the last buffer in the current packet.	This bit is controlled by software only. It controls MAL activities, and does not affect the COMMAC channel.
4		Reserved	It is assumed that this bit is set to zero by the software.

5	I	Interrupt 0 After finishing processing the current buffer, if this bit is 1, the end of buffer field in the End of Buffer Interrupt Status Register is set and the end of buffer interrupt is asserted. 1 There is no action taken by MAL once it reaches the end of the current buffer.	MAL asserts the end of buffer interrupt after it updates the buffer descriptor's status field. This bit is controlled by software only. It controls the MAL activities and does not affect the COMMAC.
6:15		COMMAC Related Data	These bits are COMMAC specific and may contain control fields generated by the software in order to control the COMMAC channel. They may also contain status fields, generated by the COMMAC channel, that will be processed by software.

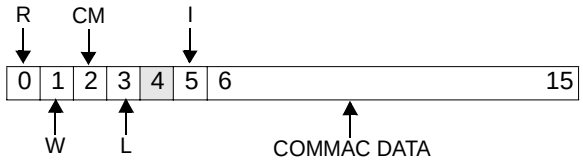


Figure 20-7. TX Status Control Field

0	R	Ready 0 Not ready 1 Ready	This bit is set by the device driver and is cleared by MAL. The device driver sets this bit after preparing the buffer for transmission. MAL clears this bit when finish processing the buffer descriptor. MAL doesn't clear the Ready bit in the case of backing-up a packet request and in case of continuous mode (see "Back Up a Packet for Transmit" on page -690 and"Continuous Mode for Transmit" on page -690).
1	W	Wrap 0 This is not the last data buffer descriptor in the buffer descriptor table. 1 This is the last data buffer descriptor in the buffer descriptor table.	After this buffer has been used, MAL will transmit data from the first descriptor buffer in the table. This bit is controlled by software only. It controls MAL activities, and does not affect the COMMAC channel.

2	CM	Continuous Mode 0 Normal Operation 1 Continuous Operation.	After this buffer descriptor is closed, the R-bit is not cleared by MAL. This ensures that the data buffer is ready for transmission when MAL next accesses this buffer descriptor. However, the R-bit is cleared if an error occurs during transmission. This bit is controlled by software only. It controls MAL activities, and does not affect the COMMAC channel.
3	L	Last 0 This is not the last buffer in the current packet. 1 This is the last buffer in the current packet.	This bit is controlled by software only. It controls MAL activities, and does not affect the COMMAC channel.
4		Reserved	It is assumed that this bit is set to zero by the software.
5	I	Interrupt 0 After finishing processing the current buffer, if this bit is 1, the end of buffer field in the End of Buffer Interrupt Status Register is set and the end of buffer interrupt is asserted. 1 There is no action taken by MAL once it reaches the end of the current buffer.	MAL asserts the end of buffer interrupt after it updates the buffer descriptor's status field. This bit is controlled by software only. It controls the MAL activities and does not affect the COMMAC.
6:15		COMMAC Related Data	These bits are COMMAC specific and may contain control fields generated by the software in order to control the COMMAC channel. They may also contain status fields, generated by the COMMAC channel, that will be processed by software.

20.7.6 RX Status/Control Field Format

The bit numbering in Figure 20-8 relates to the buffer descriptor's fullword which contains both the status/control and the length fields.

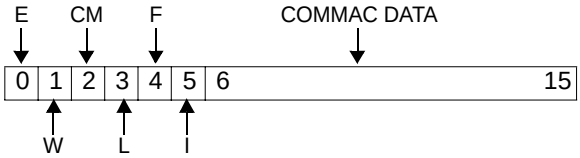


Figure 0-2. RX Status Control Field

0	E	Empty 0 The data buffer associated with this buffer descriptor has been filled with received data, or data reception has been aborted due to an error condition. 1 The data buffer associated with this buffer descriptor is empty, or reception is currently in progress.	Software is free to examine or write to any fields of this buffer descriptor. While this bit is set to Not Empty, MAL will not use this buffer descriptor again. This buffer descriptor and its associated receive buffer are owned by MAL. Once the E-bit is set, software should not write to any fields of this RX buffer descriptor. MAL clears this bit after the buffer has been filled with received data or after an error is encountered. Software sets this bit to Empty after preparing the buffer for reception. This bit controls MAL and software activities (see bit 2: continuous mode below).
1	W	Wrap 0 This is not the last data buffer descriptor in the buffer descriptor table. 1 This is the last data buffer descriptor in the buffer descriptor table.	After this buffer has been used, MAL will transmit data from the first descriptor buffer in the table. This bit is controlled by software only. It controls MAL activities, and does not affect the COMMAC channel.
2	CM	Continuous Mode 0 Normal Operation 1 Continuous Operation.	After this buffer descriptor is closed, the E-bit is not cleared by MAL. This ensures that the data buffer is ready to receive data when MAL next accesses this buffer descriptor. However, the E-bit is cleared if an error occurs during reception. This bit is controlled by software only. It controls MAL activities, and does not affect the COMMAC channel.

3	L	Last 0 This is not the last buffer in the current packet. 1 This is the last buffer in the current packet.	This bit is updated by MAL following the activity of the channel.
4	F	First 0 This is not the first buffer in the current packet. 1 This is the first buffer in the current packet.	This bit is updated by MAL following the activity of the channel.
5	I	Interrupt 0 After finishing processing the current buffer, if this bit is 1, the end of buffer field in the End of Buffer Interrupt Status Register is set and the end of buffer interrupt is asserted. 1 There is no action taken by MAL once it reaches the end of the current buffer.	MAL asserts the end of buffer interrupt after it updates the buffer descriptor's status field. This bit is controlled by software only. It controls the MAL activities and does not affect the COMMAC.
6:15		COMMAC Related Data	These bits are COMMAC specific and may contain control fields generated by the software in order to control the COMMAC channel. They may also contain status fields, generated by the COMMAC channel, that will be processed by software.

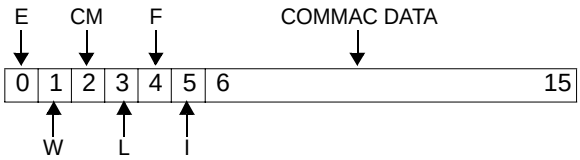


Figure 20-8. RX Status Control Field

0	E	Empty 0 The data buffer associated with this buffer descriptor has been filled with received data, or data reception has been aborted due to an error condition. 1 The data buffer associated with this buffer descriptor is empty, or reception is currently in progress.	Software is free to examine or write to any fields of this buffer descriptor. While this bit is set to Not Empty, MAL will not use this buffer descriptor again. This buffer descriptor and its associated receive buffer are owned by MAL. Once the E-bit is set, software should not write to any fields of this RX buffer descriptor. MAL clears this bit after the buffer has been filled with received data or after an error is encountered. Software sets this bit to Empty after preparing the buffer for reception. This bit controls MAL and software activities (see bit 2: continuous mode below).
1	W	Wrap 0 This is not the last data buffer descriptor in the buffer descriptor table. 1 This is the last data buffer descriptor in the buffer descriptor table.	After this buffer has been used, MAL will transmit data from the first descriptor buffer in the table. This bit is controlled by software only. It controls MAL activities, and does not affect the COMMAC channel.
2	CM	Continuous Mode 0 Normal Operation 1 Continuous Operation.	After this buffer descriptor is closed, the E-bit is not cleared by MAL. This ensures that the data buffer is ready to receive data when MAL next accesses this buffer descriptor. However, the E-bit is cleared if an error occurs during reception. This bit is controlled by software only. It controls MAL activities, and does not affect the COMMAC channel.

3	L	Last 0 This is not the last buffer in the current packet. 1 This is the last buffer in the current packet.	This bit is updated by MAL following the activity of the channel.
4	F	First 0 This is not the first buffer in the current packet. 1 This is the first buffer in the current packet.	This bit is updated by MAL following the activity of the channel.
5	I	Interrupt 0 After finishing processing the current buffer, if this bit is 1, the end of buffer field in the End of Buffer Interrupt Status Register is set and the end of buffer interrupt is asserted. 1 There is no action taken by MAL once it reaches the end of the current buffer.	MAL asserts the end of buffer interrupt after it updates the buffer descriptor's status field. This bit is controlled by software only. It controls the MAL activities and does not affect the COMMAC.
6:15		COMMAC Related Data	These bits are COMMAC specific and may contain control fields generated by the software in order to control the COMMAC channel. They may also contain status fields, generated by the COMMAC channel, that will be processed by software.

20.8 MAL Programming Notes

The following sections contain information about programming the MAL.

20.8.1 MAL Initialization

MAL initialization includes two parts: configuration and channel activation.

Configuration involves two steps:

- MAL configuration - This step is done only after a power on reset or after a MAL soft reset. The following registers are involved:
  - Configuration Register (MAL0_CFG). This register defines MAL operation on the PLB and OPB.
  - Interrupt Enable Register (MAL0_IER). This register is used to enable interrupts for various MAL error conditions.
- Channel specific configuration - This information may be changed only when the associated channel is not active. (The bit for the channel in the TX or RX Channel Active Set Register, is cleared.) The following registers are involved:
  - MAL0_RCBSx – RX Buffer Size (one register for each RX channel). This register defines the length of the RX buffers in memory.
  - MAL0_TXCTPxR or MAL0_RXCTPxR – Channel Table Pointer Register (one register for each channel). This register is programmed with the memory address of the first buffer descriptor table entry for the channel.

Setting the channel specific configuration can be done as part of MAL initialization or as part of the COMMAC initialization process. In order to activate a channel, the following actions should be taken:

- The channel has to be configured in MAL
- The related bit in Channel Active Set Register (MAL0_TXCASR or MAL0_RXCASR) has to be set
- The channel operation must be enabled (COMMAC configuration)

20.8.2 Interrupts

The MAL in the PPC440GP has five interrupt lines which are connected to the Universal Interrupt Controller (~~See "Universal Interrupt Controller" on page 9-1.~~*Universal Interrupt Controller* on page 299) Two interrupt lines, one for TX and one for RX, are used for interrupt events during packet transfer. An additional two interrupt lines, one for TX and one for RX, are used to report descriptor errors on a per-channel basis. The fifth interrupt is used to report MAL errors.

- TXEOB interrupt line is used to report end of buffer or end of packet for a specific TX channel. A bit for the related channel is set in the MAL0_TXEOBISR. See ~~"End of Buffer Interrupt Status Registers" on page 20-28~~*End of Buffer Interrupt Status Registers* on page 708.
- RXEOB interrupt line is used to report end of buffer or end of packet for a specific RX channel. A bit for the related channel is set in the MAL0_RXEOBISR. See ~~"End of Buffer Interrupt Status Registers" on page 20-28~~*End of Buffer Interrupt Status Registers* on page 708.
- TXDE interrupt line is used to indicate a descriptor error event in a specific TX channel descriptor table. A bit for the related channel is asserted in the MAL0_TXDEIR. See ~~"Descriptor Error Interrupt Registers (MAL0_TXDEIR, MAL0_RXDEIR)" on page 20-32~~*Descriptor Error Interrupt Registers (MAL0_TXDEIR, MAL0_RXDEIR)* on page 712.
- RXDE interrupt line is used to indicate a descriptor error event in a specific RX channel descriptor table. A bit for the related channel is asserted in the MAL0_RXDEIR. See ~~"Descriptor Error Interrupt Registers (MAL0_TXDEIR, MAL0_RXDEIR)" on page 20-32~~*Descriptor Error Interrupt Registers (MAL0_TXDEIR, MAL0_RXDEIR)* on page 712.
- SERR interrupt is used to report a system error indicated by MAL. For more information on handling the SERR interrupts, see ~~"Error Handling" on page 20-19~~*Error Handling* on page 699 and ~~"Error Registers" on page 20-29~~*Error Registers* on page 709.

20.8.3 Error Handling

MAL handles errors on a per-channel basis. Within a COMMAC channel, errors may arise from the COMMAC (detected as an OPB error), or from the memory access operations involved in MAL activity (detected as a PLB/descriptor error).

When a bus error occurs, MAL is notified by an OPB or PLB error signal. OPB errors are related to a specific channel and therefore channel operation is stopped. In the case of a PLB error, MAL cannot identify which channel is involved, therefore channel operation is not stopped. When a descriptor error occurs, MAL can again identify the channel involved, so channel operation is stopped. MAL stops channel operation by clearing the associated bit in the MAL0_TXCASR or MAL0_RSCASR register.

MAL keeps a record of the channels that experience errors and are made inactive. It also keeps a record of the characteristics of the first (or last) error detected (see ~~"End of Buffer Interrupt Status Registers" on page 20-28~~*End of Buffer Interrupt Status Registers* on page 708).

Note: The device driver is responsible for configuring MAL before a COMMAC can begin requesting MAL to process packets of data. The device driver should ensure that channels are not enabled during reconfiguration; otherwise, fatal errors may occur.

20.8.3.1 Error Detection

The MAL communication, both with COMMACs and with memory, is carried out via the OPB or PLB. As long as this bus communication is error-free and no descriptor errors are detected, MAL maintains normal activity with the channels set by the processor as active in the Channel Active Registers.

When an error is detected while performing a transfer for a channel, MAL asserts a maskable interrupt signal. If the identity of the channel is known (as is the case for OPB errors or descriptor errors) then MAL immediately halts the dialogue with the channel. No further transactions are made, and that channel is registered by MAL as a non-active channel. MAL resets the channel by resetting its active bit in the Channel Active Register. Software must access the Channel Active Register in order to reactivate the channel.

If the identity of the channel that caused the error is not known (as is the case for PLB errors) then MAL continues to work normally. Error resolution and channel de-activation are the responsibility of the software.

20.8.3.2 Indicated Errors

Error description is stored in the Error Status Register (MAL0_ESR), (see ~~"MAL Error Status Register (MAL0_ESR)" on page 20-29~~*MAL Error Status Register (MAL0_ESR)* on page 709).

- Descriptor Error

  A descriptor error is a data error recognized during access to the descriptor table. The error can occur during TX or RX transmission.

  For RX channels, a descriptor error occurs when MAL accesses a descriptor in which the Empty bit is cleared.

  For TX channels, a descriptor error occurs when MAL accesses a descriptor in which the Ready bit is cleared. The following cases are exceptions.

  - On access to the first buffer descriptor in a TX packet.
  - On access to a buffer descriptor that is not the last descriptor in a backed-up packet.

  As a result of this error, the following actions are taken by MAL:

  - The Active bit of the related channel is reset and the channel activity is halted until software reactivates channel activity.
  - The associated bit in the TX Descriptor Interrupt Error Register (MAL0_TXDEIR) or RX Descriptor Error Register (MAL0_RXDEIR) is set, causing a non-maskable TXDE interrupt or RXDE interrupt respectively.
  - When the channel is reactivated, MAL points to the descriptor at the head of the BD table.
- OPB Non-Fullword Error

  This error indicates that a non-fullword acknowledge was asserted by a slave.

  Following this error, the active bit of the associated channel is reset and channel activity is halted until it is reactivated by software. When the channel is reactivated, MAL points to the descriptor at the head of the BD table.
- OPB Time-Out Error

  This error indicates that an OPB time-out error was reported by the OPB arbiter.

Following this error, the active bit of the associated channel is reset and channel activity is halted until reactivated by software. When the channel is reactivated, MAL points to the descriptor at the head of the BD table.

- OPB Error

  This error indicates that an OPB error was detected.

  Following this error, the active bit of the associated channel is reset and channel activity is halted until reactivated by software. When the channel is reactivated, MAL points to the descriptor at the head of the BD table.

- PLB Error

  This error indicates that a PLB error was detected (from the PLB slave).

  In this case, MAL cannot determine which channel caused the error. Therefore, operation is not halted for any of the channels.

20.8.3.3 Error Handling Registers

MAL error handling logic includes two registers.

- Error Status Register (ESR)

  This register holds information about the error that occurred and the interrupt status. The register includes the following fields:

  Error status – This field holds the error information. The information includes the number of the channel on which the error occurred (if known) and the type of the error. The error can be either the last detected error or a locked error if "Locked error mode" is active. See ~~"Operational Error Modes" on page 20-21~~*Operational Error Modes* on page 701 for description of the Locked error mode.

  The error status field includes an "Error Valid" bit which indicates whether there is valid error information in the error status field or not. The error status field is not valid when the "Error Valid" bit is cleared (by writing 1 to this bit).

  Interrupt status – Every error detected by MAL sets a related bit in the interrupt status field. Software can clear an interrupt status bit by writing 1 to the bit to be cleared. The bits in this field are accumulative which allows more than one interrupt to be indicated in the register.)

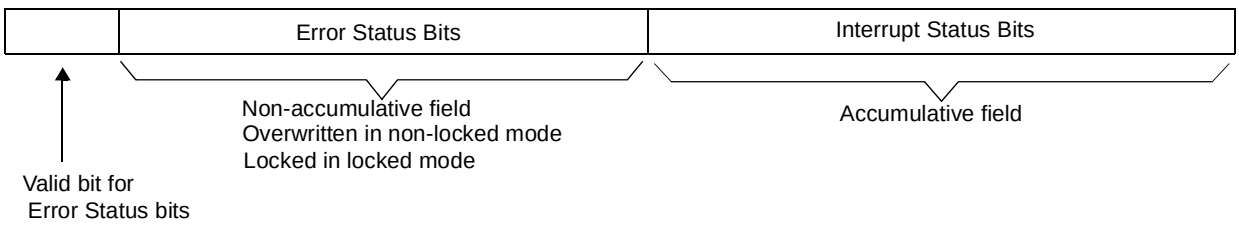


Figure 20-9. Error STatus Register Field

20.8.3.4 Operational Error Modes

MAL can operate in two different error handling modes:

- Locked Error Mode: Information about the error is written to the Error Status Register, and the Valid bit in that register is set. Information in the Error Status field of the register stays locked until software unlocks it by resetting the error Valid bit. The Interrupt Status bits of the Error Status Register are not locked in this mode, so software can find out if more errors occur. However, the Error Status field applies only to the first error that is locked.
- Non-Locked Error Mode: Information about the error is written in the Error Status Register, and the error Valid bit is set. Each new error will be overwritten, so the information in the Error Status Field is valid only for the last error that occurred.

In both modes, each error written in the error description field will set the error Valid bit, and it is the responsibility of software to reset this bit.

The error handling mode is programmed in the MAL Configuration Register (see ~~"MAL Configuration Register (MAL0_CFG)" on page 20-25~~*MAL Configuration Register (MAL0_CFG)* on page 705).

20.8.3.5 Resolution of an Error Situation

When MAL encounters an error, it reacts as follows:

- Writes information about the error in the Error Status Register (ESR). This information includes the channel ID of the channel which caused the error (if known), the bus on which the error occurred, and the kind of error that occurred.
- Resets the channel that caused the error (if known) in the Channel Active Register.
- Updates the Interrupt Status bits in the MAL0_ESR. Then, depending on the mask defined in MAL0_IER (Interrupt Enable Register), it sends an interrupt to the Universal Interrupt Controller.

After receiving an interrupt from MAL, software can analyze the error information read from the Error Status Register. Software can restart channel activity by setting the associated bit in the Channel Active Register.

When a channel is stopped and restarted, MAL starts processing descriptors from the first descriptor in the channel descriptor table. Therefore, software may also update the value of the other channel related registers (see ~~"Channel Table Pointer Registers" on page 20-34~~*Channel Table Pointer Registers* on page 714) in order to continue from the same buffer in memory.

In the case of PLB errors, MAL does not know which channel caused the error. It is the responsibility of the software to analyze the MAL error registers and the PLB slave error registers to determine which channel caused the error. Software should reset the channel within MAL, resolve the problem, and then reactivate the channel.

See ~~Figure 20.8.3.5 on page 20-21~~*Figure 20.8.3.5* on page 701 for a flow chart illustrating the steps MAL performs when resolving an error situation.

20.8.3.6 Interrupts To Software

Figure 20-10 describes MAL actions once an error is detected. Note that the actual decisions MAL makes may be in a different order than represented by this figure. In any case, the device driver should consider that all of the MAL actions are performed at the same time.

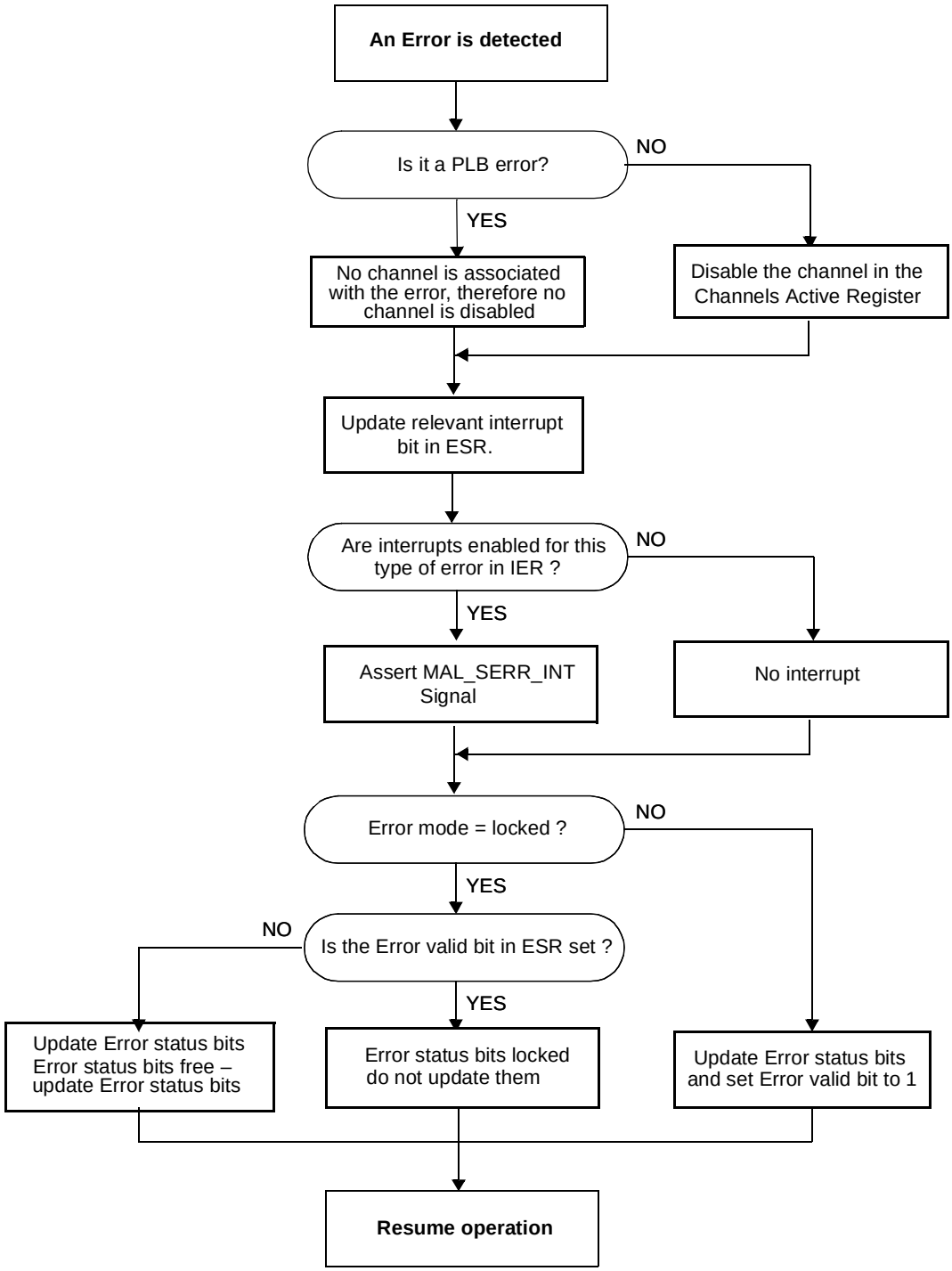


Figure 20-10. MAL Error Processing

20.9 MAL Registers

The MAL registers are Device Control Registers (DCRs).

- Unless otherwise specified, all register fields are initialized at chip reset to 0.
- Reserved fields are read as undefined and must be written as 0s.

Table 20-2. MAL Register Summary

Register	DCR Number	Access	Description	Page
MAL0_CFG	0x180	R/W	Configuration Register	705
MAL0_ESR	0x181	R/Clear	Error Status Register	709
MAL0_IER	0x182	R/W	Interrupt Enable Register	711
MAL0_TXCASR	0x184	R/W	TX Channel Active Set Register	706
MAL0_TXCARR	0x185	R/W	TX Channel Active Reset Register	706
MAL0_TXEOBISR	0x186	R/Clear	TX End of Buffer Interrupt Status Register	708
MAL0_TXDEIR	0x187	R/Clear	TX Descriptor Error Interrupt Register	712
MAL0_TXTATTRR	0x188	R/W	TX PLB Attribute Register	713
MAL0_TXBADDR	0x189	R/W	TX Descriptor Base Address Register	714
MAL0_RXCASR	0x190	R/W	RX Channel Active Set Register	706
MAL0_RXCARR	0x191	R/W	RX Channel Active Reset Register	706
MAL0_RXEOBISR	0x192	R/Clear	RX End of Buffer Interrupt Status Register	708
MAL0_RXDEIR	0x193	R/Clear	RX Descriptor Error Interrupt Register	712
MAL0_RXTATTRR	0x194	R/W	RX PLB Attribute Register	713
MAL0_RXBADDR	0x195	R/W	RX Descriptor Base Address Register	714
MAL0_TXCTP0R	0x1A0	R/W	Channel TX 0 Channel Table Pointer Register	714
MAL0_TXCTP1R	0x1A1	R/W	Channel TX 1 Channel Table Pointer Register	714

Table 20-2. MAL Register Summary

Register	DCR Number	Access	Description	Page
MAL0_TXCTP2R	0x1A2	R/W	Channel TX 2 Channel Table Pointer Register	20-3471 4
MAL0_TXCTP3R	0x1A3	R/W	Channel TX 3 Channel Table Pointer Register	20-3471 4
MAL0_RXCTP0R	0x1C0	R/W	Channel RX 0 Channel Table Pointer Register	20-3471 4
MAL0_RXCTP1R	0x1C1	R/W	Channel RX 1 Channel Table Pointer Register	20-3471 4
MAL0_RCBS0	0x1E0	R/W	Channel RX Channel 0 Buffer Size Registers	20-3671 6
MAL0_RCBS1	0x1E1	R/W	Channel RX Channel 1 Buffer Size Registers	20-3671 6

20.9.1 MAL Configuration Register (MAL0_CFG)

This register defines the operational mode of MAL. Unless a configuration change is required during system operation, the configuration register needs to be set only during system initialization.

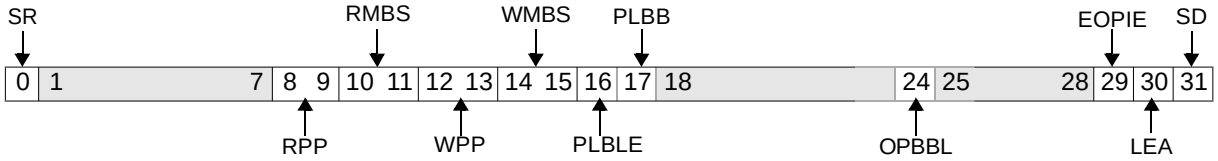


Figure 0-3. MAL Configuration Register (MAL0_CFG)

0	SR	MAL Software Reset 0 MAL reset is complete 1 Reset the MAL	Generates a general reset to MAL through a software command. After setting this bit, MAL hardware (registers, interface and internal state machines) returns to the power-on reset value. The software writes 1 to this bit in order to drive MAL to the reset state. The bit is cleared by the hardware when the reset is completed (one system clock).
1:7		Reserved	
8:9	RPP	Read PLB Priority 00 Lowest 01 10 11 Highest	Determines the priority of MAL requests on the PLB for read transactions.
10:11	RMBS	Read Max Burst Size 00 Max burst size of 4 01 Max burst size of 8 10 Max burst size of 16 11 Max burst size of 32 (recommended)	Maximum PLB burst size for read transactions. Determines the maximum data transfers (RdDAcks) per read burst transaction.
12:13	WPP	Write PLB Priority 00 Lowest 01 10 11 Highest	Determines the priority of MAL requests on the PLB for write transactions.
14:15	WMBS	Write Max Burst Size 00 Max burst size of 4 01 Max burst size of 8 10 Max burst size of 16 11 Max burst size of 32 (recommended)	Maximum PLB burst size for write transactions. Determines the maximum data transfers (WrDAcks) per write burst transaction.
16	PLBLE	PLB Lock Error 0 LOCKERROR signal not applied to the PLB slave 1 LOCKERROR signal applied to the PLB slave	When this bit is set, MAL applies the LOCKERROR signal to the PLB slave when it is the initiator during PLB transactions.

17	PLBB	PLB Burst 0 Burst transactions not allowed 1 Burst transactions allowed	When this bit is reset, MAL is not allowed to perform burst transactions.
18:23		Reserved	
24	OPBBL	OPB Bus Lock 0 OPB not locked 1 OPB locked	When this bit is set, MAL locks the OPB during data transfers to and from the COMMACs.
25:28		Reserved	
29	EOPIE	End of Packet Interrupt Enable 0 Generate interrupt on every end-of-packet only if the buffers I bit is set 1 Generate interrupt is on every end-of-packet	When this bit is set, an interrupt is generated on every end of packet (both transmit and receive). When clear, end of packet/buffer interrupt is generated only if the buffers I bit is set (1). Note: An interrupt is generated for every descriptor on which the I bit is set, regardless of the state of the EOPIE bit.
30	LEA	Locked Error Active 0 Handle errors in a non-locked mode 1 Handle errors in locked mode	Determines MAL's error handling mode. When this bit is set, MAL will handle errors in the locked mode, otherwise it will handle errors in a non-locked mode.
31	SD	MAL Scroll Descriptor 0 Do not scroll to the first descriptor of the next packet 1 Scroll to the first descriptor of the next packet	Determines whether or not MAL should scroll to the first descriptor of the next packet, following an early packet termination initiated by the related COMMAC. When set, Scrolling mode is active.

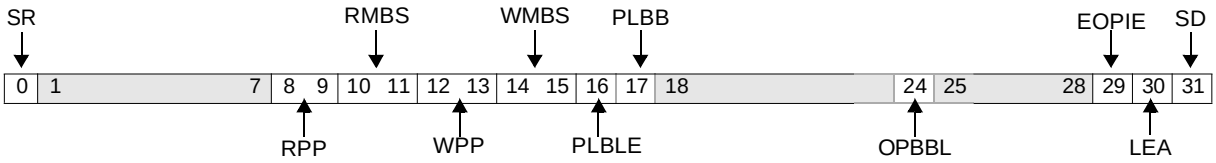


Figure 20-11. MAL Configuration Register (MAL0_CFG)

0	SR	MAL Software Reset 0 MAL reset is complete 1 Reset the MAL	Generates a general reset to MAL through a software command. After setting this bit, MAL hardware (registers, interface and internal state machines) returns to the power-on reset value. The software writes 1 to this bit in order to drive MAL to the reset state. The bit is cleared by the hardware when the reset is completed (one system clock).
1:7		Reserved	

8:9	RPP	Read PLB Priority 00 Lowest 01 10 11 Highest	Determines the priority of MAL requests on the PLB for read transactions.
10:11	RMBS	Read Max Burst Size 00 Max burst size of 4 01 Max burst size of 8 10 Max burst size of 16 11 Max burst size of 32 (recommended)	Maximum PLB burst size for read transactions. Determines the maximum data transfers (RdDAcks) per read burst transaction.
12:13	WPP	Write PLB Priority 00 Lowest 01 10 11 Highest	Determines the priority of MAL requests on the PLB for write transactions.
14:15	WMBS	Write Max Burst Size 00 Max burst size of 4 01 Max burst size of 8 10 Max burst size of 16 11 Max burst size of 32 (recommended)	Maximum PLB burst size for write transactions. Determines the maximum data transfers (WrDAcks) per write burst transaction.
16	PLBLE	PLB Lock Error 0 LOCKERROR signal not applied to the PLB slave 1 LOCKERROR signal applied to the PLB slave	When this bit is set, MAL applies the LOCKERROR signal to the PLB slave when it is the initiator during PLB transactions.
17	PLBB	PLB Burst 0 Burst transactions not allowed 1 Burst transactions allowed	When this bit is reset, MAL is not allowed to perform burst transactions.
18:23		Reserved	
24	OPBBL	OPB Bus Lock 0 OPB not locked 1 OPB locked	When this bit is set, MAL locks the OPB during data transfers to and from the COMMACs.
25:28		Reserved	
29	EOPIE	End of Packet Interrupt Enable 0 Generate interrupt on every end-of-packet only if the buffers I bit is set 1 Generate interrupt is on every end-of-packet	When this bit is set, an interrupt is generated on every end of packet (both transmit and receive). When clear, end of packet/buffer interrupt is generated only if the buffers I bit is set (1). Note: An interrupt is generated for every descriptor on which the I bit is set, regardless of the state of the EOPIE bit.
30	LEA	Locked Error Active 0 Handle errors in a non-locked mode 1 Handle errors in locked mode	Determines MAL's error handling mode. When this bit is set, MAL will handle errors in the locked mode, otherwise it will handle errors in a non-locked mode.
31	SD	MAL Scroll Descriptor 0 Do not scroll to the first descriptor of the next packet 1 Scroll to the first descriptor of the next packet	Determines whether or not MAL should scroll to the first descriptor of the next packet, following an early packet termination initiated by the related COMMAC. When set, Scrolling mode is active.

20.9.2 Channel Active Set and Reset Registers

For the Channel Active Set/Reset Registers (MAL0_TXCASR, MAL0_TXCARR, MAL0_RXCASR, MAL0_RXCARR), each bit represents its associated channel (bit 0 for channel 0, and so on). When a bit is equal to 1, the channel is been enabled for operation. When a bit is equal to 0, the channel is disabled (MAL

ignores any requests for service). If a channel is active when its enable bit is cleared, MAL stops processing the current packet. After the channel's enable bit is cleared, MAL goes back to the top of the channel descriptor table (pointed to by MAL0_TXCPTxR,MAL0_RXCPTxR).

- To enable a channel:
  - Write a 1 to its corresponding bit in the Channel Active Set Register (CASR).
  - Multiple channels can be enabled with a single CASR register write.
- To stop and reset a channel:
  - Write a 1 to its corresponding bit in the Channel Active Reset Register (CARR).
  - Writing a 0 to bits in the CARR registers has no effect on the channels.
  - Multiple channels can be reset with a single CARR register write.

MAL also clears the enable bit of a channel following an indication of an error on the channel. The CASR or CARR register(s) can be read to determine which channels are currently active.The following figures describe these registers.

Figure 0-4. TX Channel_Active Set Register (MAL0_TXCASR)

0:3	TCAS	Transmit Channel Active Set	Each bit represents its related channel (bit 0 for channel 0, and so on). When 1 is written to the bit, channel operation is enabled. MAL0 has four transmit channels.
4:31		Reserved	

Figure 0-5. TX Channel Active Reset Register (MAL0_TXCARR)

0:3	TCAR	Transmit Channel Active Reset	Each bit represents its related channel (bit 0 for channel 0, and so on). When 1 is written to the bit, channel operation is disabled. MAL0 has four transmit channels.
4:31		Reserved	

Figure 20-12. TX Channel_Active Set Register (MAL0_TXCASR)

0:3	TCAS	Transmit Channel Active Set	Each bit represents its related channel (bit 0 for channel 0, and so on). When 1 is written to the bit, channel operation is enabled. MAL0 has four transmit channels.
4:31		Reserved	

Figure 20-13. TX Channel Active Reset Register (MAL0_TXCARR)

0:3	TCAR	Transmit Channel Active Reset	Each bit represents its related channel (bit 0 for channel 0, and so on). When 1 is written to the bit, channel operation is disabled. MAL0 has four transmit channels.
4:31		Reserved	

Figure 0-6. RX Channel_Active Set Register (MAL0_RXCASR)

0:1	RCAS	Receive Channel Active Set	Each bit represents its related channel (bit 0 for channel 0, and so on). When 1 is written to the bit, channel operation is enabled. MAL0 has two receive channels.
2:31		Reserved	

Figure 20-14. RX Channel_Active Set Register (MAL0_RXCASR)

0:1	RCAS	Receive Channel Active Set	Each bit represents its related channel (bit 0 for channel 0, and so on). When 1 is written to the bit, channel operation is enabled. MAL0 has two receive channels.
2:31		Reserved	

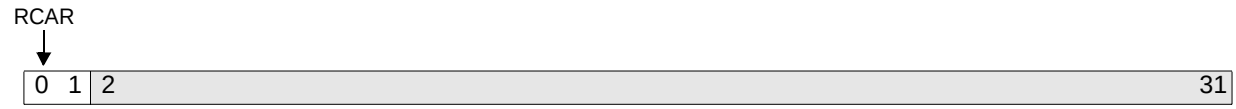


Figure 0-7. RX Channel_Active Reset Register (MAL0_RXCARR)

0:1	RCAR	Receive Channel Active Reset	Each bit represents its related channel (bit 0 for channel 0, and so on). When 0 is written to the bit, channel operation is disabled. MAL0 has two receive channels.
2:31		Reserved	

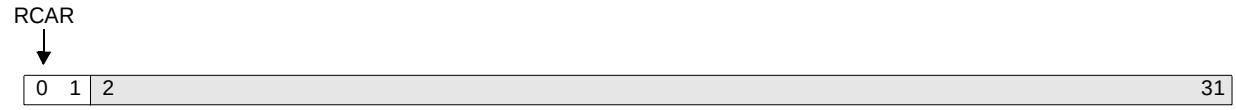


Figure 20-15. RX Channel_Active Reset Register (MAL0_RXCARR)

0:1	RCAR	Receive Channel Active Reset	Each bit represents its related channel (bit 0 for channel 0, and so on). When 0 is written to the bit, channel operation is disabled. MAL0 has two receive channels.
2:31		Reserved	

20.9.3 End of Buffer Interrupt Status Registers

Each bit in the TX End-of-Buffer Interrupt Status and RX End-of-Buffer Interrupt Status registers is related to a channel's descriptor buffer table.

The TX End-of-Buffer Interrupt Status register contains the End-of-Buffer Status bits for each TX channel. The RX End-of-Buffer Interrupt Status register contains the End-of-Buffer Status bits for the RX channels. The mechanism (as described below) for both RX and TX registers is the same.

MAL sets a channel's bit in one of the following conditions:

- When MAL finishes the processing of a buffer (writes back the status to the current descriptor), the related bit in this register is set if the I bit in the descriptor status is set.
- When MAL finishes the processing of a packet (writes back the status of the packet's last buffer) and MAL0_MCR[EOPIE] is set.

Note: In case MAL finishes the processing of a packet which is backed up, MAL doesn't consider it as an end of packet. Therefore, MAL will not set the appropriate channel bit in the End-of-Buffer Register.

- When the Bad Packet bit is set in the COMMAC channel Status halfword.

The device driver resets the interrupt by writing a 1 to the related bit. Writing a 0 has no effect.

Figure 0-8. TX End of Buffer Interrupt Status Register (MAL0_TXEOBISR)

0:3	TCEI	Transmit Channel End-of-Buffer Interrupt	Each bit represents its related channel (bit 0 for channel 0, and so on). Writing 1 to a bit clears it. MAL0 has four transmit channels.
4:31		Reserved	

Figure 20-16. TX End of Buffer Interrupt Status Register (MAL0_TXEOBISR)

0:3	TCEI	Transmit Channel End-of-Buffer Interrupt	Each bit represents its related channel (bit 0 for channel 0, and so on). Writing 1 to a bit clears it. MAL0 has four transmit channels.
4:31		Reserved	

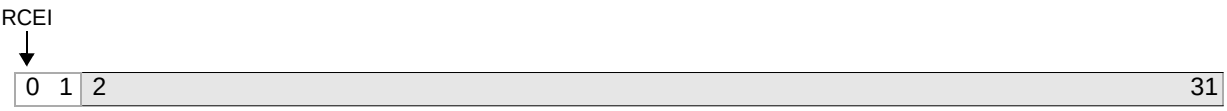


Figure 0-9. RX End of Buffer Interrupt Status Register (MAL0_RXEOBISR)

0:1	RCEI	Receive Channel End-of-Buffer Interrupt	Each bit represents its related channel (bit 0 for channel 0, and so on). Writing 1 to a bit clears it. MAL0 has two receive channels.
2:31		Reserved	

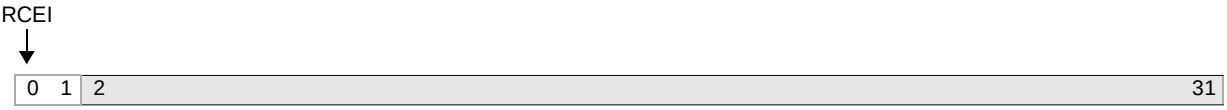


Figure 20-17. RX End of Buffer Interrupt Status Register (MAL0_RXEOBISR)

0:1	RCEI	Receive Channel End-of-Buffer Interrupt	Each bit represents its related channel (bit 0 for channel 0, and so on). Writing 1 to a bit clears it. MAL0 has two receive channels.
2:31		Reserved	

20.10 Error Registers

The following paragraphs describe MAL error registers. For more information about MAL errors see ~~"Error Handling" on page 20-19~~*Error Handling* on page 699.

20.10.1 MAL Error Status Register (MAL0_ESR)

This register holds the information about the error that occurred and the interrupts status. The register includes the following fields:

- **Error status bits** – This field holds the error information. The information includes the number of the channel on which the error occurred (if known) and the type of the error. The error can be either the last detected error or a locked error if "Locked error mode" is active. (See ~~"Operational Error Modes" on page 20-21~~*Operational Error Modes* on page 701 for description of the Locked error mode.)

The error status field includes an "Error Valid" bit which indicates whether there is error information in the error status field or not. The error status bits are not valid when the "Error Valid" bit is cleared (by writing 1 to this bit).

- **Interrupt status bits** – Every error detected by MAL sets a related bit in the interrupt status field. The interrupt status bits may be cleared by software by writing 1 to the bit to be cleared. The bits in this field are accumulative (more than one interrupt may be indicated here). These bits are masked by the IER (Interrupt Enable Register) to create a maskable interrupt (which is the MAL_SERR interrupt in the UIC).

Note: In order to reset the interrupt bits and the Error valid bit in the Error Status register, 1 must be written to the related bit. Writing 0 has no effect.

More than one bit can be cleared at a time and only R/W bits can be reset.

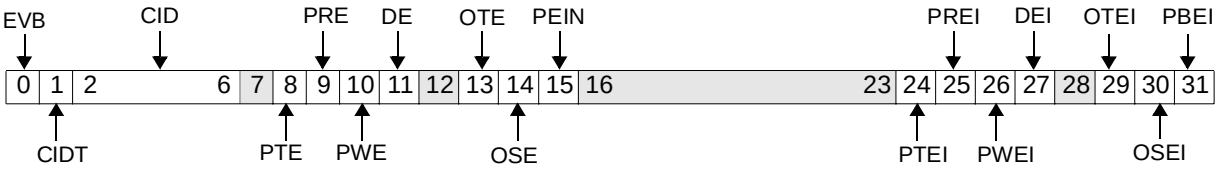


Figure 0-10. MAL Error Status Register (MAL0_ESR)

0	EVB	Error Valid Bit 0 Bit 1:15 are available for latching new error information. 1 Bits 1:15 contain last error. A new error cannot be latched.	When this bit is set, bits 1-6 include the ID of the erroneous channel (in case of OPB errors). Bits 11-15 indicate the type of error. In non-locked mode, the error indication describes the last error that had occurred. In locked mode, the error is the first one that had occurred after this bit was cleared. This bit is set when an error occurs and remains set until reset by the software. In locked mode, new errors cannot be latched in the error lock indication fields if this bit is set
1	CIDT	Channel ID Type 0 channel ID represents RX channel 1 channel ID represents TX channel	
2:6	CID	Channel ID Number	Indicates the number of the channel that caused the error. Note: An error on the PLB cannot be related to a channel. The error condition may be resolved by using the error information optionally locked in the PLB slave.
7		Reserved	
8	PTE	PLB Timeout Error 0 No error 1 PLB timeout error has occurred	Indicates the error is a PLB timeout
9	PRE	PLB Read Error 0 No error 1 PLB read error has occurred	

10	PWE	PLB Write Error 0 No error 1 PLB write error has occurred	
11	DE	Descriptor Error 0 No error 1 Non-valid descriptor	Indicates that the error is a non-valid descriptor, which is *not* the first descriptor in a TX packet.
12		Reserved	
13	OTE	OPB Timeout Error 0 No error 1 OPB timeout error has occurred	Indicates the error is an OPB timeout.
14	OSE	OPB Slave Error 0 No error 1 OPB slave error occurred	Indicates the error is an error indication asserted by an OPB slave.
15	PEIN	PLB Bus Error Indication 0 No error 1 PLB bus error occurred	When this bit is set, the detected error is a PLB error. There is no meaning to the Channel ID field in this case.
16:23		Reserved	
24	PTEI	PLB Timeout Error Interrupt 0 No error 1 PLB timeout error occurred	This bit is set following a PLB timeout error indication. Set condition for this bit generates a maskable interrupt.
25	PREI	PLB Read Error Interrupt 0 No error 1 PLB read error occurred	This bit is set following a PLB read error indication. Set condition for this bit generates a maskable interrupt.
26	PWEI	PLB Write Error Interrupt 0 No error 1 PLB write error occurred	This bit is set following a PLB write error indication. Set condition for this bit generates a maskable interrupt.
27	DEI	Descriptor Error Interrupt 0 No error 1 Descriptor data error recognized	A descriptor data error is recognized during access to the descriptor table. This error indication is asserted when a non-valid descriptor is accessed, which is *not* the first descriptor in a TX packet. Set condition for this bit generates a maskable interrupt.
28		Reserved	
29	OTEI	OPB Timeout Error Interrupt 0 No error 1 OPB time-out	This bit is set following an OPB time out error indication. Set condition for this bit generates a maskable interrupt.
30	OSEI	OPB Slave Error Interrupt 0 No error 1 OPB error from a slave	This bit is set following an OPB error indicated by the slave. Set condition for this bit generates a maskable interrupt.
31	PBEI	PLB Bus Error Interrupt 0 No error 1 PLB error indication	This bit is set following a PLB error indication (from the PLB slave). Set condition for this bit generates a maskable interrupt.

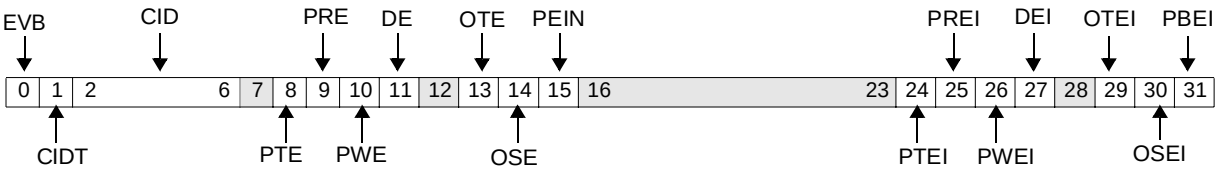


Figure 20-18. MAL Error Status Register (MAL0_ESR)

0	EVB	Error Valid Bit 0 Bit 1:15 are available for latching new error information. 1 Bits 1:15 contain last error. A new error cannot be latched.	When this bit is set, bits 1-6 include the ID of the erroneous channel (in case of OPB errors). Bits 11-15 indicate the type of error. In non-locked mode, the error indication describes the last error that had occurred. In locked mode, the error is the first one that had occurred after this bit was cleared. This bit is set when an error occurs and remains set until reset by the software. In locked mode, new errors cannot be latched in the error lock indication fields if this bit is set
1	CIDT	Channel ID Type 0 channel ID represents RX channel 1 channel ID represents TX channel	
2:6	CID	Channel ID Number	Indicates the number of the channel that caused the error. Note: An error on the PLB cannot be related to a channel. The error condition may be resolved by using the error information optionally locked in the PLB slave.
7		Reserved	
8	PTE	PLB Timeout Error 0 No error 1 PLB timeout error has occurred	Indicates the error is a PLB timeout
9	PRE	PLB Read Error 0 No error 1 PLB read error has occurred	
10	PWE	PLB Write Error 0 No error 1 PLB write error has occurred	
11	DE	Descriptor Error 0 No error 1 Non-valid descriptor	Indicates that the error is a non-valid descriptor, which is *not* the first descriptor in a TX packet.
12		Reserved	
13	OTE	OPB Timeout Error 0 No error 1 OPB timeout error has occurred	Indicates the error is an OPB timeout.
14	OSE	OPB Slave Error 0 No error 1 OPB slave error occurred	Indicates the error is an error indication asserted by an OPB slave.
15	PEIN	PLB Bus Error Indication 0 No error 1 PLB bus error occurred	When this bit is set, the detected error is a PLB error. There is no meaning to the Channel ID field in this case.
16:23		Reserved	

24	PTEI	PLB Timeout Error Interrupt 0 No error 1 PLB timeout error occurred	This bit is set following a PLB timeout error indication. Set condition for this bit generates a maskable interrupt.
25	PREI	PLB Read Error Interrupt 0 No error 1 PLB read error occurred	This bit is set following a PLB read error indication. Set condition for this bit generates a maskable interrupt.
26	PWEI	PLB Write Error Interrupt 0 No error 1 PLB write error occurred	This bit is set following a PLB write error indication. Set condition for this bit generates a maskable interrupt.
27	DEI	Descriptor Error Interrupt 0 No error 1 Descriptor data error recognized	A descriptor data error is recognized during access to the descriptor table. This error indication is asserted when a non-valid descriptor is accessed, which is *not* the first descriptor in a TX packet. Set condition for this bit generates a maskable interrupt.
28		Reserved	
29	OTEI	OPB Timeout Error Interrupt 0 No error 1 OPB time-out	This bit is set following an OPB time out error indication. Set condition for this bit generates a maskable interrupt.
30	OSEI	OPB Slave Error Interrupt 0 No error 1 OPB error from a slave	This bit is set following an OPB error indicated by the slave. Set condition for this bit generates a maskable interrupt.
31	PBEI	PLB Bus Error Interrupt 0 No error 1 PLB error indication	This bit is set following a PLB error indication (from the PLB slave). Set condition for this bit generates a maskable interrupt.

20.10.2 MAL Interrupt Enable Register (MAL0_IER)

Each bit in the following register, when it is set, enables assertion of the MAL0 SERR interrupt in the UIC when the related bit in the MAL0_ESR (interrupt bit) is set.

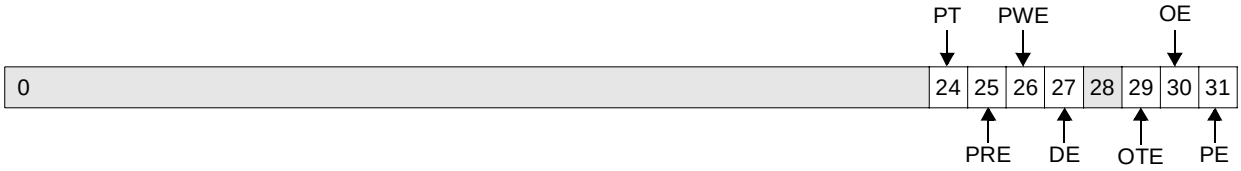


Figure 0-11. MAL Interrupt Enable Register (MAL0_IER)		
0:23		Reserved
24	PT	PLB Timeout Interrupt 0 PLB timeout interrupt disabled 1 PLB timeout interrupt enabled
25	PRE	PLB Read Error Interrupt 0 PLB read error interrupt disabled 1 PLB read error interrupt enabled
26	PWE	PLB Write Error Interrupt 0 PLB write error interrupt disabled 1 PLB write error interrupt enabled
27	DE	Descriptor Error Interrupt 0 Descriptor error interrupt disabled 1 Descriptior error interrupt enabled
28		Reserved
29	OTE	OPB Timeout Error Interrupt 0 OPB timeout error interrupt disabled 1 OPB timeout error interrupt enabled
30	OE	OPB Error Interrupt 0 OPB slave error interrupt disabled 0 OPB slave error interrupt enabled
31	PE	PLB Error Interrupt 0 PLB error interrupt disabled 1 PLB error interrupt enabled

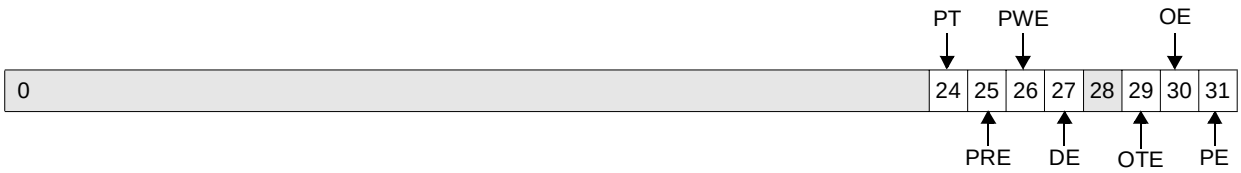


Figure 20-19. MAL Interrupt Enable Register (MAL0_IER)

0:23		Reserved
24	PT	PLB Timeout Interrupt 0 PLB timeout interrupt disabled 1 PLB timeout interrupt enabled
25	PRE	PLB Read Error Interrupt 0 PLB read error interrupt disabled 1 PLB read error interrupt enabled
26	PWE	PLB Write Error Interrupt 0 PLB write error interrupt disabled 1 PLB write error interrupt enabled
27	DE	Descriptor Error Interrupt 0 Descriptor error interrupt disabled 1 Descriptior error interrupt enabled
28		Reserved
29	OTE	OPB Timeout Error Interrupt 0 OPB timeout error interrupt disabled 1 OPB timeout error interrupt enabled
30	OE	OPB Error Interrupt 0 OPB slave error interrupt disabled 0 OPB slave error interrupt enabled
31	PE	PLB Error Interrupt 0 PLB error interrupt disabled 1 PLB error interrupt enabled

20.10.3 Descriptor Error Interrupt Registers (MAL0_TXDEIR, MAL0_RXDEIR)

Each bit in the following registers is related to a channel descriptor buffer table. Each bit indicates a descriptor data error related to a certain channel.

The TX Descriptor Error register contains the Descriptor errors bits of the TX channels. The RX Descriptor Error register contains the Descriptor errors bits of the RX channels. The mechanism (as described below) for both RX and TX registers is the same.

MAL sets a channel's bit when a descriptor data error was recognized during access to the descriptor table of a specific channel (see ~~"Descriptor Error" on page 20-19~~*Descriptor Error* on page 699).

The device driver resets the interrupt by writing a 1 to the related bit. Writing a 0 has no effect. When one or more of the TX Descriptor Error Interrupt bits is set, then the MAL_TX_DESC_ERR_INT bit is set. When one or more of the TX Descriptor Error Interrupt bits is set, the MAL TXDE interrupt in the UIC is set. When one or more of the RX Descriptor Error Interrupt bits is set, the MAL RXDE interrupt in the UIC is set.

Figure 0-12. TX Descriptor Error Interrupt Register (MAL0_TXDEIR)

0:3	TXDE	Transmit Descriptor Error Interrupt	Each bit represents its related channel (bit 0 for channel 0, and so on). When one or more bits are set to 1, MAL TXDE interrupt is set. Writing 1 to a bit clears it. MAL0 has four transmit channels.
4:31		Reserved	

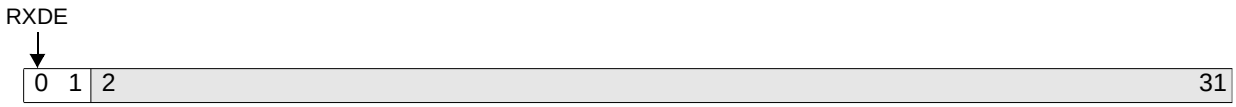


Figure 0-13. RX Descriptor Error Interrupt Register (MAL0_RXDEIR)

0:1	RXDE	Receive Descriptor Error Interrupt	Each bit represents its related channel (bit 0 for channel 0, and so on). When one or more bits are set, MAL RXDE interrupt is set. Writing 1 to a bit clears it. MAL0 has two receive channels.
2:31		Reserved	

Figure 20-20. TX Descriptor Error Interrupt Register (MAL0_TXDEIR)

0:3	TXDE	Transmit Descriptor Error Interrupt	Each bit represents its related channel (bit 0 for channel 0, and so on). When one or more bits are set to 1, MAL TXDE interrupt is set. Writing 1 to a bit clears it. MAL0 has four transmit channels.
4:31		Reserved	

Figure 20-21. RX Descriptor Error Interrupt Register (MAL0_RXDEIR)

0:1	RXDE	Receive Descriptor Error Interrupt	Each bit represents its related channel (bit 0 for channel 0, and so on). When one or more bits are set, MAL RXDE interrupt is set. Writing 1 to a bit clears it. MAL0 has two receive channels.
2:31		Reserved	

20.10.4 PLB Storage Attribute Registers (MAL0_TXTATTRR, MAL0_RXTATTRR)

These registers define the specific transfer information driven by MAL towards the PLB slaves.

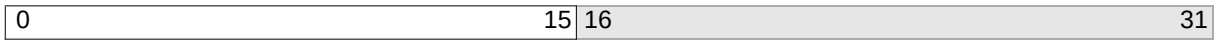


Figure 0-14. TX PLB Attribute Register (MAL0_TXTATTRR)

0:15		TX PLB Storage Attributes
16:31		Reserved

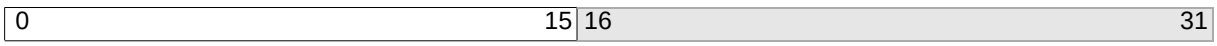


Figure 0-15. RX PLB Attribute Register (MAL0_RXTATTRR)

0:15		RX PLB Storage Attributes
16:31		Reserved

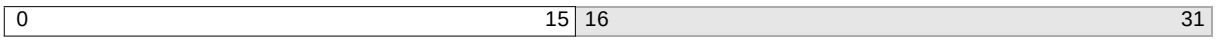


Figure 20-22. TX PLB Attribute Register (MAL0_TXTATTRR)

0:15		TX PLB Storage Attributes
16:31		Reserved

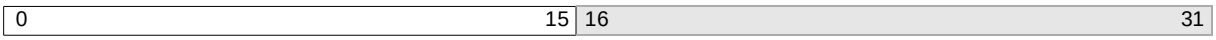


Figure 20-23. RX PLB Attribute Register (MAL0_RXTATTRR)

0:15		RX PLB Storage Attributes
16:31		Reserved

20.10.5 Channel Table Pointer Registers

Each channel has a full and separate 36-bit address space with the following limitations:

1. No data buffer is allowed to cross a 32-bit address space border.
2. No buffer descriptor table is allowed to cross a 32-bit address space border.

As the DCR bus is only 32 bits wide, each channel's table pointer needs to be configured in two DCR accesses as follows:

- Configure the 4 MSbs using a common register ((MAL0_TXBADDR or MAL0_RXBADDR))
- Configure the 32 LSbs using a channel private register (MAL0_TXCTPxR or MAL0_RXCTPxR)

20.10.5.1 Descriptor Base Address Registers (MAL0_TXBADDR, MAL0_RXBADDR)

Base address registers are gateways for updating and reading to and from all channels. MAL determines which channel should be used according to the specific TX/RX CTPR used. The flow of programming a table pointer address of 36 bits is described in the paragraphs that follow. Repeat the following steps for each channel.

Writing Flow

1. Write the four addresses' MSbs to the appropriate base address register (MAL0_TXBADDR or MAL0_RXBADDR).
2. Write the 32 LSbs to the specific channel's address register (MAL0_TXCTPxR or MAL0_RXCTPxR)

3. MAL will store the full 36-bit address combined from the above two DCR accesses in its internal memory.

Reading Flow

1. Read the 32 LSbs of the specific channel's address register.
2. MAL updates the (MAL0_TXBADDR or MAL0_RXBADDR) base address register according to the channel read in phase 1.
3. Read the 4 addresses' MSbs from bits 28 to 31 in the common base address register.

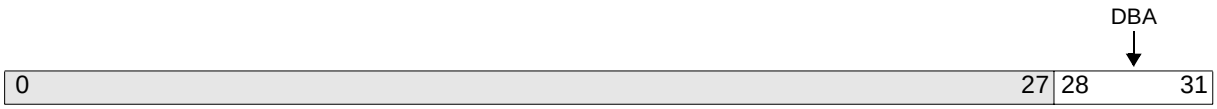


Figure 0-16. TX Descriptor Base Address Register (MAL0_TXBADDR)

0:27		Reserved
28:31	DBA	Descriptor Base Address

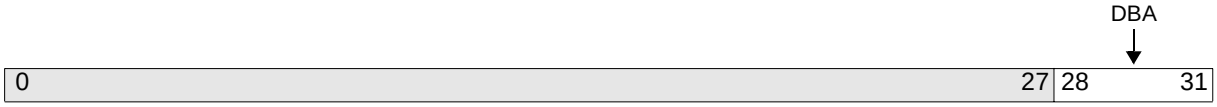


Figure 0-17. RX Descriptor Base Address Register (MAL0_RXBADDR)

0:27		Reserved
28:31	DBA	Descriptor Base Address

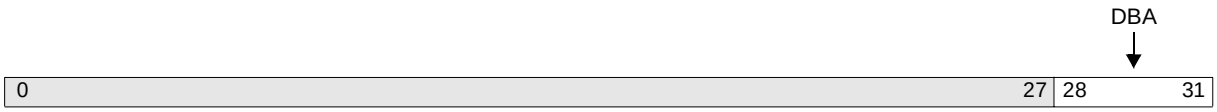


Figure 20-24. TX Descriptor Base Address Register (MAL0_TXBADDR)

0:27		Reserved
28:31	DBA	Descriptor Base Address

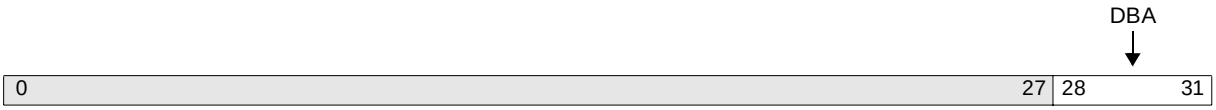


Figure 20-25. RX Descriptor Base Address Register (MAL0_RXBADDR)

0:27		Reserved
28:31	DBA	Descriptor Base Address

20.10.5.2 Channel Table Pointer Registers (MAL0_TXCTPxR, MAL0_RXCTPxR)

MAL uses RX Channel table pointer registers, one for each RX channel, and TX Channel table pointer registers, one for each TX channel. The Channel Table Pointer Registers point to the base address, in memory, of the descriptor buffer table used by each channel.

Note 1: Bits 0 to 12 of all the TXCTPxR registers are mapped to the same physical register. Writing into any of these registers overwrites the value of bits 0 to 12 in all the TXCTPxR registers. Read operation has no effect. Bits 0 to 12 of all the RXCTPxR registers are mapped to the same physical register. Writing into any of these registers overwrites the value of bits 0 to 12 in all the RXCTPxR registers. Read operation has no effect.

Note 2: When changing the value of any of the MAL0_TXCTPxR registers, both TX channels must be idle. To verify a channel is idle, check the device's Transmit Idle bit. Another way to assure that the channels are idle is to disable the channels before changing the MAL0_TXCTPxR register, and then re-enable them once the MAL0_TXCTPxR register is set to its new value.

Note 3: Although MAL supports a full 36-bit address space, no data buffer is allowed to cross a 32-bit address space border, and no buffer descriptor table is allowed to cross a 32-bit address space border.

The TX and RX Channel Table Pointer Registers have an identical format, as shown in ~~Figure 20-26~~*Figure 20-26*.and ~~Figure 20-27~~*Figure 20-27*. MAL0 has four TX channel table pointer registers, and two RX channel table pointer registers.

Figure 0-18. TX Channel Table Pointer Register (MAL0_TXCTPxR)

0:31		Channel Table Pointer	Pointer to the base address of the buffer descriptor table used by the channel. The value entered should point to a location in memory accommodating an aligned doubleword (the three least significant bits of the pointer must be 000). MAL0 has four transmit channels.

Figure 20-26. TX Channel Table Pointer Register (MAL0_TXCTPxR)

0:31		Channel Table Pointer	Pointer to the base address of the buffer descriptor table used by the channel. The value entered should point to a location in memory accommodating an aligned doubleword (the three least significant bits of the pointer must be 000). MAL0 has four transmit channels.

Figure 0-19. RX Channel Table Pointer Register (MAL0_RXCTPxR)

0:31		Channel Table Pointer	Pointer to the base address of the buffer descriptor table used by the channel. The value should point to a location in memory accommodating an aligned doubleword (the three least significant bits of the pointer must be 000). MAL0 has two receive channels.

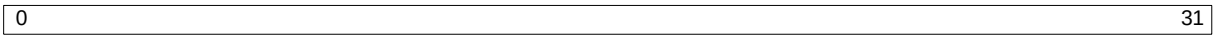


Figure 20-27. RX Channel Table Pointer Register (MAL0_RXCTPxR)

0:31		Channel Table Pointer	Pointer to the base address of the buffer descriptor table used by the channel. The value should point to a location in memory accommodating an aligned doubleword (the three least significant bits of the pointer must be 000). MAL0 has two receive channels.

The Table Pointer Registers retain their value following Soft Reset or Channel Reset.

20.10.6 RX Channel Buffer Size Register (MAL0_RCBSx)

Each RX channel has a fixed buffer length. The RX channel buffer size registers are configured by the device driver to determine the length of the buffer. This size is the maximal number of bytes that can be stored in the buffer. More than one buffer is used to store the incoming packet in case the length of the incoming packet data is more than the channel's buffer length (as defined by the channel's related register).

The buffer length for a given channel can be from 16 bytes to (4KB-16) bytes (in 16-byte steps). This length is represented by a 8-bit field. The buffer size in bytes is calculated as follows: RX_Buffer_Size_Field x 16. Figure 20-28 describes MAL0_RCBS0 register bits.

Figure 0-20. RX Channel Buffer Size Register (MAL0_RCBSx

0:23		Reserved
24:31		Receive Channel Buffer Size

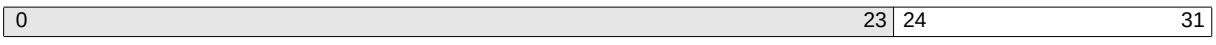


Figure 20-28. RX Channel Buffer Size Register (MAL0_RCBSx

0:23		Reserved

24:31		Receive Channel Buffer Size

21. EMAC to PHY Interface Bridge

The PPC440GP provides a bridge called the ZMII bridge that connects the ethernet media access controllers (EMACs) to standard physical devices (PHYs). Media independent interface (MII) and management data interface (MDI) signals to and from an EMAC are passed to the ZMII bridge. The ZMII bridge passes MII and MDI signals through to the PHY, or formats the MII signals to support the reduced media independent interface (RMII) or serial media independent interface (SMII).

Depending on the ZMII configuration, ZMII provides one of the following off-chip interfaces: one MII, two RMII, or two SMII. The interfaces are mutually exclusive. Interface selection is controlled by the Function Enable Register (ZMII0_FER).

RMII and SMII significantly reduce the external signal count required to support multiple EMAC ports. The interfaces share two management data interface (MDI) signals (EMCMDIO and EMCMDCLK). While the MII requires 16 additional pins for a single interface, RMII requires only seven additional pins (six data and control and a common clock) and SMII requires only four additional pins (two data, sync, and a clock). Configuring the ZMII bridge for SMII or RMII allows the use of both EMACs using a reduced number of external pins.

ZMII mode is selected by enabling bits 28:29 of Power-On Configuration Register 0 (CPC0_STRP0) as follows:

28:29 - ZMII Selection

00 - MII mode selected

01 - SMII mode selected

10 - RMII 10Mb mode selected

11 - RMII 100Mb mode selected

21.1 ZMII Features

The ZMII bridge features:

- Support for one MII PHY
- Support for two RMII PHYs
  - 10Mbps and 100Mbps data rates
  - 50mhz reference clock sourced from EMAC, or an external source, to the PHY
  - Independent 2-bit transmit and receive data paths
- Support for two SMII PHYs
  - 10Mbps and 100Mbps data rates
  - Half and full duplex operation
  - System clock sharing for multiple EMACs and PHYs
  - Independent 1-bit transmit and receive data paths
- Programmable selection of either EMAC to drive MDI
- Programmable selection of either EMAC to drive MII

- Programmable selection of either or both EMACs to drive RMII or SMII

21.2 ZMII Bridge Interface Signals

Table 21-1 lists the signals in the ZMII bridge interface. The "Pin" column lists the signals associated with package pins. Each pin is associated with one or more ZMII bridge signals, which are listed in the "MII," "RMII," and "SMII" columns. The ZMII bridge signal associated with each pin depends on the ZMII bridge interface selection. Note that MII, RMII, and SMII share the MDI signals ENETMDIO and EMCMDCLK.

Table 21-1. ZMII Bridge PHY Interface

MII	RMII	SMII	I/O	Description
EMCMDIO	EMCMDIO	EMCMDIO	I/O	MDIO data input/output
EMCMDClk	EMCMDCLK	EMCMDCLK	I	MDIO data clock
EMCRxD0	EMC0RxD0	EMC0RxD	I	MII receive data 0 RMII0 receive data 0 SMII0 receive data
EMCRxD1	EMC0RxD1	EMC1RxD	I	MII receive data 1 RMII0 receive data 1 SMII1 receive data
EMCRxD2	EMC1RxD0		I	MII receive data 2 RMII1 receive data 0
EMCRxD3	EMC1RxD1		I	MII receive data 3 RMII1 receive data 1
EMCRxDV	EMC1CRSDV		I	MII receive data valid RMII1 receive data valid
EMCRxClk			I	MII receive medium clock
EMCRxErr	EMC0RxErr		I	MII receive error RMII0 receive error
EMCCRS	EMC0CRSDV		I	MII carrier sense RMII0 carrier sense data valid
EMCTxClk	EMCRefClk	EMCRefClk	I	MII transmit clock RMII reference clock SMII reference clock
EMCCCD	EMC1RxErr		I	MII collision RMII1 receive error
EMCTxErr	EMC1TxEn		O	MII transmit error RMII1 transmit enable
EMCTxEn	EMC0TxEn	EMCSync	O	MII transmit enable RMII0 transmit enable SMII sync
EMCTxD0	EMC0TxD0	EMC0TxD	O	MII transmit data 0 RMII0 transmit data 0 SMII0 transmit data 0

Table 21-1. ZMII Bridge PHY Interface (continued)

MII	RMII	SMII	I/O	Description
EMCTxD1	EMC0TxD1	EMC1TxD	O	MII transmit data 1 RMII0 transmit data 1 SMII1 transmit data 1
EMCTxD2	EMC1TxD0		O	MII transmit data 2 RMII1 transmit data 0
EMCTxD3	EMC1TxD1		O	MII transmit data 3 RMII1 transmit data 1

21.3 EMAC-ZMII Bridge Interfaces

The ZMII bridge attaches to the EMACs on the standard MII, which consists of three sections: transmit data, receive data, and MDI. The transmit and receive data sections move data nibbles across the interface at 25 MHz and 2.5 MHz for 100 Mbps or 10 Mbps operation, respectively. Control signals define valid data, error conditions, and collision detection.

The standard MDI interface enables software to access registers in the PHY and consists of a single bidirectional data signal and a clock driven from ZMII. The MDI connection is always a pass-through interface in the ZMII Bridge.

21.3.1 MII Interface

When the ZMII bridge is configured for MII, the MII signals from one EMAC are simply passed through the ZMII bridge to a PHY. Figure 21-1 shows an EMAC connected to the ZMII bridge and a PHY. If using MII mode, only a single EMAC may be used.

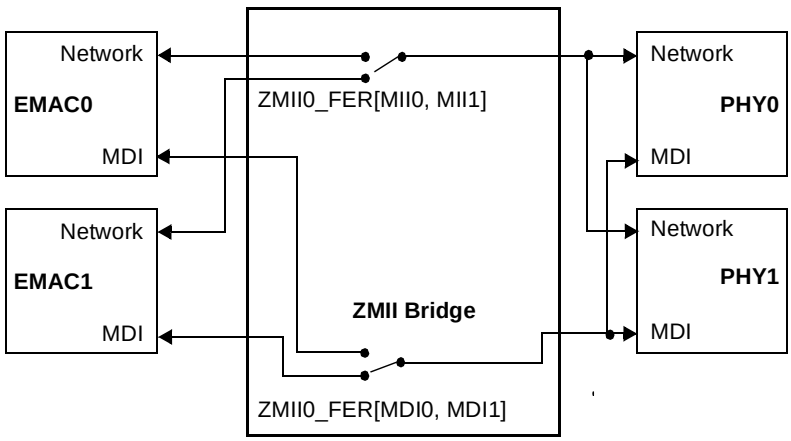


Figure 21-1. EMAC to PHY Using MII

21.3.2 RMII Interface

When the ZMII bridge is configured for RMII, the number of data signals is halved and the operating frequency is doubled to 50MHz. Each EMAC can connect through the ZMII bridge to a PHY. Figure 21-2 shows both EMACs connected to the ZMII bridge and two PHYs.

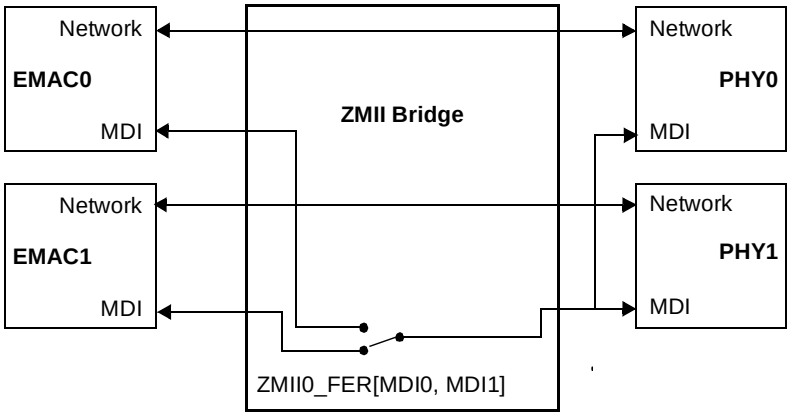


Figure 21-2. EMAC to PHY Using RMII

21.3.2.1 SMII Interface

The SMII data and control paths are similar to those of RMII. When the ZMII bridge is configured for SMII, MII signal requirements are further reduced by quartering the data width (to a single bit) and multiplying the clock frequency by 10. All signals are synchronous to a 125 MHz clock and the sync signal.

Receive information (data and control) is moved in 10-bit segments. At 100 Mbps, each segment represents a new byte. At 10 Mbps, each segment is repeated ten times. Segment boundaries are defined by a sync signal. The ZMII bridge generates a pulse on the sync signal every 10 clocks.

Transmit operates in the same way. Transmit information (data and control) is sent in 10-bit segments. At 100 Mbps, a new byte is sent every 10 clocks. At 10 Mbps, each segment is repeated 10 times; the PHY can sample any of the repeated 10-bit segments.

Figure 21-3 shows the connection of both EMACs to the ZMII bridge and two PHYs.

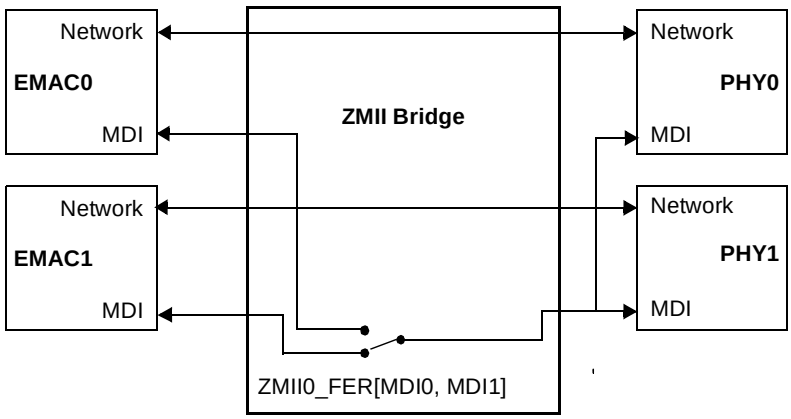


Figure 21-3. EMAC to PHY Using SMII

21.4 ZMII Bridge Registers

This section describes the registers in the ZMII bridge, which are listed in Table 21.4. The ZMII bridge registers are accessed using an OPB slave interface.

Table 21-2. Register Address List

Register	Description	Address	Access	Page
ZMII0_FER	ZMII Function Enable Register	0x1 40000780	R/W	~~21-5~~723
ZMII0_SSR	ZMII Speed Select Register	0x1 40000784	R/W	~~21-6~~724
ZMII0_SMIISR	ZMII SMII Status Register	0x1 40000788	R/W	~~21-8~~726

21.4.1 Function Enable Register (ZMII0_FER)

ZMII_FER selects the various interface conversion functions provided by the ZMII bridge. ZMII_FER also deselects (functionally isolates) an unused EMAC. The interface conversion functions are mutually exclusive, that is, ~~both~~ all EMACs must convert to the same interface (SMII, RMII, or MII) in the bridge, or be deselected (an EMAC must be deselected for MII). Mixed interface conversion is not supported.

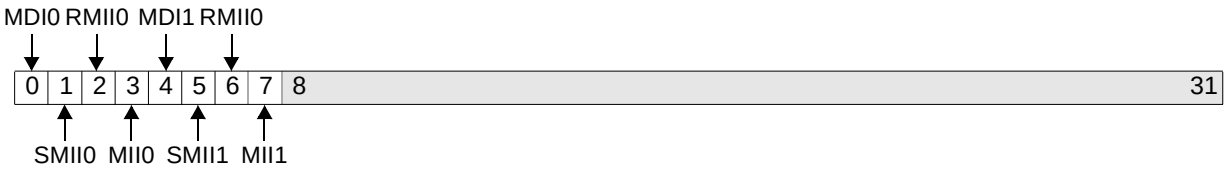


Figure 0-1. Function Enable Register (ZMII0_FER)

0	MDI0	EMAC0 MDI Enable 0 EMAC0 management data and clock are ignored. 1 EMAC0 management data and clock are driven to the PHY.	Must be 0 if MDI1 = 1.
1	SMII0	EMAC0 SMII Enable 0 EMAC0 SMII PHY interface is disabled. 1 EMAC0 SMII PHY interface is enabled.	Must be 0 if RMII0, MII0, RMII1, or MII1 is 1.
2	RMII0	EMAC0 RMII Enable 0 EMAC0 RMII PHY interface is disabled. 1 EMAC0 RMII PHY interface is enabled.	Must be 0 if SMII0, MII0, SMII1, or MII1 is 1.
3	MII0	EMAC0 MII Enable 0 EMAC0 MII PHY interface is disabled. 1 EMAC0 MII PHY interface is enabled.	Must be 0 if SMII0, RMII0, SMII1, RMII1, or MII1 is 1.
4	MDI1	EMAC1 MDI Enable 0 EMAC1 management data and clock are ignored. 1 EMAC1 management data and clock are driven to the PHY.	Must be 0 if MDI0 = 1.
5	SMII1	EMAC1 SMII Enable 0 EMAC1 SMII PHY interface is disabled. 1 EMAC1 SMII PHY interface is enabled.	Must be 0 if RMII0, MII0, RMII1, or MII1 is 1.
6	RMII1	EMAC0 RMII Enable 0 EMAC1 RMII PHY interface is disabled. 1 EMAC1 RMII PHY interface is enabled.	Must be 0 if SMII0, MII0, SMII1, or MII1 is 1.
7	MII1	EMAC0 MII Enable 0 EMAC1 MII PHY interface is disabled. 1 EMAC1 MII PHY interface is enabled.	Must be 0 if SMII0, RMII0, MII0, SMII1, or RMII1 is 1.
8:31		Reserved	

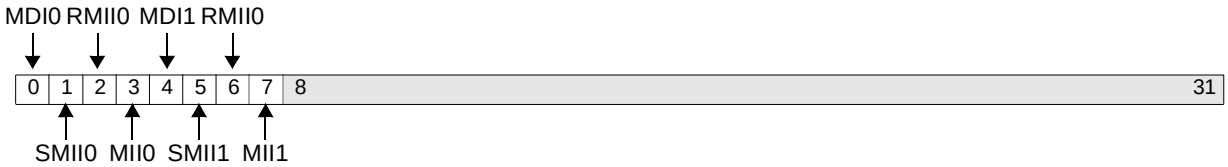


Figure 21-4. Function Enable Register (ZMII0_FER)

0	MDI0	EMAC0 MDI Enable 0 EMAC0 management data and clock are ignored. 1 EMAC0 management data and clock are driven to the PHY.	Must be 0 if MDI1 = 1.
1	SMII0	EMAC0 SMII Enable 0 EMAC0 SMII PHY interface is disabled. 1 EMAC0 SMII PHY interface is enabled.	Must be 0 if RMII0, MII0, RMII1, or MII1 is 1.
2	RMII0	EMAC0 RMII Enable 0 EMAC0 RMII PHY interface is disabled. 1 EMAC0 RMII PHY interface is enabled.	Must be 0 if SMII0, MII0, SMII1, or MII1 is 1.
3	MII0	EMAC0 MII Enable 0 EMAC0 MII PHY interface is disabled. 1 EMAC0 MII PHY interface is enabled.	Must be 0 if SMII0, RMII0, SMII1, RMII1, or MII1 is 1.
4	MDI1	EMAC1 MDI Enable 0 EMAC1 management data and clock are ignored. 1 EMAC1 management data and clock are driven to the PHY.	Must be 0 if MDI0 = 1.
5	SMII1	EMAC1 SMII Enable 0 EMAC1 SMII PHY interface is disabled. 1 EMAC1 SMII PHY interface is enabled.	Must be 0 if RMII0, MII0, RMII1, or MII1 is 1.
6	RMII1	EMAC0 RMII Enable 0 EMAC1 RMII PHY interface is disabled. 1 EMAC1 RMII PHY interface is enabled.	Must be 0 if SMII0, MII0, SMII1, or MII1 is 1.
7	MII1	EMAC0 MII Enable 0 EMAC1 MII PHY interface is disabled. 1 EMAC1 MII PHY interface is enabled.	Must be 0 if SMII0, RMII0, MII0, SMII1, or RMII1 is 1.
8:31		Reserved	

21.4.2 Speed Select Register (ZMII0_SSR)

ZMII_SSR selects the speed at which each EMAC transfer data (10 Mbps or 100 Mbps). When 10 Mbps is selected, the ZMII bridge generates a 2.5 MHz clock. When 100 Mbps is selected, the ZMII bridge generates a 25 MHz clock.

ZMII_SSR also controls whether collisions are indicated.

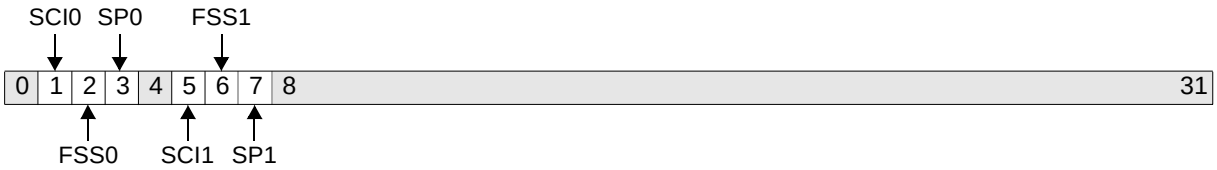


Figure 0-2. Speed Selection Register (ZMII0_SSR)

0		Reserved	
1	SCI0	EMAC0 Suppress Collision Indication 0 EMAC0 collision indication signal is asserted. 1 EMAC0 collision indication signal is deasserted.	
2	FSS0	EMAC0 Force Speed Selection 0 ZMII0_SP0 does not override IPG status field. 1 ZMII0_SP0 overrides IPG status field.	SMII only.
3	SP0	EMAC0 Speed Selection 0 10 Mbps selected. 1 100 Mbps selected.	Must be programmed for RMII; can be programmed for SMII; ignored for MII.
4		Reserved	
5	SCI1	EMAC1 Suppress Collision Indication 0 EMAC1 collision indication signal is asserted. 1 EMAC1 collision indication signal is deasserted.	
6	FSS1	EMAC1 Force Speed Selection 0 ZMII0_SP1 does not override IPG status field. 1 ZMII0_SP1 overrides IPG status field.	SMII only.
7	SP1	EMAC1 Speed Selection 0 10 Mbps selected. 1 100 Mbps selected.	Must be programmed for RMII; can be programmed for SMII; ignored for MII.
8:31		Reserved	

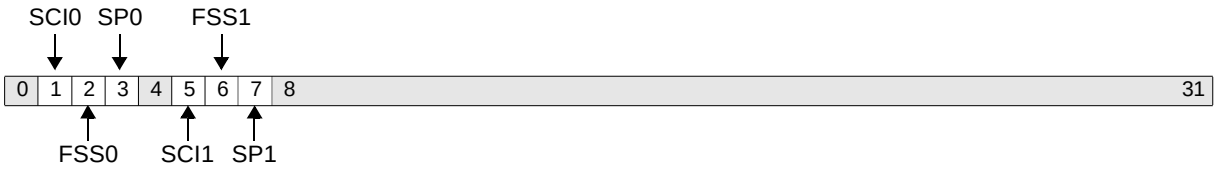


Figure 21-5. Speed Selection Register (ZMII0_SSR)

0		Reserved

1	SCI0	EMAC0 Suppress Collision Indication 0 EMAC0 collision indication signal is asserted. 1 EMAC0 collision indication signal is deasserted.	
2	FSS0	EMAC0 Force Speed Selection 0 ZMII0_SP0 does not override IPG status field. 1 ZMII0_SP0 overrides IPG status field.	SMII only.
3	SP0	EMAC0 Speed Selection 0 10 Mbps selected. 1 100 Mbps selected.	Must be programmed for RMII; can be programmed for SMII; ignored for MII.
4		Reserved	
5	SCI1	EMAC1 Suppress Collision Indication 0 EMAC1 collision indication signal is asserted. 1 EMAC1 collision indication signal is deasserted.	
6	FSS1	EMAC1 Force Speed Selection 0 ZMII0_SP1 does not override IPG status field. 1 ZMII0_SP1 overrides IPG status field.	SMII only.
7	SP1	EMAC1 Speed Selection 0 10 Mbps selected. 1 100 Mbps selected.	Must be programmed for RMII; can be programmed for SMII; ignored for MII.
8:31		Reserved	

21.4.3 SMII Status Register (ZMII0_SMIISR)

ZMII_SMIISR contains the status transferred during the last inter-packet gap in the SMII interface, at the moment it is read. This register is used only if SMII interfaces are enabled. The register bits pertain to EMAC0RxD and EMAC1RxD SMII inter frame status used to convey packet data, RX_ER, and PHY status and are valid only when RX_DV (receive data valid) is set to 0.

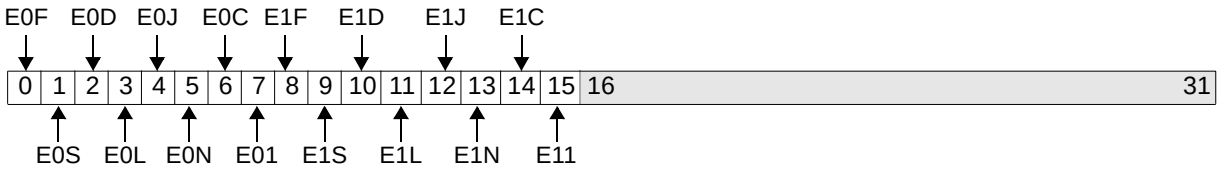


Figure 0-3. SMII Status Register (ZMII0_SMIISR)

0	E0F	EMAC0RxD from Previous Frame	Indicates whether or not the PHY detected an error somewhere in the previous frame. Should be valid in the segment immediately following a frame, and should stay valid until the first data segment of the next frame begin.
1	E0S	EMAC0RxD Speed 0 10MBit 1 100MBit	
2	E0D	EMAC0RxD Duplex 0 Half 1 Full	
3	E0L	EMAC0RxD Link 0 Down 1 Up	
4	E0J	EMAC0RxD Jabber 0 OK 1 Error	
5	E0N	EMAC0RxD Nibble 0 Invalid 1 Valid	Conveys the validity of the upper nibble of the last byte of the previous frame. Should be valid in the segment immediately following a frame, and should stay valid until the first data segment of the next frame begin.
6	E0C	EMAC0RxD False Carrier Detected	When asserted, this bit indicates that the PHY has detected a false carrier event.
7	E01	EMAC0RxD Set to 1	
8	E1F	EMAC1RxD from Previous Frame	Indicates whether or not the PHY detected an error somewhere in the previous frame. Should be valid in the segment immediately following a frame, and should stay valid until the first data segment of the next frame begin.

9	E1S	EMAC1RxD Speed 0 10MBit 1 100MBit	
10	E1D	EMAC1RxD Duplex 0 Half 1 Full	
11	E1L	EMAC1RxD Link 0 Down 1 Up	
12	E1J	EMAC1RxD Jabber 0 OK 1 Error	
13	E1N	EMAC1RxD Nibble 0 Invalid 1 Valid	Conveys the validity of the upper nibble of the last byte of the previous frame. Should be valid in the segment immediately following a frame, and should stay valid until the first data segment of the next frame begin.
14	E1C	EMAC1RxD False Carrier Detected	When asserted, this bit indicates that the PHY has detected a false carrier event.
15	E11	EMAC1RxD Set to 1	
16:31		Reserved.	

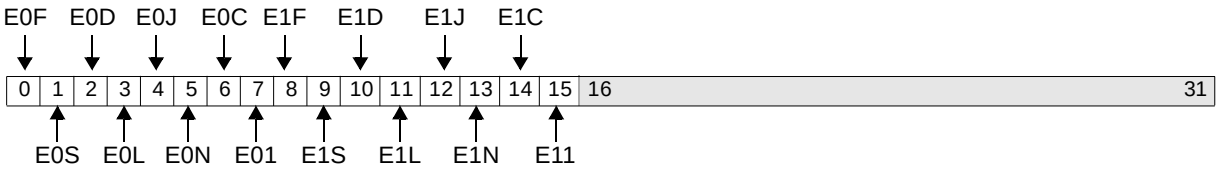


Figure 21-6. SMII Status Register (ZMII0_SMIISR)

0	E0F	EMAC0RxD from Previous Frame	Indicates whether or not the PHY detected an error somewhere in the previous frame. Should be valid in the segment immediately following a frame, and should stay valid until the first data segment of the next frame begin.
1	E0S	EMAC0RxD Speed 0 10MBit 1 100MBit	
2	E0D	EMAC0RxD Duplex 0 Half 1 Full	
3	E0L	EMAC0RxD Link 0 Down 1 Up	
4	E0J	EMAC0RxD Jabber 0 OK 1 Error	

5	E0N	EMAC0RxD Nibble 0 Invalid 1 Valid	Conveys the validity of the upper nibble of the last byte of the previous frame. Should be valid in the segment immediately following a frame, and should stay valid until the first data segment of the next frame begin.
6	E0C	EMAC0RxD False Carrier Detected	When asserted, this bit indicates that the PHY has detected a false carrier event.
7	E01	EMAC0RxD Set to 1	
8	E1F	EMAC1RxD from Previous Frame	Indicates whether or not the PHY detected an error somewhere in the previous frame. Should be valid in the segment immediately following a frame, and should stay valid until the first data segment of the next frame begin.
9	E1S	EMAC1RxD Speed 0 10MBit 1 100MBit	
10	E1D	EMAC1RxD Duplex 0 Half 1 Full	
11	E1L	EMAC1RxD Link 0 Down 1 Up	
12	E1J	EMAC1RxD Jabber 0 OK 1 Error	
13	E1N	EMAC1RxD Nibble 0 Invalid 1 Valid	Conveys the validity of the upper nibble of the last byte of the previous frame. Should be valid in the segment immediately following a frame, and should stay valid until the first data segment of the next frame begin.
14	E1C	EMAC1RxD False Carrier Detected	When asserted, this bit indicates that the PHY has detected a false carrier event.
15	E11	EMAC1RxD Set to 1	
16:31		Reserved.	

22. Ethernet Media Access Controllers

The PPC440GP provides two Ethernet media access controllers (EMACs) that are generic implementations of the Ethernet Media Access Control (MAC) protocol complying with ANSI/IEEE Std 802.3 and IEEE 802.3u supplement. EMAC supports half-duplex (CSMA/CD) and full-duplex operation for 10-Mbps and 100-Mbps operations.

Both EMACs are implemented identically, with the exception of register addresses. Because both EMACs operate identically, the rest of this chapter describes a single EMAC. The EMACs are referred to as ~~EMAC0 and~~ EMAC0and EMAC1. Except in the register summary tables in ~~"EMAC Registers" on page 22-23~~*EMAC Registers* on page 752, the EMAC registers are prefixed "EMACx_" to denote the identical implementation of registers in each EMAC.

Each EMAC provides two on-chip peripheral bus (OPB) slave interfaces. The first OPB interface provides access to the EMAC configuration and status registers. The PLB/OPB bridge enables the processor core to access these registers.

The second OPB interface is used to exchange packet information with the memory access layer (MAL). The MAL is a multi-channel, intermediate hardware layer that resides between packet-based communication cores (such as EMAC) and external memory (such as SDRAM or SRAM). The MAL transfers packet information and status between the EMACs and external memory separately for each of the three EMAC channels (one receive and two transmit). Software (a device driver) maintains a buffer descriptor ring and a set of data buffers in external memory for each channel, and manages the exchange of packet data between the data buffers and the software protocol.

The MAL performs functions such as arbitration between service requests, handling the buffer descriptor memory structure, updating the descriptor status/control fields at the end of packet transfer, and so on. EMAC supports unlimited burst length transactions on the MAL interface.

Each EMAC uses a media independent interface (MII) to communicate with the Z media independent interface (ZMII) bridge. The ZMII bridge, described in detail in ~~Chapter 21, "EMAC to PHY Interface Bridge,"~~*EMAC to PHY Interface Bridge* on page 719 communicates with standard physical devices (PHYs). The ZMII bridge supports the media independent interface (MII), the reduced pin count reduced media independent interface (RMII), and serial media independent interface (SMII). Details of PHY communication are in ~~21.3, "EMAC-ZMII Bridge Interfaces," on page 21-3~~*EMAC-ZMII Bridge Interfaces* on page 721. The ZMII bridge receives and drives the two MIIs from the two EMACs and produces one of the following off-chip interfaces: one MII, two RMII, or two SMII, depending on the ZMII configuration.

EMAC uses independent receive and transmit FIFOs. Programmable FIFO thresholds minimize overflows and underruns, and can launch integrated IEEE 802.3x pause packets for flow control.

As part of the remote monitoring (RMON) and management information base (MIB) defined in IEEE 802.3z, the EMAC contains registers that count the number of octets transmitted and received.

Figure 22-1 illustrates an EMAC in a typical Ethernet application. _

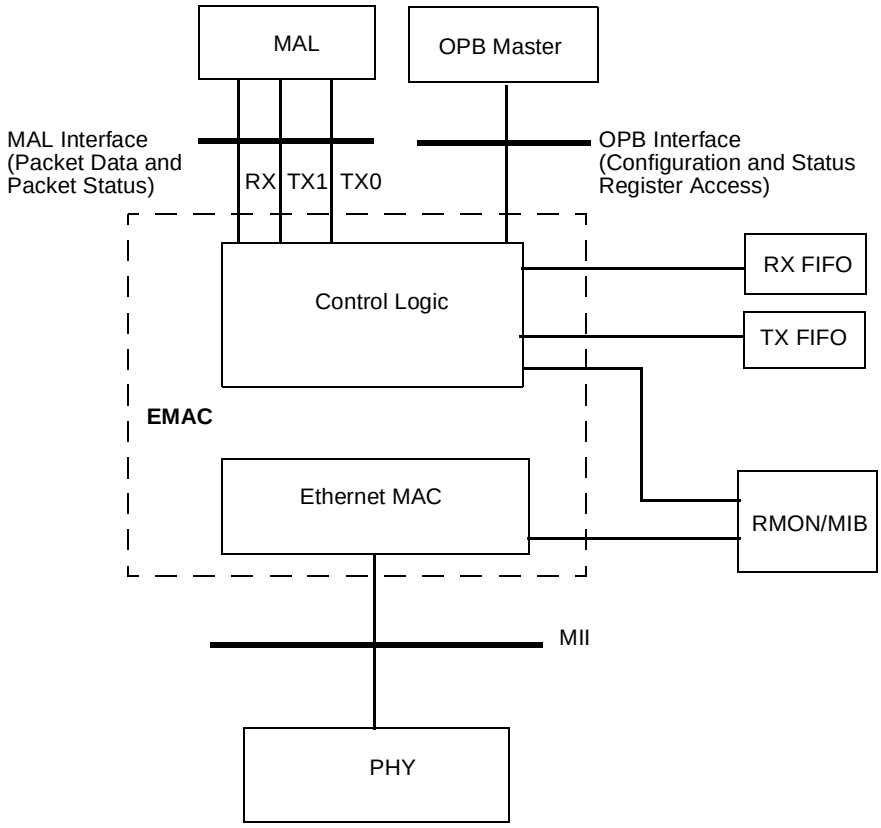


Figure 22-1. EMAC in a Typical Ethernet Application

Note: Each EMAC is connected to an OPB, to which the OPB master and MAL are also connected. EMAC transmit channels operate independently from the receive channels.

22.1 EMAC Features

Each EMAC features:

- Dual speed (10/100 Mbps) CSMA/CD (half-duplex) and full-duplex Ethernet MAC complying with ANSI/IEEE Std. 802.3 and IEEE 802.3u supplement.
- Automatic source address insertion or replacement for transmitted packets is a programmable option.
- Automatic stripping of frame padding bytes and frame check sequence (FCS) is a programmable option.

  When padding bytes are stripped, the padding and FCS field are removed. FCS stripping removes only the FCS field.
- FCS control for transmit/receive packets.
- Access to registers with support for burst processing.
- MAL for packet moving having one-cycle MAL slave latency.

- Independent, large (2 KB) transmit and (4 KB) receive FIFOs with programmable thresholds to minimize overruns and underruns.
- Multiple packet handling in transmit and receive FIFOs.
- Unicast, multicast, broadcast, and promiscuous address filtering capabilities.
- Two 64-bit hash filters for unicast and multicast packets.
- Automatic retransmission of collided packets.
- Rejection of runt packets before providing them to MAL.
- MII for connection to a variety of PHY layer devices.
- Programmable inter-packet gap to enable tuning for better system performance.
- Compliance with IEEE 802.3x standard packet-based flow control, including self-assembled control pause packet transmitting.
- Support for VLAN tag ID in compliance with IEEE Draft 802.3ac/D1.0 standard.
- VLAN tag insertion or replacement for transmit packets is a programmable option.
- Wake on LAN (WOL) handling.
- Programmable internal and external loop-back capabilities.
- Extensive error/status vector generation for each processed packet.
- Power management using a clock and power management (CPM) unit.

22.2 EMAC Operation

The EMAC hardware components and its internal structure are illustrated in the block diagram in Figure 22-2.

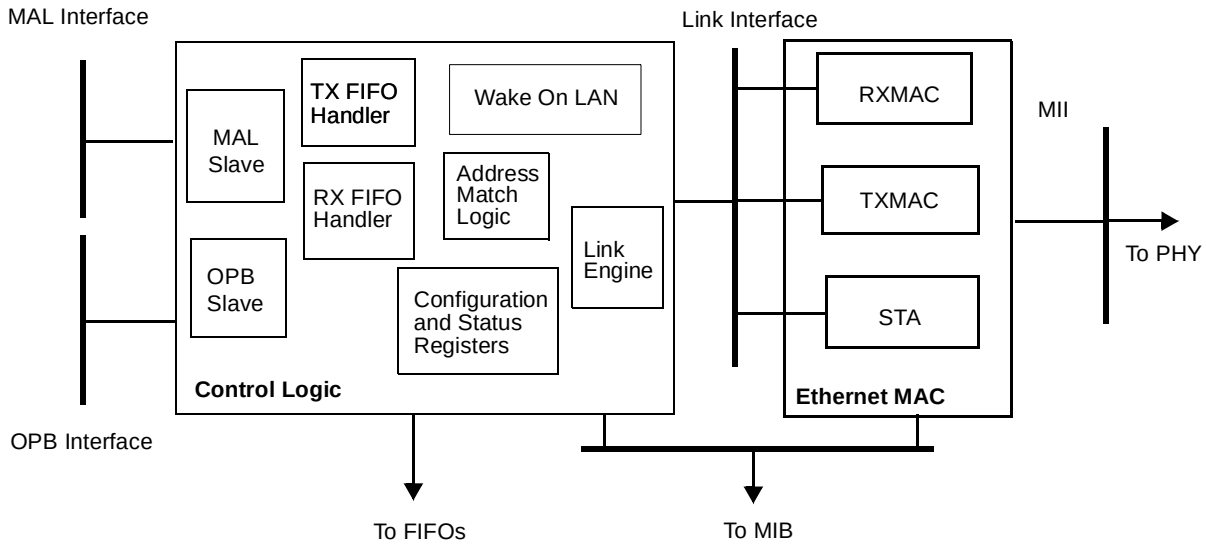


Figure 0-1. Internal EMAC Structure

The EMAC hardware components and its internal structure are illustrated in the block diagram in Figure 22-2. The EMAC is connected to a physical layer device via the MII interface defined by eMAC configuration . .

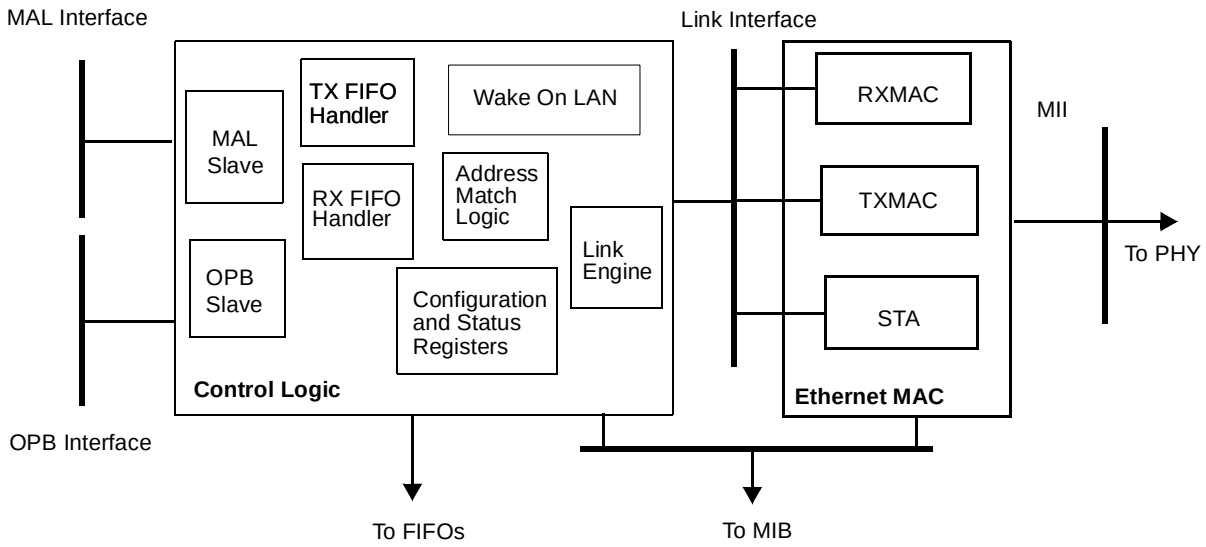


Figure 22-2. Internal EMAC Structure

The control logic sub-block implements the following functions:

- OPB slave device
- MAL slave device
- FIFO management logic
- Ethernet address and pause packet match logic
- Register file for FIFOs and Ethernet MAC handler management
- Logic for support of WOL technology
- Link engine

The Ethernet MAC sub-block implements the following functions:

- Transmit MAC Handler (TXMAC)
- Receive MAC Handler (RXMAC)
- MII management function unit (STA)

These functions are described in the following sections.

22.2.1 MAL Slave Logic

The MAL slave (MALS) logic controls MAL transactions. MALS transfers TX and RX data between MAL and the OPB on one side, and the EMAC FIFO handlers on the other side. MALS is a dedicated MAL slave.

22.2.2 OPB Slave Logic

The OPB slave (OPBS) logic controls all OPB transactions between the processor core and the EMAC configuration and status registers.

22.2.3 FIFO Management Logic

The FIFO management logic is used for data interchange handling with the attached receive (RX FIFO) and transmit (TX FIFO modules.

22.2.4 Ethernet Address Match Logic

Address match logic checks the destination address of received packets against a set of predefined addresses specified by the current address filtering mode. EMAC contains one unicast (individual address) register, two hash tables for filtering individual and group address, and logic for detecting broadcast address (all ones). EMAC supports promiscuous mode and multicast promiscuous mode.

This logic also checks the destination address of the incoming packet against a special multicast address used for control (pause) packet recognition.

All checks for address matching are performed only after the entire destination address field is received (except for promiscuous and multicast promiscuous modes).

22.2.5 Configuration and Status Registers

Configuration and status registers define the EMAC configuration and reflect error/status of recent transmitted or received packets.

22.2.6 Wake On LAN Logic

EMAC supports Wake On LAN (WOL) technology, an industry standard described in the *Wired for Management (WFM)* specification. This technology allows a sleeping or powered-off network node to be awakened with a special packet called a Magic Packet. In the PPC440GP, with WOL mode enabled, the EMAC discards all incoming packets and does not request data from the MAL for transmission. When a magic packet is detected, an interrupt is generated on UIC1 (bit 29 for EMAC0 or bit 31 for EMAC1).

22.2.7 Ethernet MAC

The Ethernet MAC logic supports media independent interface (MII).

22.2.8 EMAC Loopback Modes

EMAC supports the external and internal loop-back modes illustrated in Figure 22-3.

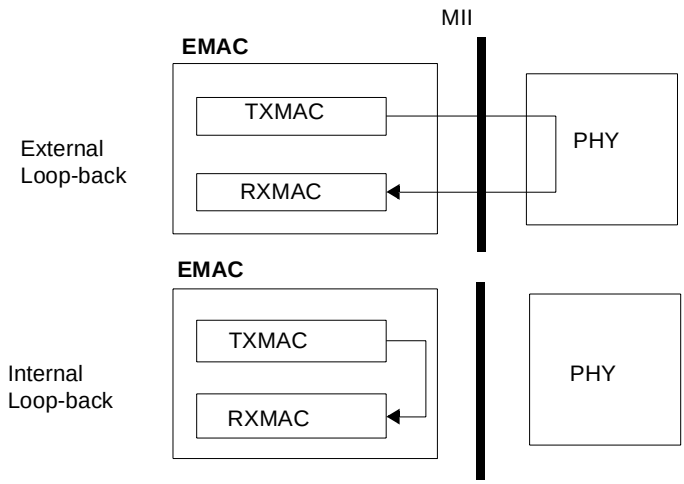


Figure 22-3. EMAC Loopback Modes

External loop-back mode uses the PHY device. To EMAC, external loop-back is identical to full-duplex operation. Configuring EMAC to operate in a full-duplex mode enables external loop-back. EMAC does not need an external loopback configuration signal.

In internal loopback mode, data from the EMAC transmit channel is routed to the EMAC receive channel. The loopback mode functions correctly with or without a connected PHY. In internal loopback mode, EMAC does not activate (monitor) any MII signals. Transmit channel signals are buffered internally to the receive channel. However, if internal loopback is used without a PHY, the EMAC transmit and receive clocks must be provided by another means. In internal loopback mode, the EMAC transmit clock and receive clock must be sourced from a single clock.

22.3 EMAC Transmit Operation

The transmit part of EMAC handles packet transmission from the MAL device to the ~~MII~~MII interface. At the end of a transmission process, EMAC provides a status/error word which allows monitoring the transmission operation.

EMAC implements dual MAL transmit channels (two transmit channels are allocated within MAL) to support efficient use of the transmit FIFO. Both channels share resources inside the EMAC. The transmit channels can be configured to independently request packets from MAL and drive them into the transmit FIFO, or to function as a single channel (in dependent mode).

22.3.1 Arbitration Between TX Channels

Because the transmit channels (referred to as TX Channel 0 and TX Channel 1) for each EMAC drive data into one FIFO, they cannot request packet data from MAL at the same time. MAL ensures that only one EMAC transmit channel for each EMAC is active at a given time.

22.3.2 Independent Mode

In independent mode, each EMAC transmit channel independently requests packets from MAL. Each channel can be configured to work in either single packet or multiple packet mode.

In single packet mode, EMACx_MR1[TR0] = 00 and EMACx_MR1[TR1] = 00. The channel requests one packet from MAL and resets EMACx_TMR0[GNP0, GNP1] as appropriate. The channel asks for service again only after EMACx_TMR0[GNP0] = 1 or EMACx_TMR0[GNP1] = 1 (set by the device driver).

In multiple packet mode, EMACx_MR1[TR0] = 01 and EMACx_MR1[TR1] = 01. After the channel finishes transferring a packet, the channel asks MAL for the next packet as soon as the other channel is in its Idle Phase and there is enough room in the FIFO. The channel continues to request more packets until one of the following events occur:

- The channel receives notification from MAL that the next buffer descriptor is not marked ready for transmission. When this occurs, the channel sets EMACx_TMR0[GNP0, GNP1] = 0, as appropriate, and waits for software to reactivate the channel by setting EMACx_TMR0[GNP0] = 1 or EMACx_TMR0[GNP1] = 1.
- A transmit error or signal quality error (SQE) occurs and the corresponding interrupt is not masked in the EMACx_ISER. After such an error, the channel sets EMACx_TMR0[GNP0, GNP1] = 0, as appropriate, and sets EMACx_ISR[DB0] = 1 or EMACx_ISR[DB1] = 1 (the EMACx_ISR field that is set depends on which channel is active) and the corresponding EMACx_ISR error. The channel does not request service again until EMACx_TMR0[GNP0] = 1 or EMACx_TMR0[GNP1] = 1 and EMACx_ISR[DB0] = 0 or EMACx_ISR[DB1] = 0 (again, depending on channel).

In independent mode, if both channels are configured to work in multiple packet mode and both EMACx_TMR0[GNP0] = 1 and EMACx_TMR0[GNP1] = 1 at the same time, the channels operate in a sequential repeating manner as long as no errors occur.

22.3.3 Dependent Mode

In dependent mode, EMACx_MR1[TR0] = 10 and EMACx_MR1[TR1] = 10. The two TX channels act as if they were one channel, sharing EMACx_TMR0[GNPD]. When EMACx_TMR0[GNPD] = 1, the channel specified by EMACx_TMR0[FC] starts requesting MAL service. Then, both channels continue to request packets from MAL in an alternating, sequential, repeating manner, until one of the following occurs:

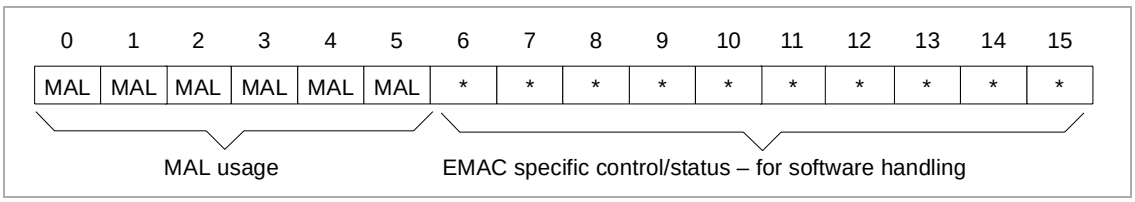
- One of the channels receives notification from MAL that the next buffer descriptor is not marked ready for transmission. When this occurs, EMAC clears EMACx_TMR0[GNPD]. At this point, neither channel requests a packet from MAL until EMACx_TMR0[GNPD] back to 1. The first channel to request a new packet from MAL when this occurs is the channel that received notification from MAL that no packets were ready to transmit (regardless of the setting of EMACx_TMR0[FC]). Further requests continue in an alternating, repeating manner.
- Either a transmit error or an SQE occurs on one of the channels and the corresponding interrupt is not masked in the EMACx_ISER. One of the following scenarios can occur.
  - If the other channel has not yet requested MAL service when the channel for which the error occurred receives notification from MAL that the transmit operation has completed, EMACx_TMR0[GNPD] is cleared (EMACx_TMR0[GNPD] = 0) and the EMACx_ISR[DBDM] is immediately set (EMAC_ISR[DBDM]=1).
  - If the other channel was receiving data from MAL, it initiates early termination. If a second packet was being transmitted on the media, it is stopped. In these cases, EMACx_TMR0[GNPD] is cleared (EMACx_TMR0[GNPD] = 0) and the EMACx_ISR[DBDM] = 1 only after notification from MAL that the transmit operation has completed has been received for the second channel.

At this point, neither channel activates a request to MAL until EMACx_TMR0[GNPD] = 1 and EMACx_ISR[DBDM] = 0. The channel specified by EMACx_TMR0[FC] is the first to request service from MAL. Subsequent requests continue in an alternating, sequential manner.

22.3.3.1 MAL TX Descriptor Control/Status Field

For each transmitted packet, MAL uses the descriptor control/status field of the buffer descriptor to provide an EMAC with control information (write), and to obtain packet status from the EMAC after transmission is complete (read). Software writes the control bits in the buffer descriptor before packet transmission, and reads the status bits from the buffer descriptor after packet transmission has completed. See ~~"Buffer Descriptor" on page 20-7~~*Buffer Descriptor* on page 687 for more information on the buffer descriptor structure.

Figure 22-4. MAL TX Descriptor Control/Status Field



Bits	Bit Name	Bit Description	Mode
0:5	MAL Usage	~~See "TX Status/Control Field Format" on page 20-14.~~See *TX Status/Control Field Format* on page 694	R
TX Control Information (Write Access)			
6	Generate FCS	0 FCS is not generated by EMAC. 1 EMAC calculates and adds the FCS field to the packet to be transmitted.	W
7	Generate padding	0 Padding is not generated by EMAC. 1 EMAC adds the padding field to the packet to be transmitted (only when Generate FCS is also set).	W
8	Insert source address	0 EMAC will not insert source address. 1 EMAC inserts the source address field into the packet to be transmitted using the content of the Individual Address High (EMACx_IAHR) and Individual Address Low (EMACx_IALR) Registers.	W
9	Replace source address	0 EMAC will not replace source address. 1 EMAC replaces the source address field in the packet to be transmitted using the content of the Individual Address High (EMACx_IAHR) and Individual Address Low (EMACx_IALR) Registers.	W
10	Insert VLAN Tag	0 EMAC will not insert a VLAN tag. 1 EMAC inserts the VLAN Tag field into the packet to be transmitted using the content of the VLAN TPID register (EMACx_VTPID).	W
11	Replace VLAN Tag	0 EMAC will not replace the VLAN tag. 1 EMAC replaces the VLAN Tag field in the packet to be transmitted using the content of the VLAN TPID register (EMACx_VTPID).	W
TX Status Information (Read Access)			
6	Bad FCS on transmitted frame	0 FCS was correct in the transmitted packet. 1 Indicates that a bad FCS was found in the transmitted packet.	R
7	Bad previous packet in dependent mode	0 Packet transmission OK. 1 Indicates that a Descriptor Error, Transmit Error, or SQE Error occurred in the previously transmitted frame. This bit will only be activated in dependent mode.	R
8	Loss of carrier sense	0 No loss of carrier. 1 During the transmission of a frame, the PHY_CRS input was de-asserted after it previously was asserted, or it was not asserted at all.	R
9	Excessive deferral	0 No excessive deferral. 1 Indicates that the current frame has been deferred for an excessive period of time. Applicable only in half duplex mode. The value of this period in bit times is calculated in the following ways: For 10/100 Mbps operation it is: 2 x (maxFrameSize x 8) bit times.	R

Bits	Bit Name	Bit Description	Mode
10	Excessive collisions	0 Less than 16 collisions. 1 Indicates that the current frame transmission had ended with a collision on the 16th consecutive attempt. Applicable only in half-duplex mode.	R
11	Late collision	0 No late collision. 1 Frame collided outside of the collision window. Applicable only in half-duplex mode.	R
12	Multiple collision	0 More than 1 but less than 16 collisions did not occur. 1 Transmitted frame collided more than once but less than 16 times. Applicable only in half-duplex mode.	R
13	Single collision	0 Single collision did not occur. 1 Activates if transmitted frame collided once. Applicable only in half-duplex mode.	R
14	Underrun	0 Underrun did not occur. 1 Frame transmission was aborted because of underrun; data from the Transmit FIFO was not valid in time to allow continuous data transmission on the MII.	R
15	SQE	0 Signal Quality Error did not occur. 1 Signal Quality Error test failed during packet transmission. Applicable only in half -duplex mode during 10 Mbps operation.	R

22.3.3.2 Early Packet Termination during Transmit

EMAC can initiate early packet termination during transmit, terminating packet transmission before MAL finishes transferring all packet data from memory to the EMAC transmit FIFO. Early packet information is typically used when error conditions force the EMAC to abort a transmission.

EMAC performs early termination on the MAL interface if any of the following conditions occur:

- Underrun in the transmit FIFO.
- Excessive collisions.
- Excessive deferral.
- Late collision.

22.3.3.3 Empty Packets

EMAC treats empty packets as if a normal packet had been written, but does not write data to the transmit FIFO. A status word of all 0s is returned after an empty packet. EMAC expects that for word-aligned packets, MAL activates the related word transfer indication during the last data transfer, rather than providing an empty packet indication.

22.3.3.4 Automatic Retransmission of Colliding Packets

EMAC automatically retransmits packets that collide on the ~~MII~~MII interface. The transmit FIFO always preserves the first 64 bytes of a packet until it receives an indication that the collision window has passed. Otherwise, if a collision was detected within the collision window, the packet is retransmitted without a new request from MAL.

22.3.3.5 Inter-Packet Gap (IPG) Tuning

EMAC supports user-programmable IPG length using the EMACx_IPGVR register, which contains one-third of the IPG value. Changing the contents of EMACx_IPGVR enables the user to adjust the fairness or aggressiveness of EMAC on the medium. Programming a lower number (but not less than four) causes EMAC becomes more aggressive on the media. This can result in EMAC capturing the network by forcing less aggressive nodes to defer. Programming a larger number of bit times causes EMAC to becomes less aggressive on the network; it might defer more often than normal. EMAC performance might decrease as the IPG period is increased from the default value, but the resulting behavior can improve media performance by reducing the occurrence of collisions.

22.3.3.6 Full-Duplex Operation

Full-duplex operation allows simultaneous transmit and receive activity on the ~~ZMII~~MII interface. Software can set EMACx_MR1[FDE] = 1 to enable full-duplex mode. During full-duplex operation, the following changes occur in EMAC functionality.

- Transmission is not deferred while receive is active.
- The IPG counter, which controls transmit deferral during the IPG between back-to-back transmits, is started when transmit activity for the first packet ends, instead of when transmit and carrier activity end.
- SQE test is not performed.
- Collision indication is ignored.

22.3.3.7 Packet Content Configuration Options

EMAC can modify the content of the packet coming from MAL before issuing it to the ~~MII~~MII interface. Figure

22-5 illustrates possible changes in the transmit packet format.

Packet as delivered to EMAC by MAL

DA	SA (optional)	Length/Type	Data	FCS (optional)

Packet as delivered by EMAC on the MII

Preamble	SPD	DA	SA	Length/Type	Data	Padding (if needed)	FCS

Preamble - combination of alternative 0s and 1s (7 bytes)
SPD - Start Of Packet delimiter (1 byte)
DA - Destination Address (6 bytes)
SA - Source Address (6 bytes)
Length/Type - length of data field / type definition (Ethernet) (2 bytes)
Data - data field including padding if needed in short packets
Padding (optional) - needed to pad the packet to the minimum packet size
FCS - Cycle Redundancy Check (4 bytes)

Figure 0-2. Transmit Packet Structure (Excluding VLAN Tagged and Control Packets)

Packet as delivered to EMAC by MAL

DA	SA (optional)	Length/Type	Data	FCS (optional)

Packet as delivered by EMAC on the MII

Preamble	SPD	DA	SA	Length/Type	Data	Padding (if needed)	FCS

Preamble - combination of alternative 0s and 1s (7 bytes)
SPD - Start Of Packet delimiter (1 byte)
DA - Destination Address (6 bytes)
SA - Source Address (6 bytes)
Length/Type - length of data field / type definition (Ethernet) (2 bytes)
Data - data field including padding if needed in short packets
Padding (optional) - needed to pad the packet to the minimum packet size
FCS - Cycle Redundancy Check (4 bytes)

Figure 0-1. Transmit Packet Structure (Excluding VLAN Tagged and Control Packets)
Figure 22-5. Transmit Packet Structure (Excluding VLAN Tagged and Control Packets)

The following options, unless mutually exclusive, can be set for each packet, and can be provided as a part of command write transactions from MAL (see ~~"MAL TX Descriptor Control/Status Field" on page 22-7~~*MAL TX Descriptor Control/Status Field* on page 735).

- Automatic data padding for short transmit packets

  EMAC pads the transmit packet with extra bytes between the data and the FCS field to reach a total length of 64 bytes (including FCS). This feature is supported only when the packet coming from the transmit FIFO does not contain the FCS. Automatic padding enables software to avoid sending padding as a part of the packet data field, and, therefore, reduces the amount of data transferred on the system bus during the short packet transmission.

- Source address insertion

  EMAC adds the source address (SA) field to the transmitted packet. EMAC uses the contents of the Individual Address High (EMACx_IAHR) and Individual Address Low (EMACx_IALR) registers for the source address value. *This option is mutually exclusive with source address replacement.*

- Source address replacement

  EMAC replaces the source address received from the transmit FIFO with the contents of EMACx_LSAH and EMACx_LSAL. *This option is mutually exclusive with source address insertion*.

- Add four FCS bytes

  EMAC calculates the FCS for the transmitted packet. The FCS is appended to data coming from the Transmit FIFO or padding bytes (if added).

- VLAN Tag insertion

  EMAC adds the content of the VLAN Tag field to the transmitted packet (see "VLAN Support" on page -747). EMAC uses the content of the VLAN TPID (EMACx_VTPID) and VLAN TCI (EMACx_VTCI) registers for the VLAN Tag value. This feature is supported only when the packet from the transmit FIFO does not contain an FCS. *This option is mutually exclusive with VLAN tag replacement*.

- VLAN Tag replacement.

  EMAC replaces the content of the VLAN Tag field in the transmitted packet. EMAC uses the contents of EMACx_VTPID and EMACx_VTCI for the VLAN Tag value. This feature is supported only when the packet from the transmit FIFO does not contain an FCS. *This option is mutually exclusive with VLAN tag insertion*.

~~Table 22-1~~*Table 22-1* summarizes the possible options for adding FCS and source address to the transmitted packet.

Table 22-1. FCS/SA Enable - Possible Configurations

Configuration Options			EMAC Action		
Generate FCS	Insert SA	Replace SA	Add FCS	Add SA	Replace SA
0	Don't care	Don't care	N	N	N
1	0	0	Y	N	N
1	0	1	Y	N	Y
1	1	0	Y	Y	N

~~Table 22-2~~*Table 22-2* summarizes the possible options for adding FCS and padding to the transmitted packet.

Table 22-2. FCS/Pad Enable - Possible Configurations

Configuration Options		EMAC Action	
Generate FCS	Generate Pad	Add FCS	Add Pad
0	Don't care	N	N
1	0	Y	N
1	1	Y	Y

~~Table 22-3~~*Table 22-3* summarizes the possible options for adding FCS and VLAN Tag to the transmitted packet.

Table 22-3. FCS/VLAN Tag Enable - Possible Configurations

Configuration Options			EMAC Action		
Generate FCS	Insert VLAN Tag	Replace VLAN Tag	Add FCS	Insert VLAN Tag	Replace VLAN Tag
0	Don't care	Don't care	N	N	N
1	0	0	Y	N	N
1	0	1	Y	N	Y
1	1	0	Y	Y	N

22.4 EMAC Receive Operation

The receive part of EMAC is responsible for receiving packets coming from the physical layer (PHY) device (via the MII) and forwarding them to the receive channel of the attached MAL. At the end of the reception process, EMAC provides a status/error word that enables software to monitor the receive operation.

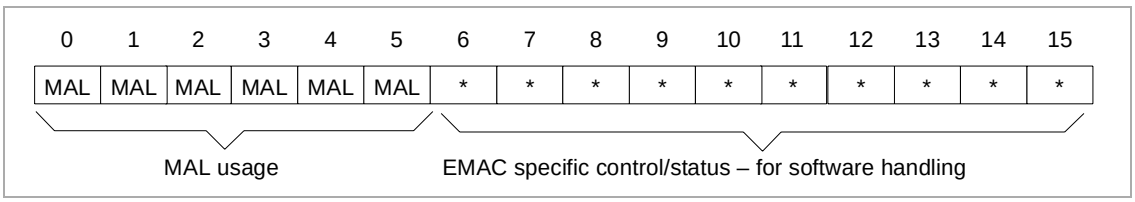
22.4.1 EMAC – MAL RX Packet Transfer Flow

EMAC initiates request for service from MAL when the number of occupied entries in the receive FIFO reaches the low water mark specified in EMACx_RWMR[RLWM].

22.4.2 MAL RX Descriptor Status

For each packet that is received, MAL obtains status from EMAC after reception is complete, and writes this information into the buffer descriptor status/control field. Software uses this information to monitor the status of received packets. See ~~"Buffer Descriptor" on page 20-7~~*Buffer Descriptor* on page 687 for more information on the buffer descriptor structure.

Figure 22-6. MAL RX Descriptor Control/Status Field



Bits	Bit Name	Bit Description	Mode
0:5	MAL Usage	See *RX Status/Control Field Format* on page 696.	R
RX Status Information (Read Access)			
6	Overrun Error	0 No overrun error. 1 EMAC detected an overrun error. An overrun error occurs if the flow of received data to the RX FIFO is corrupted because of insufficient empty space.	R
7	Pause Packet	0 Received packet is not a control pause packet. 1 Received packet is a control pause packet.	R
8	Bad Packet	0 No packet errors. 1 Early termination caused by packet error.	R
9	Runt Packet	0 Duration of PHY_RX_DV signal OK. 1 Duration of PHY_RX_DV signal greater than ShortEventMax Time constant and less than collision windows.	R
10	Short Event	0 Duration of PHY_RX_DV signal OK. 1 Duration of PHY_RX_DV signal was less than ShortEventMaxTime constant	R
11	Alignment Error	0 Received packet length OK. 1 Received packet length not an integral number of octets.	R
12	Bad FCS	0 FCS OK. 1 The FCS value does not match the FCS value calculated by EMAC.	R

Bits	Bit Name	Bit Description	Mode
13	Packet Too Long	0 Received packet length OK. 1 Received packet length exceeds maximum packet length. • 1518 octets for standard packet 1522 octets for VLAN tagged packetData following the maximum packet length is not transferred to MAL	R
14	Out of Range Error	0 Received packet length field value OK. 1 Received packet length field value greater than maximum allowed LLC data size. The maximum allowed logical link control (LLC) data size is greater than 1500 and less than 1536.	R
15	In Range Error	Refer to ~~Table 22-4~~ *Table 22-4* for a description of conditions for activating this status bit.	R

Table 22-4. In Range Length Error Behavior for Various Packet Lengths

Programmed Length (Bytes)	Actual Length	EMAC Action
Less than 46 (42 if VLAN Tagged packet)	Differs from 46 (42 in case of VLAN Tagged packet)	In range length error is activated
Less than 46 (42 if VLAN Tagged packet)	46 (42 in case of VLAN Tagged packet)	In range length error is not activated
Greater than or equal to 46 (42 if VLAN Tagged packet) and less than or equal to 1500	Equals the length field value	In range length error is not activated
Greater than or equal to 46 (42 if VLAN Tagged packet) and less than or equal to 1500	Differs from the length field value	In range length error is activated

22.4.3 Early Packet Termination during Receive

Early packet termination occurs when packet reception is aborted by EMAC before the packet data transfer to MAL is completed.

EMAC performs early termination in the following cases.

- An overrun occurs in the receive FIFO
- Packet is too long and the Receive Oversize Packet option is not enabled (EMACx_RMR[ROP] = 0)

22.4.4 Discarding Packets During Receive

Receive packets can be discarded if certain error conditions are detected. EMAC behavior depends on whether the packet to be discarded is already being output to MAL. If the packet containing the error is not being provided to MAL when the discard condition is detected, the packet is flushed from the Receive FIFO. In this case, EMAC does not provide status information to MAL. If the packet containing the error is already being output to MAL, EMAC initiates an early packet termination procedure, as described in ~~"Early Packet Termination during Receive."~~ *Early Packet Termination during Receive* on page 742

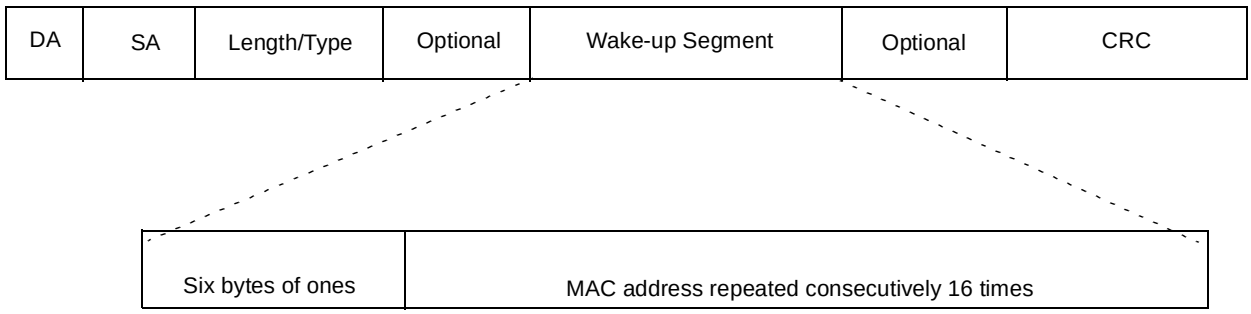
Each receive discard condition can be individually controlled using appropriate settings, as described in ~~"Receive Mode Register (EMACx_RMR)" on page 22-30~~ *Receive Mode Register (EMACx_RMR)* on page 759.

22.4.5 Wakeup On Lan (WOL) Support

WOL logic in EMAC supports WOL technology, an industry standard in the Wired for Management (WFM) Specification. WOL remotely awakens sleeping or powered-off nodes on a network. To wake up a node, a specific packet, called a *magic packet*, is sent to the network node. When so configured, EMAC monitors the incoming bit stream for magic packets.

The magic packet, also called a wake-up packet, contains a unique data field not normally expected in LAN traffic. When a WOL-enabled adapter on a powered-off client decodes this data field, a wake-up signal is generated. In the PPC440GP, with WOL mode enabled, EMAC discards all incoming packets and does not request data from the MAL for transmission. When a magic packet is detected, an interrupt is generated on UIC1 (bit 29 for EMAC0 or bit 31 for EMAC1).

Figure 22-7 shows the wake-up packet format. The key to the wake-up packet is the MAC address of the target client, which is repeated 16 times. This pattern of 16 addresses in the data field is not expected to occur in any packet except the wake-up packet.



DA - destination address (6bytes) - UAA or Broadcast address
SA - source address (6 bytes)
Length/Type - length of data field (802.3)/type definition (Ethernet) (2 bytes)
optional - for example, IP header
CRC - Cycle Redundancy Check (4 bytes)

Figure 22-7. Wake-Up Packet Format

The destination address can be a specific address, called the universally administered address (UAA), or a broadcast address. If the destination address is a UAA, the wake-up packet is sent only to the client at that address. However, because the client is powered off and is no longer transmitting, some protocols remove the client MAC address from routing tables and internal caches at other nodes. In this case, wake-up packets addressed to a target client are discarded because nodes and routers do not know where to send them. The solution to this problem is to use a broadcast address. A directed broadcast has a valid network address and a broadcast host address. Network routers and nodes forward directed broadcasts to the appropriate network, where it is seen as a MAC-level broadcast and detected by the powered-off client.

22.4.5.1 EMAC WOL Support

EMAC enters WOL mode when EMACx_MR0[WKE] = 1.

WOL mode should only be changed while EMACx_MR0[RXI] = 1 and EMACx_MR0[RXE] = 0. After EMACx_MR0[WKE] = 1, EMACx_MR0[RXE] can be set to 1.

A reset (soft or hard) should be issued before programming EMAC to WOL mode. In to WOL mode, EMAC does not propagate any received packets to MAL. Also, EMAC transmit channels do not request data from MAL.

22.5 Flow Control

For efficient system performance, each EMAC implements full-duplex flow control by handling specific MAC control packets contained in the pause opcode. EMAC supports flow control as defined in the IEEE 802.3x-1997 standard.

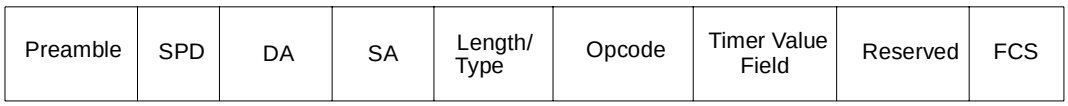
22.5.1 MAC Control Packet

The flow control mechanism enables receive FIFO control logic to automatically notify the node transmitting packets to suspend transmission for a defined period of time. The pause control packet has a fixed length defined as follows:

MinFrameSize – 32 bits (60 bytes)

MAC control packets have a unique value of 0x8808 in the length/type field, and share the same packet format as normal Ethernet packets, except that the data field consists of an opcode field and a parameter field. The opcode field contains an opcode command and the parameter field contains a value associated with the opcode command (0x0001). The only opcode command defined by IEEE 802.3x is the pause opcode; the parameter field for the pause opcode defines the pause time. MAC control packets containing a pause opcode, also called pause packets, can have a destination address equal to a reserved multicast address, or can be the address of the receive station itself. The reserved multicast address is 0x0180C2000001.

Figure 22-8 illustrates the control packet format.



Preamble - Alternating zeroes and ones (7 bytes)
SPD (Start Of Packet Delimiter), 1 byte
DA (Destination Address) = 0x0180C2000001
SA (Source Address) = Universally Administered Address (UAA)
Length/Type = 0x8808
Opcode Field = 0x0001
Timer Value Field = PauseValue
Reserved = zeroes (42 bytes)
FCS = calculated FCS

Figure 22-8. Control Packet Format

The timer value field contains the value of the delay interval in resolution of *pause_quanta*, defined in IEEE 802.3x as follows: "MAC Control Parameter[s] (pause_time) is a 2-octet, unsigned integer containing the length of time for which the receiving station is requested to inhibit data packet transmission. The field is

transmitted most-significant octet first, and least-significant octet second. The pause_time is measured in units of pause_quanta, equal to 512 bit times. The range of possible pause_time is 0 to 65535 pause_quanta."

22.5.2 Control Packet Transmission

Two options initiate the pause packet transmission from EMAC. The transmitted pause packet forces the node, with the destination address specified, to temporarily suspend the transmission of packets to EMAC.

- Software initiated

  The packet transferred to EMAC by MAL for transmission is a pause packet created by software. EMAC transmits this as a normal packet.

- Automatic flow control initiated

  The EMAC integrated flow control mechanism detects the need for and then transmits a control (pause) packet automatically. When building the control packet, EMAC obtains the SA (source address) field from EMACx_IAHR and EMACx_IALR, and the timer value from the Pause Timer Register (EMACx_PTR[TVF]). The contents of the other fields in the packet are shown in Figure 22-8.

22.5.3 Integrated Flow Control

To enable integrated flow control in full-duplex mode, set EMACx_MR1[EIFC] = 1. When the receive FIFO reaches a predefined threshold level (called a high water mark and specified by EMACx_RWMR[RHWM]), an internal request for control (pause) packet transmission is activated. EMAC sends a control (pause) packet when a new packet enters the receive FIFO and the number of vacant entries in the Receive FIFO is less than the high water mark. When the receive FIFO reaches another predefined threshold level (the low water

mark, specified by EMACx_RWMR[RLWM]), a new internal request for a pause packet transmission, with a pause timer value of 0, is activated. EMAC sends a pause packet, with a pause timer value of 0, only once, and only if a pause packet with a non-zero value in the pause timer was transmitted earlier.

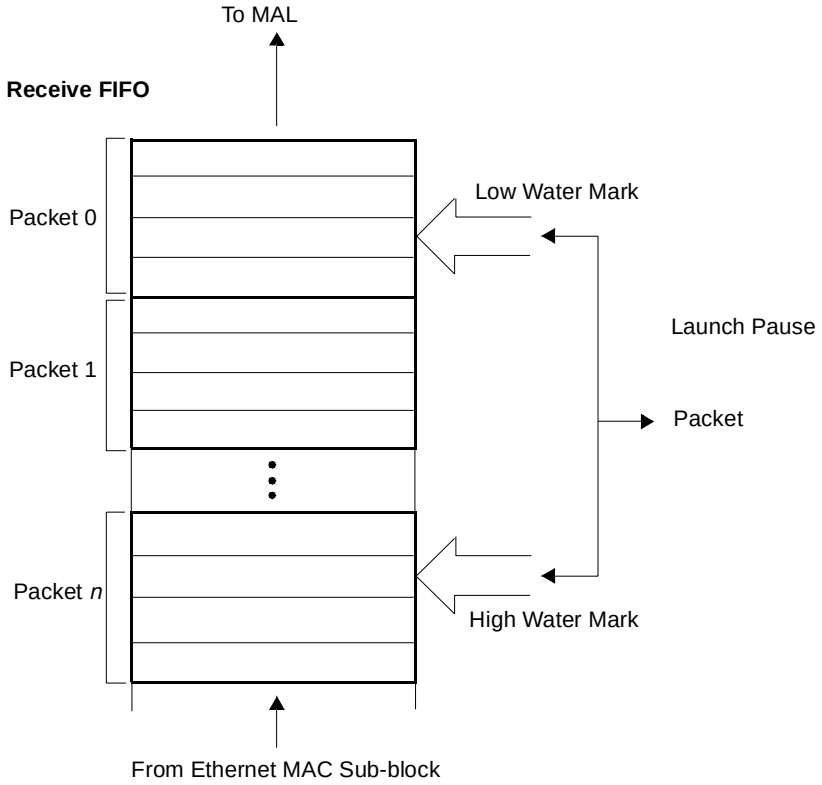


Figure 22-9. Integrated Flow Control Mechanism

22.5.4 Control Packet Reception

In the receive path, the EMAC can be configured to respond to pause packets, or ignore them, as specified by EMACx_MR1[APP]. When response to pause packets is enabled and EMAC detects a valid MAC control packet with a pause opcode, the EMAC stores the value of the timer value field. The received packet is considered a valid control packet only if no error was detected during the packet reception. If, at the end of packet reception, the packet is considered valid, EMAC launches a pause operation state machine, as specified in the IEEE P802.3x standard. Figure 22-10 illustrates the pause operation state machine.

If a control (pause) packet is received while another packet is transmitted, the ongoing transmission process is completed and the transmitter is paused. If other packets are in the transmit FIFO, their transmission is delayed until the pause timer expires. EMAC normally does not pass the MAC control packets to MAL unless EMACx_RMR[PPP] = 1.

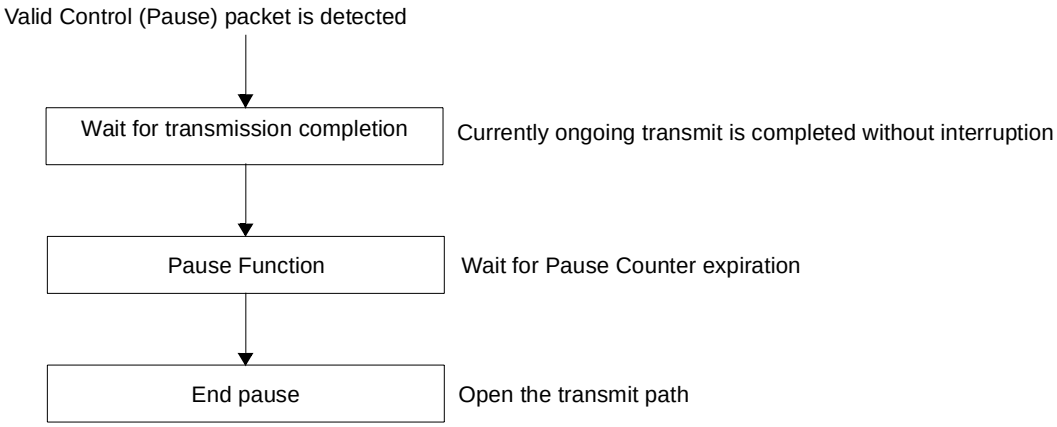


Figure 22-10. Pause Operation State Machine

In the Pause Function state, EMAC decrements its internal pause timer, which was set to the timer value field of the received control packet.

Note 1: The transmission of control (pause) packets is not affected by the reception of a receive control (pause) packet. Received control (pause) packets inhibit only the transmission of regular packets from the Transmit FIFO.

Note 2: Receipt of a new valid control (pause) packet causes the pause timer of EMAC to be reloaded with the contents of the timer value field of the recently received packet, regardless of the current pause timer setting. This indicates new pause operations take precedence over earlier pause operations.

22.6 VLAN Support

EMAC can handle VLAN tagged packets, as specified in IEEE Draft P802.3ac/D1.0a standard when EMACx_MR1[VLE] = 1.

A tagged MAC frame is an extension of the standard MAC packet. The extension for VLAN tag support consists of a 4-octet VLAN tag inserted between the end of the source address and the beginning of the length/type fields of the MAC packet.

The VLAN tag consists of two fields:

- A 2-octet constant Type field value equal to the VLAN Tag Protocol Identifier (0x8100)
- A 2-octet field containing Tag Control Information (TCI)

The MAC client data and FCS fields of the basic MAC packet follow the VLAN tag. The length of the packet is extended by four octets by the VLAN tag (up to 1522 bytes). The FCS is calculated over all fields from the destination address through the end of the MAC client data or pad (if present); that is, all fields except the preamble, SPD, and FCS.

Figure 22-11 illustrates the Tagged MAC Frame format.

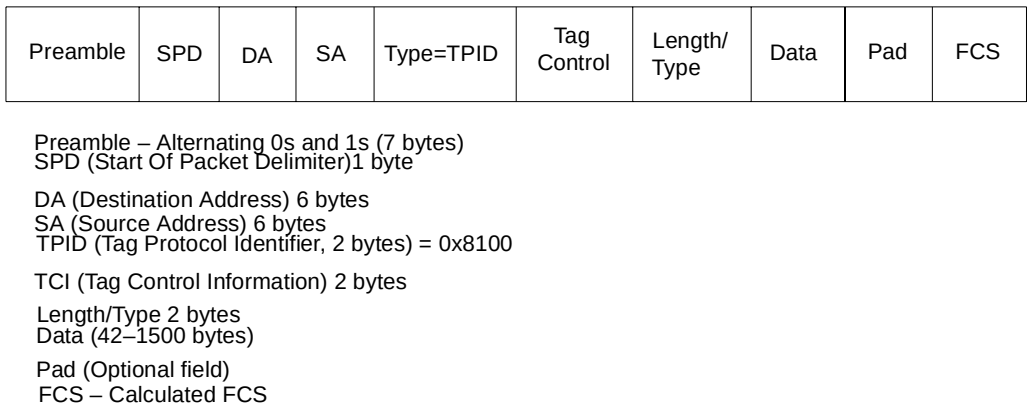


Figure 22-11. Tagged MAC Packet Format

Figure 22-12 illustrates the structure of the TCI field.

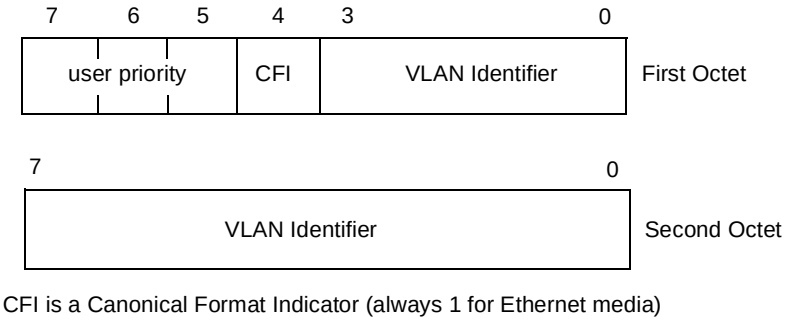


Figure 22-12. Tag Control Information Field Structure

22.6.1 VLAN Tagged Packet Transmission

When EMACx_MR1[VLE] = 1, the following configuration options are available, depending on the content of the appropriate bits in the MAL control word (See *MAL TX Descriptor Control/Status Field* on page 735.)

- The Generate FCS bit (bit 6) is not set or both Insert VLAN Tag and Replace VLAN Tag bits (bits 10 and 11, respectively) are not set: EMAC transmits the packet without any changes
- Bit 6 is set and bit 10 is also set: EMAC will insert TPID and Tag control information for the transmitting packet using the content of EMACx_VTPID and EMACx_VTCI, respectively
- Bit 6 is set and bit 11 is also set: EMAC will replace TPID and Tag control information for the transmitting packet using the content of EMACx_VTPID and EMACx_VTCI, respectively

22.6.2 VLAN Tagged Packet Reception

If EMACx_MR1[VLE] = 1, the EMAC parses the VLAN Tag unique type/length in the incoming packet during the receive process. If the VLAN Tag is equal to the value stored in EMACx_VTPID, EMAC continues the receive process and allows the received packet to contain up to 1522 octets. Otherwise, the receive process is continued unless the length is greater than 1518 bytes.

22.6.3 Address Match Mechanism

The address match (or filtering) mechanism is a hardware aid that reduces the average amount of CPU cycles required to determine whether an incoming packet should be accepted.

EMAC uses various address filters for incoming packets by using the following address recognition modes.

- Individual mode (also referred to as physical)
- Multicast mode (also referred to as group)
- Broadcast mode (an all-ones group address)
- Promiscuous mode
- Promiscuous multicast mode
- WOL mode

A flowchart for address recognition of received packets is shown in ~~Figure 22-13 on page 22-22~~*Figure 22-13* on page 751. If the least significant bit (LSb) of the first byte of the destination address (DA) is 0, the packet is considered individual. If the first bit received is 1, the packet is considered multicast. When the DA field contains all 1s, the packet is broadcast, a special type of multicast.

22.6.3.1 Non-WOL Mode

When EMAC operates in single individual mode (EMACx_RMR[IAE] = 1), the DA of the received packet is compared to the physical address stored in EMACx_IAHR and EMACx_IALR.

When EMAC operates in multiple individual mode (EMACx_RMR[MIAE] = 1), EMAC performs a calculation on the contents of the DA field (logical address filter) to determine whether or not to accept the packet.

When EMAC operates in promiscuous mode (EMACx_RMR[PME] = 1), all properly formed packets are received, regardless of the content of the DA field.

When EMAC operates in multicast promiscuous mode (EMACx_RMR[PMME] = 1), all multicast packets are received, regardless of the content of the DA field.

When EMAC operates in broadcast address mode (EMACx_RMR[BAE] = 1), EMAC performs an address compare on received packets with broadcast addresses.

When EMAC operates in multicast address mode (EMACx_RMR[MAE] = 1), EMAC performs a calculation on the contents of the DA field (logical address filter) as in multiple individual mode, in order to determine whether or not the packet should be accepted.

The logical address filter hardware implements a hash code searching technique commonly used by programmers. The hardware maps the DA of the incoming packet into one of 64 categories corresponding to 64 bits stored in the EMACx_IAHT1–EMACx_IAHT4 or EMACx_GAHT1–EMACx_GAHT4 registers. The

hardware accepts or rejects the packet, depending on the state of the corresponding bit in the EMACx_IAHT1–EMACx_IAHT4 or EMACx_GAHT1–EMACx_GAHT4 registers corresponding to the selected category.

Figure 22-14 on page 22-23*Figure 22-14* on page 752 shows the details of the hardware mapping algorithm. The example depicts multiple individual address mode, but with changes can be used for the multicast address mode.

If the most significant bit (MSb) of an incoming address is 0, the address is individual and is passed to the individual address filter. If the MSb of an incoming address is 1, the address is multicast and is passed to the multicast address filter. The individual/multicast address filter is a 64-bit mask composed of the EMACx_IAHT1–EMACx_IAHT4 or EMACx_GAHT1–EMACx_GAHT4 registers (each register contains 16 bits of a 64-bit mask). The incoming address is sent through the FCS circuit. After the 48 address bits have gone through the FCS circuit, the high-order 6 bits of the resulting FCS (32-bit CRC) are used to select a the 64-bit positions in the individual/multicast address filter. If the selected filter bit is 1, the address is accepted.

Note: The individual/multicast address filter ensures only that there is a possibility that the incoming packet belongs to this node. To determine if the packet belongs to the node, the incoming individual/multicast address propagated to the main memory is compared by software to the list of logical addresses to be accepted by this node.

For software, the task of mapping an individual/multicast address to one of 64 bit positions requires a program that uses the same CRC algorithm to calculate the hash.

22.6.3.2 WOL Mode

In WOL mode (EMACx_MR0[WKE] = 1), EMAC operates only with the broadcast or individual address in the destination address field.

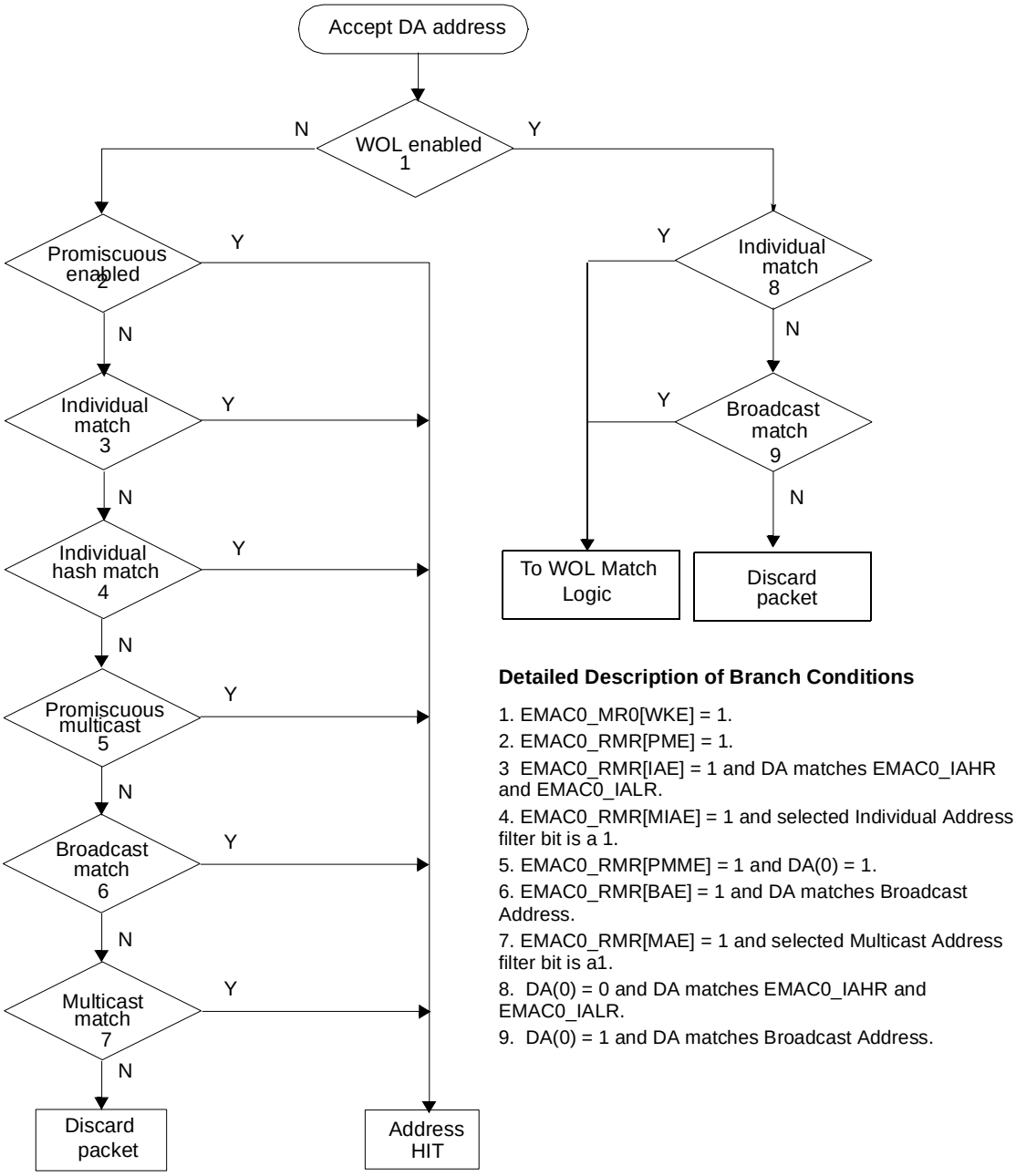


Figure 22-13. Receive Address Recognition Flowchart

The example in Figure 22-14 shows that the received individual address maps into category 24, and that bit 8 in EMACx_IAHT2 is set. The match indication is activated and the packet should be accepted.

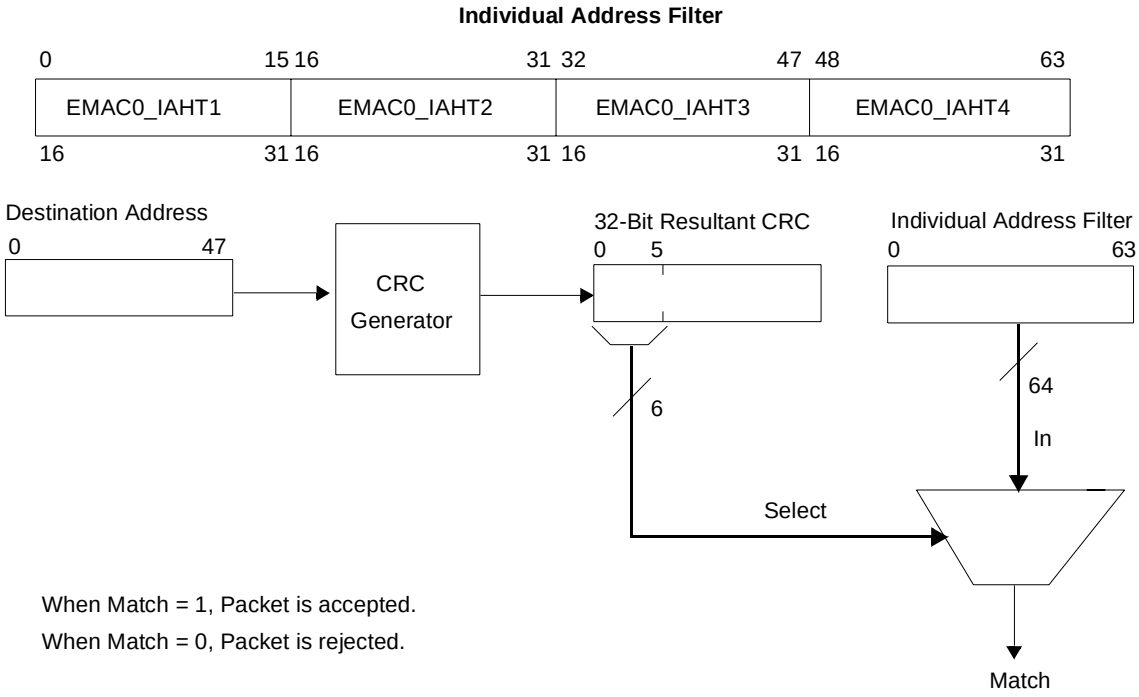


Figure 22-14. Ethernet Address Filter Operation

22.7 EMAC Registers

This section describes the EMAC registers, which are listed in Table 22-5 for EMAC0 and in Table 22-6 for EMAC1. In the register descriptions, the registers are prefixed "EMACx" to denote the identical register implementations in each EMAC. The EMAC registers are accessed using the OPB slave interface. Access to these memory-mapped I/O (MMIO) registers should be word-aligned.

Note: It is required to perform a soft reset of EMAC before initialization as shown in EMACx Mode Register 0 (EMACx_MR0[SRST]).

Table 22-5. ~~EMAC1~~ EMAC0 Register Summary

Register	Address	Write Access	Power-on Reset Value	Access	Page
~~EMAC1_MR0~~EMAC0_MR0	0x1 ~~40000900~~40000800	See description in ~~"Scenario 1" on page 22-46~~"Scenario 1" on page 22-46 *Scenario 1* on page 774	0xC0000000	R/W	~~22-282~~ 2-28756
~~EMAC1_MR1~~EMAC0_MR1	0x1 ~~40000904~~40000804	Reset	0x00000000	R/W	~~22-262~~ 2-26755
~~EMAC1_TMR0~~EMAC0_TMR0	0x1 ~~40000908~~40000808	See description on Page 756	0x00000000	R/W	~~22-282~~ 2-28756
~~EMAC1_TMR1~~EMAC0_TMR1	0x1 ~~4000090C~~40000 80C	See description on Page 758	0x380F0000	R/W	~~22-292~~ 2-29758
~~EMAC1_RMR~~ EMAC0_RMR	0x1 ~~40000910~~40000810	Reset	0x00000000	R/W	~~22-302~~ 2-30759
~~EMAC1_ISR~~EMAC0_ISR	0x1 ~~40000914~~40000814	Always	0x00000000	R/W	~~22-312~~ 2-31760
~~EMAC1_ISER~~ EMAC0_ISER	0x1 ~~40000918~~40000818	Reset	0x00000000	R/W	~~22-342~~ 2-34762
~~EMAC1_IAHR~~EMAC0_IAHR	0x1 ~~4000091C~~40000 81C	Reset, R, T	0x00000000	R/W	~~22-362~~ 2-36764
~~EMAC1_IALR~~EMAC0_IALR	0x1 ~~40000920~~40000820	Reset, R, T	0x00000000	R/W	~~22-362~~ 2-36765
~~EMAC1_VTPID~~EMAC0_VTPID	0x1 ~~40000924~~40000824	Reset, R, T	0x00008808	R/W	~~22-372~~ 2-37765
~~EMAC1_VTCI~~EMAC0_VTCI	0x1 ~~40000928~~40000828	Reset, R, T	0x00000000	R/W	~~22-372~~ 2-37766
~~EMAC1_PTR~~EMAC0_PTR	0x1 ~~4000092C~~40000 82C	Reset, T	0x0000FFFF	R/W	~~22-382~~ 2-38766
~~EMAC1_IAHT1~~EMAC0_IAHT1	0x1 ~~40000930~~40000830	Reset, R	0x00000000	R/W	~~22-382~~ 2-38767
~~EMAC1_IAHT2~~EMAC0_IAHT2	0x1 ~~40000934~~40000834	Reset, R	0x00000000	R/W	~~22-382~~ 2-38767
~~EMAC1_IAHT3~~EMAC0_IAHT3	0x1 ~~40000938~~40000838	Reset, R	0x00000000	R/W	~~22-382~~ 2-38767

Note: See ~~"Reset and Initialization" on page 22-45~~"Reset and Initialization" on page 22-45 *Reset and Initialization* on page 773 for definitions of letters in the Write Access column.

Table 22-5. ~~EMAC1~~ EMAC0 Register Summary (continued)

Register	Address	Write Access	Power-on Reset Value	Access	Page
~~EMAC1_IAHT4~~EMAC0_IAHT4	0x1~~4000093C~~40000 83C	Reset, R	0x00000000	R/W	~~22-382~~ 2-3876 7
~~EMAC1_GAHT1~~EMAC0_GAHT1	0x1~~40000940~~40000 840	Reset, R	0x00000000	R/W	~~22-392~~ 2-3976 7
~~EMAC1_GAHT2~~EMAC0_GAHT2	0x1~~40000944~~40000 844	Reset, R	0x00000000	R/W	~~22-392~~ 2-3976 7
~~EMAC1_GAHT3~~EMAC0_GAHT3	0x1~~40000948~~40000 848	Reset, R	0x00000000	R/W	~~22-392~~ 2-3976 7
~~EMAC1_GAHT4~~EMAC0_GAHT4	0x1~~4000094C~~40000 84C	Reset, R	0x00000000	R/W	~~22-392~~ 2-3976 7
~~EMAC1_LSAH~~EMAC0_LSAH	0x1~~40000950~~40000 850	Not applicable	0x00000000	R	~~22-392~~ 2-3976 7
~~EMAC1_LSAL~~EMAC0_LSAL	0x1~~40000954~~40000 854	Not applicable	0x00000000	R	~~22-392~~ 2-3976 8
~~EMAC1_IPGVR~~EMAC0_IPGVR	0x1~~40000958~~40000 858	Reset, T	0x00000004	R/W	~~22-402~~ 2-4076 8
~~EMAC1_STACR~~EMAC0_STACR	0x1~~4000095C~~40000 85C	See description on Page 769	0x00008000	R/W	~~22-402~~ 2-4076 9
~~EMAC1_TRTR~~EMAC0_TRTR	0x1~~40000960~~40000 860	See description on Page 770	0x00000000	R/W	~~22-412~~ 2-4177 0
~~EMAC1_RWMR~~EMAC0_RWMR	0x1~~40000964~~40000 864	Reset	0x04001000	R/W	~~22-422~~ 2-4277 0
~~EMAC1_OCTX~~EMAC0_OCTX	0x1~~40000968~~40000 868	Not applicable	0x00000000	R	~~22-432~~ 2-4377 1
~~EMAC1_OCRX~~EMAC0_OCRX	0x1~~4000096C~~40000 86C	Not applicable	0x00000000	R	~~22-432~~ 2-4377 2

Note: See ~~"Reset and Initialization" on page 22-45~~"Reset and Initialization" on page 22-45 *Reset and Initialization* on page 773 for definitions of letters in the Write Access column.

Table 22-6. EMAC1 Register Summary

Register	Address	Write Access	Power-on Reset Value	Access	Page
EMAC1_MR0	0x1 40000900	See description in ~~"Scenario 1" on page 22-46~~*Scenario 1* on page 774	0xC0000000	R/W	~~22-287~~ 756
EMAC1_MR1	0x1 40000904	Reset	0x00000000	R/W	~~22-267~~ 755
EMAC1_TMR0	0x1 40000908	See description on Page 756	0x00000000	R/W	~~22-287~~ 756
EMAC1_TMR1	0x1 4000090C	See description on Page 758	0x380F0000	R/W	~~22-297~~ 758
EMAC1_RMR	0x1 40000910	Reset	0x00000000	R/W	~~22-307~~ 759
EMAC1_ISR	0x1 40000914	Always	0x00000000	R/W	~~22-317~~ 760
EMAC1_ISER	0x1 40000918	Reset	0x00000000	R/W	~~22-347~~ 762
EMAC1_IAHR	0x1 4000091C	Reset, R, T	0x00000000	R/W	~~22-367~~ 764
EMAC1_IALR	0x1 40000920	Reset, R, T	0x00000000	R/W	~~22-367~~ 765
EMAC1_VTPID	0x1 40000924	Reset, R, T	0x00008808	R/W	~~22-377~~ 765
EMAC1_VTCI	0x1 40000928	Reset, R, T	0x00000000	R/W	~~22-377~~ 766
EMAC1_PTR	0x1 4000092C	Reset, T	0x0000FFFF	R/W	~~22-387~~ 766
EMAC1_IAHT1	0x1 40000930	Reset, R	0x00000000	R/W	~~22-387~~ 767
EMAC1_IAHT2	0x1 40000934	Reset, R	0x00000000	R/W	~~22-387~~ 767
EMAC1_IAHT3	0x1 40000938	Reset, R	0x00000000	R/W	~~22-387~~ 767
EMAC1_IAHT4	0x1 4000093C	Reset, R	0x00000000	R/W	~~22-387~~ 767
EMAC1_GAHT1	0x1 40000940	Reset, R	0x00000000	R/W	~~22-397~~ 767
EMAC1_GAHT2	0x1 40000944	Reset, R	0x00000000	R/W	~~22-397~~ 767
EMAC1_GAHT3	0x1 40000948	Reset, R	0x00000000	R/W	~~22-397~~ 767
EMAC1_GAHT4	0x1 4000094C	Reset, R	0x00000000	R/W	~~22-397~~ 767
EMAC1_LSAH	0x1 40000950	Not applicable	0x00000000	R	~~22-397~~ 767
EMAC1_LSAL	0x1 40000954	Not applicable	0x00000000	R	~~22-397~~ 768

Note: See ~~"Reset and Initialization" on page 22-45~~*Reset and Initialization* on page 773 for definitions of letters in the Write Access column.

Table 22-6. EMAC1 Register Summary (continued)

Register	Address	Write Access	Power-on Reset Value	Access	Page
EMAC1_IPGVR	0x1 40000958	Reset, T	0x00000004	R/W	~~22-40~~768
EMAC1_STACR	0x1 4000095C	See description on Page 769	0x00008000	R/W	~~22-40~~769
EMAC1_TRTR	0x1 40000960	See description on Page 770	0x00000000	R/W	~~22-41~~770
EMAC1_RWMR	0x1 40000964	Reset	0x04001000	R/W	~~22-42~~770
EMAC1_OCTX	0x1 40000968	Not applicable	0x00000000	R	~~22-43~~771
EMAC1_OCRX	0x1 4000096C	Not applicable	0x00000000	R	~~22-43~~772

Note: See ~~"Reset and Initialization" on page 22-45~~*Reset and Initialization* on page 773 for definitions of letters in the Write Access column.

22.7.1 Mode Register 0 (EMACx_MR0)

EMACx_MR0 defines the operating modes of the EMAC, which can be changed at any time during EMAC operation.

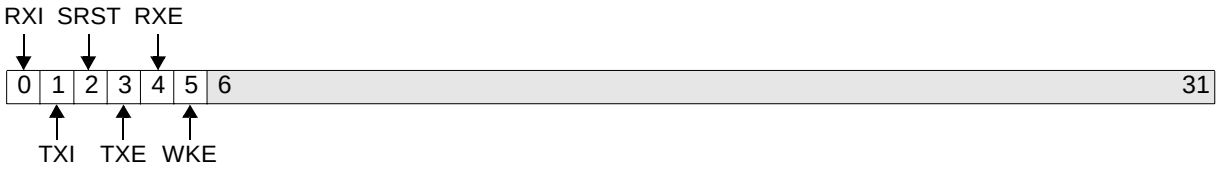


Figure 0-3. Mode Register 0 (EMACx_MR0)

0	RXI	Receive MAC Idle 0 RX MAC processing packet 1 RX MAC idle; RX packet processing complete	Read-only
1	TXI	Transmit MAC Idle 0 TX MAC processing packet 1 TX MAC idle; TX packet processing complete	Read-only

2	SRST	EMAC Software Reset 0 EMAC reset is complete 1 Reset the EMAC	Generates a general reset to EMAC through a software command. After setting this bit, EMAC hardware (registers, interface and internal state machines) returns to the power-on reset value. The software writes 1 to this bit in order to drive EMAC to the reset state. The bit is cleared by the hardware when the reset is completed. If EMACx_MR0[SRST] = 1, writing to any EMAC register, and reading any other bit in this register, is not supported.
3	TXE	Transmit MAC Enable 0 TX MAC is disabled 1 TX MAC is enabled	
4	RXE	Receive MAC Enable 0 RX MAC is disabled 1 RX MAC is enabled	
5	WKE	Wake-Up Enable 0 Incoming packets are not examined for wake-up packet 1 Examine incoming packets for wake-up packet	Software can change EMACx_MR0[WKE] only while EMACx_MR0[RXI] = 1 and EMACx_MR0[RXE] = 0.
6:31		Reserved	

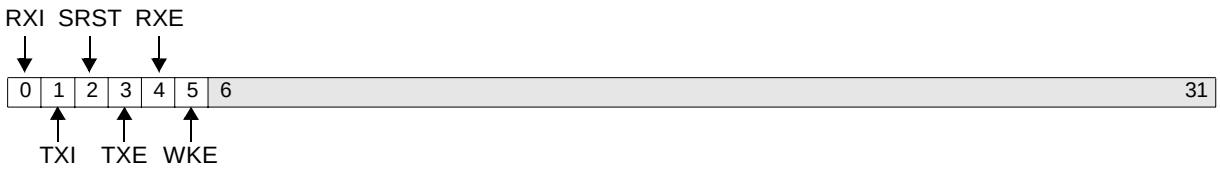


Figure 22-15. Mode Register 0 (EMACx_MR0)

0	RXI	Receive MAC Idle 0 RX MAC processing packet 1 RX MAC idle; RX packet processing complete	Read-only
1	TXI	Transmit MAC Idle 0 TX MAC processing packet 1 TX MAC idle; TX packet processing complete	Read-only
2	SRST	EMAC Software Reset 0 EMAC reset is complete 1 Reset the EMAC	Generates a general reset to EMAC through a software command. After setting this bit, EMAC hardware (registers, interface and internal state machines) returns to the power-on reset value. The software writes 1 to this bit in order to drive EMAC to the reset state. The bit is cleared by the hardware when the reset is completed. If EMACx_MR0[SRST] = 1, writing to any EMAC register, and reading any other bit in this register, is not supported.

3	TXE	Transmit MAC Enable 0 TX MAC is disabled 1 TX MAC is enabled	
4	RXE	Receive MAC Enable 0 RX MAC is disabled 1 RX MAC is enabled	
5	WKE	Wake-Up Enable 0 Incoming packets are not examined for wake-up packet 1 Examine incoming packets for wake-up packet	Software can change EMACx_MR0[WKE] only while EMACx_MR0[RXI] = 1 and EMACx_MR0[RXE] = 0.
6:31		Reserved	

22.7.2 Mode Register 1 (EMACx_MR1)

EMACx_MR1 defines the EMAC operating modes, which can be changed only after a reset.

Software can use EMACx_MR1[FDE] and proper programming of the attached PHY to activate external loop-back mode.

When EMACx_MR1[FDE, ILE] = 1, EMAC wraps transmitted packets back to the receive FIFO without accessing the MII.

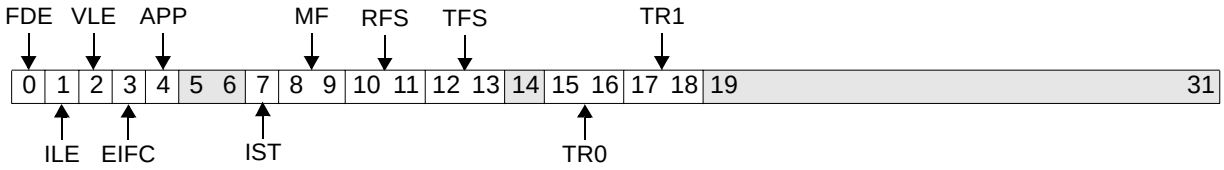


Figure 0-4. Mode Register 1 (EMACx_MR1)

0	FDE	Full-Duplex Enable 0 Disable simultaneous transmit and receive 1 Enable simultaneous transmit and receive	
1	ILE	Internal Loop-back Enable 0 No wrap back 1 Transmitted packets wrapped back to receive FIFO	Full Duplex must also be set (EMACx_MRI[FDE]=1).
2	VLE	VLAN Enable 0 Disable processing of VLAN Tags 1 Enable processing of VLAN Tags	
3	EIFC	Enable Integrated Flow Control 0 Disable integrated flow control mechanism 1 Enable integrated flow control mechanism	Refer to "Flow Control" on page 22-15 for more details. Set EMACx_MR1[EIFC] = 0 in half-duplex mode.

4	APP	Allow Pause Packet 0 Disables processing of incoming control (pause) packets 1 Enables processing of incoming control (pause) packets	
5:6		Reserved	Always zero
7	IST	Ignore SQE test 0 Wait for end of SQE test period before activation of valid signal 1 Do not wait for end of SQE test period before activation of valid signal	EMACx_MR1[IST] = 0 only during half-duplex operation on 10 Mbps media.
8:9	MF	Medium Frequency 00 10 Mbps (Ethernet mode) 01 100 Mbps (Fast Ethernet mode) 10 Reserved 11 Reserved	Defines the possible operational frequency on the MII interface.
10:11	RFS	Receive (RX) FIFO Size 00 512 bytes 01 1 KB 10 2 KB 11 4 KB	
12:13	TFS	Transmit (TX) FIFO Size 00 Reserved 01 1 KB 10 2 KB 11 Reserved	
14		Reserved	Always 0
15:16	TR0	Transmit Request 0 00 Single packet 01 Multiple packets 10 Dependent mode (bits 17:18 must also be programmed to 10) 11 Reserved	Defines the different modes for using transmit channel 0 of EMAC.
17:18	TR1	Transmit Request 1 00 Single packet 01 Multiple packets 10 Dependent mode (bits 15:16 must also be programmed to 10) 11 Reserved	Defines the different modes for using transmit channel 1 of EMAC.
19:31		Reserved	

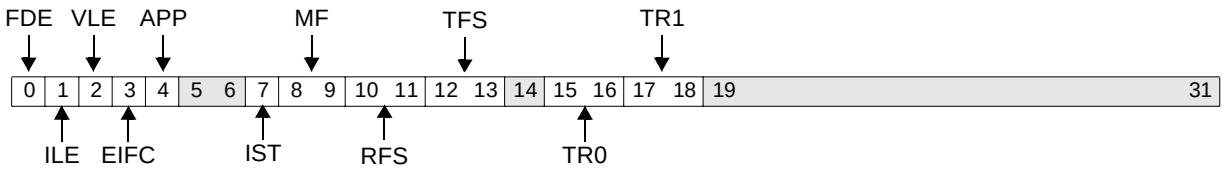


Figure 22-16. Mode Register 1 (EMACx_MR1)

0	FDE	Full-Duplex Enable 0 Disable simultaneous transmit and receive 1 Enable simultaneous transmit and receive	
1	ILE	Internal Loop-back Enable 0 No wrap back 1 Transmitted packets wrapped back to receive FIFO	Full Duplex must also be set (EMACx_MRI[FDE]=1).
2	VLE	VLAN Enable 0 Disable processing of VLAN Tags 1 Enable processing of VLAN Tags	
3	EIFC	Enable Integrated Flow Control 0 Disable integrated flow control mechanism 1 Enable integrated flow control mechanism	Refer to "Flow Control" on page -744 for more details. Set EMACx_MR1[EIFC] = 0 in half-duplex mode.
4	APP	Allow Pause Packet 0 Disables processing of incoming control (pause) packets 1 Enables processing of incoming control (pause) packets	
5:6		Reserved	Always zero
7	IST	Ignore SQE test 0 Wait for end of SQE test period before activation of valid signal 1 Do not wait for end of SQE test period before activation of valid signal	EMACx_MR1[IST] = 0 only during half-duplex operation on 10 Mbps media.
8:9	MF	Medium Frequency 00 10 Mbps (Ethernet mode) 01 100 Mbps (Fast Ethernet mode) 10 Reserved 11 Reserved	Defines the possible operational frequency on the MII interface.
10:11	RFS	Receive (RX) FIFO Size 00 512 bytes 01 1 KB 10 2 KB 11 4 KB	
12:13	TFS	Transmit (TX) FIFO Size 00 Reserved 01 1 KB 10 2 KB 11 Reserved	
14		Reserved	Always 0

15:16	TR0	Transmit Request 0 00 Single packet 01 Multiple packets 10 Dependent mode (bits 17:18 must also be programmed to 10) 11 Reserved	Defines the different modes for using transmit channel 0 of EMAC.
17:18	TR1	Transmit Request 1 00 Single packet 01 Multiple packets 10 Dependent mode (bits 15:16 must also be programmed to 10) 11 Reserved	Defines the different modes for using transmit channel 1 of EMAC.
19:31		Reserved	

22.7.3 Transmit Mode Register 0 (EMACx_TMR0)

EMACx_TMR0 defines EMAC operating modes during transmit operations (see ~~"EMAC Transmit Operation" on page 22-5~~*EMAC Transmit Operation* on page 734).

EMACx_TMR0[GNP0, GNP1, GNPD] are self-clearing. Writing 0 to these fields has no effect.

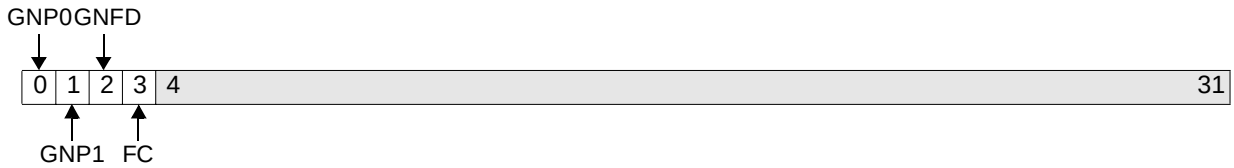


Figure 0-5. Transmit Mode Register 0 (EMACx_TMR0)

0	GNP0	Get New Packet 0 0 Writing 0 has no effect. 1 Packet ready for transmission on TX Channel 0	EMACx_TMR0[GNP0] = 0 if EMAC is programmed in dependent mode.
1	GNP1	Get New Packet 1 0 Writing 0 has no effect. 1 Packet ready for transmission on TX Channel 1	EMACx_TMR0[GNP1] = 0 if EMAC is programmed in dependent mode.
2	GNPD	Get New Packet for Dependent Mode 0 Writing 0 to this bit has no effect 1 Packet ready for transmission in dependent mode	EMACx_TMR0[GNPD] = 0 if EMAC is not programmed in dependent mode. EMACx_TMR0[GNPD] = 1 activates the EMAC transmit path in dependent mode.
3	FC	First Channel 0 Activate TX Channel 0 first when GNPD is 1 1 Activate TX Channel 1 first when GNPD is 1	EMACx_TMR0[FC] is only meaningful in dependent mode, after resetting EMACx_ISR[DBDM]. EMACx_TMR0[FC] = 0 if EMAC is not programmed in dependent mode.
4:31		Reserved	

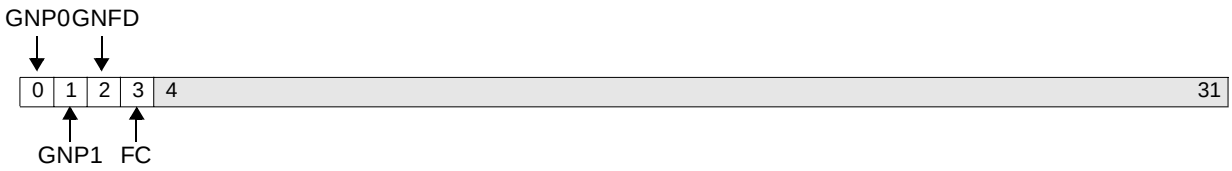


Figure 22-17. Transmit Mode Register 0 (EMACx_TMR0)

0	GNP0	Get New Packet 0 0 Writing 0 has no effect. 1 Packet ready for transmission on TX Channel 0	EMACx_TMR0[GNP0] = 0 if EMAC is programmed in dependent mode.
1	GNP1	Get New Packet 1 0 Writing 0 has no effect. 1 Packet ready for transmission on TX Channel 1	EMACx_TMR0[GNP1] = 0 if EMAC is programmed in dependent mode.
2	GNPD	Get New Packet for Dependent Mode 0 Writing 0 to this bit has no effect 1 Packet ready for transmission in dependent mode	EMACx_TMR0[GNPD] = 0 if EMAC is not programmed in dependent mode. EMACx_TMR0[GNPD] = 1 activates the EMAC transmit path in dependent mode.
3	FC	First Channel 0 Activate TX Channel 0 first when GNPD is 1 1 Activate TX Channel 1 first when GNPD is 1	EMACx_TMR0[FC] is only meaningful in dependent mode, after resetting EMACx_ISR[DBDM]. EMACx_TMR0[FC] = 0 if EMAC is not programmed in dependent mode.
4:31		Reserved	

22.7.4 Transmit Mode Register 1 (EMACx_TMR1)

EMACx_TMR1 defines conditions for activation of MAL service requests during transmit operations (see ~~"EMAC Transmit Operation" on page 22-5~~*EMAC Transmit Operation* on page 734).

22.7.4.1 Low-Priority Requests

EMAC requests low priority service from MAL when the number of vacant entries in the transmit FIFO exceeds the decimal transmit low request (TLR) value. EMACx_TMR1[TLR] must be at least 17.

To avoid a deadlock, the sum of EMACx_TMR1[TLR] and EMACx_TRTR[TRT] must be at least 4 smaller than the transmit FIFO size specified by EMACx_MR1[TFS].

22.7.4.2 Urgent-Priority Requests

EMAC requests urgent priority service from MAL if the following conditions occur:

- EMAC begins transmitting the packet to the media before the entire packet is placed in the TX FIFO
- The number of vacant entries for the currently transmitting packet exceeds the decimal TUR value

Software must coordinate the value of EMACx_TMR1[TUR] with the value of EMACx_MR1[TFS].The value of EMACx_TMR1[TUR] must be smaller than that of EMACx_MR1[TFS] so that the array address encoded in EMACx_TMR1[TUR] can access the full 66-bit wide array.

The binary value of EMACx_TMR1[TUR] must be greater than that of EMACx_TMR1[TLR].

The EMACx_TMR1 contents can be changed only when EMACx_TMR0[GNP0, GNP1, GNPD] = 0.

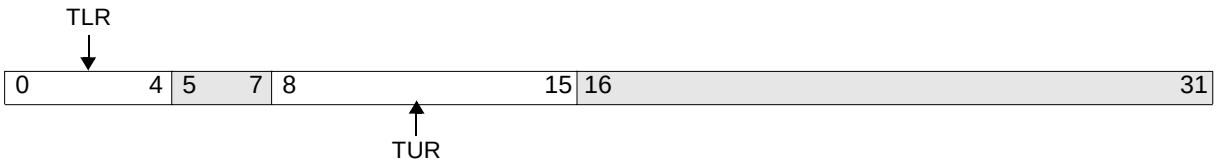


Figure 0-6. Transmit Mode Register 1 (EMACx_TMR1)

0:4	TLR	Transmit Low Request
5:7		Reserved
8:15	TUR	Transmit Urgent Request
16:31		Reserved

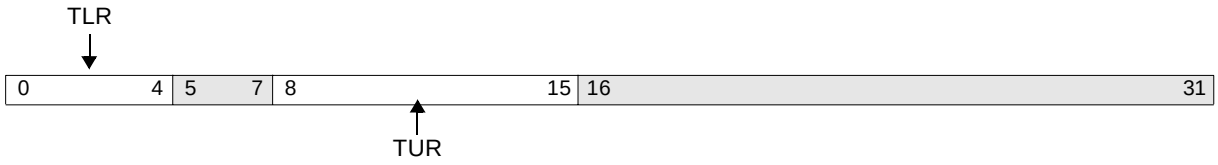


Figure 22-18. Transmit Mode Register 1 (EMACx_TMR1)

0:4	TLR	Transmit Low Request
5:7		Reserved
8:15	TUR	Transmit Urgent Request
16:31		Reserved

22.7.5 Receive Mode Register (EMACx_RMR)

EMACx_RMR defines EMAC operating modes during receive operations.

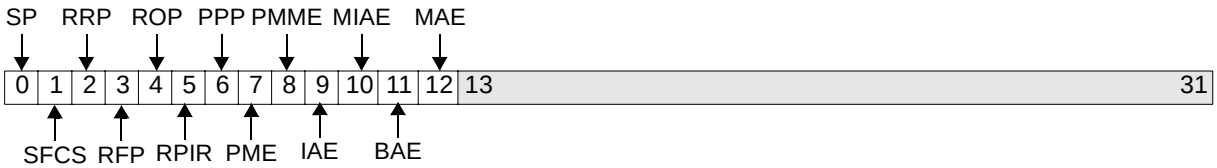


Figure 0-7. Receive Mode Register (EMACx_RMR)

0	SP	Strip Padding 0 Do not strip pad bytes from the received packet. 1 Strip pad/FCS bytes from the received packet.
1	SFCS	Strip FCS 0 Do not strip FCS bytes from the received packet. 1 Strip FCS bytes from the received packet.
2	RRP	Receive Runt Packets 0 Discard packets less than 64 bytes in length. 1 Receive packets less than 64 bytes in length.
3	RFP	Allow Receive Packets with a FCS Error 0 Discard packets containing a FCS error. 1 Receive packets containing a FCS error.
4	ROP	Receive Oversize Packet 0 Discard packets that activate Packet Is Too Long error. 1 Receive packets that activate Packet Is Too Long error.
5	RPIR	Receive Packets with In Range Error 0 Discard packets that activate In Range Error. 1 Receive packets that activate In Range Error.
6	PPP	Propagate Pause Packet 0 Do not propagate incoming pause packet to MAL; remove packet from FIFO. 1 Propagate incoming pause packet to MAL.
7	PME	Promiscuous Mode Enable 0 Do not enable promiscuous mode. 1 Accept all packets.

8	PMME	Promiscuous Multicast Mode Enable 0 Do not accept all multicast packets. 1 Accept all multicast packets.
9	IAE	Individual Address Enable 0 Do not compare address of received packets with content of individual address register. 1 Compare address of received packets with content of individual address register.
10	MIAE	Multiple Individual Address Enable 0 Do not compare address of received packets with hash table of individual addresses. 1 Compare address of received packets with hash table of individual addresses.
11	BAE	Broadcast Address Enable 0 Do not compare address of received packets with broadcast addresses. 1 Compare address of received packets with broadcast addresses.
12	MAE	Multicast Address Enable 0 Do not compare address of received packets with multicast addresses. 1 Compare address of received packets with multicast addresses.
13:31		Reserved

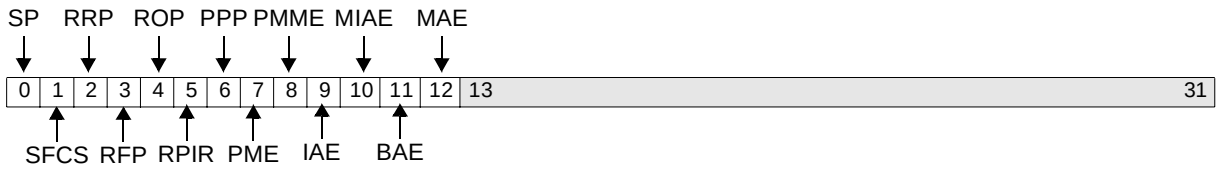


Figure 22-19. Receive Mode Register (EMACx_RMR)

0	SP	Strip Padding 0 Do not strip pad bytes from the received packet. 1 Strip pad/FCS bytes from the received packet.
1	SFCS	Strip FCS 0 Do not strip FCS bytes from the received packet. 1 Strip FCS bytes from the received packet.
2	RRP	Receive Runt Packets 0 Discard packets less than 64 bytes in length. 1 Receive packets less than 64 bytes in length.
3	RFP	Allow Receive Packets with a FCS Error 0 Discard packets containing a FCS error. 1 Receive packets containing a FCS error.

4	ROP	Receive Oversize Packet 0 Discard packets that activate Packet Is Too Long error. 1 Receive packets that activate Packet Is Too Long error.
5	RPIR	Receive Packets with In Range Error 0 Discard packets that activate In Range Error. 1 Receive packets that activate In Range Error.
6	PPP	Propagate Pause Packet 0 Do not propagate incoming pause packet to MAL; remove packet from FIFO. 1 Propagate incoming pause packet to MAL.
7	PME	Promiscuous Mode Enable 0 Do not enable promiscuous mode. 1 Accept all packets.
8	PMME	Promiscuous Multicast Mode Enable 0 Do not accept all multicast packets. 1 Accept all multicast packets.
9	IAE	Individual Address Enable 0 Do not compare address of received packets with content of individual address register. 1 Compare address of received packets with content of individual address register.
10	MIAE	Multiple Individual Address Enable 0 Do not compare address of received packets with hash table of individual addresses. 1 Compare address of received packets with hash table of individual addresses.
11	BAE	Broadcast Address Enable 0 Do not compare address of received packets with broadcast addresses. 1 Compare address of received packets with broadcast addresses.
12	MAE	Multicast Address Enable 0 Do not compare address of received packets with multicast addresses. 1 Compare address of received packets with multicast addresses.
13:31		Reserved

22.7.6 Interrupt Status Register (EMACx_ISR)

Each EMAC generates a distinct interrupt event indication. The event indication signal is driven out of the EMAC to UIC1 (interrupt 28 ~~(EMAC0)~~ for EMAC0 and ~~UIC1~~ interrupt 30 ~~(~~for EMAC1). This interrupt is generated from the content of the EMACx_ISR. The content of the EMACx_ISR is first ANDed with the corresponding mask bits in the EMACx_ISER; the resulting bits are then logically ORed to produce the interrupt signal. Thus, if any of the resulting bits is a 1, an interrupt is generated.

Note: EMAC activates its interrupt signal only after an indication that status for the current packet was accepted by MAL (with the exception of "MMA Operation Succeed/MMA Operation Failed," which causes unconditional activation of interrupt, if it is not masked).

The interrupt indication is cleared by writing 1 to the related bit in the EMACx_ISR; writing 0 has no effect.

The event indication signal is cleared when all non-masked event indication bits are cleared.

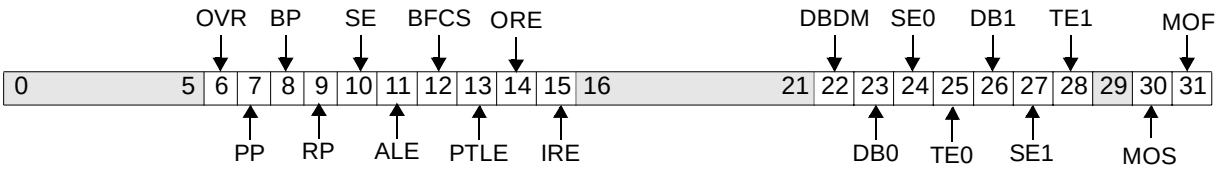


Figure 0-8. Interrupt Status Register (EMACx_ISR)

0:5		Reserved	
6	OVR	Overrun 0 No overrun error 1 Overrun error during reception of recent packet	
7	PP	Pause Packet 0 Received packet is not a control pause packet 1 Received packet is a control pause packet	
8	BP	Bad Packet 0 Receive operation OK 1 Early termination was initiated because of a packet error	
9	RP	Runt Packet 0 No Runt packets received 1 Runt packet received	Set when EMACx_RMR[RRP] = 1 and the duration of PHY_RX_DV signal was greater than ShortEventMaxTime constant and less than the collision window.
10	SE	Short Event 0 No short events 1 Duration of PHY_RX_DV signal less than ShortEventMaxTime constant	
11	ALE	Alignment Error 0 No alignment error in received packet 1 Alignment error in received packet	The packet contained an odd number of nibbles (4 bits).
12	BFCS	Bad FCS 0 No FCS error in received packet 1 Packet with an FCS error received	Set if EMACx_RMR[RFP] = 1.

13	PTLE	Packet Too Long Error 0 No oversized packets received 1 Oversized packet received	Set if EMACx_RMR[ROP] = 1 and the received packet length exceeded the maximum allowed value: • 1518 octets for standard packet (checked only if the length/type field of the transmitted packet contained length value) • 1522 octets for VLAN tagged packet (checked only if the length/type field of the transmitted packet contained length value and jumbo support is disabled)
14	ORE	Out Of Range Error 0 Received packet length field value OK 1 Received packet length field value greater than the maximum allowed LLC data size	Indicates that received packet has a length field value greater than the maximum allowed logical link control (LLC) data size (greater than 1500 and less than 1536).
15	IRE	In Range Error 0 Received packet does not contain an In Range Error 1 Received packet contains an In Range Error	
16:21		Reserved	
22	DBDM	Dead Bit Dependent Mode 0 No transmit error or SQE in dependent mode 1 Transmit error or SQE has occurred while in dependent mode	If EMACx_ISR[DBDM] = 1, EMAC does not request MAL service, even if EMACx_TMR0[GNPD] = 1. EMACx_ISR[DBDM] does not affect EMAC interrupt.
23	DB0	Dead Bit 0 0 No transmit error or SQE for TX Channel 0 while not in dependent mode 1 Transmit error or SQE has occurred for TX Channel 0 while not in dependent mode	If EMACx_ISR[DB0] = 1, EMAC does not request service for TX Channel 0 from MAL, even if EMACx_TMR0[GNP0] = 1. EMACx_ISR[DB0] does not affect EMAC interrupt.
24	SE0	Signal Quality Error 0 0 No SQEs on TX Channel 0 1 SQE test failure during transmission of a packet from TX Channel 0	Applicable only in half-duplex mode during 10 Mbps operations; 0 in all other modes.
25	TE0	Transmit Error 0 0 TX Channel 0 transmission OK 1 TX Channel 0 transmission aborted	EMAC aborts the transmitted packet if one of the following events takes place: • Late collision detection • Excessive collision detection • Excessive deferral • TX FIFO underrun • Loss of carrier sense

26	DB1	Dead Bit 1 0 No transmit error or SQE for TX Channel 1 while not in dependent mode 1 Transmit error or a SQE has occurred for TX Channel 1 while not in dependent mode	If this bit is set, EMAC does not request MAL service for TX Channel 1 even if EMACx_TMR1[GNP1] = 1. EMACx_ISR[DB1] does not affect EMAC interrupt.
27	SE1	Signal Quality Error 1 0 No SQE on TX Channel 1 1 SQE test failure during transmission of a packet from TX Channel 1	Applicable only in half-duplex mode during 10 Mbps operations; 0 in all other modes.
28	TE1	Transmit Error 1 0 TX Channel 1 transmission OK 1 TX Channel 1 transmission aborted	EMAC aborts the transmitted packet if one of the following events takes place: • Late collision detection • Excessive collision detection • Excessive deferral • TX FIFO underrun • Loss of carrier sense
29		Reserved	Always 0
30	MOS	MMA Operation Succeeded 0 MMA_CONTROL addressed on the OPB 1 PHY transfer valid	The device driver should poll assertion of EMACx_ISR[MOS] or EMACx_ISR[MOF] before issuing a new command or before using data read from the PHY.
31	MOF	MMA Operation Failed 0 MMA_CONTROL addressed on the OPB 1 PHY transfer *not* valid	The device driver should poll assertion of EMACx_ISR[MOF] or EMACx_ISR[MOS] before issuing a new command or before using data read from the PHY.

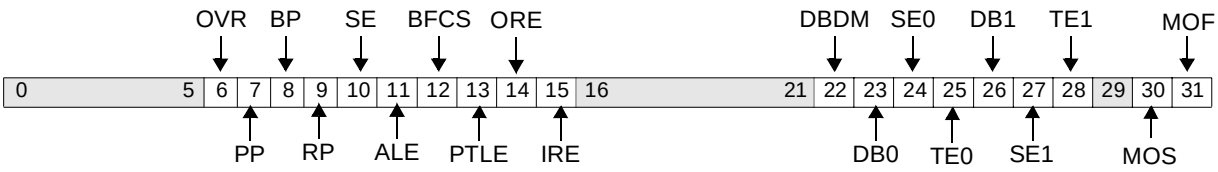


Figure 22-20. Interrupt Status Register (EMACx_ISR)

0:5		Reserved
6	OVR	Overrun 0 No overrun error 1 Overrun error during reception of recent packet
7	PP	Pause Packet 0 Received packet is not a control pause packet 1 Received packet is a control pause packet
8	BP	Bad Packet 0 Receive operation OK 1 Early termination was initiated because of a packet error

9	RP	Runt Packet 0 No Runt packets received 1 Runt packet received	Set when EMACx_RMR[RRP] = 1 and the duration of PHY_RX_DV signal was greater than Short-EventMaxTime constant and less than the collision window.
10	SE	Short Event 0 No short events 1 Duration of PHY_RX_DV signal less than ShortEventMaxTime constant	
11	ALE	Alignment Error 0 No alignment error in received packet 1 Alignment error in received packet	The packet contained an odd number of nibbles (4 bits).
12	BFCS	Bad FCS 0 No FCS error in received packet 1 Packet with an FCS error received	Set if EMACx_RMR[RFP] = 1.
13	PTLE	Packet Too Long Error 0 No oversized packets received 1 Oversized packet received	Set if EMACx_RMR[ROP] = 1 and the received packet length exceeded the maximum allowed value: • 1518 octets for standard packet (checked only if the length/type field of the transmitted packet contained length value) • 1522 octets for VLAN tagged packet (checked only if the length/type field of the transmitted packet contained length value and jumbo support is disabled)
14	ORE	Out Of Range Error 0 Received packet length field value OK 1 Received packet length field value greater than the maximum allowed LLC data size	Indicates that received packet has a length field value greater than the maximum allowed logical link control (LLC) data size (greater than 1500 and less than 1536).
15	IRE	In Range Error 0 Received packet does not contain an In Range Error 1 Received packet contains an In Range Error	
16:21		Reserved	
22	DBDM	Dead Bit Dependent Mode 0 No transmit error or SQE in dependent mode 1 Transmit error or SQE has occurred while in dependent mode	If EMACx_ISR[DBDM] = 1, EMAC does not request MAL service, even if EMACx_TMR0[GNPD] = 1. EMACx_ISR[DBDM] does not affect EMAC interrupt.
23	DB0	Dead Bit 0 0 No transmit error or SQE for TX Channel 0 while not in dependent mode 1 Transmit error or SQE has occurred for TX Channel 0 while not in dependent mode	If EMACx_ISR[DB0] = 1, EMAC does not request service for TX Channel 0 from MAL, even if EMACx_TMR0[GNP0] = 1. EMACx_ISR[DB0] does not affect EMAC interrupt.
24	SE0	Signal Quality Error 0 0 No SQEs on TX Channel 0 1 SQE test failure during transmission of a packet from TX Channel 0	Applicable only in half-duplex mode during 10 Mbps operations; 0 in all other modes.

25	TE0	Transmit Error 0 0 TX Channel 0 transmission OK 1 TX Channel 0 transmission aborted	EMAC aborts the transmitted packet if one of the following events takes place: • Late collision detection • Excessive collision detection • Excessive deferral • TX FIFO underrun • Loss of carrier sense
26	DB1	Dead Bit 1 0 No transmit error or SQE for TX Channel 1 while not in dependent mode 1 Transmit error or a SQE has occurred for TX Channel 1 while not in dependent mode	If this bit is set, EMAC does not request MAL service for TX Channel 1 even if EMACx_TMR1[GNP1] = 1. EMACx_ISR[DB1] does not affect EMAC interrupt.
27	SE1	Signal Quality Error 1 0 No SQE on TX Channel 1 1 SQE test failure during transmission of a packet from TX Channel 1	Applicable only in half-duplex mode during 10 Mbps operations; 0 in all other modes.
28	TE1	Transmit Error 1 0 TX Channel 1 transmission OK 1 TX Channel 1 transmission aborted	EMAC aborts the transmitted packet if one of the following events takes place: • Late collision detection • Excessive collision detection • Excessive deferral • TX FIFO underrun • Loss of carrier sense
29		Reserved	Always 0
30	MOS	MMA Operation Succeeded 0 MMA_CONTROL addressed on the OPB 1 PHY transfer valid	The device driver should poll assertion of EMACx_ISR[MOS] or EMACx_ISR[MOF] before issuing a new command or before using data read from the PHY.
31	MOF	MMA Operation Failed 0 MMA_CONTROL addressed on the OPB 1 PHY transfer *not* valid	The device driver should poll assertion of EMACx_ISR[MOF] or EMACx_ISR[MOS] before issuing a new command or before using data read from the PHY.

22.7.7 Interrupt Status Enable Register (EMACx_ISER)

EMACx_ISER indicates which conditions in the EMACx_ISR can generate an interrupt.

Each masking bit in the EMACx_ISER corresponds to a related bit in the EMACx_ISR. If a mask bit is set to 1, the corresponding status bit, when set, causes an interrupt to be generated. Setting a mask bit to 0 suppresses interrupt generation for the associated condition.

Mask bits for reserved bits in the EMACx_ISR are not implemented, have no effect on write, and return 0 on read.

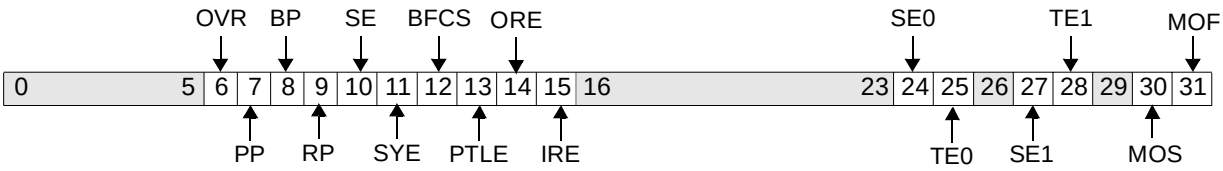


Figure 0-9. Interrupt Status Enable Register (EMACx_ISER)

0:5		Reserved

6	OVR	Overrun 0 Overrun error will not generate an interrupt. 1 Overrun error will generate an interrupt.
7	PP	Pause Packet 0 Received control pause packet will not generate an interrupt. 1 Received control pause packet will generate an interrupt.
8	BP	Bad Packet 0 Early termination on received packet will not generate an interrupt. 1 Early termination on received packet will generate an interrupt.
9	RP	Runt Packet 0 Received runt packet will not generate an interrupt. 1 Received runt packet will generate an interrupt.
10	SE	Short Event 0 Short event during receive will not generate an interrupt. 1 Short event during receive will generate an interrupt.
11	ALE	Alignment Error 0 Alignment error in received packet will not generate an interrupt. 1 Alignment error in received packet will generate an interrupt.
12	BFCS	Bad FCS 0 FCS error in received packet will not generate an interrupt. 1 FCS error in received packet will generate an interrupt.
13	PTLE	Packet Too Long Error 0 Oversized packets received will not generate an interrupt. 1 Oversized packet received will generate an interrupt.
14	ORE	Out Of Range Error 0 Out of range error on received packet will not generate an interrupt. 1 Out of range error on received packet will generate an interrupt.

15	IRE	In Range Error 0 In range error on received packet will not generate an interrupt. 1 In range error on received packet will generate an interrupt.
16:23		Reserved
24	SE0	SQE Error 0 0 SQE error on TX Channel 0 will not generate an interrupt. 1 SQE error on TX Channel 0 will generate an interrupt.
25	TE0	Transmit Error 0 0 TX error on TX Channel 0 will not generate an interrupt. 1 TX error on TX Channel 0 will generate an interrupt.
26		Reserved
27	SE1	SQE Error 1 0 SQE error on TX Channel 1 will not generate an interrupt. 1 SQE error on TX Channel 1 will generate an interrupt.
28	TE1	Transmit Error 1 0 TX error on TX Channel 1 will not generate an interrupt. 1 TX error on TX Channel 1 will generate an interrupt.
29		Reserved
30	MOS	MMA Operation Succeeded 0 Successful MMA Operation with a PHY will not generate an interrupt. 1 Successful MMA Operation with a PHY will generate an interrupt.
31	MOF	MMA Operation Failed 0 Unsuccessful MMA Operation with a PHY will not generate an interrupt. 1 Unsuccessful MMA Operation with a PHY will generate an interrupt.

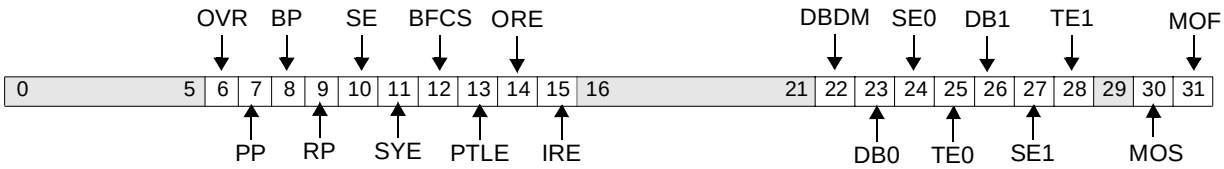


Figure 22-21. Interrupt Status Enable Register (EMACx_ISER)

0:5		Reserved
6	OVR	Overrun 0 Overrun error will not generate an interrupt. 1 Overrun error will generate an interrupt.
7	PP	Pause Packet 0 Received control pause packet will not generate an interrupt. 1 Received control pause packet will generate an interrupt.
8	BP	Bad Packet 0 Early termination on received packet will not generate an interrupt. 1 Early termination on received packet will generate an interrupt.
9	RP	Runt Packet 0 Received runt packet will not generate an interrupt. 1 Received runt packet will generate an interrupt.
10	SE	Short Event 0 Short event during receive will not generate an interrupt. 1 Short event during receive will generate an interrupt.
11	ALE	Alignment Error 0 Alignment error in received packet will not generate an interrupt. 1 Alignment error in received packet will generate an interrupt.
12	BFCS	Bad FCS 0 FCS error in received packet will not generate an interrupt. 1 FCS error in received packet will generate an interrupt.
13	PTLE	Packet Too Long Error 0 Oversized packets received will not generate an interrupt. 1 Oversized packet received will generate an interrupt.
14	ORE	Out Of Range Error 0 Out of range error on received packet will not generate an interrupt. 1 Out of range error on received packet will generate an interrupt.

15	IRE	In Range Error 0 In range error on received packet will not generate an interrupt. 1 In range error on received packet will generate an interrupt.
16:21		Reserved
22	DBDM	Dead Bit Dependent Mode 0 Dead bit dependent mode will not generate an interrupt 1 Dead bit dependent mode will generate an interrupt
23	DB0	Dead Bit 0 0 Dead bit 0 will not generate an interrupt 1 Dead bit 0 will generate an interrupt
24	SE0	SQE Error 0 0 SQE error on TX Channel 0 will not generate an interrupt. 1 SQE error on TX Channel 0 will generate an interrupt.
25	TE0	Transmit Error 0 0 TX error on TX Channel 0 will not generate an interrupt. 1 TX error on TX Channel 0 will generate an interrupt.
26	DB1	Dead Bit 1 0 Dead bit 1 will not generate an interrupt 1 Dead bit 1 will generate an interrupt
27	SE1	SQE Error 1 0 SQE error on TX Channel 1 will not generate an interrupt. 1 SQE error on TX Channel 1 will generate an interrupt.
28	TE1	Transmit Error 1 0 TX error on TX Channel 1 will not generate an interrupt. 1 TX error on TX Channel 1 will generate an interrupt.
29		Reserved
30	MOS	MMA Operation Succeeded 0 Successful MMA Operation with a PHY will not generate an interrupt. 1 Successful MMA Operation with a PHY will generate an interrupt.
31	MOF	MMA Operation Failed 0 Unsuccessful MMA Operation with a PHY will not generate an interrupt. 1 Unsuccessful MMA Operation with a PHY will generate an interrupt.

22.7.8 Individual Address High (EMACx_IAHR)

EMACx_IAHR contains the high-order halfword of the station unique individual address.

During packet reception, if EMAC is programmed in individual address match mode (EMACx_RMR[IAE] = 1), the contents of EMACx_IAHR are concatenated with the content of EMACx_IALR to form a composite address that is compared with the destination address of the received packet. If addresses match, the packet is transferred to MAL.

During packet transmission, EMACx_IAHR is used in source address inclusion/replacement and as the source address field in the self-assembled control (pause) packet.

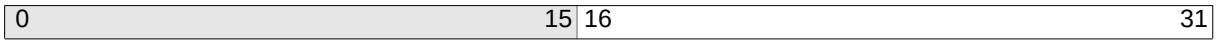


Figure 0-10. Individual Address High Register (EMACx_IAHR)

0:15		Reserved	
16:31		High-order halfword of the station unique individual address	This field contains bits 0:15 of the destination address (bit 0 is the most significant bit).

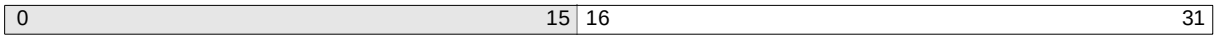


Figure 22-22. Individual Address High Register (EMACx_IAHR)

0:15		Reserved	
16:31		High-order halfword of the station unique individual address	This field contains bits 0:15 of the destination address (bit 0 is the most significant bit).

22.7.9 Individual Address Low (EMACx_IALR)

EMACx_IALR contains the low-order word of the station unique individual address.

During packet reception, EMACx_IALR is compared with the corresponding address bits of the received packet.

During packet transmission, EMACx_IALR is used in source address inclusion/replacement and as the source address field in the self-assembled control (pause) packet.

Figure 0-11. Individual Address Low Register (EMACx_IALR)

0:31		Low-order bits of Receive Individual Address or Transmit Source Address

Figure 22-23. Individual Address Low Register (EMACx_IALR)

0:31		Low-order bits of Receive Individual Address or Transmit Source Address

22.7.10 VLAN TPID Register (EMACx_VTPID)

EMACx_VTPID contains the value of the VLAN TPID (Tag Protocol Identifier) field.

During packet reception, packet bytes 13 and 14 are compared to the content of this register to check whether the packet is tagged with a VLAN ID.

During packet transmission, EMAC uses EMACx_VTPID when VLAN Tag replacement or VLAN Tag inclusion mode is chosen.

The value of this register must be a Type field (8100).

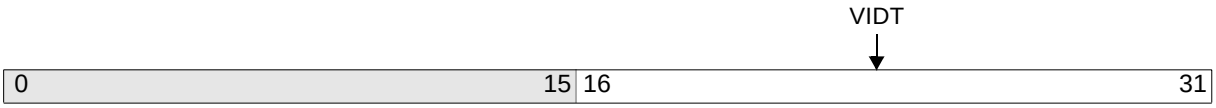


Figure 0-12. VLAN TPID Register (EMACx_VTPID)

0:15		Reserved
16:31	VIDT	VLAN ID tag

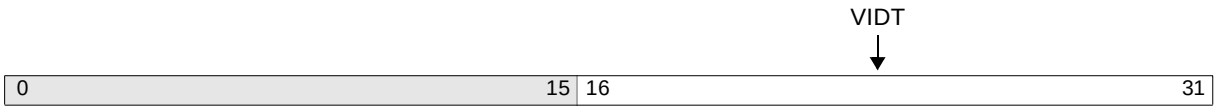


Figure 22-24. VLAN TPID Register (EMACx_VTPID)

0:15		Reserved
16:31	VIDT	VLAN ID tag

22.7.11 VLAN TCI Register (EMACx_VTCI)

EMACx_VTCI contains the value of the VLAN TCI (Tag Control Information) field.

During packet transmission, EMAC uses EMACx_VTCI when VLAN Tag replacement or VLAN Tag inclusion mode is chosen.

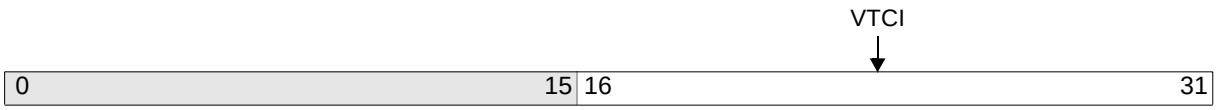


Figure 0-13. VLAN TCI Register (EMACx_VTCI)

0:15		Reserved
16:31	VTCI	VLAN TCI tag

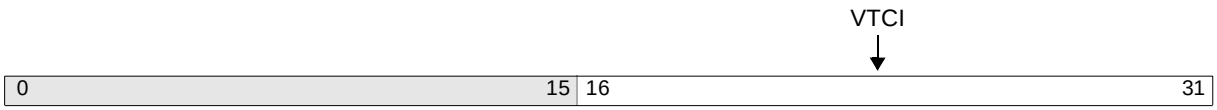


Figure 22-25. VLAN TCI Register (EMACx_VTCI)

0:15		Reserved
16:31	VTCI	VLAN TCI tag

22.7.12 Pause Timer Register (EMACx_PTR)

EMACx_PTR defines the time period for which the pause function is enabled. EMAC uses EMACx_PTR[TVR] as the timer value field of control (pause) packets (see ~~"Control Packet Transmission" on page 22-16~~*Control Packet Transmission* on page 745). Each bit corresponds to 512 bit times.

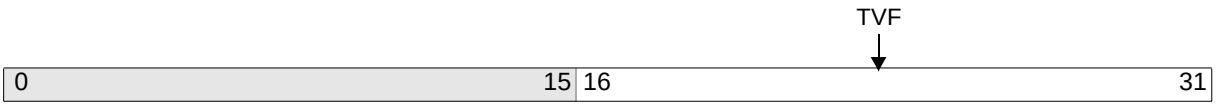


Figure 0-14. Pause Timer Register (EMACx_PTR)

0:15		Reserved
16:31	TVF	Timer Value Field

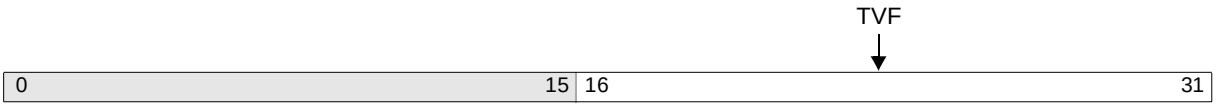


Figure 22-26. Pause Timer Register (EMACx_PTR)

0:15		Reserved
16:31	TVF	Timer Value Field

22.7.13 Individual Address Hash Tables 1–4 (EMACx_IAHT1–EMACx_IAHT4)

These registers are used in the hash table function of the multiple individual addressing mode.

See ~~"Address Match Mechanism" on page 22-20~~*Address Match Mechanism* on page 749 for more information. See ~~Figure 22-14 on page 22-23~~*Figure 22-14* on page 752 for bit mapping information.

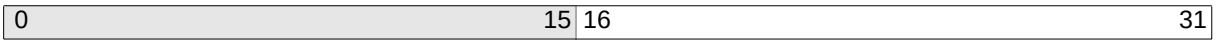


Figure 0-15. Individual Address Hash Tables 1–4 (EMACx_IAHT1–EMACx_IAHT4)

0:15		Reserved
16:31		Individual Address Hash Number

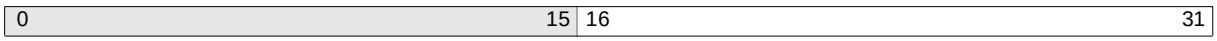


Figure 22-27. Individual Address Hash Tables 1–4 (EMACx_IAHT1–EMACx_IAHT4)

0:15		Reserved
16:31		Individual Address Hash Number

22.7.14 Group Address Hash Tables 1–4 (EMACx_GAHT1–EMACx_GAHT4)

These registers are used in the hash table function of the group addressing mode.

See ~~"Address Match Mechanism" on page 22-20~~*Address Match Mechanism* on page 749 for more information. See ~~Figure 22-14 on page 22-23~~*Figure 22-14* on page 752 for bit mapping information.

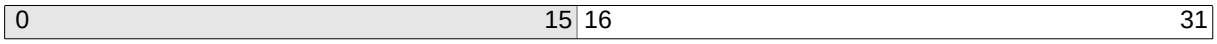


Figure 0-16. Group Address Hash Tables 1–4 (EMACx_GAHT1–EMACx_GAHT4)

0:15		Reserved
16:31		Group Address Hash Number

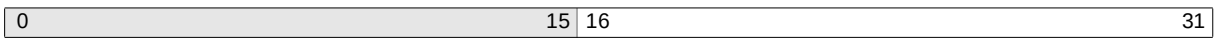


Figure 22-28. Group Address Hash Tables 1–4 (EMACx_GAHT1–EMACx_GAHT4)

0:15		Reserved
16:31		Group Address Hash Number

22.7.15 Last Source Address High (EMACx_LSAH)

EMACx_LSAH contains the high-order halfword of the source address of the last "good" received packet. The packet is considered to be "good" if EMAC is programmed to provide this packet to MAL.

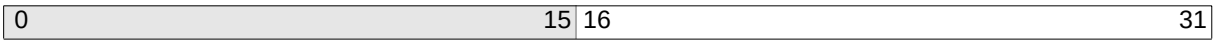


Figure 0-17. Last Source Address High Register (EMACx_LSAH)

0:15		Reserved
16:31		Last Source Address High-Order Halfword

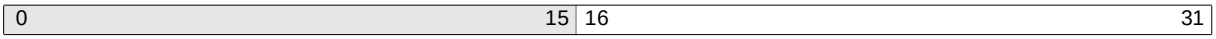


Figure 22-29. Last Source Address High Register (EMACx_LSAH)

0:15		Reserved
16:31		Last Source Address High-Order Halfword

22.7.16 Last Source Address Low (EMACx_LSAL)

EMACx_LSAL contains the low-order word of the source address of the last "good" received packet. The packet is considered "good" if EMAC is programmed to provide this packet to MAL.

Figure 0-18. Last Source Address Low Register (EMACx_LSAL)

0:31		Last Source Address Low-Order Word

Figure 22-30. Last Source Address Low Register (EMACx_LSAL)

0:31		Last Source Address Low-Order Word

22.7.17 Inter-Packet Gap Value Register (EMACx_IPGVR)

EMACx_IPGVR contains the value of one-third of the inter-packet gap (IPG) for the next packet to be transmitted. ("Frame" is synonymous with "packet.")

The resolution of each bit is 8-bit times. The minimum value in the register is four, causing a minimum.

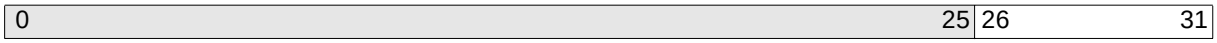


Figure 0-19. inter-Packet Gap Value Register (EMACx_IPGVR)

0:25		Reserved
26:31		Inter-Packet Gap

Figure 22-31. inter-Packet Gap Value Register (EMACx_IPGVR)

0:25		Reserved
26:31		Inter-Packet Gap

22.7.18 STA Control Register (EMACx_STACR)

EMACx_STACR controls the MII Management interface. The software must follow the following steps during access to the EMACx_STACR:

1. Software polls EMACx_STACR[OC], waiting for it to be set by EMAC.

EMAC sets EMACx_STACR[OC] = 0 when the EMACx_STACR is written to.

EMAC then sets EMACx_STACR[OC] = 1 to indicate that the data has been written to the PHY, or the data read from the PHY is valid. The device driver should poll for EMACx_STACR[OC] = 1 before issuing a new command, or before using data read from the PHY.

2. The software can perform read/write access to the EMACx_STACR.
3. EMAC clears EMACx_STACR[OC] (sets EMACx_STACR[OC] = 0) and starts activity on the MII management interface.
4. Return to step 1.

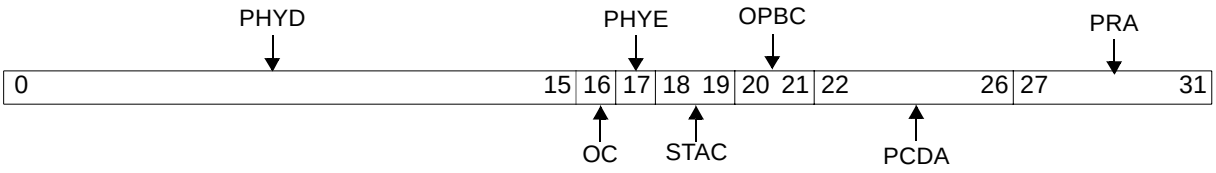


Figure 0-20. STA Control Register (EMACx_STACR)

0:15	PHYD	PHY data	Data to be sent to the PHY if the command is a write, or data is read from the PHY if the command is a read.
16	OC	Operation Complete 0 EMACx_STACR is addressed 1 PHY data transfer complete	
17	PHYE	PHY Error 0 Successful read transaction 1 Read transaction was not successful	EMACx_STACR[PHYE] = 0 when a read is successful.
18:19	STAC	STA Command 00 Reserved 01 Read 10 Write 11 Reserved	EMAC sets EMACx_STACR[STAC] = 0 when the command is completed.
20:21	OPBC	OPB Bus Clock Frequency 00 50 MHz 01 66 MHz 10 83 MHz 11 100 MHz	EMACx_STACR[OPBC] is used to generate the Management Data Clock (EMCMDClk). When the operational frequency differs from those in the list, then the next greater frequency should be chosen.
22:26	PCDA	PHY Command Destination Address	
27:31	PRA	PHY Register Address	

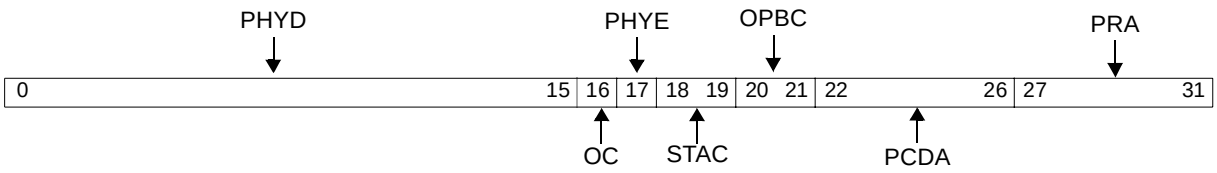


Figure 22-32. STA Control Register (EMACx_STACR)

0:15	PHYD	PHY data	Data to be sent to the PHY if the command is a write, or data is read from the PHY if the command is a read.
16	OC	Operation Complete 0 EMACx_STACR is addressed 1 PHY data transfer complete	
17	PHYE	PHY Error 0 Successful read transaction 1 Read transaction was not successful	EMACx_STACR[PHYE] = 0 when a read is successful.
18:19	STAC	STA Command 00 Reserved 01 Read 10 Write 11 Reserved	EMAC sets EMACx_STACR[STAC] = 0 when the command is completed.
20:21	OPBC	OPB Bus Clock Frequency 00 50 MHz 01 66 MHz 10 83 MHz 11 100 MHz	EMACx_STACR[OPBC] is used to generate the Management Data Clock (EMCMDCIk). When the operational frequency differs from those in the list, then the next greater frequency should be chosen.
22:26	PCDA	PHY Command Destination Address	
27:31	PRA	PHY Register Address	

22.7.19 Transmit Request Threshold Register (EMACx_TRTR)

EMACx_TRTR defines the conditions that cause EMAC to initiate transmission to the Ethernet MAC sub-block, and for requesting service from MAL.

EMACx_TRTR[TRT] defines the number of occupied entries in the transmit FIFO that should be written before the transmit FIFO control logic initiates a transmit request to the Ethernet MAC sub-block.

If an entire packet is already located in the transmit FIFO, EMAC initiates a transmit regardless of the programmed value.

The software must coordinate the value of EMACx_TRTR[TRT] with the transmit FIFO size specified in EMACx_MR1[TFS].

To avoid deadlock, the sum of EMACx_ TMR1[TLR] and EMACx_TRTR[TRT] must be smaller, by at least 4, than the transmit FIFO specified in EMACx_MR1[TFS].

To avoid an underrun, program this threshold to a high enough value.

In half-duplex mode, in case of collision, to allow packet re-transmission without involving MAL, EMAC preserves the necessary space in the Transmit FIFO unless it gets an indication that the collision window has elapsed.

The EMACx_TRTR can be written only while EMACx_MR0[TXI] = 1.

Figure 0-21. Transmit Request Threshold Register (EMACx_TRTR)

0:4	TRT	Transmit Request Threshold The following number of bytes must be placed in the Transmit FIFO before initiating a transmit request. 00000 64 bytes 00001 128 bytes 00010 192 bytes 00011 256 bytes . . . 11111 2048 bytes
5:31		Reserved

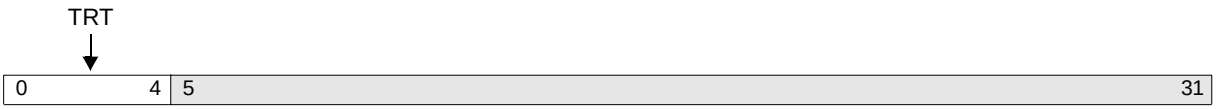


Figure 22-33. Transmit Request Threshold Register (EMACx_TRTR)

0:4	TRT	Transmit Request Threshold The following number of bytes must be placed in the Transmit FIFO before initiating a transmit request. 00000 64 bytes 00001 128 bytes 00010 192 bytes 00011 256 bytes . . . 11111 2048 bytes
5:31		Reserved

22.7.20 Receive Low/High Water Mark Register (EMACx_RWMR)

The EMACx_RWMR defines the conditions that cause the EMAC to activate a low or urgent priority MAL request, and that manage flow control.

EMAC activates a low priority request if the number of occupied entries in the receive FIFO is greater than or equal to the content of EMACx_RWMR[RLWM] (the receive low water mark is reached). A request for a pause packet with a pause_value of 0 is also issued when the receive low water mark is reached.

Software must coordinate the value of EMACx_RWMR[RLWM] with the value of EMACx_MR1[RFS]. EMACx_RWMR[RLWM] should be smaller than EMACx_MR1[RFS] and larger than the MAL burst length.

Note: In the PPC440GP, the MAL burst length is 16 words for all channels.

If the entire packet is already in the receive FIFO, EMAC initiates a low priority request regardless of the programmed value.

EMAC activates an urgent priority request if the number of occupied entries in the Receive FIFO is greater than or equal to EMACx_RWMR[RHWM] (the receive high water mark is reached). A request for a pause packet is also issued when the receive high water mark is reached.

Software must coordinate the value of EMACx_RWMR[RHWM] with the value of EMACx_MR1[RFS]. EMACx_RWMR[RHWM] should be greater than the value of EMACx_RWMR[RLWM] and less then the size of the receive FIFO.

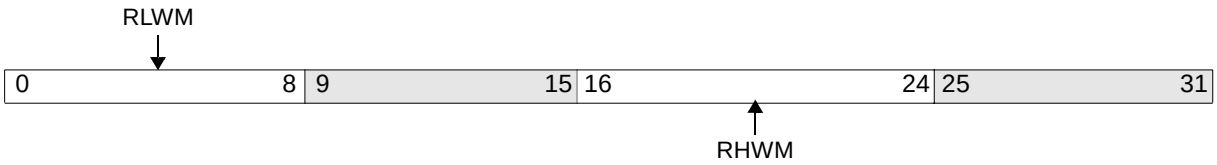


Figure 0-22. Receive Low/High Water Mark Register (EMACx_RWMR)

0:8	RLWM	Receive Low Water Mark
9:15		Reserved
16:24	RHWM	Receive High Water Mark
25:31		Reserved

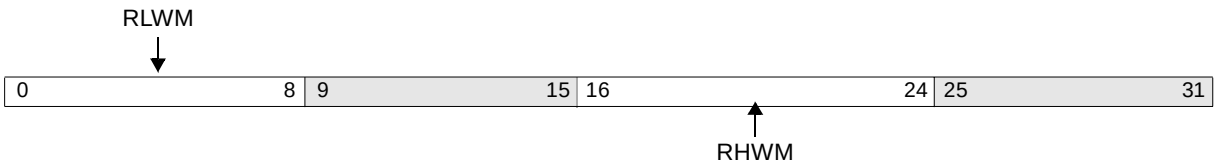


Figure 22-34. Receive Low/High Water Mark Register (EMACx_RWMR)

0:8	RLWM	Receive Low Water Mark
9:15		Reserved
16:24	RHWM	Receive High Water Mark

25:31		Reserved

22.7.21 Number of Octets Transmitted (EMACx_OCTX)

The read-only EMACx_OCTX register contains the number of transmitted octets.

Figure 0-23. Number of Octets Transmitted (EMACx_OCTX)

0:31	OCTX	Number of octets (bytes) transmitted.

Figure 22-35. Number of Octets Transmitted (EMACx_OCTX)

0:31	OCTX	Number of octets (bytes) transmitted.

22.7.22 Number of Octets Received (EMACx_OCRX)

The read-only EMACx_OCRX register contains the number of received octets.

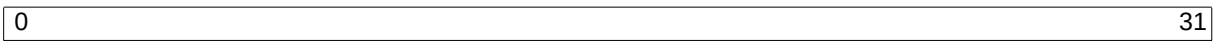


Figure 0-24. Number of Octets Received (EMACx_OCRX)

0:31	OCRX	Number of octets (bytes) received.

Figure 22-36. Number of Octets Received (EMACx_OCRX)

0:31	OCRX	Number of octets (bytes) received.

~~0.1 MII~~

22.8 MII Interface

EMAC implements all MII functionality in accordance with Clause 22 in the IEEE Std. 802.3u.

The MII interface is a reconciliation sublayer interface which allows a variety of PHYs to be attached to the EMAC Ethernet MAC without future upgrade problems.

22.8.1 MII Station Management Interface

The EMAC MII station management unit (STA) implements a specific protocol and a special packet format to exchange management packets with the registers of the attached PHY. EMAC automatically generates MII management packets, which conform to Clause 22 in IEEE Std. 802.3u. EMAC uses the EMACx_STACR for generation of the management packet. illustrates the interface.

22.9 MAL – EMAC Packet Transfer Flow

The packet transfer flow consists of three phases. These three phases are used to define the details of the EMAC-MAL protocol.

1. Packet phase - EMAC initiates a packet transfer operation. The packet transfer is started by a command write. During command write MAL provides control information for EMAC on a per-packet basis. Following the command write, MAL begins the data transfer, during which MAL transfers data between the buffers located in the system's memory and EMAC. In transmit, the data is transferred from the system's memory to EMAC, while in receive, the data is transferred from EMAC to the system's memory buffers.
   - EMAC remains in the packet phase until the data transfer has been completed or a ready status can be returned to MAL. The packet phase ends when EMAC deasserts the FRAME signal associated with the related channel (receive/transmit).
   - The packet phase is defined by activity of an appropriate FRAME signal.
2. Status phase - This is the second phase of the packet transfer. Following the de-assertion of the FRAME signal, EMAC switches to the status phase. At this stage, EMAC uses an appropriate signal as a request for service which is interpreted by MAL as a request for status read.

3. Idle phase - EMAC moves into the idle phase following a reset or after status was transferred (end of status phase). During the idle phase, EMAC cannot send any signals to MAL, nor can MAL send any active signals to EMAC. EMAC exits the idle phase by asserting the FRAME signal (and entering the packet phase described above). Idle phase can be skipped when EMAC operates in multiple transfer mode.

Figure 22-37 illustrates the different phases in the EMAC-MAL communication.

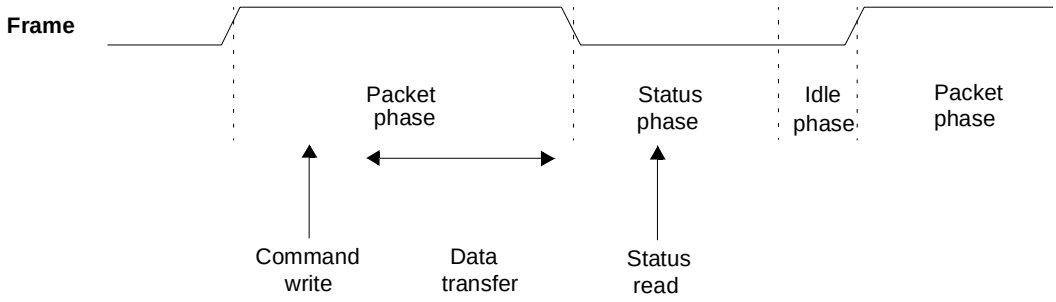


Figure 22-37. EMAC-MAL Communication Phases

During the packet and status phases EMAC signals a request for service by driving its arbitration level signal to a non-idle level.

22.10 Programming Notes

Certain combinations in device drivers are not allowed when writing to EMAC registers. When creating device drivers, ensure that the following guidelines are used:

- In dependent mode, EMACx_MR1[TR0] must be equal EMACx_MR1[TR1]
- When internal loopback is enabled (EMACx_MR1[ILE] = 1), EMAC must be configured in full-duplex mode (EMACx_MR1[FDE] =1)
- EMACx_MR1[IST] =0 only when EMACx_MR1[MF] = 0:10 and EMACx_MR1[FDE] =0
- In dependent mode, EMACx_ISER[SE0, TE0] must equal EMACx_ISER[SE1, TE1]
- EMACx_MR1[EIFC] =0 if EMACx_MR1[FDE] =0
- EMACx_TMR1[TLR] must be greater than the MAL burst size in entities (6 for MAL)
- EMACx_TMR1[TUR] must be greater or equal to EMACx_TMR1[TLR] and less than the Transmit FIFO size in entries (EMACx_MR1[TFS])
- To avoid deadlock, the sum of EMACx_TMR1[TLR] and the EMACx_TRTR[TRT] must be at least four less than the Transmit FIFO size specified in EMACx_MR1[TFS]
- EMACx_RWMR[RLWM] must be greater than the MAL burst size in entities (six for MAL)
- EMACx_RWMR[RHWM] must be greater than EMACx_RWMR[RLWM]
- EMACx_RWMR[RHWM] must be less than the Receive FIFO size in entities (EMACx_MR1[RFS])

22.10.1 Reset and Initialization

The EMAC must be initialized after a reset, or before performing configuration changes. The following types of reset operations can be applied to EMAC.

- Hard Reset. When RESET input is asserted, EMAC aborts all on-going activities unconditionally, initializes all internal state machines, counters, registers, and flushes transmit and receive FIFOs. To be recognized, the reset signal must be asserted for at least two cycles of the slowest clock domain inside EMAC (indicating that the hard reset must be at least 800 ns).
- Soft Reset. Software first should reset the appropriate MAL channels and then begin a soft reset by setting EMACx_MR0[SRST] = 1. In response to the soft reset, EMAC aborts all on-going activities unconditionally, initializes all internal state machines, counters, registers, and flushes transmit and receive FIFOs. After EMAC finishes all activities related to the soft reset processing, EMACx_MR0[SRST] = 0.
- Smart Reset. The software initializes smart reset mode by writing 0 to EMACx_MR0[TXE] or EMACx_MR0[RXE], or to both. In this case, the Ethernet MAC sub-block completes on-going activity (receive, transmit, or both) and then goes to the related Idle state (indicated by setting either EMACx_MR0[TXI] = 1 or EMACx_MR0[RXI] = 1, or both). In this case, the control logic sub-block of EMAC is still accessible for OPB and MAL transactions.

Before performing the necessary configuration changes in EMAC, the software must follow one of the following scenarios. Then the EMAC can be properly configured.

22.10.1.1 Scenario 1

- Hard/soft reset was activated.
- During hard/soft reset, EMACx_MR0[TXE] and EMACx_MR0[RXE] are reset.
- Software detects that the EMACx_MR0[SRST] is reset (after soft reset only).
- Software keeps EMACx_TMR0[GNP0, GNP1] = 0.
- The software can change one or more fields in registers marked with a Reset write access mode in ~~Table 22-5, "EMAC0 Register Summary," on page 22-23~~*Table 22-5 EMAC0 Register Summary* on page 752 (actually, all EMAC registers are accessible in this scenario).
- The software initializes EMACx_TMR0[GNP0, GNP1] as appropriate.
- The software configures EMACx_MR0[TXE, RXE].

22.10.1.2 Scenario 2

- Software sets EMACx_MR0[TXE] = 0.
- The TXMAC component of the Ethernet MAC sub-block completes on-going activity and then sets EMACx_MR0[TXI] = 1 to enter the related Idle state.
- Software detects EMACx_MR0[TXI] = 1.
- Software performs the necessary EMAC configuration, keeping EMACx_MR0[TXE] = 0. The software can access only part of the EMAC registers marked with write access mode T in ~~Table 22-5, "EMAC0 Register Summary," on page 22-23~~*Table 22-5 EMAC0 Register Summary* on page 752.
- After all configuration is done, software can set EMACx_MR0[TXE] = 1.

Note: When Scenario 2 occurs, EMAC can still receive packets if EMACx_MR0[RXE] = 1. Scenarios 2 and 3 can occur simultaneously.

22.10.1.3 Scenario 3

- Software sets EMACx_MR0[RXE] = 0.

- The RXMAC component of the Ethernet MAC sub-block completes on-going activity and then sets EMACx_MR0[RXI] = 1 to enter the related Idle state.
- Software detects EMACx_MR0[RXI] = 1.
- Software performs the necessary EMAC configuration, keeping EMACx_MR0[RXE] = 0. The software can access only part of EMAC registers marked with write access mode R in ~~Table 22-5, "EMAC0 Register Summary," on page 22-23~~*Table 22-5 EMAC0 Register Summary* on page 752.
- After all configuration is done, software can set EMACx_MR0[RXE] = 1.

Note: When Scenario 3 occurs, EMAC can still transmit packets if EMACx_MR0[TXE] = 1. Scenarios 2 and 3 can occur simultaneously.

23. Serial Port Operations

The PPC440GP contains two universal asynchronous receiver/transmitters (UARTs) which provide two full-duplex serial interfaces to support communications with serial peripheral devices. Each UART is compatible with the National Semiconductor (NS) 16750 chip, and includes a 64-byte send and a 64-byte receive FIFO.

Features of the UART include:

- Compatible with the NS 16750
- 64-byte send FIFO, 64-byte receive FIFO
- Full duplex operation
- Programmable baud rate generator
- Supports 5- to 8-bit word size, 1 or 2 stop bits, even, odd, or no parity
- One 8-wire interface (UART0) and one 4-wire interface (UART1)
- Hardware flow control is selectable

The UART performs serial-to-parallel conversion on data characters received from a peripheral device, and parallel-to-serial conversion on data characters received from the processor. The processor can read the complete status of the UART at any time during the functional operation. Status information reported includes the type and condition of the transfer operations being performed by the UART, as well as any error conditions, such as parity, overrun, framing, and break interrupt.

This UART is functionally identical to NS16750 in character mode (on power up it will be in this mode), and can be put into FIFO mode to relieve the processor of excessive software overhead. Here, internal FIFOs are activated allowing 64 bytes (plus 3 bits per byte of error data in the RCVR FIFO) to be stored in both receive and transmit modes.

The source of the UART serial clock input is selected in Chip Control Register ~~0~~ 0(CPC0_CR0[U0EC:U1EC]) bits 9 and 10. Either the internal serial clock or an external serial clock can be selected. A programmable baud rate generator is included that is capable of dividing the UART serial clock input by a divisor of 1 to (2^{16} – 1) and producing the 16× clock required for driving the UART internal transmitter/receiver logic. The internal serial clock input is derived from the PLB clock by a divisor specified in CPC0_CR0[UDIV]. See ~~Chapter 13, "Clocking," on page 13-1~~*Section 13 Clocking* on page 407 for additional information.

The UART has an interrupt system that can be programmed to the user's requirements, helping to minimize the computing required to handle the communications link. UART interrupts are capable of triggering an interrupt request to the PPC440GP interrupt controller.

23.1 Functional Description

- Runs NS 16750 software
- Registers are identical to the NS16750 register set
- After reset, all registers are identical to the NS16750 register set
- Complete status reporting capability
- Transmitter and receiver are each buffered with 64-byte FIFOs when FIFO mode selected
- Can add/delete standard asynchronous communication bits such as start, stop, and parity to/from the serial data

- When in character mode, holding and shift registers eliminate the need for precise synchronization between the processor and serial data
- Full prioritized interrupt system controls
- Independently controlled transmit, receive, line status, and data set interrupts
- Programmable baud rate generator divides the UART serial clock input by 1 to (2^{16}–1) and generates the 16x clock:

  Baud rate (bps) = (Serial Clock Input) / (16 x Decimal Divisor)
- Receiver uses 5-way oversampling as follows: it samples each serial bit five times, and if at least three of the samples are 1's, the bit is determined to be a 1, otherwise it is a 0
- Fully programmable serial-interface characteristics:
  - 5-, 6-, 7-, or 8-bit characters
  - Even, odd, or no parity bit generation and detection
  - 1-,1.5-, or 2-stop bit generation
  - Variable baud rate
- Line break generation and detection, and false start bit detection
- Internal diagnostic capability:
  - Loopback controls for communications link fault isolation
  - Break, parity, overrun, framing error simulation

23.2 Serial Port Clocking

Each of the two PPC440GP UARTs can be clocked individually from an external serial clock or from a single internally generated serial clock. The internally generated serial clock is derived from the PLB clock, and may only be an integer (n) fraction of that clock where n is in the range from 1 to 32.

The choice of serial clock frequency affects the serial communications error rate. If an external clock of 1.8432 MHz (or some multiple of this frequency) is used, the error rate approaches zero. However, when using the internally generated clock only certain clock frequencies are possible, which results in a small, non-zero error rate in all cases.

The optimum serial clock frequency is determined from the following relationship:

Serial Clock = Baud Rate $\times$ 16 $\times$ UART Divisor

Acceptable baud rates are always integral multiples of 300 (for example, 1200 = 4 $\times$ 300). Table 23-1 shows optimum UART divisor and divide ratios for a range of possible baud rates. This information is provided for various clock frequencies. Note that the serial clock frequency must be less than half the OPB frequency. The

UART divisor is programmed in UARTx_DLM and UARTx_DLL (see ~~"Divisor Latch LSB and MSB Registers (UARTx_DLL, UARTx_DLM)" on page 23-17~~*Divisor Latch LSB and MSB Registers (UARTx_DLL, UARTx_DLM)* on page 791). The value range is 1 to (2^16-1)=65535.

Table 23-1. Baud Rate Settings

Ideal Baud Rate	ClkGen Serial Div	Serial Clock	UART Divisor	Actual Baud Rate	Baud Error
PLB Clock = 75Hz					
1200	21	3.5714e+06	186	1200.076805	0.0064004
2400	21	3.5714e+06	93	2400.153610	0.0064004
4800	16	4.6875e+06	61	4802.766393	0.0576332
9600	8	9.3750e+06	61	9605.532787	0.0576332
19200	7	1.0714e+07	35	19132.653061	0.3507653
28800	18	4.1667e+06	9	28935.185185	0.4693930
33600	28	2.6786e+06	5	33482.142857	0.3507653
38400	11	6.8182e+06	11	38739.669421	0.8845558
57600	9	8.3333e+06	9	57870.370370	0.4693930
115200	20	3.7500e+06	2	117187.500000	1.7252604
307200	15	5.0000e+06	1	312500.000000	1.7252604
PLB Clock = 83.3Hz					
1200	20	4.1667e+06	217	1200.076757	0.0063964
2400	10	8.3333e+06	217	2400.153514	0.0063964
4800	5	1.6667e+07	217	4800.307028	0.0063964
9600	32	2.6042e+06	17	9574.141774	0.2693565
19200	17	4.9020e+06	16	19148.283548	0.2693565
28800	18	4.6296e+06	10	28935.184028	0.4693890
33600	31	2.6882e+06	5	33602.149194	0.0063964
38400	17	4.9020e+06	8	38296.567096	0.2693565
57600	18	4.6296e+06	5	57870.368056	0.4693890
115200	9	9.2593e+06	5	115740.736111	0.4693890
307200	6	1.3889e+07	3	289351.840278	5.8099478
PLB Clock = 100Hz					
1200	28	3.5714e+06	186	1200.076805	0.0064004
2400	28	3.5714e+06	93	2400.153610	0.0064004
4800	14	7.1429e+06	93	4800.307220	0.0064004
9600	7	1.4286e+07	93	9600.614439	0.0064004
19200	25	4.0000e+06	13	19230.769231	0.1602564
28800	31	3.2258e+06	7	28801.843318	0.0064004
33600	31	3.2258e+06	6	33602.150538	0.0064004
38400	27	3.7037e+06	6	38580.246914	0.4693930

Table 23-1. Baud Rate Settings (continued)

Ideal Baud Rate	ClkGen Serial Div	Serial Clock	UART Divisor	Actual Baud Rate	Baud Error
57600	12	8.3333e+06	9	57870.370370	0.4693930
115200	6	1.6667e+07	9	115740.740741	0.4693930
307200	20	5.0000e+06	1	312500.000000	1.7252604
PLB Clock = 125Hz					
1200	30	4.1667e+06	217	1200.076805	0.0064004
2400	15	8.3333e+06	217	2400.153610	0.0064004
4800	22	5.6818e+06	74	4798.832924	0.0243141
9600	22	5.6818e+06	37	9597.665848	0.0243141
19200	11	1.1364e+07	37	19195.331695	0.0243141
28800	17	7.3529e+06	16	28722.426471	0.2693525
33600	29	4.3103e+06	8	33674.568966	0.2219314
38400	29	4.3103e+06	7	38485.221675	0.2219314
57600	17	7.3529e+06	8	57444.852941	0.2693525
115200	17	7.3529e+06	4	114889.705882	0.2693525
307200	25	5.0000e+06	1	312500.000000	1.7252604
PLB Clock = 133.3Hz					
1200	32	4.1667e+06	217	1200.076805	0.0064004
2400	28	4.7619e+06	124	2400.153610	0.0064004
4800	28	4.7619e+06	62	4800.307220	0.0064004
9600	28	4.7619e+06	31	9600.614439	0.0064004
19200	14	9.5238e+06	31	19201.228878	0.0064004
28800	17	7.8431e+06	17	28835.063436	0.1217480
33600	31	4.3011e+06	8	33602.150537	0.0064004
38400	7	1.9048e+07	31	38402.457756	0.0064004
57600	5	2.6667e+07	29	57471.264366	0.2234994
115200	24	5.5556e+06	3	115740.740738	0.4693930
307200	27	4.9383e+06	1	308641.975301	0.4693930
PLB Clock = 150Hz					
1200	13	1.1538e+07	601	1199.923205	0.0063996
2400	21	7.1429e+06	186	2400.153610	0.0064004
4800	21	7.1429e+06	93	4800.307220	0.0064004
9600	16	9.3750e+06	61	9605.532787	0.0576332
19200	8	1.8750e+07	61	19211.065574	0.0576332
28800	13	1.1538e+07	25	28846.153846	0.1602564
33600	9	1.6667e+07	31	33602.150538	0.0064004
38400	7	2.1429e+07	35	38265.306122	0.3507653

Table 23-1. Baud Rate Settings (continued)

Ideal Baud Rate	ClkGen Serial Div	Serial Clock	UART Divisor	Actual Baud Rate	Baud Error
57600	18	8.3333e+06	9	57870.370370	0.4693930
115200	9	1.6667e+07	9	115740.740741	0.4693930
307200	30	5.0000e+06	1	312500.000000	1.7252604
PLB Clock = 166.6Hz					
1200	20	8.3333e+06	434	1200.076757	0.0063964
2400	20	8.3333e+06	217	2400.153514	0.0063964
4800	10	1.6667e+07	217	4800.307028	0.0063964
9600	5	3.3333e+07	217	9600.614055	0.0063964
19200	32	5.2083e+06	17	19148.283548	0.2693565
28800	19	8.7719e+06	19	28855.031163	0.1910804
33600	31	5.3763e+06	10	33602.149194	0.0063964
38400	17	9.8039e+06	16	38296.567096	0.2693565
57600	18	9.2593e+06	10	57870.368056	0.4693890
115200	18	9.2593e+06	5	115740.736111	0.4693890
307200	17	9.8039e+06	2	306372.536765	0.2693565

23.3 UART Registers

UART registers are accessed via memory to 0x14000_0xYY where X=2 for UART0 and X=3 for UART1.

Table 23-2. UART Configuration Registers

Register	Address	Access	Description	Page
UARTx_RBR	0x14000_0X00 [1]	R	UART x Receiver Buffer Register	783
UARTx_THR	0x14000_0X00 [1]	W	UART x Transmitter Holding Register	783
UARTx_IER	0x14000_0X01 [1]	R/W	UART x Interrupt Enable Register	783
UARTx_IIR	0x14000_0X02	R	UART x Interrupt Identification Register	784
UARTx_FCR	0x14000_0X02	W	UART x FIFO Control Register	786
UARTx_LCR	0x14000_0X03	R/W	UART x Line Control Register	787
UARTx_MCR	0x14000_0X04	R/W	UART x Modem Control Register	788
UARTx_LSR	0x14000_0X05	R/W	UART x Line Status Register	789
UARTx_MSR	0x14000_0X06	R/W	UART x Modem Status Register	790
UARTx_SCR	0x14000_0X07	R/W	UART x Scratch Register	791
UARTx_DLL	0x14000_0X00 [1]	R/W	UART x Divisor Latch (LSB)	791
UARTx_DLM	0x14000_0X01 [1]	R/W	UART x Divisor Latch (MSB)	791

1. UARTx_LCR[DLAB] controls the function accessed through registers 0x14000_0X00 and 0x14000_0X01. When UARTx_LCR[DLAB] is 0, access is enabled to the Receiver/Transmitter registers and the Interrupt Enable register. When UARTx_LCR[DLAB] is a 1, access is enabled to the Divisor Latch registers.

The system programmer may access any of the UART registers via the processor. These registers control all UART operations including transmission and reception of data. In PPC440GP there are two UARTs, designated 0 (8-wire interface) and 1 (4-wire interface). In the following sections, the registers are specified with a generic name where x represents 0 or 1. For example, the Line Control Register appears as a UARTx_LCR.

For UART1, two of the four wires are TX and RX. The remaining two wires can be programmed as a combination of DTR and DSR, or CTS and RTS in CPC0_CR0(DCS:RDS). DCD and RI are not available on the 4-wire interface.

23.3.1 Receiver Buffer Registers (UARTx_RBR)

~~Figure 23-1~~Figure 23-1 describes UARTx_RBR bit definitions.

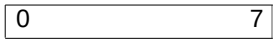


Figure 0-1. UART Receiver Buffer Registers (UARTx_RBR)

0:7		Data bit
Note: UARTx_RBR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.		

Figure 23-1. UART Receiver Buffer Registers (UARTx_RBR)

0:7		Data bit
Note: UARTx_RBR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.		

23.3.2 Transmitter Holding Registers (UARTx_THR)

~~Figure 23-2~~Figure 23-2 describes UARTx_THR bit definitions.

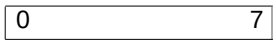


Figure 0-2. UART Transmitter Holding Registers (UARTx_THR)

0:7		Data bit
Note: UARTx_THR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.		

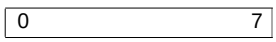


Figure 23-2. UART Transmitter Holding Registers (UARTx_THR)

0:7		Data bit
Note: UARTx_THR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.		

23.3.3 Interrupt Enable Registers (UARTx_IER)

Five UART interrupts on four priority levels are enabled via the Interrupt Enable Register, UARTx_IER. Any of the five interrupts can be used to surface a UART interrupt to the PPC440GP interrupt controller. Each interrupt can be enabled by setting its appropriate bit. Resetting UARTx_IER[4:7] totally disables the UART interrupt system. Disabling an interrupt prevents it from being shown as active in the UARTx_IIR and prevents it from signaling a UART interrupt to the PPC440GP interrupt controller. See ~~Table 23-3, "Interrupt Priority Level," on page 23-9~~*Table 23-3 Interrupt Priority Level* on page 785. ~~Figure 23-3~~Figure 23-3 describes UARTx_IER bit definitions.

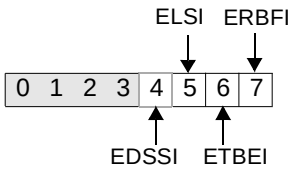


Figure 0-3. UART Interrupt Enable Registers (UARTx_IER)

0:3		Reserved	Always 0.
4	EDSSI	Modem Status Interrupt 0 Disable modem status interrupt 1 Enable modem status interrupt	
5	ELSI	Receiver Line Status Interrupt enable 0 Disable receiver line status interrupt 1 Enable receiver line status interrupt	
6	ETBEI	Transmitter Holding Register Empty Interrupt enable 0 Disable transmitter holding register empty interrupt 1 Enable transmitter holding register empty interrupt	
7	ERBFI	Received Data Available Interrupt enable 0 Disable received data available interrupt 1 Enable received data available interrupt	In FIFO mode, timeout interrupts follow the enable/disable state of ERBFI.
Note: UARTx_IER is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.			

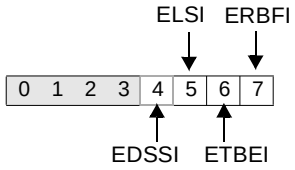


Figure 23-3. UART Interrupt Enable Registers (UARTx_IER)

0:3		Reserved	Always 0.
4	EDSSI	Modem Status Interrupt 0 Disable modem status interrupt 1 Enable modem status interrupt	
5	ELSI	Receiver Line Status Interrupt enable 0 Disable receiver line status interrupt 1 Enable receiver line status interrupt	
6	ETBEI	Transmitter Holding Register Empty Interrupt enable 0 Disable transmitter holding register empty interrupt 1 Enable transmitter holding register empty interrupt	
7	ERBFI	Received Data Available Interrupt enable 0 Disable received data available interrupt 1 Enable received data available interrupt	In FIFO mode, timeout interrupts follow the enable/disable state of ERBFI.
Note: UARTx_IER is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.			

23.3.4 Interrupt Identification Registers (UARTx_IIR)

The UART prioritizes interrupts into four levels which are recorded in the Interrupt Identification Register. The interrupt types in the order of their priority are as follows:

1. Receiver line status
2. Received data available and character timeout indication
3. Transmitter holding register empty
4. Modem status

Table 23-3 lists the interrupt priority levels.

Table 23-3. Interrupt Priority Level

IIR Bit 4	IIR Bit 5	IIR Bit 6	Priority Level	Interrupt Type	Interrupt Source	Interrupt Reset Control
0	1	1	1	Receiver Line Status	Overrun, Parity or Framing Error, or Break Interrupt.	Read LSR.
0	1	0	2	Received Data Available	Receiver data available or trigger level reached.	Read RBR, or FIFO drops below trigger level.
1	1	0	2	Character Timeout Indication	No characters have been removed from or input to the receiver FIFO during the last four character times and it contains at least one character during this time.	Read RBR.
0	0	1	3	Transmitter Holding Register Empty	Transmitter Holding Register Empty.	Read IIR (if source of interrupt) or write THR.
0	0	0	4	Modem Status	Clear to Send, Data Set Ready, Ring Indicator or Data Carrier Detect.	Read MSR.

When the processor accesses UARTx_IIR, the UART records new interrupts, but does not change its current contents until the access by the processor is complete. The UART indicates the highest priority interrupt pending to the PPC440GP interrupt controller via the IIR. ~~Figure 23-4~~Figure 23-4 describes UARTx_IRR bit definitions~~.~~.

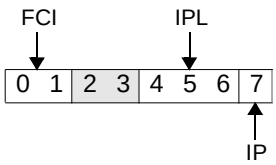


Figure 0-4. UART Interrupt Identification Registers (UARTx_IIR)

0:1	FCI	FIFO Control Indicator 00 FIFOs disabled (UARTx_FCR[FC] = 0) 01 Reserved 10 Reserved 11 FIFOs enabled (UARTx_FCR[FC] = 1)
2:3		Reserved
4:6	IPL	Interrupt Priority Level 000 Priority level 4 001 Priority level 3 010 Priority level 2 011 Priority level 1 100 Reserved 101 Reserved 110 Priority level 2 111 Reserved Note: Priority 1 is highest priority.

7	IP	Interrupt Pending 0 Interrupt is pending 1 No interrupt pending	When set to 0, IIR contents can be used as a pointer to the appropriate interrupt service routine.
Note: UARTx_IIR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.			

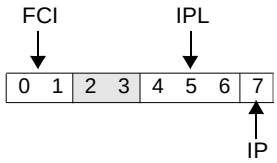


Figure 23-4. UART Interrupt Identification Registers (UARTx_IIR)

0:1	FCI	FIFO Control Indicator 00 FIFOs disabled (UARTx_FCR[FC] = 0) 01 Reserved 10 Reserved 11 FIFOs enabled (UARTx_FCR[FC] = 1)	
2:3		Reserved	
4:6	IPL	Interrupt Priority Level 000 Priority level 4 001 Priority level 3 010 Priority level 2 011 Priority level 1 100 Reserved 101 Reserved 110 Priority level 2 111 Reserved	Note: Priority 1 is highest priority.
7	IP	Interrupt Pending 0 Interrupt is pending 1 No interrupt pending	When set to 0, IIR contents can be used as a pointer to the appropriate interrupt service routine.
Note: UARTx_IIR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.			

23.3.5 FIFO Control Registers (UARTx_FCR)

The FIFO control register has the same address as the IIR and is a write-only register. This register is used to perform FIFO control operations such as selecting the type of DMA signaling, setting the receiver FIFO trigger levels, clearing the FIFOs, and enabling the FIFO.

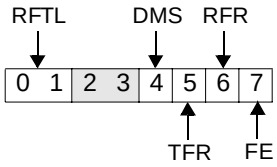


Figure 0-5. UART FIFO Control Registers (UARTx_FCR)

0:1	RFTL	Receiver FIFO Trigger Level 00 01 byte 01 16 bytes 10 32 bytes 11 56 bytes	
2:3		Reserved	
4	DMS	DMA Mode Select 0 Mode 0 = single transfer 1 Mode 1 = multiple transfers	Select single or multiple transfer mode if UARTx_FCR[7] = 1.
5	TFR	Transmitter FIFO Reset 0 Operation complete 1 Reset the transmitter FIFO	A 1 written to this bit clears all bytes in the transmitter FIFO and resets all of its counter logic to 0. The transmitter shift register is not cleared. This bit is self-clearing.
6	RFR	Receiver FIFO Reset 0 Operation complete 1 Reset the receiver FIFO	A 1 written to this bit clears all bytes in the receiver FIFO and resets all of its counter logic to 0. The receiver shift register is not cleared. This bit is self-clearing.
7	FE	FIFO Enable 0 Disable FIFOs 1 Enable FIFOs	When set to 1, both the receiver and transmitter FIFOs are enabled. When set to 0, both receiver and transmitter FIFOs are reset. Data is automatically cleared from both FIFOs when changing to and from FIFO and 16450 modes. Programming other bits will be ignored if this bit is not a 1.
Note: UARTx_FCR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.			

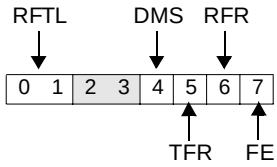


Figure 23-5. UART FIFO Control Registers (UARTx_FCR)

0:1	RFTL	Receiver FIFO Trigger Level 00 01 byte 01 16 bytes 10 32 bytes 11 56 bytes	
2:3		Reserved	
4	DMS	DMA Mode Select 0 Mode 0 = single transfer 1 Mode 1 = multiple transfers	Select single or multiple transfer mode if UARTx_FCR[7] = 1.
5	TFR	Transmitter FIFO Reset 0 Operation complete 1 Reset the transmitter FIFO	A 1 written to this bit clears all bytes in the transmitter FIFO and resets all of its counter logic to 0. The transmitter shift register is not cleared. This bit is self-clearing.
6	RFR	Receiver FIFO Reset 0 Operation complete 1 Reset the receiver FIFO	A 1 written to this bit clears all bytes in the receiver FIFO and resets all of its counter logic to 0. The receiver shift register is not cleared. This bit is self-clearing.
7	FE	FIFO Enable 0 Disable FIFOs 1 Enable FIFOs	When set to 1, both the receiver and transmitter FIFOs are enabled. When set to 0, both receiver and transmitter FIFOs are reset. Data is automatically cleared from both FIFOs when changing to and from FIFO and 16450 modes. Programming other bits will be ignored if this bit is not a 1.
Note: UARTx_FCR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.			

23.3.6 Line Control Registers (UARTx_LCR)

The system programmer uses the line control register (LCR) to specify the format of the asynchronous data communications exchange and to set the Divisor Latch Access bit. The contents of the LCR can also be read by the processor. The read capability simplifies system programming, and eliminates the need for separate storage of the line characteristics in system memory.

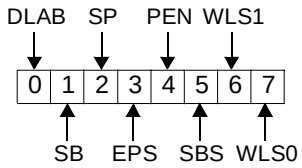


Figure 0-6. UART Line Control Registers (UARTx_LCR)

0	DLAB	Divisor Latch Access Bit 0 Address RBR, THR and IER with LTADR2-0 for read or write operation 1 Address Divisor Latches with LTADR2-0 for read or write operation	
1	SB	Set Break 0 Disable Break 1 Enable Break	Causes a break condition to be transmitted to the UART when the core is receiving. SOUT is forced to the spacing state (0). This bit acts only on SOUT and has no effect on the transmitter logic.
2	SP	Sticky Parity 0 Disable sticky parity 1 Enable sticky parity	If UARTx_LCR[EPS] = 1 and UARTx_LCR[PEN] = 1, the parity bit is transmitted and checked as 0. If UARTx_LCR [EPS] = 0 and UARTx_LCR[PEN] = 1,the parity bit is transmitted and checked as 1.
3	EPS	Even Parity Select 0 Generate odd parity 1 Generate even parity	This bit is significant only if UARTx_LCR[PEN] = 1.
4	PEN	Parity Enable 0 Disable parity checking 1 Enable parity checking	
5	SBS	Stop Bit Select 0 Characters have 1 stop bit 1 Characters have 1.5 or 2 stop bits	If UARTx_LCR[WPS1,WPS0] = 00, characters have 1.5 stop bits. For any other value of UARTx_LCR[WPS1,WPS0], characters have 2 stop bits. The receiver checks the first stop bit only, regardless of how many stop bits are selected.
6	WLS1	Word Length Select Bits 1 00 Use 5-bit characters 01 Use 6-bit characters 10 Use 7-bit characters 11 Use 8-bit characters	
7	WLS0	Word Length Select Bits 0 00 Use 5-bit characters 01 Use 6-bit characters 10 Use 7-bit characters 11 Use 8-bit characters	

Note: UARTx_LCR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.

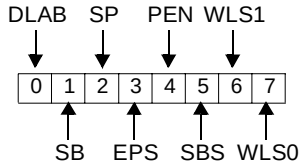


Figure 23-6. UART Line Control Registers (UARTx_LCR)

0	DLAB	Divisor Latch Access Bit 0 Address RBR, THR and IER with LTADR2-0 for read or write operation 1 Address Divisor Latches with LTADR2-0 for read or write operation	
1	SB	Set Break 0 Disable Break 1 Enable Break	Causes a break condition to be transmitted to the UART when the core is receiving. SOUT is forced to the spacing state (0). This bit acts only on SOUT and has no effect on the transmitter logic.
2	SP	Sticky Parity 0 Disable sticky parity 1 Enable sticky parity	If UARTx_LCR[EPS] = 1 and UARTx_LCR[PEN] = 1, the parity bit is transmitted and checked as 0. If UARTx_LCR [EPS] = 0 and UARTx_LCR[PEN] = 1,the parity bit is transmitted and checked as 1.
3	EPS	Even Parity Select 0 Generate odd parity 1 Generate even parity	This bit is significant only if UARTx_LCR[PEN] = 1.
4	PEN	Parity Enable 0 Disable parity checking 1 Enable parity checking	
5	SBS	Stop Bit Select 0 Characters have 1 stop bit 1 Characters have 1.5 or 2 stop bits	If UARTx_LCR[WPS1,WPS0] = 00, characters have 1.5 stop bits. For any other value of UARTx_LCR[WPS1,WPS0], characters have 2 stop bits. The receiver checks the first stop bit only, regardless of how many stop bits are selected.
6	WLS1	Word Length Select Bits 1 00 Use 5-bit characters 01 Use 6-bit characters 10 Use 7-bit characters 11 Use 8-bit characters	
7	WLS0	Word Length Select Bits 0 00 Use 5-bit characters 01 Use 6-bit characters 10 Use 7-bit characters 11 Use 8-bit characters	
Note: UARTx_LCR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.			

23.3.7 Modem Control Registers (UARTx_MCR)

The interface between the modem, data set, or peripheral device emulating a modem, and the UART, is controlled by the Modem Control Register (UARTx_MCR).

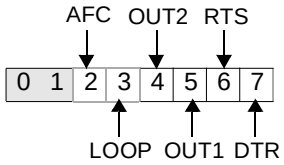


Figure 0-7. UART Modem Control Registers (UARTx_MCR)

0:1		Reserved	Always 0.
2	AFC	Auto Flow Control 0 Hardware flow control disabled 1 Hardware flow control enabled	
3	LOOP	Loopback Mode 0 Disabled 1 Enabled	Provides a local loopback feature for diagnostic testing of the UART. The following occurs: 1. SOUT is set to the marking state (logic 1) SIN is disconnected. 2. The output of the transmitter shift register feeds the input of the receiver shift register. 3. The four modem control inputs $\overline{DSR}$, $\overline{CTS}$, $\overline{RI}$, and $\overline{DCD}$ are disconnected. 4. The four modem control outputs $\overline{DTR}$, $\overline{RTS}$, $\overline{OUT1}$, and $\overline{OUT2}$ are set to a logic 1 (their inactive state). 5. The four modem control outputs are connected internally to the four modem control inputs. Transmitted data is immediately received to verify the UART transmit and receive data paths. Receiver and transmitter interrupts are operational. Their sources are external to the UART. Also operational are the modem control interrupts, but their source is the low-order 4 bits of UARTx_MCR instead of the modem control inputs to the UART. UARTx_IER still controls the interrupts.
4	OUT2	User Output 2 0 $\overline{OUT2}$ inactive (1) 1 $\overline{OUT2}$ active (0)	The $\overline{OUT2}$ bit may be written and read but it provides no function
5	OUT1	User Output 1 0 $\overline{OUT1}$ inactive (1) 1 $\overline{OUT1}$ active (0)	The $\overline{OUT1}$ bit may be written and read but it provides no function
6	RTS	Request To Send 0 $\overline{RTS}$ inactive (1) 1 $\overline{RTS}$ active (0)	

7	DTR	Data Terminal Ready 0 $\overline{\text{DTR}}$ inactive (1) 1 $\overline{\text{DTR}}$ active (0)
Note: UARTx_MCR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.		

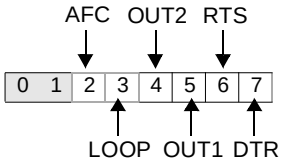


Figure 23-7. UART Modem Control Registers (UARTx_MCR)

0:1		Reserved	Always 0.
2	AFC	Auto Flow Control 0 Hardware flow control disabled 1 Hardware flow control enabled	
3	LOOP	Loopback Mode 0 Disabled 1 Enabled	Provides a local loopback feature for diagnostic testing of the UART. The following occurs: 1. SOUT is set to the marking state (logic 1) SIN is disconnected. 2. The output of the transmitter shift register feeds the input of the receiver shift register. 3. The four modem control inputs $\overline{\text{DSR}}$, $\overline{\text{CTS}}$, $\overline{\text{RI}}$, and $\overline{\text{DCD}}$ are disconnected. 4. The four modem control outputs $\overline{\text{DTR}}$, $\overline{\text{RTS}}$, $\overline{\text{OUT1}}$, and $\overline{\text{OUT2}}$ are set to a logic 1 (their inactive state). 5. The four modem control outputs are connected internally to the four modem control inputs. Transmitted data is immediately received to verify the UART transmit and receive data paths. Receiver and transmitter interrupts are operational. Their sources are external to the UART. Also operational are the modem control interrupts, but their source is the low-order 4 bits of UARTx_MCR instead of the modem control inputs to the UART. UARTx_IER still controls the interrupts.
4	OUT2	User Output 2 0 $\overline{\text{OUT2}}$ inactive (1) 1 $\overline{\text{OUT2}}$ active (0)	The $\overline{\text{OUT2}}$ bit may be written and read but it provides no function
5	OUT1	User Output 1 0 $\overline{\text{OUT1}}$ inactive (1) 1 $\overline{\text{OUT1}}$ active (0)	The $\overline{\text{OUT1}}$ bit may be written and read but it provides no function
6	RTS	Request To Send 0 $\overline{\text{RTS}}$ inactive (1) 1 $\overline{\text{RTS}}$ active (0)	
7	DTR	Data Terminal Ready 0 $\overline{\text{DTR}}$ inactive (1) 1 $\overline{\text{DTR}}$ active (0)	
Note: UARTx_MCR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.			

23.3.8 Line Status Registers (UARTx_LSR)

Information concerning the data transfer is held for the processor in this register. Bits 3 through 6 are conditions that produce a receiver line status interrupt whenever the condition corresponding to the active bit is detected and the interrupt is enabled. This register is intended for read operations only and writing is not recommended.

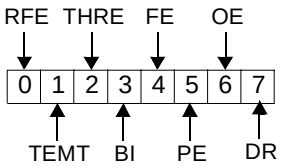


Figure 0-8. UART Line Status Registers (UARTx_LSR)

0	RFE	Receiver FIFO Error Indicator 0 In FIFO mode, reset to 0 when the processor reads the UARTx_LSR, provided there are no subsequent errors in the FIFO. 1 There are one or more instances of parity error, framing error or break indication in the FIFO.	Always 0 in 16450 mode.
1	TEMT	Transmitter Empty Indicator 0 Reset to 0 whenever the THR or the transmitter shift register contain a character. In FIFO mode, it is reset to 0 whenever the transmitter FIFO or the transmitter shift register contain a character. 1 Set to 1 when the THR and the Transmitter shift register are both empty. In FIFO mode, it is set to 1 when the transmitter FIFO and the transmitter shift register are both empty.	
2	THRE	Transmitter Holding Register Empty Indicator 0 Concurrent reset to 0 with the loading of the THR by the processor. In FIFO mode it is reset to 0 when at least one byte is written to the transmitter FIFO. 1 Set to 1 when the UART is ready to accept a new character for transmission. In FIFO mode, this bit is set when the transmitter FIFO is empty.	When UARTx_IER[THRE] = 1, the UART issues an interrupt to the interrupt controller. This bit is set to 1 when a character is transferred from the THR to the transmitter shift register.

3	BI	Break Interrupt Indicator. 0 Reset to 0 whenever processor reads Line Status Register (LSR). 1 Set to 1 whenever the received data input is held at the spacing level (0) for longer than a full word transmission time.	The full word transmission time is the time required for the start bit, data bits (can be 5–8 bits), parity and stop bits. In FIFO mode, this error is revealed to the processor when the character this error is associated with is at the top of the FIFO. Only one 0 character is loaded into the receiver FIFO when a break occurs. After the next valid start bit is received and has gone into the marking state, the next character transfer is enabled. Error causes a Receiver Line Status Interrupt.
4	FE	Framing Error Indicator. 0 Reset to 0 whenever processor reads LSR. 1 Set to 1 whenever stop bit following the last data bit or parity bit is detected as 0 (spacing level).Indicates that a valid stop bit was not found in the received character.	Error causes a Receiver Line Status Interrupt.
5	PE	Parity Error Indicator. 0 Reset to 0 whenever processor reads UARTx_LSR. 1 Indicates that the received data character does not have the correct parity as determined by the even parity select bit (UARTx_LCR.[EPS]). Set to 1 upon detection of a parity error.	In FIFO mode, this error is revealed to the processor when the character this error is associated with is at the top of the FIFO. Error causes a Receiver Line Status Interrupt.
6	OE	Overrun Error Indicator. 0 Reset to 0 whenever processor reads UARTx_LSR. 1 Data in the RBR was read by the processor before the next character was transferred into the UARTx_RBR, hence the original data was lost.	In FIFO mode, if the incoming data continues to fill the FIFO beyond the trigger level, an OE occurs only after the FIFO is completely full and the entire next character has been received in the receiver shift register. The processor is informed of the OE immediately upon occurrence. The character in the shift register will be overwritten and will not be transferred to the FIFO. Error causes a Receiver Line Status Interrupt.
7	DR	Receiver Data Ready Indicator. 0 Reset to 0 when all data has been read from the receiver FIFO or the UARTx_RBR. 1 An entire incoming character has been received into the UARTx_RBR or receiver FIFO.	
Note: UARTx_LSR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.			

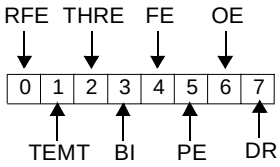


Figure 23-8. UART Line Status Registers (UARTx_LSR)

0	RFE	Receiver FIFO Error Indicator 0 In FIFO mode, reset to 0 when the processor reads the UARTx_LSR, provided there are no subsequent errors in the FIFO. 1 There are one or more instances of parity error, framing error or break indication in the FIFO.	Always 0 in 16450 mode.
1	TEMT	Transmitter Empty Indicator 0 Reset to 0 whenever the THR or the transmitter shift register contain a character. In FIFO mode, it is reset to 0 whenever the transmitter FIFO or the transmitter shift register contain a character. 1 Set to 1 when the THR and the Transmitter shift register are both empty. In FIFO mode, it is set to 1 when the transmitter FIFO and the transmitter shift register are both empty.	
2	THRE	Transmitter Holding Register Empty Indicator 0 Concurrent reset to 0 with the loading of the THR by the processor. In FIFO mode it is reset to 0 when at least one byte is written to the transmitter FIFO. 1 Set to 1 when the UART is ready to accept a new character for transmission. In FIFO mode, this bit is set when the transmitter FIFO is empty.	When UARTx_IER[THRE] = 1, the UART issues an interrupt to the PPC440GP interrupt controller. This bit is set to 1 when a character is transferred from the THR to the transmitter shift register.
3	BI	Break Interrupt Indicator. 0 Reset to 0 whenever processor reads Line Status Register (LSR). 1 Set to 1 whenever the received data input is held at the spacing level (0) for longer than a full word transmission time.	The full word transmission time is the time required for the start bit, data bits (can be 5–8 bits), parity and stop bits. In FIFO mode, this error is revealed to the processor when the character this error is associated with is at the top of the FIFO. Only one 0 character is loaded into the receiver FIFO when a break occurs. After the next valid start bit is received and has gone into the marking state, the next character transfer is enabled. Error causes a Receiver Line Status Interrupt.
4	FE	Framing Error Indicator. 0 Reset to 0 whenever processor reads LSR. 1 Set to 1 whenever stop bit following the last data bit or parity bit is detected as 0 (spacing level).Indicates that a valid stop bit was not found in the received character.	Error causes a Receiver Line Status Interrupt.
5	PE	Parity Error Indicator. 0 Reset to 0 whenever processor reads UARTx_LSR. 1 Indicates that the received data character does not have the correct parity as determined by the even parity select bit (UARTx_LCR.[EPS]). Set to 1 upon detection of a parity error.	In FIFO mode, this error is revealed to the processor when the character this error is associated with is at the top of the FIFO. Error causes a Receiver Line Status Interrupt.

6	OE	Overrun Error Indicator. 0 Reset to 0 whenever processor reads UARTx_LSR. 1 Data in the RBR was read by the processor before the next character was transferred into the UARTx_RBR, hence the original data was lost.	In FIFO mode, if the incoming data continues to fill the FIFO beyond the trigger level, an OE occurs only after the FIFO is completely full and the entire next character has been received in the receiver shift register. The processor is informed of the OE immediately upon occurrence. The character in the shift register will be overwritten and will not be transferred to the FIFO. Error causes a Receiver Line Status Interrupt.
7	DR	Receiver Data Ready Indicator. 0 Reset to 0 when all data has been read from the receiver FIFO or the UARTx_RBR. 1 An entire incoming character has been received into the UARTx_RBR or receiver FIFO.	
Note: UARTx_LSR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.			

23.3.9 Modem Status Registers (UARTx_MSR)

The processor can monitor the present state of the modem (or peripheral device) control lines by reading the Modem Status Register (UARTx_MSR). In addition, the UARTx_MSR has four bits to indicate if any of the modem (or peripheral device) control lines have changed state.

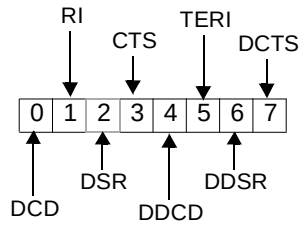


Figure 0-9. UART Modem Status Registers (UARTx_MSR)

0	DCD	Data Carrier Detect	In loopback mode (UARTx_MCR[LB] is 1), it is equivalent to UARTx_MCR[OUT2].
1	RI	Ring Indicator	In loopback mode (UARTx_MCR[LB] is 1), it is equivalent to UARTx_MCR[OUT1].
2	DSR	Data Set Ready	In loopback mode (UARTx_MCR[LB] is 1), it is equivalent to UARTx_MCR[DTR].
3	CTS	Clear To Send	In loopback mode (UARTx_MCR[LB] is 1), it is equivalent to UARTx_MCR[RTS].
4	DDCD	Delta Data Carrier Detect 0 Set when processor reads the Modem Status Register 1 $\overline{\text{DCD}}$ input changed state	Indicates that the $\overline{\text{DCD}}$ input to the UART has changed state since the processor last read the Modem Status Register. A modem status interrupt is generated.

5	TERI	Trailing Edge of Ring Indicator 0 Set when processor reads the Modem Status Register 1 $\overline{RI}$ input changed from 0 to 1	Indicates that the $\overline{RI}$ input to the UART changed from 0 to 1 since the processor last read the Modem Status Register. A modem status interrupt is generated.
6	DDSR	Delta Data Set Ready 0 Set when processor reads the Modem Status Register 1 $\overline{DSR}$ input changed state	Indicates that the $\overline{DSR}$ input to the UART has changed state since the processor last read the Modem Status Register. A modem status interrupt is generated.
7	DCTS	Delta Clear To Send 0 Set when processor reads the Modem Status Register 1 $\overline{CTS}$ input changed state	Indicates that the $\overline{CTS}$ input to the UART has changed state since the processor last read the Modem Status Register. A modem status interrupt is generated.
Note: UARTx_MSR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.			

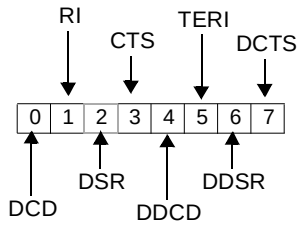


Figure 23-9. UART Modem Status Registers (UARTx_MSR)

0	DCD	Data Carrier Detect	In loopback mode (UARTx_MCR[LB] is 1), it is equivalent to UARTx_MCR[OUT2].
1	RI	Ring Indicator	In loopback mode (UARTx_MCR[LB] is 1), it is equivalent to UARTx_MCR[OUT1].
2	DSR	Data Set Ready	In loopback mode (UARTx_MCR[LB] is 1), it is equivalent to UARTx_MCR[DTR].
3	CTS	Clear To Send	In loopback mode (UARTx_MCR[LB] is 1), it is equivalent to UARTx_MCR[RTS].
4	DDCD	Delta Data Carrier Detect 0 Set when processor reads the Modem Status Register 1 $\overline{DCD}$ input changed state	Indicates that the $\overline{DCD}$ input to the UART has changed state since the processor last read the Modem Status Register. A modem status interrupt is generated.
5	TERI	Trailing Edge of Ring Indicator 0 Set when processor reads the Modem Status Register 1 $\overline{RI}$ input changed from 0 to 1	Indicates that the $\overline{RI}$ input to the UART changed from 0 to 1 since the processor last read the Modem Status Register. A modem status interrupt is generated.
6	DDSR	Delta Data Set Ready 0 Set when processor reads the Modem Status Register 1 $\overline{DSR}$ input changed state	Indicates that the $\overline{DSR}$ input to the UART has changed state since the processor last read the Modem Status Register. A modem status interrupt is generated.
7	DCTS	Delta Clear To Send 0 Set when processor reads the Modem Status Register 1 $\overline{CTS}$ input changed state	Indicates that the $\overline{CTS}$ input to the UART has changed state since the processor last read the Modem Status Register. A modem status interrupt is generated.
Note: UARTx_MSR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.			

23.3.10 Scratchpad Registers (UARTx_SCR)

A scratchpad register intended for use by the programmer as a temporary data location is provided in this UART. It does not control the UART operation in any way.

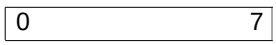


Figure 0-10. Scratchpad Registers (UARTx_SCR)

0:7		Data bits
Note: UARTx_SCR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.		

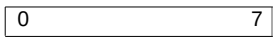


Figure 23-10. Scratchpad Registers (UARTx_SCR)

0:7		Data bits
Note: UARTx_SCR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.		

23.3.11 Divisor Latch LSB and MSB Registers (UARTx_DLL, UARTx_DLM)

The divisor latches are used to program the UART divisor used in generating the baud clock. A 16-bit divisor may be programmed through these registers. Access to these registers is provided by setting UARTx_LCR[DLAB] = 1. These registers have a power-on reset value of 0.

Figure 0-11. UART Baud-Rate Divisor Latch (MSB) Registers (UARTx_DLM)

0:7		Data bits
Note: UARTx_DLM is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.		

Figure 23-11. UART Baud-Rate Divisor Latch (MSB) Registers (UARTx_DLM)

0:7		Data bits
Note: UARTx_DLM is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.		

Figure 0-12. UART Baud-Rate Divisor Latch (LSB) Registers (UARTx_DLL)

8:15		Data bits
Note: UARTx_DLL is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.		

Figure 23-12. UART Baud-Rate Divisor Latch (LSB) Registers (UARTx_DLL)

8:15		Data bits
Note: UARTx_DLL is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.		

The UART divisor is calculated using the following formula:

UART Divisor = Serial Input Clock/(16 x Baud Rate)

For example, if the serial input clock= 11.0592MHz and a baud rate of 9600bps is required:

UART Divisor = Serial Input Clock/(16 x Baud Rate)

= 11,059,200/(16x9600)

= 72 = 0x48

For this example, UARTx_DLM should be programmed to 0 and UARTx_DLL register should be programmed to 0x48. Due to the error introduced by rounding, some baud rates cannot be generated at certain serial input clock frequencies. Table 23-4 lists some common baud rates and their corresponding Divisor Latch register values with a serial input clock of 11.0592MHz.

Table 23-4. Divisor Latch Settings for Certain Baud Rates

Baud Rate (bps)	Divisor Latch MSB	Divisor Latch LSB
9600	0x00	0x48
19200	0x00	0x24
28800	0x00	0x18
38400	0x00	0x12
57600	0x00	0x0C

23.4 FIFO Operation

This section discusses the various modes of operation including interrupt mode, polled mode, and sleep mode.

23.4.1 Interrupt Mode

23.4.1.1 Receiver

Receiver interrupts occur as described below when the receiver FIFO and receiver interrupts are enabled by setting UARTx_FCR[FE] = 1 and UARTx_IER[ERBFI] = 1.

The received data available interrupt is issued when the number of characters in the FIFO has reached the trigger level programmed into UARTx_FCR. This interrupt is reset to 0 when the FIFO character count drops below this trigger level.

The received data available indicator is issued when the number of characters in the FIFO has reached the trigger level programmed into UARTx_FCR. This indicator is reset to 0 when the FIFO character count drops below this trigger level.

The receiver line status interrupt (UARTx_IIR = 0xC6) is a top priority interrupt, whereas the received data available interrupt (UARTx_IIR = 0xC4) is a second priority interrupt.

Data Ready (UARTx_LSR[DR]) is set as soon as a character is transferred from the shift register to the receiver FIFO. This bit is reset when the FIFO is empty.

Receiver timeout interrupts will occur as described below when the receiver FIFO and receiver interrupts are enabled by setting UARTx_FCR[FE] = 1 and UARTx_IER[ERBFI] = 1.

A FIFO timeout will occur when:

At least one character is in the receiver FIFO, no serial characters have been received for four serial character time periods, and the processor has not read the FIFO for four serial character time periods. A serial character time period is as follows:

1/(baud rate) x (# start bits + word length + # parity bits + # stop bits)

For example, the serial character time period for an 8-bit word with one parity bit, two stop bits at 56K baud is as follows:

1/(56000) x (1 + 8 + 1 + 2) = 214.3µs

So the timeout would occur after 857.1µs, if the above conditions hold.

When a timeout interrupt has occurred, it is cleared and the timer is reset when the processor reads one character from the receiver FIFO.

When a timeout interrupt has not occurred, the timer is reset after a new serial character is received or the processor reads the receiver FIFO.

23.4.1.2 Transmitter

Transmitter interrupts occur, as described below, when the transmitter FIFO and transmitter interrupts are enabled by setting UARTx_FCR[FE] = 1 and UARTx_IER[ETBEI] = 1.

The transmitter holding register interrupt (UARTx_IIR = 0xC2) occurs when transmit FIFO is empty, and is cleared as soon as the transmitter holding register is written to or the IIR is read. One to 16 characters may be written to the transmitter FIFO while servicing this interrupt.

The transmitter FIFO empty indications are delayed by one character time minus the last stop bit time whenever the following event occurs: UARTx_LSR[THRE] = 1 and there were less than two bytes simultaneously present in the transmit FIFO since the last UARTx_LSR[THRE] = 1. If UARTx_FCR[FE] = 1 (FIFOs enabled), the first transmitter interrupt after changing UARTx_FCR[FE] is immediate.

Receiver FIFO trigger level interrupts, received data available interrupts, and character timeouts all have equivalent second interrupt priority. Current transmitter holding register empty interrupt and Transmit FIFO empty have equivalent third interrupt priority.

23.4.2 Polled Mode

When UARTx_FCR[FE] = 1 (FIFOs enabled), and UARTx_IER[5:7] are all set to 0 (interrupts disabled), the UART is in FIFO polled mode of operation. The receiver and transmitter are controlled separately, so either can be in polled mode of operation. In polled mode, the user program must check the UARTx_LSR to see the status of the receiver and/or transmitter.

UARTx_LSR[3:6] specifies which errors (if any) have occurred. Character status errors are handled in the same way as in interrupt mode. Since UARTx_IER[ELSI] = 0, the IIR is not affected. UARTx_LSR[DR] is set as long as there is at least one character in the receiver FIFO. UARTx_LSR[THRE] indicates if the transmitter FIFO is empty. UARTx_LSR[TEMT] indicates if the transmitter FIFO and the transmitter shift register are empty. UARTx_LSR[RFE] indicates if there are any errors in the receiver FIFO.

In FIFO polled mode, there are no character timeout or trigger levels; however, the FIFOs are still capable of holding characters.

23.4.3 UART and Sleep Mode

Both UARTs can be placed in sleep mode via the UART sleep bits in the CPC0_ER register (CPC0_ER[UART0:UART1]). The most common usage would be to save a little power if one or both of the UARTs were not going to be used.

Using sleep mode dynamically requires careful software control to make sure the UARTs are idle before putting them to sleep.

23.5 DMA Operation

The DMA controller can be configured to perform DMA operations using UART0, which appears as an 8-bit peripheral to the DMA controller. When selected, the UART receiver is internally wired to the DMAReq and DMAAck signals of DMA channel 2, and the transmitter is internally wired to the DMAReq and DMAAck signals of DMA channel 3.

The UART can be operated in FIFO mode or non-FIFO mode. In FIFO mode, the transfers can be done as single transfers (DMA mode 0) or multiple transfers (DMA mode 1), depending on the setting of the DMS field in the FIFO Control Register (FCR). In non-FIFO mode, DMA transfers are performed using single transfers, using the UART's DMA mode 0. This section describes proper UART0 DMA programming. For more information on general DMA programming, see ~~Chapter 19, "Direct Memory Access Controller," on page 19-1~~*Section 19 Direct Memory Access Controller* on page 655.

23.5.1 Control Register 0 (CPC0_CR0)

Only CPC0_CR0 fields related to UART are shown in Figure 23-13. Other non-related functions are not shown here.

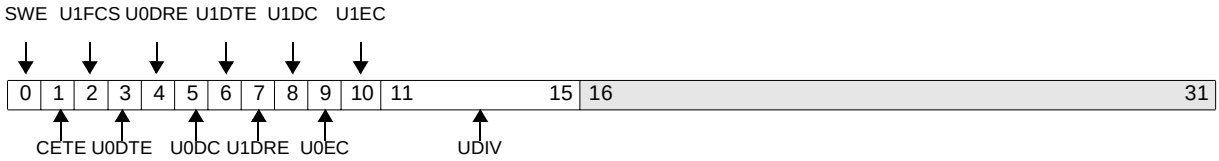


Figure 23-13. Control Register 0 (CPC0_CR0)

0	SWE	System Write Enable 0 DCR updates to the SYS0 and SYS1 registers are disabled 1 DCR updates to the SYS0 and SYS1 registers are enabled
1	CETE	CPU EXT Timer Enable 0 CPU timers increment at CPU clock frequency 1 CPU timers increment at TMR_CLK frequency

2	U1FCS	UART 1 Flow Control Select 0 Configure UART1 as DSR and DTR 1 Configure UART1 as CTS and RTS
3	U0DTE	UART 0 DMA Transmit Enable 0 UART0 DMA transmit is disabled 1 UART0 DMA transmit is enabled
4	U0DRE	UART 0 DMA Receive Enable 0 UART0 DMA receive is disabled 1 UART0 DMA receive is enabled
5	U0DC	UART 0 DMA Clear 0 U0DTE and U0DRE are not cleared when UART receives a corresponding terminal count. 1 U0DTE and U0DRE are cleared when UART receives a corresponding terminal count.
6	U1DTE	UART 1 DMA Transmit Enable 0 UART1 DMA transmit is disabled 1 UART1 DMA transmit is enabled
7	U1DRE	UART 1 DMA Receive Enable 0 UART1 DMA receive is disabled 1 UART1 DMA receive is enabled
8	U1DC	UART 1 DMA Clear 0 U1DTE and U1DRE are not cleared when UART receives a corresponding terminal count. 1 U1DTE and U1DRE are cleared when UART receives a corresponding terminal count.
9	U0EC	UART 0 EXT Clock Enable 0 UART0 uses the internal serial clock 1 UART0 uses the external serial clock
10	U1EC	UART 1 EXT Clock Enable 0 UART1 uses the internal serial clock 1 UART1 uses the external serial clock
11:15	UDIV	UART Divider 00000 - divider = 1 00001 - divider = 2 00010 - divider = 3 ..... 11110 - divider = 31 11111 - divider = 32 These bits control the bus frequency divider ratio between the PLB and UART serial clock frequencies. Each UART can be individually configured to use the serial clock that is divided down according to these bits, or to use an externally provided serial clock. For example, if the PLB clock is running at 266MHz, a divider value of 20 will set the serial clock frequency at 13.3MHz. **Note:** Maximum serial clock frequency allowed is slightly less than 1/2 OPB frequency.
16:31		Reserved

23.5.2 Transmitter DMA Mode

The four DMA channels on the PPC440GP can be used to move data to and from the two UARTs. The UART0 transmit channel is tied to DMA channel 1 and the UART1 transmit channel is tied to DMA channel 3. Use of the DMA with the UARTs is controlled by the UxDTE bits of the CPC0_CR0 register.

When the DMA mode bit in the UART FIFO control register, UARTx_FCR[DMS], is 0 the DMA request goes active when the FIFOs are disabled or the FIFOs are enabled and there are no characters in the TX FIFO or Transmit Holding Register (THR). Once activated, the DMA request goes inactive after the first character is loaded into the TX FIFO or THR.

When UARTx_FCR[DMS] is 1 the DMA request goes active, when FIFOs are enabled and there is at least one unfilled position in the TX FIFO. DMA request goes inactive when the TX FIFO is completely full. To operate in this mode the corresponding PPC440GP DMA Channel Control Register must be configured to accept DMA requests from an internal source. Setting the Peripheral Location (PL) bit of DMA0_CRx to 1 configures the DMA channel to accept DMA requests from UARTs on the OPB.

Table 23-5 lists required register settings for UART0 transmit transfers. The UART0 transmit channel is tied to DMA channel 1. Other DMA registers and register fields must be programmed appropriately, see ~~"Direct Memory Access Controller" on page 19-1~~*Direct Memory Access Controller* on page 655 for more information.

Table 23-5. UART 0 Transmitter DMA Mode Register Field Settings

Register [Field]	Meaning
CPC0_CR0[U0DTE]=1	UART0 DMA transmit channel is enabled using DMA channel 1
CPC0_CR0[U0DC]	Set to 0 to not clear CPC0_CR0[DTE] enable when terminal count is reached, set to 1 to clear enable when terminal count is reached.
DMA0_CR1[TD]=0	DMA channel 1 transfer direction is from memory to peripheral.
DMA0_CR1[PL]=1	DMA channel 1 peripheral is on the OPB (UART0).
DMA0_CR1[PW]=000	Peripheral width is byte (8 bits).
DMA0_CR1[TM]=00	DMA Channel 1 is in peripheral mode.
DMA0_CR1[PWC]=000010	Peripheral wait cycles, how long the internal DMAck is active. Three cycles are required.
DMA0_CR1[PHC]=000	Peripheral hold cycles are 0.
DMA0_CR1[ETD]=1	EOT/TC is programmed as terminal count output.
UART0_FCR[DMS]	Set to 0 for a single DMA transfer or 1 for multiple DMA transfers.

Note: When using DMA channel 1 for UART0 transmitter transfers, external DMA transfers cannot be performed on this channel.

Table 23-6 lists required register settings for UART 1 transmit transfers. The UART1 transmit channel is tied to DMA channel 3. Other DMA registers and register fields must be programmed appropriately, see ~~"Direct Memory Access Controller" on page 19-1~~*Direct Memory Access Controller* on page 655 for more information.

Table 23-6. UART 1 Transmitter DMA Mode Register Field Settings

Register [Field]	Meaning
CPC0_CR0[U1DTE]=1	UART 1 DMA transmit channel is enabled using DMA channel 3.
CPC0_CR0[U1DC]	Set to 0 to not clear ~~CPC0_CR0~~CPC0_CR0UART1[DTE] enable when terminal count is reached, set to 1 to clear enable when terminal count is reached.
DMA0_CR3[TD]=0	DMA channel 3 transfer direction is from memory to peripheral.
DMA0_CR3[PL]=1	DMA channel 3 peripheral is on the OPB (UART0).
DMA0_CR3[PW]=000	Peripheral width is byte (8 bits).
DMA0_CR3[TM]=00	DMA channel 3 is in peripheral mode.
DMA0_CR3[PWC]=000010	Peripheral wait cycles, how long the internal DMAck is active. Three cycles are required.
DMA0_CR3[PHC]=000	Peripheral hold cycles are 0.
DMA0_CR3[ETD]=1	EOT/TC is programmed as terminal count output.
UART1_FCR[DMS]	Set to 0 for a single DMA transfer or 1 for multiple DMA transfers.

Note: When using DMA channel 3 for UART 1 transmitter transfers, external DMA transfers cannot be performed on this channel.

23.5.3 Receiver DMA Mode

The four DMA channels on the PPC440GP can be used to move data to and from the two UARTs. The UART0 receive channel is tied to DMA channel 0 and the UART1 receive channel is tied to DMA channel 2. Use of the DMA with the UARTs is controlled by the UxDRE bits of the CPC0_CR0 register.

When the DMA mode bit in the UART FIFO control register, UARTx_FCR[DMS], is 0 the DMA request goes active when there is at least one character in the RX FIFO or Receive Buffer Register (RBR). Once activated, the DMA request goes inactive when there are no more characters in the FIFO or RBR.

When UARTx_FCR[DMS] is 1 the DMA request goes active when the FIFOs are enabled and the trigger level or the timeout has been reached. DMA request goes inactive when there are no more characters in the FIFO or RBR. To operate in this mode the corresponding ~~440GP~~ PPC440GP DMA Channel Control Register must be configured to accept DMA requests from an internal source. Setting the Peripheral Location (PL) bit of DMA0_CRx to 1configures the DMA channel to accept DMA requests from UARTs on the OPB.

Table 23-7 lists required register settings for UART0 receive transfers. The UART0 receive channel is tied to DMA channel 0. Other DMA registers and register fields must be programmed appropriately, see ~~"Direct Memory Access Controller" on page 19-1~~*Direct Memory Access Controller* on page 655 for more information.

Table 23-7. UART0 Receiver DMA Mode Register Field Settings

Register [Field]	Meaning
CPC0_CR0[U0DRE]=1	UART0 DMA receiver channel is enabled using DMA channel 0.
CPC0_CR0[U0DC]	Set to 0 to not clear CPC0_CR0[DRE] enable when terminal count is reached, set to 1 to clear enable when terminal count is reached.
DMA0_CR0[TD]=1	DMA channel 0 transfer direction is from peripheral to memory.
DMA0_CR0[PL]=1	DMA channel 0 peripheral is on the OPB (UART0).
DMA0_CR0[PW]=000	Peripheral width is byte (8 bits).
DMA0_CR0[TM]=00	DMA channel 0 is in peripheral mode.
DMA0_CR0[PWC]=000010	Peripheral wait cycles, how long the internal DMAAck is active. Three cycles are required.
DMA0_CR0[PHC]=000	Peripheral hold cycles are 0.
DMA0_CR0[ETD]=1	EOT/TC is programmed as terminal count output.
UART0_FCR[DMS]	Set to 0 for a single DMA transfer or 1 for multiple DMA transfers.

Note: When using DMA channel 0 for UART0 receiver transfers, external DMA transfers cannot be performed on this channel.

Table 23-8 lists required register settings for UART1 receive transfers. The UART1 receive channel is tied to DMA channel 2. Other DMA registers and register fields must be programmed appropriately, see "Direct Memory Access Controller" on page -655 for more information.

Table 23-8. UART1 Receiver DMA Mode Register Field Settings

Register [Field]	Meaning
CPC0_CR0[U1DRE]=1	UART1 DMA receiver channel is enabled using DMA channel 2.
CPC0_CR0[UIDC]	Set to 0 to not clear CPC0_CR0[DRE] enable when terminal count is reached, set to 1 to clear enable when terminal count is reached.
DMA0_CR2[TD]=1	DMA Channel 2 transfer direction is from peripheral to memory.
DMA0_CR2[PL]=1	DMA Channel 2 peripheral is on the OPB (UART0).
DMA0_CR2[PW]=00	Peripheral width is byte (8 bits).
DMA0_CR2[TM]=00	DMA Channel 2 is in peripheral mode.
DMA0_CR2[PWC]=000010	Peripheral wait cycles, how long the internal DMAAck is active. Three cycles are required.
DMA0_CR2[PHC]=000	Peripheral hold cycles are 0.
DMA0_CR2[ETD]=1	EOT/TC is programmed as terminal count output.
UART1_FCR[DMS]	Set to 0 for a single DMA transfer or 1 for multiple DMA transfers.

Note: When using DMA channel 2 for UART1 receiver transfers, external DMA transfers cannot be performed on this channel.

25. GPIO Operations

This chapter describes the General Purpose I/O (GPIO) controller located on the on-chip peripheral bus (OPB) of the PPC440GP. Thirty two of the functional I/O were implemented using the GPIO controller. Function pins that are not used in the application can be used as GPIOs.

25.1 GPIO Controller Overview

The GPIO Controller is an OPB macro that controls up to 32 bidirectional module I/O pins with user-programmable functions. To reduce the quantity of module I/O on the PPC440GP package there are no dedicated general purpose I/Os. Each of the GPIOs has been assigned a function pin such as an interrupt. However, if not all of the pins are used functionally they can be used as GPIO. The ~~chip configuration~~ GPIO control register ~~CPC0_GPIO~~ 0 (CPC0_GPIO) controls whether a pin is used for a defined function or as a GPIO. ~~See "GPIO Configuration Register (CPC0_GPIO)" on page 25-3.~~See *GPIO Configuration Register (CPC0_GPIO)* on page 825 ~~for more information.~~

All module I/O inputs are synchronized to the OPBClk before being stored in the Input register.

All registers, except the Input Register, are both read and write accessible. The Input register is read-only.

Registers provide direct control of all GPIO Controller functions.

All register bits and core input and output signals maintain a bit-for-bit correspondence. For example, GPIO_Out[20] is controlled by GPIO0_OR Register Bit 20.

25.2 Features

The GPIO Controller has the following features:

- Direct control of all functions from registers programmed via memory-mapped addresses
- Control of 32 bidirectional GPIO module pins
  - Each GPIO output has programmable three-state control
  - Each GPIO output is separately programmable to emulate an open drain driver
  - Each GPIO input is observable from a register bit

25.3 GPIO Interface Signals

The selection between a pin being a designated function or a GPIO is controlled by the ~~Chip configuration~~ GPIO control register 0 (CPC0_GPIO). Once a pin is selected as a GPIO, it is controlled by a corresponding bit in 4 registers; the GPIO Input Register (GPIO_IR), GPIO Output Register, GPIO_OR, GPIO Output Tri-state Control register, GPIO_TCR, and GPIO Output Open Drain Control register, GPIO_ODR.

~~Figure 25-1~~*Figure 25-1* shows how the GPIO macro is multiplexed with a function pin.

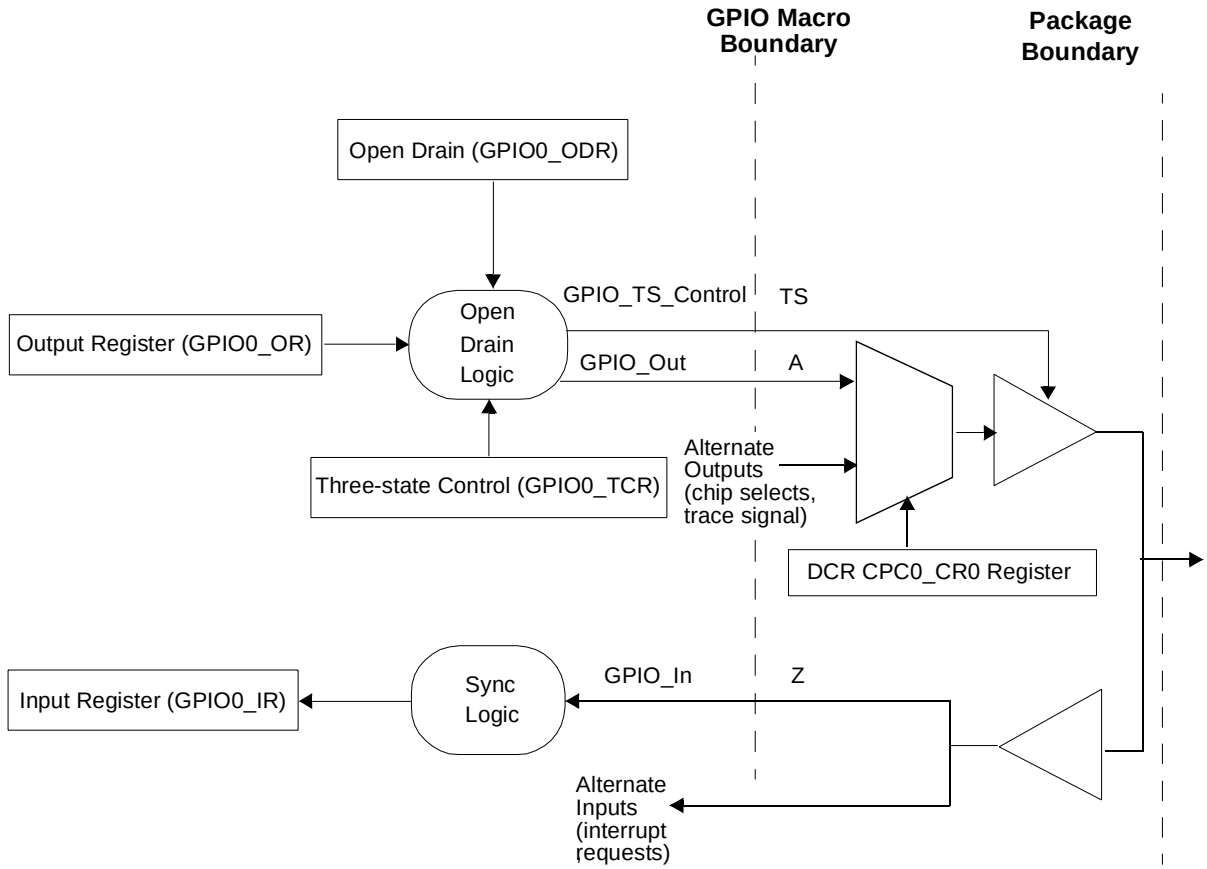


Figure 0-1. GPIO Functional Block Diagram

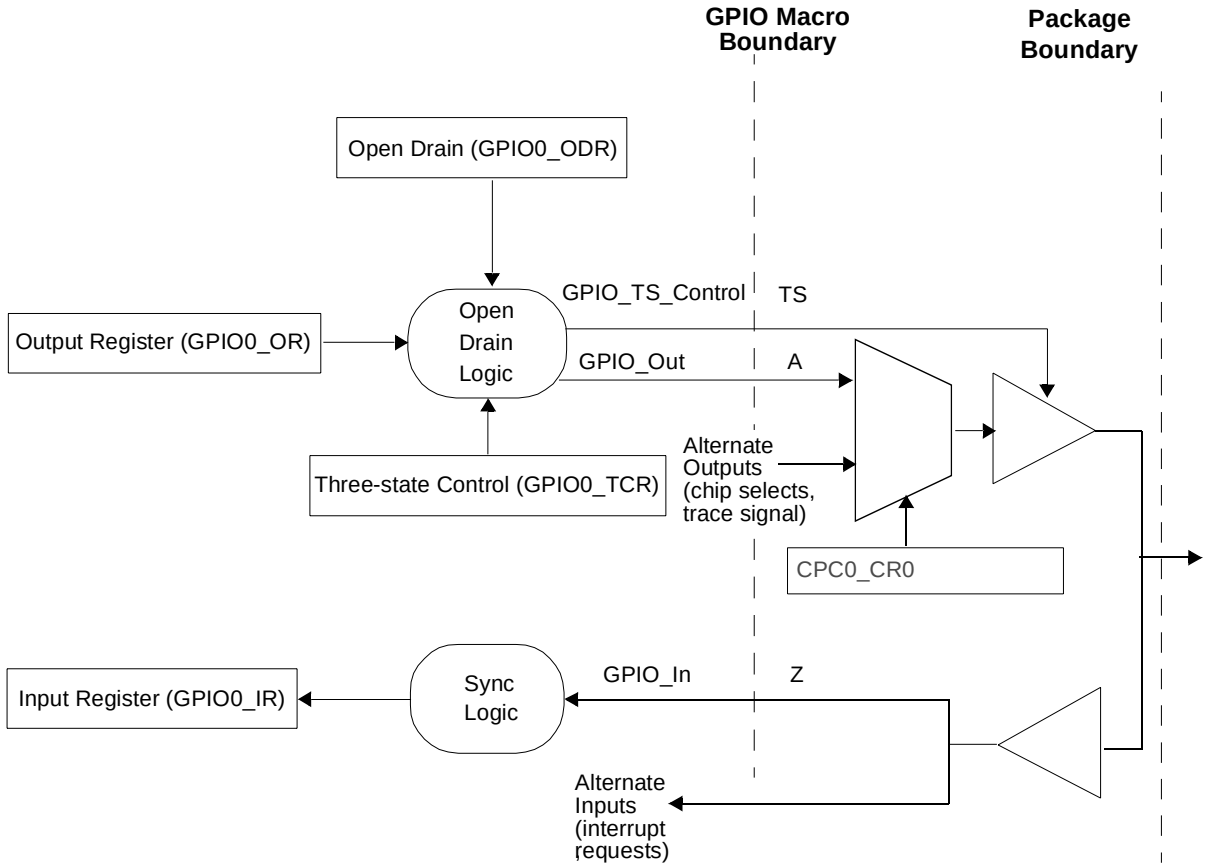


Figure 25-1. GPIO Functional Block Diagram

Note 1: All the registers within the GPIO controller are synchronous with the OPBClk, and all GPIO inputs are synchronized to OPBClk before being stored. Pin multiplexing is controlled by the setting of CPC0_GPIO[0:18] as follows:

Function Inputs - UART, Interrupts

Function Outputs - UART, Trace

Note 2: For GPIO bits 0-17 these pins default to function pins. If these pins are used as GPIO the default state must be accounted for in the design. For example GPIO bits 0-10 default to inputs as interrupts. If they are used as GPIO inputs simply setting the appropriate bits in the CPC0_GPIO register is sufficient. However, if they are used as GPIO outputs they will power-up as floating and may need to have a pull-up or pull-down, to maintain an inactive state on attached logic, until they can be configured by software. Conversely GPIO bits that default to function outputs may conflict with external logic if they are used as inputs; until they can be configured.

25.3.1 External Module Signals

The GPIO signals do not have dedicated module pins. The module pins, used by the GPIO signals, are shared (multiplexed) with other signals. The GPIO ~~Control Register~~ control register 0 (CPC0_GPIO) controls the way in which the signal on a shared pin is interpreted. The following sections provide details of the CPC0_GPIO register bit settings and the function of the GPIO registers.

The GPIO signals are multiplexed with ~~twelve~~ eleven interrupt signals, four UART1 signals and two IIC1 signals. Multiplexing is controlled by the setting of CPC0_GPIO[0:18] as follows. In all cases when a pin is used as a GPIO, it is necessary to configure the appropriate bit in this register as well as the GPIO registers located in the GPIO controller.

25.3.1.1 GPIO Configuration Register (CPC0_GPIO)

~~Figure 25-2~~Figure 25-2 describes CPC0_GPIO register bits.

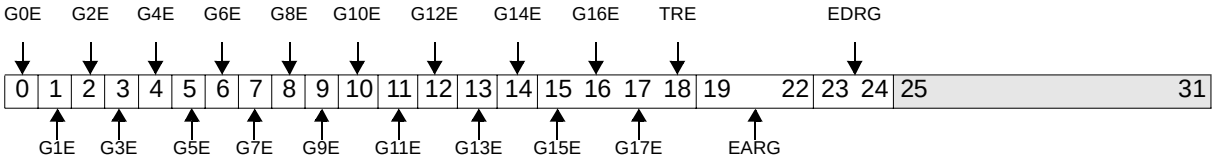


Figure 0-2. GPIO Configuration Register (CPC0_GPIO)

0	G0E	GPIO 0 Enable 0 Enable interrupt IRQ00 as an interrupt 1 Enable interrupt IRQ00 as GPIO0
1	G1E	GPIO 1 Enable 0 Enable interrupt IRQ01 as an interrupt 1 Enable interrupt IRQ01 as GPIO1
2	G2E	GPIO 2 Enable 0 Enable interrupt IRQ02 as an interrupt 1 Enable interrupt IRQ02 as GPIO2
3	G3E	GPIO 3 Enable 0 Enable interrupt IRQ03 as an interrupt 1 Enable interrupt IRQ03 as GPIO3
4	G4E	GPIO 4 Enable 0 Enable interrupt IRQ04 as an interrupt 1 Enable interrupt IRQ04 as GPIO4
5	G5E	GPIO 5 Enable 0 Enable interrupt IRQ05 as an interrupt 1 Enable interrupt IRQ05 as GPIO5
6	G6E	GPIO 6 Enable 0 Enable interrupt IRQ06 as an interrupt 1 Enable interrupt IRQ06 as GPIO6

7	G7E	GPIO 7 Enable 0 Enable interrupt IRQ07 as an interrupt 1 Enable interrupt IRQ07 as GPIO7	
8	G8E	GPIO 8 Enable 0 Enable interrupt IRQ08 as an interrupt 1 Enable interrupt IRQ08 as GPIO8	
9	G9E	GPIO 9 Enable 0 Enable interrupt IRQ09 as an interrupt 1 Enable interrupt IRQ09 as GPIO9	
10	G10E	GPIO 10 Enable 0 Enable interrupt IRQ010 as an interrupt 1 Enable interrupt IRQ010 as GPIO10	
11	G11E	GPIO 11 Enable 0 Enable interrupt IRQ011 as an interrupt 1 Enable interrupt IRQ011 as GPIO11	
12	G12E	GPIO 12 Enable 0 Enable UART1RX as UART1RX 1 Enable UART1RX as GPIO 12	
13	G13E	GPIO 13 Enable 0 Enable UART1TX as UART1TX 1 Enable UART1TX as GPIO 13	
14	G14E	GPIO 14 Enable 0 Enable UART1DSR_CTS as UART1DSR_CTS 1 Enable UART1DSR_CTS as GPIO 14	
15	G15E	GPIO 15 Enable 0 Enable UART1RTS_DTR as UART1RTS_DTR 1 Enable UART1RTS_DTR as GPIO 15	
16	G16E	GPIO 16 Enable 0 Enable IIC1SCL as IIC1SCL 1 Enable IIC1SCL as GPIO 16	
17	G17E	GPIO 17 Enable 0 Enable IIC1SDA as IIC1SDA 1 Enable IIC1SDA as GPIO 17	
18	TRE	GPIO Trace Enable 0 GPIO18-31 are enabled 1 GPIO18-31 are disabled	Trace interface cannot be used when GPIO is enabled. Trace interface can be used when GPIO is disabled.

19:22	EARG	EBC Address Receiver Gating 0000 Receiver gating disabled 0001 Disable address bus inputs 0:1 0010 Disable address bus inputs 0:2 0011 Disable address bus inputs 0:3 .. .. .. 1111 Disable address bus inputs 0:15	Encoded value for gating external bus controller address bus
23:24	EDRG	EBC Data Receiver Gating 00 Receiver gating disabled 01 Disable data bus input bytes 1, 2, 3 10 Disable data bus input bytes 2, 3 11 Disable data bus input byte 3	Encoded value for gating external bus controller data bus
25:31		Reserved	

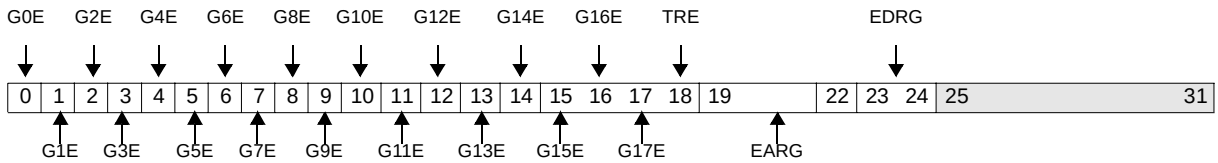


Figure 25-2. GPIO Configuration Register (CPC0_GPIO)

0	G0E	GPIO 0 Enable 0 Enable interrupt IRQ00 as an interrupt 1 Enable interrupt IRQ00 as GPIO0
1	G1E	GPIO 1 Enable 0 Enable interrupt IRQ01 as an interrupt 1 Enable interrupt IRQ01 as GPIO1
2	G2E	GPIO 2 Enable 0 Enable interrupt IRQ02 as an interrupt 1 Enable interrupt IRQ02 as GPIO2
3	G3E	GPIO 3 Enable 0 Enable interrupt IRQ03 as an interrupt 1 Enable interrupt IRQ03 as GPIO3
4	G4E	GPIO 4 Enable 0 Enable interrupt IRQ04 as an interrupt 1 Enable interrupt IRQ04 as GPIO4
5	G5E	GPIO 5 Enable 0 Enable interrupt IRQ05 as an interrupt 1 Enable interrupt IRQ05 as GPIO5
6	G6E	GPIO 6 Enable 0 Enable interrupt IRQ06 as an interrupt 1 Enable interrupt IRQ06 as GPIO6
7	G7E	GPIO 7 Enable 0 Enable interrupt IRQ07 as an interrupt 1 Enable interrupt IRQ07 as GPIO7

8	G8E	GPIO 8 Enable 0 Enable interrupt IRQ08 as an interrupt 1 Enable interrupt IRQ08 as GPIO8	
9	G9E	GPIO 9 Enable 0 Enable interrupt IRQ09 as an interrupt 1 Enable interrupt IRQ09 as GPIO9	
10	G10E	GPIO 10 Enable 0 Enable interrupt IRQ010 as an interrupt 1 Enable interrupt IRQ010 as GPIO10	
11	G11E	GPIO 11 Enable 0 Disable GPIO11 1 Enable GPIO11	
12	G12E	GPIO 12 Enable 0 Enable UART1RX as UART1RX 1 Enable UART1RX as GPIO 12	
13	G13E	GPIO 13 Enable 0 Enable UART1TX as UART1TX 1 Enable UART1TX as GPIO 13	
14	G14E	GPIO 14 Enable 0 Enable UART1DSR_CTS as UART1DSR_CTS 1 Enable UART1DSR_CTS as GPIO 14	
15	G15E	GPIO 15 Enable 0 Enable UART1RTS_DTR as UART1RTS_DTR 1 Enable UART1RTS_DTR as GPIO 15	
16	G16E	GPIO 16 Enable 0 Enable IIC1SCL as IIC1SCL 1 Enable IIC1SCL as GPIO 16	
17	G17E	GPIO 17 Enable 0 Enable IIC1SDA as IIC1SDA 1 Enable IIC1SDA as GPIO 17	
18	TRE	GPIO Trace Enable 0 GPIO18-31 are enabled 1 GPIO18-31 are disabled	Trace interface cannot be used when GPIO is enabled. Trace interface can be used when GPIO is disabled.
19:22	EARG	EBC Address Receiver Gating 0000 Receiver gating disabled 0001 Disable address bus inputs 0:1 0010 Disable address bus inputs 0:2 0011 Disable address bus inputs 0:3 .. .. .. 1111 Disable address bus inputs 0:15	Encoded value for gating external bus controller address bus
23:24	EDRG	EBC Data Receiver Gating 00 Receiver gating disabled 01 Disable data bus input bytes 1, 2, 3 10 Disable data bus input bytes 2, 3 11 Disable data bus input byte 3	Encoded value for gating external bus controller data bus
25:31		Reserved	

25.4 GPIO Registers

All GPIO registers are 32-bi memory-mapped registers and are accessed via load/store instructions at the address of the register. The GPIO0_IR register is read-only; all other registers are both read and write accessible. The registers are only active for a particular bit when that bit is selected as a GPIO in the CPC0_GPIO register.

Table 25-1 contains a summary of all GPIO registers.

Table 25-1. GPIO Register Summary

Register	Description	Address	Access	Page
GPIO0_OR	GPIO Output Register	0x1 40000700	R/W	~~25-5~~827
GPIO0_TCR	GPIO Three-State Control Register	0x1 40000704	R/W	~~25-6~~828
GPIO0_ODR	GPIO Open Drain Register	0x1 40000718	R/W	~~25-6~~828
GPIO0_IR	GPIO Input Register	0x1 4000071C	R	~~25-7~~829

25.4.1 GPIO Register Reset Values

When a system reset occurs, each register in the GPIO macro, except GPIO0_IR, is reset to 0. All outputs are in the high-impedance state. GPIO0_IR is not reset because it is always clocked and always follows the GPIO_In state.

Note: The CPC0_GPIO register needs to be initialized since it is not initialized by reset. During system or chip reset this register defaults to 0x0000 2000. However its value is not affected by a core reset.

25.4.2 GPIO Output Register (GPIO0_OR)

The state of each bit in the GPIO0_OR Register may be reflected in the corresponding GPIO controller macro output signal GPIO_Out. ~~Figure 25-3~~*Figure 25-3* describes GPIO0_OR register bits.

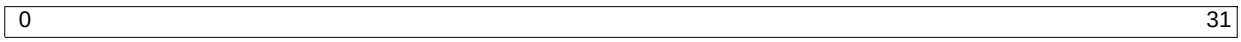


Figure 0-3. GPIO Output Register (GPIO0_OR)

0:31		GPIO0_OR register bits

Figure 25-3. GPIO Output Register (GPIO0_OR)

0:31		GPIO0_OR register bits

25.4.3 GPIO Three-State Control Register (GPIO0_TCR)

Each bit in the GPIO0_TCR Register controls the corresponding GPIO_TS_Control macro output signal. When 0, GPIO_TS_Control forces the module I/O drivers into the high-impedance state.

The state of the GPIO_Out signal driving Pin A (Driver Data Input) of the module I/O three-state driver is irrelevant when the driver is in the high impedance state. High impedance take precedence over data output signals. ~~Figure 25-4~~*Figure 25-4* describes GPIO0_TCR register bits.

Figure 0-4. GPIO Three-State Register (GPIO0_TCR)

0:31		GPIO0_TCR register bits

Figure 25-4. GPIO Three-State Register (GPIO0_TCR)

0:31		GPIO0_TCR register bits

25.4.4 GPIO Open Drain Register (GPIO0_ODR)

The GPIO Open Drain Register configures the module I/O three-state driver to emulate open drain drivers on a bit-by-bit basis. This is done by controlling the GPIO_Out and GPIO_TS_Control signals, via the GPIO0_OR and GPIO0_TCR registers, respectively. ~~Figure 25-5~~*Figure 25-5* describes GPIO0_ODR register bits.

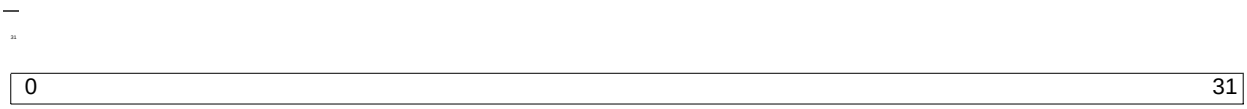


Figure 0-5. GPIO Open Drain Register (GPIO0_ODR)

0:31		GPIO0_ODR register bits

Figure 25-5. GPIO Open Drain Register (GPIO0_ODR)

0:31		GPIO0_ODR register bits

When programmed to 1, each bit in the GPIO0_ODR register forces the corresponding GPIO_Out signal to 0 and the corresponding GPIO_TS_Control signal to the inverted GPIO_Out signal. In this case, the value of the corresponding bit in the GPIO0_TCR register is ignored. The open drain logic is shown in the table below.

When emulating an open drain driver, the module I/O driver never drives a 1 level. It either drives a 0 level or it is in the high impedance state emulating an open drain 1 level.

Table 25-2. GPIO0_ODR Control Logic

GPIO0_ODR Bit	GPIO0_OR Bit	GPIO_Out Macro Output	GPIO0_TCR Bit	GPIO_TS_Control Macro Output	Module I/O Three-State Driver
0	x	x	0	0	Forced to high-impedance state
0	0	0	1	1	Driving 0
0	1	1	1	1	Driving 1
1	0	0	x	1	Driving 0
1	1	0	x	0	Forced to high-impedance state

25.4.5 GPIO Input Register (GPIO0_IR)

The state of each bit in the GPIO0_IR Register reflects the corresponding GPIO Controller macro input signal GPIO_In. All GPIO_In signals are synchronized to OPBClk before being stored in the GPIO0_IR Register. The GPIO0_IR Register is read-only and does not change during a read access.

All bi-directional three-state drivers must be in the high-impedance state in order to receive valid data as input from off chip.~~Figure 25-6~~*Figure 25-6* describes GPIO0_IR register bits.

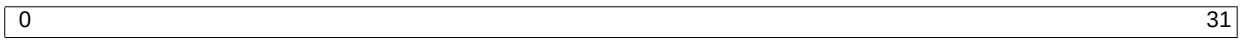


Figure 0-6. GPIO Input Register (GPIO0_IR)

0:31		GPIO register bits

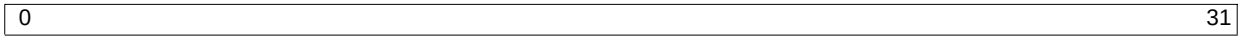


Figure 25-6. GPIO Input Register (GPIO0_IR)

0:31		GPIO register bits

24. IIC Bus Interface

The PPC440GP provides two inter-integrated circuit (IIC) bus interfaces, IIC0 and IIC1. These interfaces comply with the specifications contained in the Phillips ® Semiconductors document The I2C bus and how to use it (including specifications) (1995 update).

Each IIC bus has a two wire, bi-directional, open-drain, low-speed serial interface. The serial clock (IIC0SClk and IIC1SClk) and serial data (IIC0SDA and IIC1SDA) lines are bidirectional, to support multiple bus masters and to mix high- and low-speed devices on the same bus.

Each IIC interface (referred to as IIC to distinguish it from the Phillips I 2 C bus) supports the following standard and enhanced features:

- 100-kHz and 400-kHz operation
- 8-bit data transfers
- 7-bit and 10-bit addressing
- Slave transmitter and receiver
- Master transmitter and receiver
- Multiple bus masters

24.1 Addressing

The IIC interfaces support 7-bit and 10-bit addressing for master and slave transfers. Addressing is described in detail in ~~"IICx Low Master Address Register (IICx_LMADR)" on page 24-5~~*IICx Low Master Address Register (IICx_LMADR)* on page 805, ~~"IICx High Master Address Register (IICx_HMADR)" on page 24-5~~*IICx High Master Address Register (IICx_HMADR)* on page 806, ~~"IICx Low Slave Address Register (IICx_LSADR)" on page 24-14~~*IICx Low Slave Address Register (IICx_LSADR)* on page 814, and ~~"IICx High Slave Address Register (IICx_HSADR)" on page 24-14~~*IICx High Slave Address Register (IICx_HSADR)* on page 814.

Descriptions of addressing modes and address formats follow.

24.1.1 Addressing Modes

For master transfers, the address mode (AMD) field of the IIC Control register (IICx_CNTL) controls whether 7-bit or 10-bit addresses are used. If IICx_CNTL[AMD] = 0, addresses contain 7 bits; if IICx_CNTL[AMD] = 1, addresses contain 10 bits.

For slave transfers, the contents of the IICx High Slave Address register (IICx_HSADR) determines whether 7-bit or 10-bit addressing is used. If IICx_HSADR = 0b00000000, 7-bit addressing is used. If 10-bit addressing is to be used for slave transfers, IICx_HSADR = 0b11110*yy*x, where *yy* contains the high-order bits of the 10-bit address, and x is a don't care.

Programming Note: For slave transfers, IICx_CNTL[AMD] does not control addressing mode.

24.1.2 Seven-Bit Addresses

Figure 24-1 illustrates a 7-bit address. For master transfers, the address bits 0 through 6 (A0:A6) are read from IICx_LMADR. For slave transfers, A0:A6 are read from IICx_LSADR. Bit 7 of address byte 0 contains a transfer type bit provided by the IIC interface.

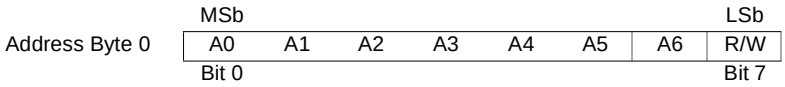


Figure 24-1. 7-Bit Addressing

24.1.3 Ten-Bit Addresses

Figure 24-2 illustrates a 10-bit address. A0:A1 of address byte 0 are read from IICx_HMADR[A6:A7] (for master transfers) or IICx_HSADR[A6:A7] (for slave transfers). These are the two highest-order address bits transmitted on the IIC bus. Bit 7 of address byte 0 contains a transfer type bit provided by the IIC interface.

For 10-bit addressing for master or slave transfers, respectively, IICx_HMADR[A0:A4] and IICx_HSADR [A0:A4] must contain 0b11110.

The low-order byte of the 10-bit address, contained in A0:A7 of address byte 1, are read from IICx_LMADR or IICx_LSADR for master or slave transfers, respectively.

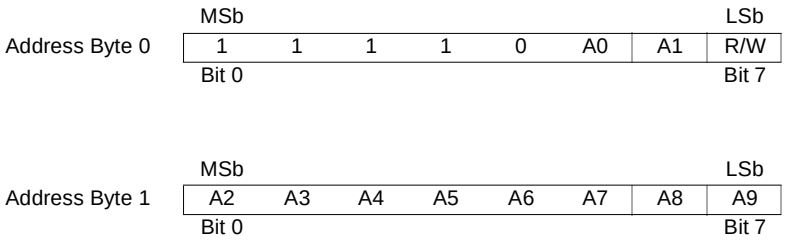


Figure 24-2. 10-Bit Addressing

24.2 IIC Registers

IICx registers are accessed via memory locations 0x1 4000 0XYY, where X=4 for the IIC0 interface and X=5 for the IIC1 interface. YY identifies the register within the IIC0 or IIC1.

Table 24-1 lists the IICx registers. Descriptions of the registers, in the listed order, follow in ~~"IIC Register Descriptions" on page 24-3~~*IIC Register Descriptions* on page 803.

Table 24-1. IIC Registers

Register	Description	Address	Offset	Access	Effect of Reset	Bits	Page
IICx_MDBUF	IICx Master Data Buffer	0x14000 0X00	0x0	R/W	Cleared	8,16	~~24-3~~803
IICx_SDBUF	IICx Slave Data Buffer	0x14000 0X02	0x2	R/W	Cleared	8,16	~~24-4~~804
IICx_LMADR	IICx Low Master Address	0x14000 0X04	0x4	R/W	No	8	~~24-5~~805
IICx_HMADR	IICx High Master Address	0x14000 0X05	0x5	R/W	No	8	~~24-5~~806
IICx_CNTL	IICx Control	0x14000 0X06	0x6	R/W	Cleared	8	~~24-6~~806
IICx_MDCNTL	IICx Mode Control	0x14000 0X07	0x7	R/W	Cleared	8	~~24-8~~808
IICx_STS	IICx Status	0x14000 0X08	0x8	R/W	Cleared	8	~~24-10~~809
IICx_EXTSTS	IICx Extended Status	0x14000 0X09	0x9	R/W	Cleared	8	~~24-11~~811
IICx_LSADR	IICx Low Slave Address	0x14000 0X0A	0xA	R/W	No	8	~~24-14~~814
IICx_HSADR	IICx High Slave Address	0x14000 0X0B	0xB	R/W	No	8	~~24-14~~814
IICx_CLKDIV	IICx Clock Divide	0x14000 0X0C	0xC	R/W	Cleared	8	~~24-15~~815
IICx_INTRMSK	IICx Interrupt Mask	0x14000 0X0D	0xD	R/W	Cleared	8	~~24-16~~816
IICx_XFRCNT	IICx Transfer Count	0x14000 0X0E	0xE	R/W	Cleared	8	~~24-17~~817
IICx_XTCNTLSS	IICx Extended Control and Slave Status	0x14000 0X0F	0xF	R/W	Cleared	8	~~24-18~~818
IICx_DIRECTCNTL	IICx Direct Control	0x14000 0X10	0x10	R/W	0x0f	4	~~24-20~~820

24.3 IIC Register Descriptions

The following sections contains the bit definitions for the various registers in the IIC interface.

24.3.1 IICx Master Data Buffer (IICx_MDBUF)

The IICx Master Data Buffer (IICx_MDBUF) is a 1-byte × 4-byte first-in/first-out (FIFO) buffer. Data placed in IICx_MDBUF is written onto the IIC bus when performing a master write operation. Data received from the IIC bus is read from IICx_MDBUF when performing a master read operation.

Figure 24-3 describes the IICx_MDBUF register bits.

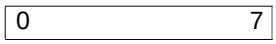


Figure 0-1. IICx Master Data Buffer (IICx_MDBUF)

0		Data bit
1		Data bit
2		Data bit
3		Data bit
4		Data bit
5		Data bit
6		Data bit
7		Data bit

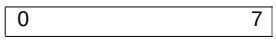


Figure 24-3. IICx Master Data Buffer (IICx_MDBUF)

0		Data bit
1		Data bit
2		Data bit
3		Data bit
4		Data bit
5		Data bit
6		Data bit
7		Data bit

Halfword reads from and writes to the IIC bus assume the MSB comes first. For halfword writes, the first byte written to the IICx_MDBUF is the MSB, byte 0. The followed byte written is the LSB, byte 1. For halfword reads, the first byte received is MSB, byte 0, and the following byte is LSB, byte 1.

If a byte is read from the IICx_MDBUF while the FIFO is empty, obsolete data is read. Check the master buffer status, IICx_STS[MDBS], before reading IICx_MDBUF.

If a byte is written to IICx_MDBUF while the FIFO is full, the byte is discarded and not placed into the FIFO. Check the master buffer full status, IICx_STS[MDBF], before writing IICx_MDBUF.

IICx_MDBUF is cleared (flushed and set to empty) whenever the IIC interface is reset, or IICx_MDCNTL[FMDB] = 1.

24.3.2 IICx Slave Data Buffer (IICx_SDBUF)

The IICx Slave Data Buffer (IICx_SDBUF) is a 1-byte 4-byte first-in/first-out (FIFO) buffer. The data contained in IICx_SDBUF is either received from the IIC bus when the IIC interface is addressed as a slave during a write, or is written on the IIC bus when the IIC interface is addressed as a slave during a read operation.

Figure 24-4 describes the IICx_SDBUF register bits.

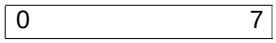


Figure 0-2. IICx Slave Data Buffer (IICx_SDBUF)

0		Data bit
1		Data bit
2		Data bit
3		Data bit
4		Data bit
5		Data bit
6		Data bit
7		Data bit

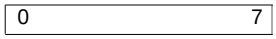


Figure 24-4. IICx Slave Data Buffer (IICx_SDBUF)

0		Data bit
1		Data bit
2		Data bit
3		Data bit
4		Data bit
5		Data bit
6		Data bit
7		Data bit

Halfword reads from and writes to the IIC bus assume the MSB comes first. For halfword writes, the first byte written to the IICx_SDBUF is the MSB, byte 0. The followed byte written is the LSB, byte 1. For halfword reads, the first byte received is MSB, byte 0, and the following byte is LSB, byte 1.

If a byte is read from the IICx_SDBUF while the FIFO is empty, obsolete data is read. Check the slave buffer status, IICx_STS[SBDD], before reading IICx_SDBUF.

If a byte is written to IICx_SDBUF while the FIFO is full, the byte is discarded and not placed into the FIFO. Check the master buffer full status, IICx_STS[MDBF], before writing IICx_SDBUF.

IICx_SDBUF is cleared (flushed and set to empty) whenever the IIC interface is reset, or IICx_MDCNTL[FSDB] = 1.

24.3.3 IICx Low Master Address Register (IICx_LMADR)

The IICx Low Master Address (IICx_LMADR) and IICx High Master Address Register (IICx_HMADR) form addresses that the IIC interface transmits on the IIC bus.

When in 7-bit address mode, IICx_CNTL[AMD] = 0, IICx_LMADR[A0:A6] is the address transmitted on the IIC bus; IICx_LMADR[A7] is a don't care. When in 10-bit address mode, IICx_CNTL[AMD] = 1, IICx_LMADR[A0:A7] is the second byte of the address transmitted on the IIC bus.

Figure 24-5 describes the IICx_LMADR register bits.

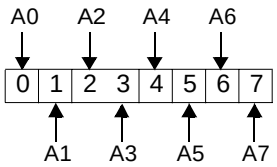


Figure 0-3. IICx Low Master Address Register (IICx_LMADR)

0	A0	Address bit 0	
1	A1	Address bit 1	
2	A2	Address bit 2	
3	A3	Address bit 3	
4	A4	Address bit 4	
5	A5	Address bit 5	
6	A6	Address bit 6	LSb for 7-bit addresses
7	A7	Address bit 7	LSb for 10-bit addresses; don't care for 7-bit addresses

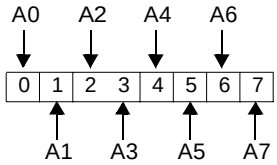


Figure 24-5. IICx Low Master Address Register (IICx_LMADR)

0	A0	Address bit 0	
1	A1	Address bit 1	
2	A2	Address bit 2	
3	A3	Address bit 3	
4	A4	Address bit 4	
5	A5	Address bit 5	
6	A6	Address bit 6	LSb for 7-bit addresses
7	A7	Address bit 7	LSb for 10-bit addresses; don't care for 7-bit addresses

24.3.4 IICx High Master Address Register (IICx_HMADR)

IICx High Master Address Register (IICx_HMADR) provides the upper address bits in 10-bit addressing mode.

When in 10-bit address mode, IICx_CNTL[AMD] = 1, IICx_HMADR must be programmed to 0b1111 0yyz, where yy are the high-order bits of a 10-bit address and z is a don't care.

IICx_HMADR[A5:A6] are the two most significant bits of the 10-bit address. IICx_HMADR[A7] is a don't care.

Figure 24-6 describes the IICx_HMADR register bits.

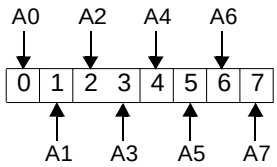


Figure 0-4. IICx High Master Address Register (IICx_HMADR)

0	A0	Address bit 0	1 for 10-bit addresses
1	A1	Address bit 1	1 for 10-bit addresses
2	A2	Address bit 2	1 for 10-bit addresses
3	A3	Address bit 3	1 for 10-bit addresses
4	A4	Address bit 4	0 for 10-bit addresses
5	A5	Address bit 5	MSb for 10-bit addresses
6	A6	Address bit 6	Next to MSb for 10-bit addresses

7	A7	Address bit 7	Don't care for 10-bit addresses

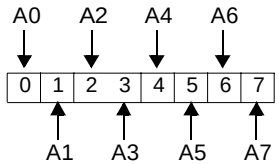


Figure 24-6. IICx High Master Address Register (IICx_HMADR)

0	A0	Address bit 0	1 for 10-bit addresses
1	A1	Address bit 1	1 for 10-bit addresses
2	A2	Address bit 2	1 for 10-bit addresses
3	A3	Address bit 3	1 for 10-bit addresses
4	A4	Address bit 4	0 for 10-bit addresses
5	A5	Address bit 5	MSb for 10-bit addresses
6	A6	Address bit 6	Next to MSb for 10-bit addresses
7	A7	Address bit 7	Don't care for 10-bit addresses

24.3.5 IICx Control Register (IICx_CNTL)

The IICx Control Register (IICx_CNTL) starts and stops IIC interface master transfers on the IIC bus. When a transfer begins, the IIC interface uses the values in IICx_CNTL to determine the type and size of the transfer.

> **Programming Note:** IICx_CNTL *must* be the last register programmed. Whenever IICx_CNTL[PT] = 1, the IIC interface attempts to perform the requested transfer using values set in other registers.

During and after transfers, the IICx_STS and IICx_EXTSTS registers can be read to determine the state of the IIC interface and the IIC bus.

Only IICx_CNTL[PT] is cleared when a requested master transfer is complete; the remaining bits are not affected.

Figure 24-7 describes the IICx_CNTL register bits.

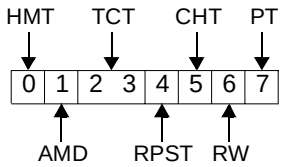


Figure 0-5. IICx Control Register (IICx_CNTL)

0	HMT	Halt Master Transfer 0 Normal transfer operation. 1 Issue Stop signal on the IIC bus as soon as possible to halt master transfer.	If no transfer is in progress, no action is taken. IICx_CNTL[PT] needs not be set. If IICx_MDCNTL[EINT] = 1, an interrupt is generated.
1	AMD	Addressing Mode 0 Use 7-bit addressing. 1 Use 10-bit addressing.	Does not affect slave transfers.
2:3	TCT	Transfer Count 00 Transfer one byte. 01 Transfer two bytes. 10 Transfer three bytes. 11 Transfer four bytes.	
4	RPST	Repeated Start 0 Normal start operation 1 Use repeated Start function to start transfer.	
5	CHT	Chain Transfer 0 Transfer is only or last transfer. 1 Transfer is one of a sequence of transfers (but not last in sequence).	Completion of a requested transfer causes a Stop condition to be generated on the IIC bus.
6	RW	Read/Write 0 Transfer is a write. 1 Transfer is a read.	
7	PT	Pending Transfer 0 Most recent requested transfer is complete. 1 Start transfer if bus is free.	

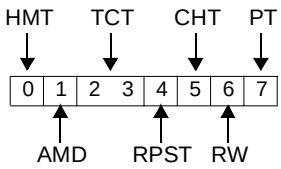


Figure 24-7. IICx Control Register (IICx_CNTL)

0	HMT	Halt Master Transfer 0 Normal transfer operation. 1 Issue Stop signal on the IIC bus as soon as possible to halt master transfer.	If no transfer is in progress, no action is taken. IICx_CNTL[PT] needs not be set. If IICx_MDCNTL[EINT] = 1, an interrupt is generated.
1	AMD	Addressing Mode 0 Use 7-bit addressing. 1 Use 10-bit addressing.	Does not affect slave transfers.
2:3	TCT	Transfer Count 00 Transfer one byte. 01 Transfer two bytes. 10 Transfer three bytes. 11 Transfer four bytes.	
4	RPST	Repeated Start 0 Normal start operation 1 Use repeated Start function to start transfer.	
5	CHT	Chain Transfer 0 Transfer is only or last transfer. 1 Transfer is one of a sequence of transfers (but not last in sequence).	Completion of a requested transfer causes a Stop condition to be generated on the IIC bus.
6	RW	Read/Write 0 Transfer is a write. 1 Transfer is a read.	
7	PT	Pending Transfer 0 Most recent requested transfer is complete. 1 Start transfer if bus is free.	

Table 24-2 summarizes IIC interface operation for settings of IICx_CNTL[HMT, RPST, CHT, PT]. x represents a don't care in the RPST, CHT and PT columns.

Table 24-2. IIC Response to IICx_CNTL Field Settings

IICx_CNTL Fields				Resulting Action on IIC Bus and Inside IIC Interface
HMT	RPST	CHT	PT	
0	x	x	0	No action taken
0	0	1	1	Start, Transfer, ACK on last byte, Pause
0	0	0	1	Start, Transfer, NACK on last byte, Stop
1	x	x	x	NACK on current byte, Stop
0	1	x	1	Start, Transfer, NACK on last byte, Wait

Settings of IICx_CNTL[HMT, RPST, CHT, PT] result in the following actions.

Start	IIC Start condition generated, if the IIC interface was stopped or waiting.
Stop	IIC Stop condition generated; IIC interface enters the Stop condition.
ACK	IIC Acknowledge condition generated.
NACK	IIC Not Acknowledge condition generated, if performing a read.
Transfer	Requested bytes are transferred.
Pause	IIC interface enters the Pause state.
Wait	IIC interface enters the Wait state.

IICx_CNTL[HMT] overrides IICx_CNTL[RPST, CHT].

IICx_CNTL[RPST, PT] overrides IICx_CNTL[CHT].

24.3.6 IICx Mode Control Register (IICx_MDCNTL)

The IICx Mode Control Register (IICx_MDCNTL) sets the major modes of operation on the IIC bus. In addition, IICx_MDCNTL can force the data buffers into the empty state.

In typical applications, IICx_MDCNTL is configured once, during software initialization. Applications providing complex error handling may reconfigure this register more often.

> **Programming Note:** IICx_CLKDIV must be initialized before IICx_MDCNTL. IICx_LSADR and IICx_HSADR should also be configured before IICx_MDCNTL.

Note that the IIC hardware does not implement time-out functions on the IIC bus. Such functions must be implemented, in software, by setting IICx_CNTL[HMT] = 1, or setting IICx_XTCNTLSS[SRST] = 1.

Regarding IICx_MDCNTL[HSCL], a "slave not ready" condition occurs during a slave receive operation, if there is no space in the slave data buffer at the start of a write operation, or if the slave data buffer fills during the write. In a slave transmit operation, a slave not ready condition occurs if there is no data in the slave data buffer at the start of a read operation, or if the slave data buffer becomes empty during the read.

Using IICx_MDCNTL[HSCL] to handle slave not ready conditions can affect system performance. A slave holding the IICSCL signal low guarantees data delivery to or from the requesting master, but prevents other masters from performing transfers over the IIC bus. There is no general rule for handling slave not ready conditions; each system has its own requirements.

Figure 24-8 describes the IICx_MDCNTL register bits.

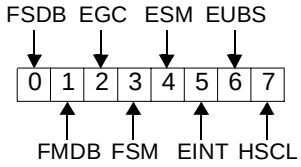


Figure 0-6. IICx Mode Control Register (IICx_MDCNTL)

0	FSDB	Flush Slave Data Buffer 0 Normal operation 1 Set slave data buffer to empty.	Cleared after buffer is emptied.
1	FMDB	Flush Master Data Buffer 0 Normal operation 1 Set master data buffer to empty.	Cleared after buffer is emptied.
2	EGC	Enable General Call 0 Ignore general call on IIC bus. 1 Respond to general call on IIC bus.	IICx_MDCNTL[ESM] overrides this field; if IICx_MDCNTL[ESM] = 1, a general call is ignored.
3	FSM	Fast/Standard Mode 0 IIC transfers run at 100 kHz (standard mode). 1 IIC transfers run at 400 kHz (fast mode).	
4	ESM	Enable Slave Mode 0 Slave transfers are ignored. 1 Slave transfers are enabled.	Program IICx_LSADR and IICx_HSADR before setting this field.
5	EINT	Enable Interrupt 0 Interrupts are disabled. 1 Enables interrupts for interrupts enabled in IICx_NTRMSK.	
6	EUBS	Exit Unknown IIC Bus State 0 Normal operation. 1 IIC bus control state machine exits unknown bus state, if in an unknown state.	If the IIC bus control state machine is in a known state, setting IICx_MDCNTL[EUBS] = 1 has no effect.
7	HSCL	Hold IIC Serial Clock Low 0 If slave is not ready, issue a NACK in response to slave transfer request. 1 If slave is not ready, hold the IICSCL signal low until slave is ready.	This field is used only when in slave mode.

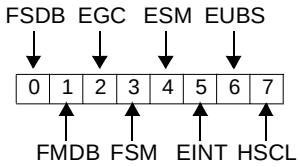


Figure 24-8. IICx Mode Control Register (IICx_MDCNTL)

0	FSDB	Flush Slave Data Buffer 0 Normal operation 1 Set slave data buffer to empty.	Cleared after buffer is emptied.
1	FMDB	Flush Master Data Buffer 0 Normal operation 1 Set master data buffer to empty.	Cleared after buffer is emptied.
2	EGC	Enable General Call 0 Ignore general call on IIC bus. 1 Respond to general call on IIC bus.	IICx_MDCNTL[ESM] overrides this field; if IICx_MDCNTL[ESM] = 1, a general call is ignored.
3	FSM	Fast/Standard Mode 0 IIC transfers run at 100 kHz (standard mode). 1 IIC transfers run at 400 kHz (fast mode).	
4	ESM	Enable Slave Mode 0 Slave transfers are ignored. 1 Slave transfers are enabled.	Program IICx_LSADR and IICx_HSADR before setting this field.
5	EINT	Enable Interrupt 0 Interrupts are disabled. 1 Enables interrupts for interrupts enabled in IICx_NTRMSK.	
6	EUBS	Exit Unknown IIC Bus State 0 Normal operation. 1 IIC bus control state machine exits unknown bus state, if in an unknown state.	If the IIC bus control state machine is in a known state, setting IICx_MDCNTL[EUBS] = 1 has no effect.
7	HSCL	Hold IIC Serial Clock Low 0 If slave is not ready, issue a NACK in response to slave transfer request. 1 If slave is not ready, hold the IICSCL signal low until slave is ready.	This field is used only when in slave mode.

24.3.7 IICx Status Register (IICx_STS)

The IICx Status register (IICx_STS) contains the state of the IIC interface and the status of any previously requested master transfers.

During and after transfers, software can read the IICx_STS and IICx_EXTSTS registers to determine the state of the IIC interface and the IIC bus.

Programming Note: IICx_STS should be the first register read by an interrupt or error handler routine. IICx_STS can also be read in a polling loop if the IIC interrupts are not used.

All IICx_STS bit fields except for IICx_STS[SSS] must be cleared before requesting another master transfer. IICx_STS[SSS] is not affected by master transactions.

Figure 24-9 describes the IICx_STS register bits.

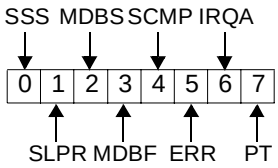


Figure 0-7. IICx Status Register (IICx_STS)

0	SSS	Slave Status Set 0 No slave operations are in progress. 1 Slave operation is in progress.	Read-only; this field is set when any of the following fields are set: IICx_XTCNTLSS[SRC, SRS, SWC, SWS].
1	SLPR	Sleep Request 0 Normal operation. 1 Sleep mode (CPC0_ER[IIC] = 1).	Read-only. The IIC interface is awakened when a start signal is detected on the IIC bus or when the CPC0_ER[IICx] is cleared.
2	MDBS	Master Data Buffer Status 0 Master data buffer is empty. 1 Master data buffer contains data.	Read-only.
3	MDBF	Master Data Buffer Full 0 Master data buffer is not full. 1 Master data buffer is full.	Read-only.
4	SCMP	Stop Complete 0 No request to halt transfer, or master data transfer, is complete. 1 Request to halt transfer, or master data transfer, is complete.	To clear IICx_STS[SCMP], set IICx_STS[SCMP] = 1.
5	ERR	Error 0 No error has occurred. 1 One of the following fields is set: IICx_EXTSTS[LA, ICT, XFRA] = 1.	Read-only.
6	IRQA	IRQ Active 0 No IIC interrupt has been sent to the universal interrupt controller (UIC). 1 An IIC interrupt has been sent to the UIC.	To clear IICx_STS[IRQA], set IICx_STS[IRQA] = 1. If IICx_MDCNTL[EINT] = 0, then IICx_STS[IRQA] is not set.
7	PT	Pending Transfer 0 No transfer is pending, or transfer is in progress. 1 Transfer is pending.	Read-only.

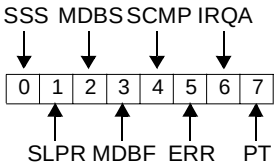


Figure 24-9. IICx Status Register (IICx_STS)

0	SSS	Slave Status Set 0 No slave operations are in progress. 1 Slave operation is in progress.	Read-only; this field is set when any of the following fields are set: IICx_XTCNTLSS[SRC, SRS, SWC, SWS].
1	SLPR	Sleep Request 0 Normal operation. 1 Sleep mode (CPC0_ER[IIC] = 1).	Read-only. The IIC interface is awakened when a start signal is detected on the IIC bus or when the CPC0_ER[IICx] is cleared.
2	MDBS	Master Data Buffer Status 0 Master data buffer is empty. 1 Master data buffer contains data.	Read-only.
3	MDBF	Master Data Buffer Full 0 Master data buffer is not full. 1 Master data buffer is full.	Read-only.
4	SCMP	Stop Complete 0 No request to halt transfer, or master data transfer, is complete. 1 Request to halt transfer, or master data transfer, is complete.	To clear IICx_STS[SCMP], set IICx_STS[SCMP] = 1.
5	ERR	Error 0 No error has occurred. 1 One of the following fields is set: IICx_EXTSTS[LA, ICT, XFRA] = 1.	Read-only.
6	IRQA	IRQ Active 0 No IIC interrupt has been sent to the universal interrupt controller (UIC). 1 An IIC interrupt has been sent to the UIC.	To clear IICx_STS[IRQA], set IICx_STS[IRQA] = 1. If IICx_MDCNTL[EINT] = 0, then IICx_STS[IRQA] is not set.
7	PT	Pending Transfer 0 No transfer is pending, or transfer is in progress. 1 Transfer is pending.	Read-only.

The Error and Pending Transfer, IICx_STS[ERR, PT], bit fields indicate the success or failure of the requested transfer. Table 24-3 lists the transfer status for all combinations of the IICx_STS[ERR,PT] bit fields.

Table 24-3. IICx_STS[ERR, PT] Decoding

ERR	PT	Status
0	0	Requested transfer completed without errors
0	1	Requested transfer is in progress; no errors were detected
1	0	Requested transfer is complete, but not all data was transferred
1	1	Requested transfer is in progress; but an error was detected

Programming Note: No action regarding a master transfer should be taken unless all pending transfers have completed, IICx_STS[PT] = 0.

If an error requires the IIC interface to send a Stop, the Stop Complete bit field is set, IICx_STS[SCMP] = 1. Note that slave operations should be serviced regardless of the state of a requested master transfer.

IIC1_MDCNTL[EUBS] must be set after a reset before IIC interface can be placed in sleep mode. The IIC interface is placed in sleep mode by setting the CPC0_ER[IICx] via software. Awaking the IIC interface is possible directly through software by clearing the CPC0_ER[IICx] or indirectly by detecting a Start condition on the IIC bus. When a Start condition is detected, the IIC interface is awakened, clearing the CPC0_ER[IICx] and the IICx_STS[SLPR].

The IICx_STS[MDBS, MDBF] contain the current status of the Master Data Buffer, IICx_MDBUF. When the IICx_MDBUF contains data, IICx_STS[MDBS] is set. When the IICx_MDBUF is full, IICx_STS[MDBF] is set.

The state of the IICx_MDBUF is not instantly recorded by the IICx_STS[MDBS, MDBF]. The delay depends on the size of the buffer access. For halfword accesses, these fields are valid on the third OPB clock following the transfer. For byte accesses, these fields are valid on the second OPB clock following the transfer.

24.3.8 IICx Extended Status Register (IICx_EXTSTS)

The IICx Extended Status register (IICx_EXTSTS) reports additional IIC status.

During and after transfers, software can read the IICx_STS and IICx_EXTSTS registers to determine the state of the IIC interface and the IIC bus.

Figure 24-10 describes the IICx_EXTSTS register bits.

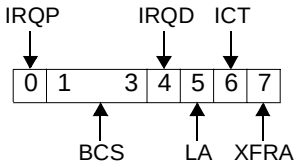


Figure 0-8. IICx Extended Status Register (IICx_EXTSTS)

0	IRQP	IRQ Pending 0 No IRQ is pending. 1 An IRQ is active, another IRQ is on-deck, and another interrupt-generating condition has occurred.	• IICx_EXTSTS[IRQP] might be set momentarily while an IRQ moves from the Pending to the On-deck state. • An interrupt remains pending, IICx_EXTSTS[IRQP]=1, until the current on-deck interrupt becomes active, IICx_STS[IRQD]=0 and IICx_STS[IRQA]=1. • Writing 1 to IICx_EXTSTS[IRQP] clears the field. • When the IIC interrupt is disabled, IICx_MDCNTL[IRQP] = 0, IICx_EXTSTS[IRQP] should be ignored.

1:3	BCS	Bus Control State 000 Unused; if this value is read an error occurred. 001 Slave-selected state; the IIC interface has detected and decoded a slave transfer request on the IIC bus. 010 Slave Transfer state; the IIC interface has detected but has not decoded a slave transfer request on the IIC bus. 011 Master Transfer state; entered after a master transfer request has started on the IIC bus. 100 Free Bus state; the bus is free and no transfer request is pending. 101 Busy Bus state; the bus is busy. 110 Unknown state; value after IIC reset. 111 Unused; if this value is read an error occurred.	Read-only.
4	IRQD	IRQ On-Deck 0 No IRQ is on-deck. 1 An interrupt is active, and another interrupt-generating condition has occurred.	• IICx_EXTSTS[IRQD] might be set momentarily while an IRQ moves from the On-deck to the Active state. • An interrupt remains on-deck, IICx_EXTSTS[IRQD = 1, until the current active interrupt is no longer active, IICx_STS[IRQA] = 0. • If IICx_EXTSTS[IRQP] = 1, IICx_EXTSTS[IRQD] is set on the next OPB clock. • Writing 1 to IICx_EXTSTS[IRQD] clears the field. • When the IIC interrupt is disabled, IICx_MDCNTL[IRQP]=0, IICx_EXTSTS[IRQD] should be ignored.
5	LA	Lost Arbitration 0 Normal operation. 1 Loss of arbitration has ended the requested master transfer.	• If arbitration is lost, any requested master transaction may have terminated prematurely. Read data may be incomplete and not all write data may have been written. • If arbitration is lost during a repeat start, the master may not own the IIC bus.
6	ICT	Incomplete Transfer 0 Normal operation. 1 Some of the bytes of the requested master transfer were not transferred.	For an incomplete transfer, read the transfer count, IICx_XFRCNT, to determine how bytes were transferred.

7	XFRA	Transfer Aborted 0 No transfer is pending, or transfer is in progress. 1 A requested master transfer was aborted by a NACK during the transfer of the address byte, or was aborted because arbitration was lost. Lost arbitration can be caused by the loss of data during the transfer of the second or subsequent data byte.	Transfer aborted. When set to a 1, a requested master transfer was aborted by a NOT acknowledge during the transfer of the address byte. It is also set to a 1 when a requested master transfer loses data. Lost arbitration can be caused by the loss of data during the transfer of the second or subsequent data byte.

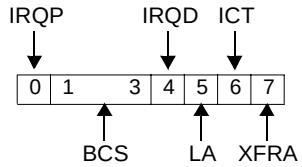


Figure 24-10. IICx Extended Status Register (IICx_EXTSTS)

0	IRQP	IRQ Pending 0 No IRQ is pending. 1 An IRQ is active, another IRQ is on-deck, and another interrupt-generating condition has occurred.	• IICx_EXTSTS[IRQP] might be set momentarily while an IRQ moves from the Pending to the On-deck state. • An interrupt remains pending, IICx_EXTSTS[IRQP]=1, until the current on-deck interrupt becomes active, IICx_STS[IRQD]=0 and IICx_STS[IRQA]=1. • Writing 1 to IICx_EXTSTS[IRQP] clears the field. • When the IIC interrupt is disabled, IICx_MDCNTL[IRQP] = 0, IICx_EXTSTS[IRQP] should be ignored.
1:3	BCS	Bus Control State 000 Unused; if this value is read an error occurred. 001 Slave-selected state; the IIC interface has detected and decoded a slave transfer request on the IIC bus. 010 Slave Transfer state; the IIC interface has detected but has not decoded a slave transfer request on the IIC bus. 011 Master Transfer state; entered after a master transfer request has started on the IIC bus. 100 Free Bus state; the bus is free and no transfer request is pending. 101 Busy Bus state; the bus is busy. 110 Unknown state; value after IIC reset. 111 Unused; if this value is read an error occurred.	Read-only.

4	IRQD	IRQ On-Deck 0 No IRQ is on-deck. 1 An interrupt is active, and another interrupt-generating condition has occurred.	• IICx_EXTSTS[IRQD] might be set momentarily while an IRQ moves from the On-deck to the Active state. • An interrupt remains on-deck, IICx_EXTSTS[IRQD = 1, until the current active interrupt is no longer active, IICx_STS[IRQA] = 0. • If IICx_EXTSTS[IRQP] = 1, IICx_EXTSTS[IRQD] is set on the next OPB clock. • Writing 1 to IICx_EXTSTS[IRQD] clears the field. • When the IIC interrupt is disabled, IICx_MDCNTL[IRQP]=0, IICx_EXTSTS[IRQD] should be ignored.
5	LA	Lost Arbitration 0 Normal operation. 1 Loss of arbitration has ended the requested master transfer.	• If arbitration is lost, any requested master transaction may have terminated prematurely. Read data may be incomplete and not all write data may have been written. • If arbitration is lost during a repeat start, the master may not own the IIC bus.
6	ICT	Incomplete Transfer 0 Normal operation. 1 Some of the bytes of the requested master transfer were not transferred.	For an incomplete transfer, read the transfer count, IICx_XFRCNT, to determine how bytes were transferred.
7	XFRA	Transfer Aborted 0 No transfer is pending, or transfer is in progress. 1 A requested master transfer was aborted by a NACK during the transfer of the address byte, or was aborted because arbitration was lost. Lost arbitration can be caused by the loss of data during the transfer of the second or subsequent data byte.	Transfer aborted. When set to a 1, a requested master transfer was aborted by a NOT acknowledge during the transfer of the address byte. It is also set to a 1 when a requested master transfer loses data. Lost arbitration can be caused by the loss of data during the transfer of the second or subsequent data byte.

IICx_EXTSTS[IRQP, IRQD] and IICx_STS[IRQA] provide a FIFO for storing interrupts. A new interrupt is considered pending, and remains pending while an on-deck interrupt is present. Once the on-deck interrupt becomes active, the pending interrupt moves on-deck, and remains on-deck until there is no active interrupt. When the active interrupt is cleared, the on-deck (initially pending) interrupt becomes active.

Programming Note: An active interrupt remains active until software clears it.

IICx_EXTSTS[BCS] indicates the state of the IIC interface. The field is read-only.

IICx_EXTSTS[LA, ICT, XFRA] are cleared when IICx_EXTSTS[XFRA] = 1.

Writing 1 to IICx_EXTSTS[IRQP, IRQD, LA, ICT, XFRA] clears these fields.

24.3.9 IICx Low Slave Address Register (IICx_LSADR)

The IICx Low Slave Address Register (IICx_LSADR) and IICx_ High Slave Address Register (IICx_HSADR) program the slave address of the IIC interface. IICx_HSADR is used only for 10-bit addressing, and is not programmed in 7-bit addressing mode.

When 7-bit addressing is used, IICx_LSADR is written with the slave address; IICx_HSADR must be written with zeros. For 7-bit addressing, IICx_LSADR[A0:A6] contain the address transmitted on the IIC bus; IICx_LSADR[A7] is a don't care.

When 10-bit addressing is used, IICx_LSADR[A0:A7] contain the second address byte transmitted on the IIC bus.

Figure 24-5 describes the IICx_LSADR register bits.

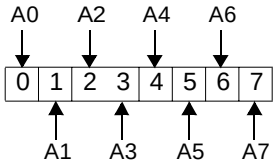


Figure 0-9. IICx Low Slave Address Register (IICx_LSADR)

0	A0	Address bit 0	
1	A1	Address bit 1	
2	A2	Address bit 2	
3	A3	Address bit 3	
4	A4	Address bit 4	
5	A5	Address bit 5	
6	A6	Address bit 6	LSb for 7-bit addresses
7	A7	Address bit 7	LSb for 10-bit addresses; don't care for 7-bit addresses

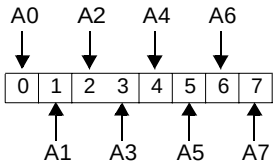


Figure 24-11. IICx Low Slave Address Register (IICx_LSADR)

0	A0	Address bit 0
1	A1	Address bit 1
2	A2	Address bit 2
3	A3	Address bit 3

4	A4	Address bit 4	
5	A5	Address bit 5	
6	A6	Address bit 6	LSb for 7-bit addresses
7	A7	Address bit 7	LSb for 10-bit addresses; don't care for 7-bit addresses

24.3.10 IICx High Slave Address Register (IICx_HSADR)

For 7-bit addressing, set IICx High Slave Address Register (IICx_HSADR) to 0.

To enable 10-bit slave addressing, IICx_HSADR must be programmed to 0b1111 0*yyx*, where *yy* are the high-order bits of a 10-bit address and *x* is a don't care.

Programming Note: IICx_HSADR is used only for 10-bit addressing, and should be set to 0 for 7-bit addressing mode.

Thus, in 10-bit address mode, IICx_HSADR[A6:A7] contain the two highest -order bits of the 10-bit address; IICx_HSADR[A7] is a don't care. IICx_LSADR contains the low-order byte of the 10-bit address.

Figure 24-12 describes the IICx_HSADR register bits.

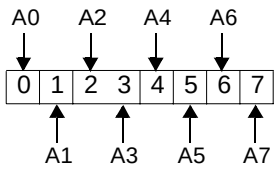


Figure 0-10. IICx High Slave Address Register (IICx_HSADR)

0	A0	Address bit 0	1 for 10-bit addresses
1	A1	Address bit 1	1 for 10-bit addresses
2	A2	Address bit 2	1 for 10-bit addresses
3	A3	Address bit 3	1 for 10-bit addresses
4	A4	Address bit 4	0 for 10-bit addresses
5	A5	Address bit 5	MSb for 10-bit addresses
6	A6	Address bit 6	Next to MSb for 10-bit addresses
7	A7	Address bit 7	Don't care for 10-bit addresses

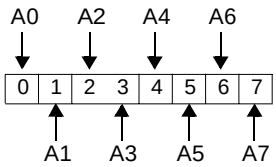


Figure 24-12. IICx High Slave Address Register (IICx_HSADR)

0	A0	Address bit 0	1 for 10-bit addresses
1	A1	Address bit 1	1 for 10-bit addresses
2	A2	Address bit 2	1 for 10-bit addresses
3	A3	Address bit 3	1 for 10-bit addresses
4	A4	Address bit 4	0 for 10-bit addresses
5	A5	Address bit 5	MSb for 10-bit addresses
6	A6	Address bit 6	Next to MSb for 10-bit addresses
7	A7	Address bit 7	Don't care for 10-bit addresses

Thus, in 10-bit address mode, bits 0:6 are used to decode the first address byte that was transmitted on the IIC bus, and bit 7 is in a don't care.

24.3.11 IICx Clock Divide Register (IICx_CLKDIV)

The IICx Clock Divide Register (IICx_CLKDIV) establishes a reference between the OPB clock and the IIC bus serial clock.

> **Programming Note:** IICx_CLKDIV must be initialized before IICx_MDCTRL. Until IICx_CLKDIV is initialized, all IIC bus activity is ignored.

Figure 24-13 describes the IICx_CLKDIV register bits.

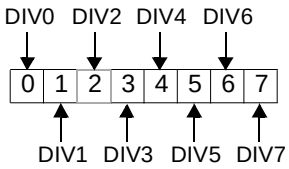


Figure 0-11. IICx Clock Divide Register (IICx_CLKDIV)

0	DIV0	Divisor bit 0
1	DIV1	Divisor bit 1
2	DIV2	Divisor bit 2
3	DIV3	Divisor bit 3
4	DIV4	Divisor bit 4

5	DIV5	Divisor bit 5
6	DIV6	Divisor bit 6
7	DIV7	Divisor bit 7

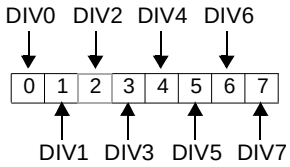


Figure 24-13. IICx Clock Divide Register (IICx_CLKDIV)

0	DIV0	Divisor bit 0
1	DIV1	Divisor bit 1
2	DIV2	Divisor bit 2
3	DIV3	Divisor bit 3
4	DIV4	Divisor bit 4
5	DIV5	Divisor bit 5
6	DIV6	Divisor bit 6
7	DIV7	Divisor bit 7

IICx_CLKDIV divides PPC440GP's on-chip peripheral bus (OPB) clock to form the base clock for the IIC bus.

Table 22-4 lists the divisor values for several OPB frequency ranges. These divisor values apply for standard and fast mode. Select the divisor value by matching the OPB clock frequency to the corresponding frequency range in Table 22-4. For example, if the OPB clock frequency is 50MHz, select a divisor value of 0x4.

Table 0-1. IICx Clock Divide Programming

OPB Frequency Range (MHz)	Divisor Value
20	0x1
$20 < f \leq 30$	0x2
$30 < f \leq 40$	0x3
$40 < f \leq 50$	0x4
$50 < f \leq 60$	0x5

Table 24-4. IICx Clock Divide Programming

OPB Frequency Range (MHz)	DivisorValue
20	0x1
20 < *f* ≤ 30	0x2
30 < *f* ≤ 40	0x3
40 < *f* ≤ 50	0x4
50 < *f* ≤ 60	0x5
60 < *f* ≤ 70	0x6
70 < *f* ≤ 80	0x7
80 < *f* ≤ 90	0x8
90 < *f* ≤ 100	0x9
100 < *f* ≤ 110	0xA
110 < *f* ≤ 120	0xB
120 < *f* ≤ 130	0xC
130 < *f* ≤ 140	0xD
140 < *f* ≤ 150	0xE

24.3.12 IICx Interrupt Mask Register (IICx_INTRMSK)

The IICx Interrupt Mask Register (IICx_INTRMSK) specifies which conditions can generate an IIC interrupt when the IIC interrupt is enabled, IICx_MDCNTL[EINT]=1.

Figure 24-14 describes the IICx_INTRMSK register bits.

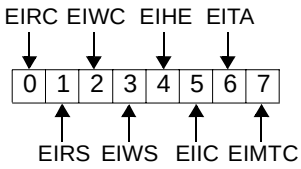


Figure 0-12. IICx Interrupt Mask Register (IICx_INTRMSK)

0	EIRC	Enable IRQ on Slave Read Complete 0 Disable 1 Enable	The interrupt is activated upon receipt of a Stop during a slave read on the IIC bus. IICx_XTCNTLSS[SRC] = 1 indicates a Slave Read Complete.
1	EIRS	Enable IRQ on Slave Read Needs Service 0 Disable 1 Enable	The interrupt is activated upon receipt of a slave read on the IIC bus and the slave buffer was empty or went empty and more data was requested on the IIC bus. **Note:** IICx_XTCNTLSS[SRS] = 1 indicates a Slave Read Needs Service.

2	EIWC	Enable IRQ on Slave Write Complete 0 Disable 1 Enable	The interrupt is activated upon receipt of a Stop during a slave write on the IIC bus. **Note:** IICx_XTCNTLSS[SWC] = 1 indicates a Slave Write Compete.
3	EIWS	Enable IRQ on Slave Write Needs Service 0 Disable 1 Enable	The interrupt is activated when the slave buffer becomes full during a slave write on the IIC bus. **Note:** IICx_XTCNTLSS[SWS] = 1 indicates a Slave Write Needs Service.
4	EIHE	Enable IRQ on Halt Executed 0 Disable 1 Enable	
5	EIIC	Enable IRQ on Incomplete Transfer 0 Disable 1 Enable	
6	EITA	Enable IRQ on Transfer Aborted 0 Disable 1 Enable	
7	EIMTC	Enable IRQ on Requested Master Transfer Complete 0 Disable 1 Enable	

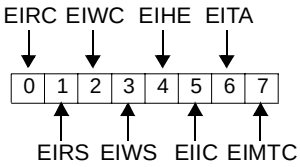


Figure 24-14. IICx Interrupt Mask Register (IICx_INTRMSK)

0	EIRC	Enable IRQ on Slave Read Complete 0 Disable 1 Enable	The interrupt is activated upon receipt of a Stop during a slave read on the IIC bus. IICx_XTCNTLSS[SRC] = 1 indicates a Slave Read Complete.
1	EIRS	Enable IRQ on Slave Read Needs Service 0 Disable 1 Enable	The interrupt is activated upon receipt of a slave read on the IIC bus and the slave buffer was empty or went empty and more data was requested on the IIC bus. **Note:** IICx_XTCNTLSS[SRS] = 1 indicates a Slave Read Needs Service.
2	EIWC	Enable IRQ on Slave Write Complete 0 Disable 1 Enable	The interrupt is activated upon receipt of a Stop during a slave write on the IIC bus. **Note:** IICx_XTCNTLSS[SWC] = 1 indicates a Slave Write Compete.

3	EIWS	Enable IRQ on Slave Write Needs Service 0 Disable 1 Enable	The interrupt is activated when the slave buffer becomes full during a slave write on the IIC bus. **Note:** IICx_XTCNTLSS[SWS] = 1 indicates a Slave Write Needs Service.
4	EIHE	Enable IRQ on Halt Executed 0 Disable 1 Enable	
5	EIIC	Enable IRQ on Incomplete Transfer 0 Disable 1 Enable	
6	EITA	Enable IRQ on Transfer Aborted 0 Disable 1 Enable	
7	EIMTC	Enable IRQ on Requested Master Transfer Complete 0 Disable 1 Enable	

24.3.13 IICx Transfer Count Register (IICx_XFRCNT)

The IICx Transfer Count Register (IICx_XFRCNT) reports the number of bytes transferred on the IIC bus during a master or a slave operation.

Figure 24-15 describes the IICx_XFRCNT register bits.

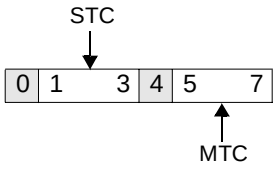


Figure 0-13. IICx Transfer Count Register (IICx_XFRCNT)

0		Reserved
1:3	STC	Slave Transfer Count 000 0 bytes transferred 001 1 byte transferred 010 2 bytes transferred 011 3 bytes transferred 100 4 bytes transferred 101 Reserved 110 Reserved 111 Reserved
4		Reserved

5:7	MTC	Master Transfer Count 000 0 bytes transferred 001 1 byte transferred 010 2 bytes transferred 011 3 bytes transferred 100 4 bytes transferred 101 Reserved 110 Reserved 111 Reserved

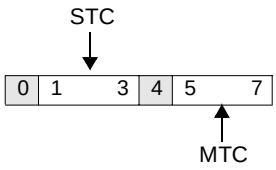


Figure 24-15. IICx Transfer Count Register (IICx_XFRCNT)

0		Reserved
1:3	STC	Slave Transfer Count 000 0 bytes transferred 001 1 byte transferred 010 2 bytes transferred 011 3 bytes transferred 100 4 bytes transferred 101 Reserved 110 Reserved 111 Reserved
4		Reserved
5:7	MTC	Master Transfer Count 000 0 bytes transferred 001 1 byte transferred 010 2 bytes transferred 011 3 bytes transferred 100 4 bytes transferred 101 Reserved 110 Reserved 111 Reserved

IICx_XFRCNT[MTC] is cleared when there is a pending transfer, IICx_CNTL[PT] = 1.

IICx_XFRCNT[STC] is cleared when:

- A slave operation starts on the IIC bus
- Software indicates the slave does not need service by clearing IICx_XTCNTLSS[SRS] or IICx_XTCNTLSS[SWS].

24.3.14 IICx Extended Control and Slave Status Register (IICx_XTCNTLSS)

The IICx Extended Control and Slave Status Register (IICx_XTCNTLSS) provides additional control of IIC interface functions and reports the status of slave operations.

Figure 24-16 describes the IICx_XTCNTLSS register bits.

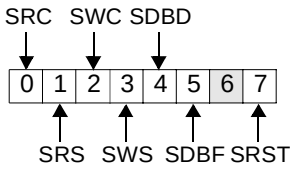


Figure 0-14. IICx Extended Control and Slave Status Register (IICx_XTCNTLSS)

0	SRC	Slave Read Complete 0 Normal operation, or IICx_MDCNTL[HSCL] = 0, IICx_SDBUF is empty, and a read operation is in progress. 1 A NACK or Stop condition was received over the IIC bus, or a repeated Start condition ended a read operation.	Check whether the read operation emptied IICx_SDBUF.
1	SRS	Slave Read Needs Service 0 Normal operation or slave read does not need service. 1 IICx_SDBUF is empty, and a read operation was requested on the IIC bus. The set condition may also indicate that IICx_SDBUF is empty due to a slave read and additional data is requested by the master.	1. If IICx_MDCNTL[HSCL]=0 and IICx_SDBUF contains no data, the slave will issue a NACK and IICx_XTCNTLSS[SRS] is set. 2. If ICx_MDCNTL[HSCL]=0, and IICx_SDBUF contains data, the slave will send the data. IICx_XTCNTLSS[SRS] is not set unless the master request additional data. 3. If IICx_MDCNTL[HSCL]=0, and IICx_SDBUF contains no data, the slave will hold IICSCL low to indicate the slave is busy. IICx_XTCNTLSS[SRS] is set until the IICx_SDBUF is filled. Once filled, IICSCL is released, IICx_XTCNTLSS[SRS] is cleared, and the slave sends the data. 4. If IICx_MDCNTL[HSCL]=1, and IICx_SDBUF contains data, the slave will send the data. IICx_XTCNTLSS[SRS] is not set unless the master requests additional data.

2	SWC	Slave Write Complete 0 Normal operation or slave write in progress. 1 A Stop signal was received during a write operation, or a repeated Start condition ended a write operation.	
3	SWS	Slave Write Needs Service 0 Normal operation or slave write does not need service. 1 IICx_SDBUF is full during a slave write.	1. If IICx_MDCNTL[HSCL] = 1 and IICx_SDBUF is full, the slave will hold IICSCL low to indicate the slave is busy. IICx_XTCNTLSS[SWS] is set until IICx_SDBUF is empty. Once empty, IICSCL is released, IICx_XTCNTLSS[SWS] is cleared, and the slave receives the data. 2. If IICx_MDCNTL[HSCL] = 0 and IICx_SDBUF is full, the slave will issue a NACK and IICx_XTCNTLSS[SWS] is set.
4	SDBD	Slave Data Buffer Has Data 0 IICx_SDBUF is empty 1 IICx_SDBUF contains data	Read-only
5	SDBF	Slave Data Buffer Full 0 IICx_SDBUF is not full 1 IICx_SDBUF is full	Read-only
6		Reserved	
7	SRST	Soft Reset 0 Normal operation 1 Soft reset	

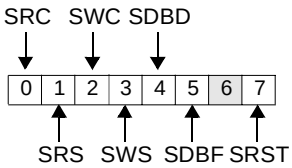


Figure 24-16. IICx Extended Control and Slave Status Register (IICx_XTCNTLSS)

0	SRC	Slave Read Complete 0 Normal operation, or IICx_MDCNTL[HSCL] = 0, IICx_SDBUF is empty, and a read operation is in progress. 1 A NACK or Stop condition was received over the IIC bus, or a repeated Start condition ended a read operation.	Check whether the read operation emptied IICx_SDBUF.

1	SRS	Slave Read Needs Service 0 Normal operation or slave read does not need service. 1 IICx_SDBUF is empty, and a read operation was requested on the IIC bus. The set condition may also indicate that IICx_SDBUF is empty due to a slave read and additional data is requested by the master.	1. If IICx_MDCNTL[HSCL]=0 and IICx_SDBUF contains no data, the slave will issue a NACK and IICx_XTCNTLSS[SRS] is set. 2. If ICx_MDCNTL[HSCL]=0, and IICx_SDBUF contains data, the slave will send the data. IICx_XTCNTLSS[SRS] is not set unless the master request additional data. 3. If IICx_MDCNTL[HSCL]=0, and IICx_SDBUF contains no data, the slave will hold IICSCL low to indicate the slave is busy. IICx_XTCNTLSS[SRS] is set until the IICx_SDBUF is filled. Once filled, IIC-SCL is released, IICx_XTCNTLSS[SRS] is cleared, and the slave sends the data. 4. If IICx_MDCNTL[HSCL]=1, and IICx_SDBUF contains data, the slave will send the data. IICx_XTCNTLSS[SRS] is not set unless the master requests additional data.
2	SWC	Slave Write Complete 0 Normal operation or slave write in progress. 1 A Stop signal was received during a write operation, or a repeated Start condition ended a write operation.	
3	SWS	Slave Write Needs Service 0 Normal operation or slave write does not need service. 1 IICx_SDBUF is full during a slave write.	1. If IICx_MDCNTL[HSCL] = 1 and IICx_SDBUF is full, the slave will hold IICSCL low to indicate the slave is busy. IICx_XTCNTLSS[SWS] is set until IICx_SDBUF is empty. Once empty, IICSCL is released, IICx_XTCNTLSS[SWS] is cleared, and the slave receives the data. 2. If IICx_MDCNTL[HSCL] = 0 and IICx_SDBUF is full, the slave will issue a NACK and IICx_XTCNTLSS[SWS] is set.
4	SDBD	Slave Data Buffer Has Data 0 IICx_SDBUF is empty 1 IICx_SDBUF contains data	Read-only
5	SDBF	Slave Data Buffer Full 0 IICx_SDBUF is not full 1 IICx_SDBUF is full	Read-only
6		Reserved	
7	SRST	Soft Reset 0 Normal operation 1 Soft reset	

Writing a 1 to IICx_XTCNTLSS[SRC, SRS, SWC, SWS] clears these fields.

The IICx_XTCNTLSS[SBSS, SDBF] contain the current status of the Slave Data Buffer, IICx_SDBUF. When the IICx_SDBUF contains data, IICx_XTCNTLSS[SDBD] is set. When the IICx_SDBF is full, IICx_XTCNTLSS[SDBF] is set.

The state of the IICx_SDBUF is not instantly recorded by the IICx_XTCNTL[SDBD, SDBF]. The delay depends on the size of the buffer access. For half-word accesses, these fields are valid on the third OPB clock following the transfer. For byte accesses, these fields are valid on the second OPB clock following the transfer.

If any of the following fields: IICx_XTCNTLSS[SRC, SRS, SWC, SWS] = 1 and IICx_MDCNTL[HSCL] = 0; no new slave operations will be accepted over the IIC bus. A NACK is issued until IICx_XTCNTLSS[SRS] or IICx_XTCNTLSS[SRS], and IICx_XTCNTLSS[SRS] or IICx_XTCNTLSS[SRS], are cleared.

Soft reset, IICx_XTCNTLSS[SRST], provides a last means of recovery from IIC interface or IIC bus failure. Once enabled, soft reset completely resets the IIC interface. All IIC registers are affected. All transmissions from the IIC interface are terminated. Enabling soft reset during an IIC transmission may improperly terminate the transmission and hang the IIC bus.

24.3.15 IICx Direct Control Register (IICx_DIRECTCNTL)

The IICx Direct Control Register (IICx_DIRECTCNTL), which controls and monitors the IIC serial clock (IICSCL) and serial data (IICSDA) signal, is used for error recovery when a malfunction is detected on the IIC interface.

Figure 24-17 describes the IICx_DIRECTCNTL register bits.

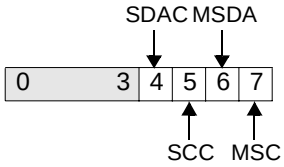


Figure 0-15. IICx Direct Control Register (IICx_DIRECTCNTL)

0:3		Reserved	
4	SDAC	IICSDA Output Control Directly controls the IICSDA output. 0 IICSDA is a logic 0 1 IICSDA is a logic 1	
5	SCC	IICSCL Output Control Directly controls the IICSCL output 0 IICSCL is a logic 0 1 IICSCL is a logic 1	
6	MSDA	Monitor IICSDA Used to monitor the IICSDA input 0 IICSDA is a logic 0 1 IICSDA is a logic 1	Read-only
7	MSC	Monitor IICSCL. Used to monitor the IICSCL input. 0 IICSCL is a logic 0 1 IICSCL is a logic 1	Read-only

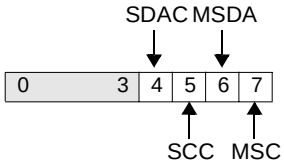


Figure 24-17. IICx Direct Control Register (IICx_DIRECTCNTL)

0:3		Reserved	
4	SDAC	IICSDA Output Control Directly controls the IICSDA output. 0 IICSDA is a logic 0 1 IICSDA is a logic 1	
5	SCC	IICSCL Output Control Directly controls the IICSCL output 0 IICSCL is a logic 0 1 IICSCL is a logic 1	
6	MSDA	Monitor IICSDA Used to monitor the IICSDA input 0 IICSDA is a logic 0 1 IICSDA is a logic 1	Read-only
7	MSC	Monitor IICSCL. Used to monitor the IICSCL input. 0 IICSCL is a logic 0 1 IICSCL is a logic 1	Read-only

IICx_DIRECTCNTL[SDAC, SCC] can be written to control the IICSDA and IICSCL signals. When controlling the IICSDA and IICSCL signals directly, the IIC controller must be placed in the reset state, IICx_XTCNTL[SRST] = 1.

IICx_DIRECTCNTL[MSDA, MSC] are used to verify that IICx_DIRECTCNTL[SDAC, SCC] were written successfully, and that the IICSCL signals can be controlled. If IICx_DIRECTCNTL[MSDA, MSC] do not correspond to IICx_DIRECTCNTL[SDAC, SCC], respectively, toggle IICSCL repeatedly to regain control.

IICx_DIRECTCNTL[SDAC, SCC, MDSA, MSC] = 1 after a chip or system reset. A Soft Reset, IICx_XTCNTLSS[SRST] = 1, does not affect the state of IICx_DIRECRCNTL.

24.4 Interrupt Handling

Service request on the IIC interface can be monitored by polling the IICx_STS[SSS, PT] or by using the IIC interrupt to the UIC. When the IICx_MDCNTL[EINT] is enabled, the IIC interface generates an interrupt to the UIC if an unmasked IIC interrupt condition occurs. The interrupt is recorded by the UIC if UIC0_ER[IICx] is enabled. The conditions that generate an IIC interrupt to the UIC are determined by the interrupt mask settings in IICx_INTMSK.

The IIC interface can queue up to three interrupts since there is one interrupt for master operations and two for slave operations. The current interrupt is referred to as the *active* interrupt. The first interrupt in the queue is the on-deck interrupt; the second queued interrupt is called the pending interrupt. The queue holds multiple

interrupts until the active interrupt is cleared by writing a 1 to IICx_STS[IRQA]. When an active interrupt is cleared, the on-deck interrupt becomes the active interrupt and the pending interrupt becomes the on-deck interrupt.

When multiple interrupts occur, the status of the active interrupt and the queued interrupt merge making it impossible to determine which of the conditions originated the active interrupt. An interrupt handle should therefore save the contents of the status registers (IICx_STS and IICx_XTCTLSS) and handle all conditions set. Once handled, the status conditions can be cleared by overwriting the status registers with their saved content. If another IIC interrupt condition occurs before clearing the status register, the status bit of this condition is preserved since status bits are cleared when written with a 1.

When multiple interrupts occur, the status of the active interrupt and the queued interrupt merge making it impossible to determine which of the conditions originated the active interrupt. An interrupt handle should therefore save the contents of the status registers (IICx_STS and IICx_XTCTLSS) and handle all conditions set. Once handled, the status conditions can be cleared by overwriting the status registers with their saved content. If another IIC interrupt condition occurs before clearing the status register, the status bit of this condition is preserved since status bits are cleared when written with a 1.

Under certain conditions, the IIC interface merges slave read (write) needs service and slave read (write) complete interrupts into one interrupt. If a slave read (write) needs service interrupt is active, or queued, and a slave read (write) complete interrupt occurs, and IICx_XTCNTLSS has not yet been read, the two interrupts are merged into a single interrupt. This merge function is performed in the IIC interface logic, and is not under software control.

24.5 General Considerations

1. After a reset, the IIC interface enters the unknown IIC bus state. This state is exited when either activity is seen on the bus or when the exit unknown IIC bus state bit, in the mode control register, is set to a 1. If the IIC interface is being used in a single master system as the master, then the exit unknown IIC bus state bit must be used to force the logic out of the unknown state.
2. Once a byte is written into either the master or slave buffer, a total of four OPB clock periods must occur before the data can be read. Flushing the master or slave buffer also requires four OPB clock periods to complete.
3. IICx_DIRECTCNTL [MSDA,MSC]are used to verify that IICx_DIRECTCNTL [SDAC,SCC] were written successfully,and that the IICSCL signals can be controlled.If IICx_DIRECTCNTL [MSDA, MSC] do not correspond to IICx__DIRECTCNTL [SDAC,SCC],respectively,toggle IICSCL repeatedly to regain control.
4. The master and slave buffers are 4×1 byte-wide FIFOs. Exercise care when using master and slave buffers. As an example, consider the case where one byte of data is written on the IIC bus. The data is first written into the master or slave buffer. After four OPB clock cycles the data is placed on the IIC bus. There is no way to verify data in the buffer without disturbing the IIC transaction. The act of verification requires removing the data. If the data is removed, invalid data is placed on the IIC bus.
5. Use care when monitoring the IICx_XCNTLSS[SDBD] or to IICx_STS[MDBS] to determine when data is present. These bits are set to 1 when the buffer contains data in any stage. Consider the case where the master buffer is empty prior to being loaded with a byte received over the IIC bus. The byte enters the fourth stage of the buffer and the IICx_STS[MDBS] is set to 1. Stages 1, 2, and 3 do not contain data. Therefore, the data is not available for four OPB clock cycles. Any attempt to prematurely read the data yields invalid data.

6. When responding to a slave needs service request, manage the data first. Read data out of the slave buffer, IICx_SDB, for slave reads or write data into the IICx_SDB for slave writes. Next clear the slave needs service request. For reads, clear IICx_XTCNTLSS[SRS]. For writes, clear IICx_XTCNTLSS[SWS]. Last, clear the active interrupt, IICx_STS[IRQA] =0.
7. There is no timeout function implemented in the IIC interface. If this type of error recovery function is needed, it must be implemented in software.
8. Avoid the situations listed in Section 7.2 of the *Phillips Semiconductors I^2C Specification,* dated 1995. For your convenience, the section is summarized as follows:

   If multiple masters can be simultaneously involved in a transfer to the same address, or device, then the design of the system must be done in such a way that arbitration between:

   - A repeated Start condition and a data bit does not occur.
   - A Stop condition and a data bit does not occur.
   - A repeated Start condition and a Stop condition does not occur.

An example of this situation occurs if one master writes 1 byte while another master writes 2 bytes to the same device.

26. External Bus Master Interface ~~(EBMI)~~

The On-Chip Peripheral Bus (OPB)-attached External Bus Master Interface (EBMI) controller transfers data between the OPB and the External Peripheral Bus (EXPB) under the direction of an external master device. The EBMI is a slave on the EXPB and a master on the OPB. This controller is necessary for any implementation of external master devices that must access system resources (SDRAM memory, on-chip SDRAM, PCI address space, etc.) on the OPB or Processor Local Bus (PLB).

For the PPC440GP, the EBMI must share the EXPB with an on-chip external bus controller (EBC) and must arbitrate for access to the external bus.

The EBMI has a device control register (DCR) interface to allow a processor to read and write internal EBMI registers specific to the EBMI. An external master may also read and write internal EBMI registers using the EXPB Special Cycle.

26.1 Feature List

The EBMI controller has the following features:

- EXPB slave device; OPB master device
- 32-bit data/ 36-bit address OPB master interface
  - Single word (32-bit) halfword, byte, or burst reads
  - Single word, halfword, byte, and burst writes
  - 32/16/8-bit OPB slave support
- Programmable 20 to 32-bit EXPB address size
- 32-bit EXPB data interface
  - Support 8, 16, or 32-bit external master
- Data packing on burst writes, up to 8 words
- Read prefetching support
  - No prefetching
  - 1 word
  - 8 word
  - 16 word
- Dual 32-byte buffers
- Support for OPB at 1, 2, 3, or 4 times the frequency of the external bus
- Directly supports one external master
- Class 2 sleep support
- Programmable by external master

26.2 Physical Implementation

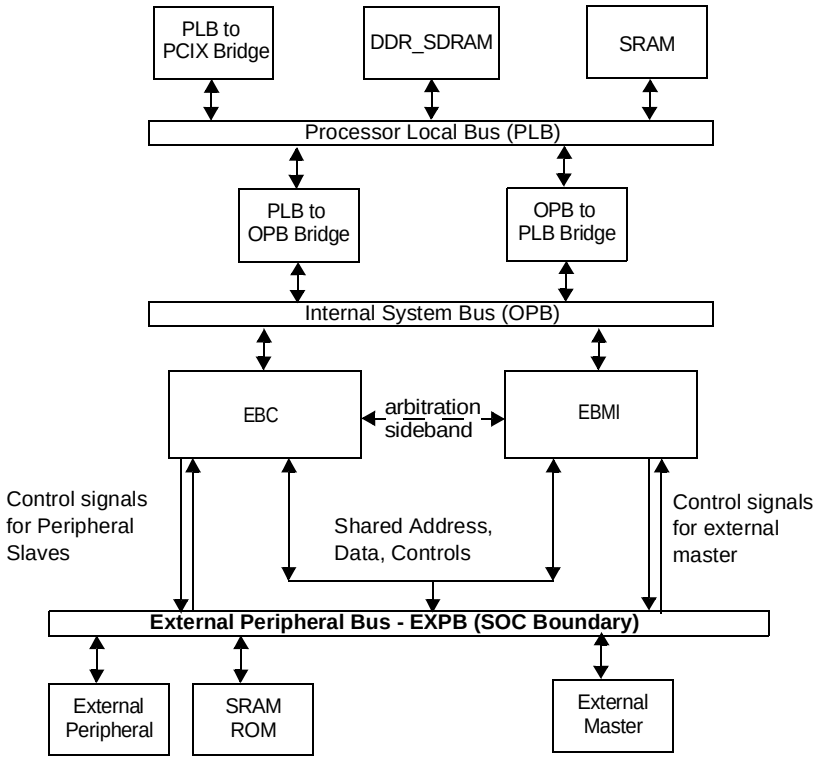


Figure 26-1. EXPB Implementation in a SOC

26.2.1 Signals

Figure 26-2 illustrates the signal I/O for the EBMI core.

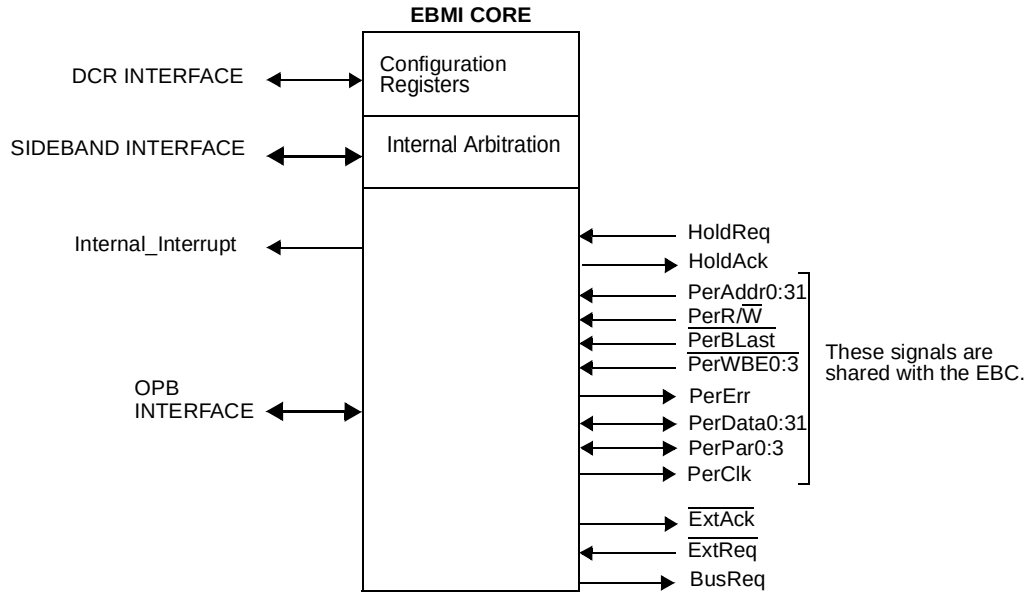


Figure 26-2. EBMI Core I/O Diagram

Table 26-1 provides a summary of all EXPB signals at the SOC I/Os used by the external master, followed by a brief description and page reference for detailed functional description.

Note: All EBMI I/Os are synchronous to the rising edge of PerClk.

Table 26-1. Summary of EXPB Signals for External Master

Signal Name	External Master State when HoldAck=0	Active State	I/O	Description	Page
PerAddr0:31	high-Z	—	O	External master address bus. PerAddr0 is the most significant bit.	26-38 34
PerData0:31	high-Z	—	I/O	External master data bus. PerData0 is the most significant bit.	26-48 34
PerPar0:3	high-Z	—	I/O	External master parity bus. The EBMI implements odd parity.	26-48 34
PerR/$\overline{W}$	high-Z	—	O	External master read not write	26-48 34
$\overline{\text{PerBLast}}$	high-Z	Low	O	External bus burst Last	26-48 34
$\overline{\text{PerWBE0:3}}$	high-Z	Low	O	External master byte enable input	26-48 35
HoldReq	operable	High	O	External master hold request	26-58 36
HoldAck	operable	High	I	External master hold acknowledge (grant)	26-68 36
$\overline{\text{ExtAck}}$	operable	Low	I	External bus cycle transfer acknowledge	26-68 36
$\overline{\text{ExtReq}}$	operable	Low	O	External bus cycle request valid	26-68 36

Table 26-1. Summary of EXPB Signals for External Master (continued)

PerErr	high-Z	High	I	External bus read data error and parity error indication	26-68 36
BusReq	operable	High	I	Bus preempt request from the EBCO	26-68 36
PerClk	EXT	—	I	Output from PPC440GP	26-68 37

26.2.1.1 PerAddr0:31 (External Master Address Bus)

The EXPB address is forwarded to the internal system bus. A master may choose to implement any number of address bits between 20 and 32. The upper address bits are appended with the appropriate value as programmed in EBM0_UAR to match the 36-bit bus width of the internal system bus.

During Special Cycles, the address bus is used to select internal EBMI facilities and is not forwarded onto the internal system bus.

The least significant bit of the address is bit 31. For a 32-bit address external master, the most significant bit of the address bus is bit 0. For a 28-bit address external master, the most significant bit of the address bus is bit 4. For a 20-bit address external master the most significant bit of the address bus is bit 12.

The address must be driven with $\overline{\text{ExtReq}}$ and held until $\overline{\text{ExtAck}}$ is observed active. The lower two bits of the address must point to the address of the first byte being transferred. During multi-beat transfers, PerAddr0:31 is ignored by the EBMI after the first $\overline{\text{ExtAck}}$ is driven active.

The external master must not allow its address to exceed its upper address range during multi-beat operations. For example, if a 32-bit external master with 20 bits of addressing begins a multi-beat operation at 0xFFFF4, it must terminate the burst after 3 beats.

26.2.1.2 PerData0:31 (External Master Data Bus)

The EXPB data bus is used to transfer data between the internal system bus or internal EBMI facilities and the external master. Bit 0 is the most significant bit, and bit 31 is the least significant bit. When subdivided into bytes, Bits 0:7 represent the most significant byte on the bus.

A master may interface to the EXPB as a byte, halfword, or word width device. A byte master uses PerData0:7 and PerPar0. A halfword master uses PerData0:15 and PerPar0:1. A word master uses PerData0:31 and PerPar0:3. The EBMI must be programmed via EBM0_CTL[EXSZ] to indicate the size of the external master. When interfaced to byte and half word external masters, the EBMI ignores the unused byte lanes.

The external master must drive write data coincident with $\overline{\text{ExtReq}}$ when it is asserted. Write data must remain driven until the EBMI asserts $\overline{\text{ExtAck}}$. For multi-beat transfers, the data must switch to the next beat the cycle after $\overline{\text{ExtAck}}$ has been asserted.

26.2.1.3 PerPar0:3 (External Master Data Bus Parity)

All EXPB data transfers (including Special Cycles) must include odd-byte parity if parity checking is enabled. Parity checking is enabled by programming EBM0_CTL[PDIS]=0.

26.2.1.4 PerR/$\overline{W}$ (External Master Read Not Write)

When $\overline{\text{ExtReq}}$ is asserted, the external master must drive this signal after the master has been granted the bus (as indicated by HoldAck=1) to indicate whether the current operation is a read (high) or a write (low).

26.2.1.5 $\overline{\text{PerBLast}}$ (External Bus Burst Last)

This signal must be asserted by the external master to indicate the last beat of a multi-beat transfer. It is asserted with the last beat of write data or when requesting the last beat of read data. If the transfer is a single beat operation, $\overline{\text{PerBLast}}$ must be asserted with the assertion of $\overline{\text{ExtReq}}$.

26.2.1.6 $\overline{\text{PerWBE0:3}}$ (External Master Byte Enable Input)

These signals must be asserted with $\overline{\text{ExtReq}}$ to indicate which bytes to write for a single-beat write operation. A 1-byte master uses only $\overline{\text{PerWBE0}}$. A 2-byte master uses only $\overline{\text{PerWBE0:1}}$. A 4-byte master uses all 4 bits.

At least one byte enable bit used by a master must be asserted for single-beat read operations in order to distinguish it from a Special Cycle read.

For multi beat operations ($\overline{\text{PerBLast}}$ asserted when $\overline{\text{ExtReq}}$ is asserted), the $\overline{\text{PerWBE0:3}}$ bits used by the external master must be zero.

To indicate a Special Cycle, all byte enable bits used by the master are de-asserted, and $\overline{\text{PerBLast}}$ is asserted when $\overline{\text{ExtReq}}$ is asserted.

~~Table 26-2~~*Table 26-2* below shows the allowable $\overline{\text{PerWBE0:3}}$, bus address, and master size combinations for write operations.

Table 26-2. Allowable PerWBE/Address/Master Size combinations

PerAddr30:31	$\overline{\text{PerWBE0:3}}$	Number of Bytes Xfered	Size of Master (bytes)	Location of Data on EXPB
00	(0:3) = 0000	4	4	PerData0:31
01	(0:3) = 1000	3	4	PerData8:31
00	(0:3) = 0001	3	4	PerData0:23
10	(0:3) = 1100	2	4	PerData16:31
01	(0:3) = 1001	2	4	PerData8:23
00	(0:3) = 0011	2	4	PerData0:15
11	(0:3) = 1110	1	4	PerData24:31
10	(0:3) = 1101	1	4	PerData16:23
01	(0:3) = 1011	1	4	PerData8:15
00	(0:3) = 0111	1	4	PerData0:7
aa[1]	(0:3) = 1111	4	4	PerData0:31
00	(0:1) = 00	2	2	PerData0:15
00	(0:1) = 01	1	2	PerData0:7
01	(0:1) = 10	1	2	PerData8:15
10	(0:1) = 00	2	2	PerData0:15

Table 26-2. Allowable PerWBE/Address/Master Size combinations (continued)

PerAddr30:31	PerWBE0:3	Number of Bytes Xfered	Size of Master (bytes)	Location of Data on EXPB
10	(0:1) = 01	1	2	PerData0:7
11	(0:1) = 10	1	2	PerData0:15
aa[1]	(0:1) = 11	2	2	PerData0:15
00	(0) = 0	1	1	PerData0:7
01	(0) = 0	1	1	PerData0:7
10	(0) = 0	1	1	PerData0:7
11	(0) = 0	1	1	PerData0:7
aa[1]	(0) = 1	1	1	PerData0:7

Note: 1. Lower 2-bits of the EBMI offset address for the Special Cycle address.

26.2.1.7 HoldReq (External Master Hold Request)

This signal is asserted by the external master to indicate that it requests ownership of the EXPB. This signal must remain active while a transfer is in progress and must not be deasserted when ExtReq is active. After completing at least one transfer, the external master should get off the bus as soon as possible upon seeing BusReq go active. This is done to avoid affecting system performance if the CPU is doing reads/writes to the EBC.

26.2.1.8 HoldAck (External Master Hold Acknowledge (Grant)

This signal is asserted by the EBMI to indicate that the external master has been granted ownership of the EXPB. This signal remains asserted until the cycle after the external master has de-asserted HoldReq. This signal is always driven by the EBMI. EBM0_CTL[EXSC] must be initialized with a DCR write to indicate the master size before EBMI asserts HoldAck.

26.2.1.9 ExtReq (External Bus Cycle Request Valid)

The external master asserts ExtReq to indicate the start of a new transfer. PerAddr0:31, PerR/W, PerBLast, and PerWBEn bits used by the master must be valid for this transfer when ExtReq is asserted. For write operations, PerData and PerPar bits used by the external master must also be valid. ExtReq must remain asserted until the data transfer has completed as indicated by ExtAck asserted by the EBMI. The external master must deassert ExtReq when the data transfer has completed.

26.2.1.10 ExtAck (External Bus Cycle Transfer Acknowledge)

This signal is asserted by the EBMI to indicate that read data is available or that write data has been accepted. This signal is always driven by the EBMI. For each data transfer, ExtAck is active for only one PerClk cycle.

26.2.1.11 PerErr (External Bus Parity Error Indication)

PerErr is used to signal write parity errors only when parity checking is enabled by setting EBM0_CTL[PPEN]=1. For a single-beat transfer, PerErr is asserted by the EBMI the same cycle ExtAck is asserted to indicate that a parity error has been detected on the data write operation.

For a multi-beat write transfer, PerErr is asserted by the EBMI two cycles after ExtAck is asserted for the data beat that had the error.

This signal is also asserted by the EBMI when ExtAck is asserted to indicate an error occurred during the read operation. If the current operation is a mult-beat transfer, PerErr continues to be asserted with ExtAck for all subsequent elements of the burst until the transfer is terminated by the de-assertion of ExtReq.

26.2.1.12 BusReq (Bus Preempt Request From The EBC)

This signal is asserted by the EBMI to indicate that the EBC needs ownership of the bus. When this is asserted, the external master should complete the current transaction if it is doing a single-beat operation and de-assert HoldReq. If the external master is doing a burst operation, the burst should be terminated as soon as possible. This signal is always driven by the EBMI.

Failure to relinquish bus ownership when BusReq is asserted can adversely affect system performance. Deadlock occurs if the master never relinquishes the bus.

26.2.1.13 PerClk (EXPB Clock)

PerClk is an output from the PPC440GP and is not used by the EBMI core. All EXPB transactions occur synchronous to PerClk.

26.2.2 Arbitration

To gain control of the EXPB, the external master requests the bus by placing an active level on the HoldReq input. The EBMI indicates that the bus is available by activating the HoldAck signal to the external master. The external master must keep HoldReq asserted for as long as it requires the EXPB. The external master may de-assert HoldReq the same cycle it de-asserts ExtReq.

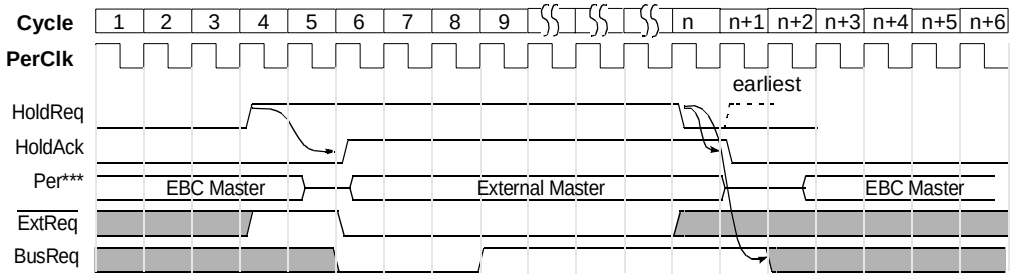
The PLB-to-OPB bridge will not accept a read or burst write command from the CPU or any other PLB master unless it can guarantee that the operation can be forwarded and accepted by the EBC. The EBC may not accept an operation off the OPB unless it also has ownership of the EXPB; consequently, the EBMI and the PLB-to-OPB bridge logic communicates with an internal arbitration protocol to ensure that the EXPB bus is free to be used by the EBC when a read or burst write command is accepted by the PLB-to-OPB bridge. The external master participates in this protocol by relinquishing the EXPB as soon as possible after BusReq is asserted by the EBC. The EBM0_FAIR register allows software to tune this protocol to meet system needs.

BusReq is asserted by the EBMI to indicate that the EBC requires control of the EXPB. The EBC cannot reclaim ownership of the EXPB until the external master releases it by negating HoldReq.

Note: The BusReq may be asserted regardless of bus ownership.

There is no requirement for when the external master must relinquish the bus after BusReq has been asserted; however, system performance may be compromised if other devices are not allowed fair access to the EXPB. The external master must complete the current transfer and relinquish the bus by deasserting HoldReq when BusReq is asserted.

The external master must High-Z the PerAddr0:31, PerR/$\overline{W}$, $\overline{PerBLast}$, $\overline{PerWBE0:3}$, PerData0:31, and PerPar0:3 bus signals when HoldAck is deasserted (the cycle after HoldReq is deasserted). ~~Figure 26-3~~*Figure 26-3* shows the EXPB bus arbitration protocol. The EBMI takes a minimum of two cycles to assert HoldAck after HoldReq is asserted; however, the assertion of HoldAck potentially takes much longer if the EBC has ownership of the EXPB.

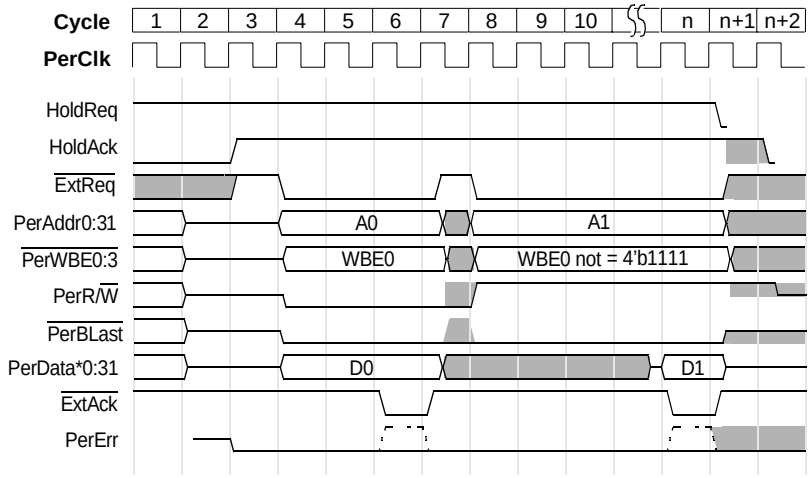


***External Bus Control Signals Enable/Ownership: PerAddr0:31, PerR/$\overline{W}$, PerBLast, PerWBE0:3, PerData0:31, PerPar0:3.

Figure 26-3. EXPB Bus Arbitration

26.2.3 Single Transfers

A single-beat write or read transfer is indicated by the external master by the assertion of $\overline{PerBLast}$ when $\overline{ExtReq}$ is asserted. The external master drives PerAddr0:31, $\overline{PerWBE0:3}$, and PerR/$\overline{W}$ with $\overline{ExtReq}$. Write data and data parity (if enabled) must also be driven with $\overline{ExtReq}$ for a write operation.



*PerAddr0:31 may be 20 to 32 bits wide. PerData0:31 includes PerPar0:3

Figure 26-4. Single Beat Write Followed by Read

The EBMI interface indicates that the write data has been accepted or that read data is available by asserting $\overline{ExtAck}$ for one cycle.

Single-beat read operations normally cause a 4-byte read operation to be forwarded to the OPB. If the single-beat prefetch mode is disabled in the EBMI, the EBMI reads only the bytes that are indicated in the PerWBE0:3 bits. At least one PerWBE0:3 bit must be asserted; otherwise, the operation is decoded as a Special Cycle read operation. When the read data is available, the EBMI drives the read data with ExtAck asserted. For more information, see the EBM0_CTL register.

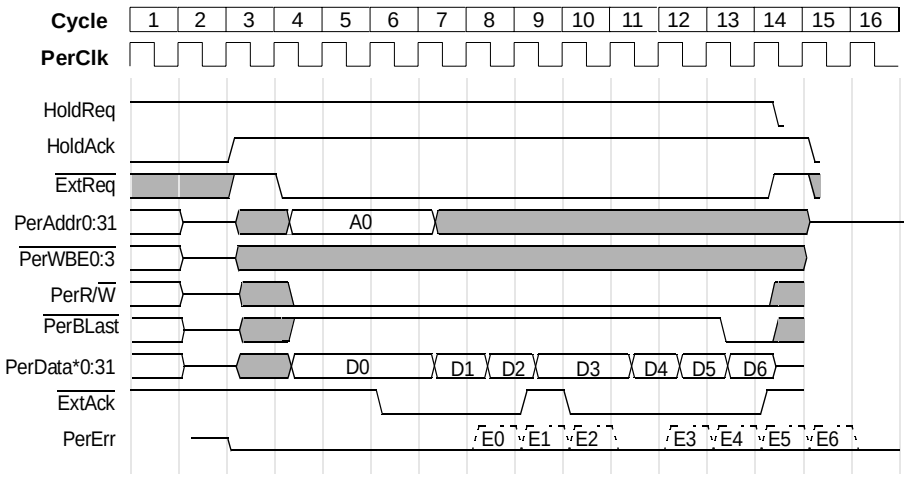
When using single beat reads to poll a memory location, the external master must force the read valid bits to reset to avoid reading stale data from the buffer (see ~~See "Read Valid Bits" on page 26-12~~*Read Valid Bits on page 842*).

26.2.4 Burst Transfers

A multi-beat (burst) write or read transfer is indicated by the external master deasserting PerBLast at the start of the transfer. Burst transfers must always occur on an address boundary equal to the size of the bus master. The PerWBE0:3 bits used by the external master must be zero during a burst transfer. The external master must drive/accept the next beat of data onto the bus the cycle after ExtAck is asserted by the EBMI. The EBMI may assert ExtAck in the cycle following the external master driving ExtReq. The external master must assert PerBLast to indicate that the last beat of data is available/wanted and that the burst transfer will terminate.

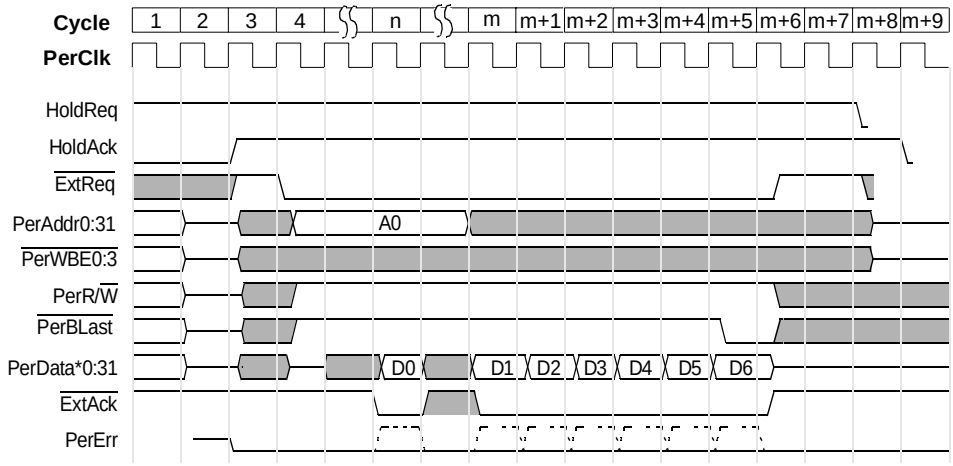
The EBMI may not terminate the burst data transfer; however, it will pace the data transfer by delaying the assertion of ExtAck.

When write data parity checking is enabled in the EBMI, the PerErr is asserted by the EBMI for a burst write operation two cycles after ExtAck is asserted if a parity error is detected on the burst write data transfer. The external master should monitor PerErr for two cycles after the end of a burst write transfer to check for a parity error indication if parity checking is enabled in the EBM0_CTL register.



*PerAddr0:31 may be 20 to 32 bits wide. PerData0:31 includes PerPar0:3.

Figure 26-5. Burst Write



*PerAddr0:31 may be 20 to 32 bits wide. PerData0:31 includes PerPar0:3

Figure 26-6. Burst Read

26.2.5 Special Cycle

A Special Cycle is executed by the external master to access internal EBMI configuration and status registers. An external master performs a Special Cycle transfer by asserting ExtReq while at the same time driving all implemented PerWBE0:3 bits to high when is asserted and driving PerBLast low. The lower 10-bits of PerAddr0:31 are used to select a specific register.

A Special Cycle is always a 32-bit data operation that is used to read or write the facilities implemented as DCR accessible registers in the EBMI. 16-bit masters must execute two single-beat Special Cycles in a row using the same address and PerWBE0:3) driven to 2'b11. 8-bit masters must execute four single-beat Special Cycles in a row using the same address and PerWBE0 driven to 1'b1. There must be at least one idle cycle between the individual special cycle transfers. EBM0_BESR[4] is set if a 16-bit or 8-bit external master inserts an OPB transfer request between the required 2 or 4 special cycle operations. The unexpected OPB transfer request is forwarded to the OPB as normal; however, future special cycle operations may complete incorrectly.

Only the lower 10-bits of the address are used for the Special Cycle to access the DCR addressable facilities within the EBMI. The upper address bits are ignored.

It data parity checking is enabled in the EBMI and a parity error is detected during a Special Cycle write operation, the target register is not written and the EBMI asserts PerErr coincident with ExtAck. The write operation will not execute to the DCR register. 8- and 16-bit external masters must complete all 32 bits of the Special Cycle write even if an error has been indicated by the EBMI when the error is not on the last data item of the 32-bit special cycle write.

~~Figure 26-8~~*Figure 26-8* below shows an external 32-bit master performing a special cycle write followed by a Special Cycle read. Figure 26-8 below shows special cycle operations being performed by an 8-bit external master.

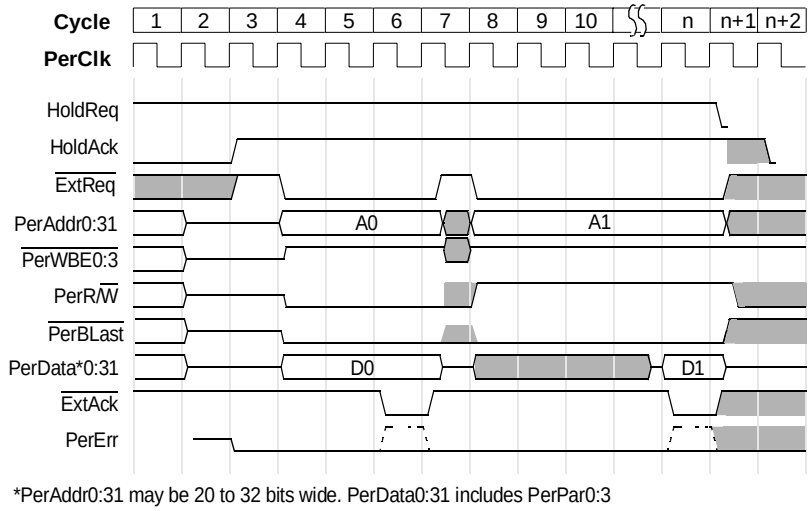


Figure 26-7. Special Cycle Write followed by Read

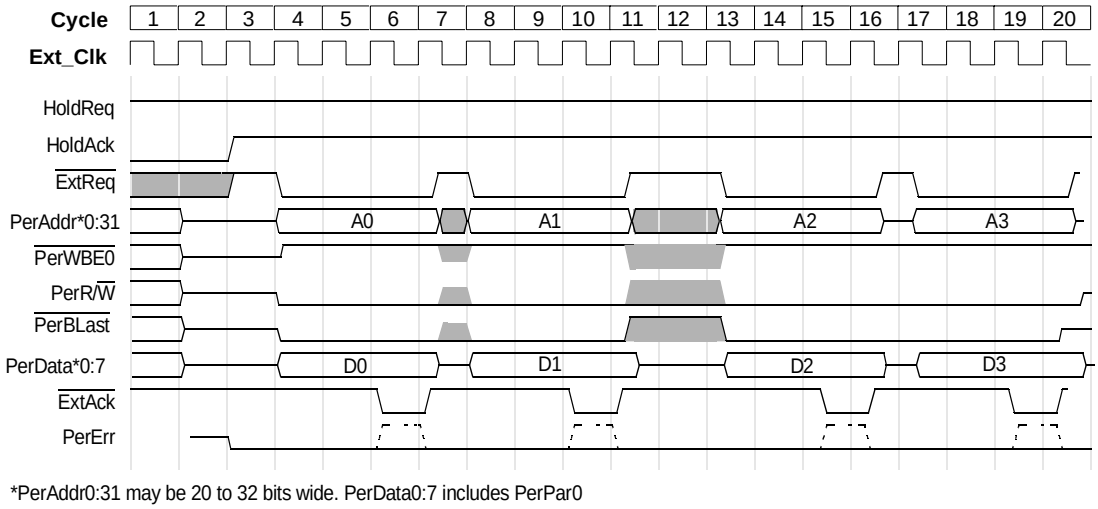


Figure 26-8. Special Cycle Write By 8-Bit Master

26.2.6 Error Reporting

The PerErr pin is driven by the EBMI to indicate a detected parity error condition on a data write from the external master. All external masters are required to generate parity for write data when parity checking is enabled in the EBMI. The EBMI is not informed if a parity error is detected on the read data; however, the system architecture may require that the external master indicate that a parity error was detected/driven by the EBMI. If the EBMI detects a parity error during a write operation and EBM0_BEMR[EPE]=0, an interrupt is generated.

The PerErr pin is driven by the EBMI to also indicate that an error occurred during the data read transaction. Possible sources for a read error would be an uncorrectable ECC error or a read from an address that does not exist. ExtAck is asserted with PerErr to allow the data transfer to complete. The external master should not use the read data if PerErr is asserted. If burst prefetching is enabled (EBM0_CTL[BPF]), PerErr may be asserted for good data if other unused data later in the buffer received an error.

An external interrupt pin is one method that an external master can use to indicate to the CPU that it detected a read parity error.

26.2.7 Interrupt Signal

The EBMI is connect to the UIC1_Irq[6]. An active "high" interrupt is asserted to the system logic when an unmasked error has been detected and is driven directly out of a latch. The signal remains asserted until the error condition is cleared by writing zeros to EBM0_BESR or by masking the error condition in EBM0_BEMR.

26.3 EBMI Buffer Management

The EBMI has two 32-byte data buffers. External master read operations read data from one buffer while the other is being filled from the OPB. External master write operations write data into one buffer while the other is written to its destination.

26.3.1 Read Operations

For read operations, the EBMI marks one buffer as the current read buffer. Read operations use data from the current buffer and do not issue another read if the current read buffer address matches the word address on the EXPB and the requested data bytes or the entire buffer is marked valid. When the current read buffer address does not match the EXPB word address or if the data in the buffer is not marked valid, the read operation forces a fetch from the address indicated by the external master. If the fetch is for a single-beat read, the EXPB word address becomes the current read buffer address and the appropriate data bytes are marked valid when the fetch is complete. If the fetch is for a burst read, the EXPB word address becomes the starting current read buffer address and the entire buffer is marked valid when the fetch is complete. When the entire buffer is marked valid, an EXPB word address matches the current read buffer address if it falls within the 32 or 16 byte range of the buffer.

26.3.1.1 OPB Read Prefetch

Burst read requests may cause a read prefetch at the start of the request to fill a read buffer. The size of the prefetch is programmed by software using EBM0_CTL[BPF]. The prefetch is programmed to 32 or 16 bytes and is used to optimize internal bus utilization. The EBMI fetches the next buffer of data when the EBMI has asserted ExtAck for the last beat of data in the current read buffer. The buffer burst prefetch may also be disabled with EBM0_CTL[BPF] so only one beat of data the size of the transfer on the EXPB is fetched at a time.

If the burst prefetch ahead mode bit BPFA is enabled in EBM0_CTL and prefetching is enabled, the EBMI continues to prefetch ahead for burst read operations as long as PerBLast is deasserted. The EBMI prefetches ahead into the other buffer for the next 32 or 16 bytes of data as soon as possible after initial prefetch has been initiated. When the data from the initial prefetch has been returned to the external master, the EBMI continues to assert ExtAck and return data from the next buffer if the data is ready and prefetched ahead into the other buffer that was emptied.

A single word (4-bytes) is read at the word address indicated by the external master to service single-beat, non-burst requests. This can be disabled in EBM0_CTL with mode bit SPFD to allow only exact reads using the EXPB write byte enables.

The best EXPB bandwidth can be achieved for burst read operations if BPFA is enabled with RDER (Ready Early Indication) in EBM0_CTL. If mode bit BPFA is disabled, the EBMI will not prefetch the next data buffer until all data from the initial prefetch has been returned to the external master.

26.3.2 Read Valid Bits

The EBMI maintains the read valid bits for each buffer. The read valid bits indicate which bytes at the current read buffer address are valid and if the entire buffer is valid. The prefetched read data in a buffer is used only if the appropriate read valid bits are set. The read valid bits are set when the fetch read data is available in the buffer. The read valid bits are reset under the following conditions:

- Any single-beat or burst write, or Special Cycle write operation resets all read valid bits in both buffers.
- The beginning of a new bus tenure (HoldAck just asserted) resets all read valid bits in both buffers unless EBM0_CTL[DRST]=1.
- Any single-beat read operation that requires a new buffer fetch resets all read valid bits in both buffers.
- A read buffer that received an internal failure indication from the OPB has its read valid bits cleared When the current EXPB transaction is ended. This will force a new OPB read operation if the read is repeated.

26.3.3 Read Early Mode

To reduce read latency for burst read operations, the EBMI may assert $\overline{\text{ExtAck}}$ to the external master as soon as the first beat of read data is available. To enable this function, software must set RDER=1 in EBM0_CTL to enable the read early mode. To avoid a buffer underrrun condition, the EBMI core de-asserts $\overline{\text{ExtAck}}$ and 'paces' the burst read operation if the burst read is suspended or delayed. Write Operations

~~0.0.1 Write Operations~~

Single beat write operations ping/pong between the two EBMI buffers. Data is not packed into the buffer and is delivered to its destination as soon as it is acknowledged on the EXPB.

Burst write operations are packed into the buffer until either 32 bytes of data have been received or the burst transfer has completed on the EXPB.

26.4 EBMI Registers

The EBMI registers are programmed using the **mfdcr** and **mtdcr** instructions. The EBMI registers are accessed using an indirect addressing method.

Table 26-3. EBMI DCR Addresses

Mnemonic	Name	DCR Number	Access	Page
EBM0_CFGADDR	EBMI Address Register	0x14	R/W	26-15 845
EBM0_CFGDATA	EBMI Data Register	0x15	R/W	26-15 845

The following table lists the indirectly accessed EBMI configuration and status registers. A read-modify-write sequence should be used to write registers with reserved fields. During this sequence, software must not alter the contents of any reserved fields.

Table 26-4. External Bus Configuration and Status Registers

Mnemonic	Name	Offset	Access	Page
EBM0_CTL	EBMI Control Register	0x00	R/W	26-15 845
EBM0_LCNT	EBMI OPB Latency Count Register	0x01	R/W	26-17 847
EBM0_BEAR	EBMI Bus Error Address Register	0x02	R/W	26-18 848
EBM0_BESR	EBMI Bus Error Status Register	0x03	R/W	26-18 848
EBM0_BEMR	EBMI Bus Error Mask Register	0x04	R/W	26-19 849
EBM0_UAR	EBMI OPB Upper Address Register	0x05	R/W	26-20 850
EBM0_UAM	EBMI OPB Upper Address Mask	0x06	R/W	26-20 850
EBM0_SLPMD	EBMI Sleep Mode Register	0x07	R/W	26-21 851
EBM0_FAIR	EBMI Fairness Control Register	0x08	R/W	26-24 853
EBM0_CID	EBMI Core ID Register	0x11	R	26-22 852

An external master may also use a Special Cycle read or write operation on the EXPB to access the EBMI registers. PerAddr27:31 is used to select the register.

If a Special Cycle operation occurs during the same cycle as DCR operation from the CPU, the DCR operation has priority and executes first.

Note: Indirect addressing is not used for Special Cycle operations, and all operations are targeted to the EBMI registers only.

Table 26-5 lists register values after reset.

Table 26-5. Register Contents After Reset

Register	Access	Reset Value	Comment	Page
Access Registers				
EBM0_CFGADDR	R/W	0x0000_0000		26-15 845
EBM0_CFGDATA	R/W	0x0000_0000		26-15 845

Table 26-5. Register Contents After Reset (continued)

Register	Access	Reset Value	Comment	Page
Configuration and Status Registers				
EBM0_CTL	R/W	0x8840_0000		26-15 845
EBM0_LCNT	R/W	0x0000_0000		26-17 847
EBM0_BEAR	R	0x0000_0000		26-18 848
EBM0_BESR	R/W	0x0000_0000		26-18 848
EBM0_BEMR	R/W	0x0000_0000		26-19 849
EBM0_UAR	R/W	0x0000_0000		26-20 850
EBM0_UAM	R/W	0x0000_0000		26-20 850
EBM0_SLPMD	R/W	0x07C0_0000		26-21 851
EBM0_FAIR	R/W	0xFFF0_0000		26-24 853
EBM0_CID	R	0x3250_1rrr	rrr is the IBM internal revision number of the core	26-22 852

26.4.1 Register Descriptions

26.4.1.1 EBMI Configuration Address Register (EBM0_CFGADDR)

~~Figure 26-9~~*Figure 26-9* shows EBM0_CFGADDR bit definitions. This register is written by software to indirectly address the EBMI registers.

Figure 0-1. EBMI Configuration Address Register (EBM0_CFGADDR)

0:26		
27:31	DCRA	DCR Address Offset

Figure 26-9. EBMI Configuration Address Register (EBM0_CFGADDR)

0:26		
27:31	DCRA	DCR Address Offset

26.4.1.2 EBMI DCR Data Register (EBM0_CFGDATA)

This register is a data port written by software after it writes EBM0_CFGDATA to read or write the other EBMI registers.

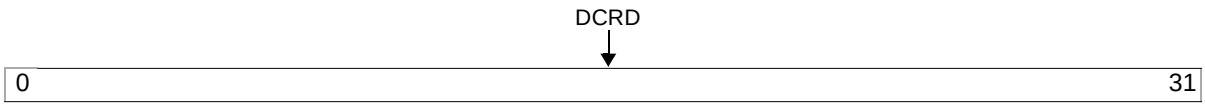


Figure 0-2. EBMI DCR Data Register (EBM0_CFGDATA)

0:31	DCRD	DCR Data Port

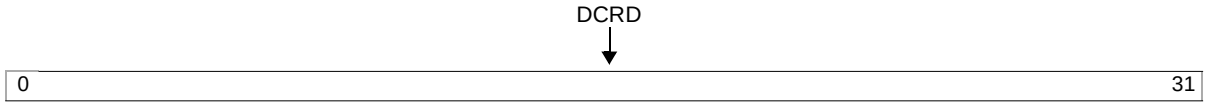


Figure 26-10. EBMI DCR Data Register (EBM0_CFGDATA)

0:31	DCRD	DCR Data Port

26.4.1.3 EBMI Control Register (EBM0_CTL)

~~Figure 26-11~~ *Figure 26-11* shows EBM0_CTL bit definitions.

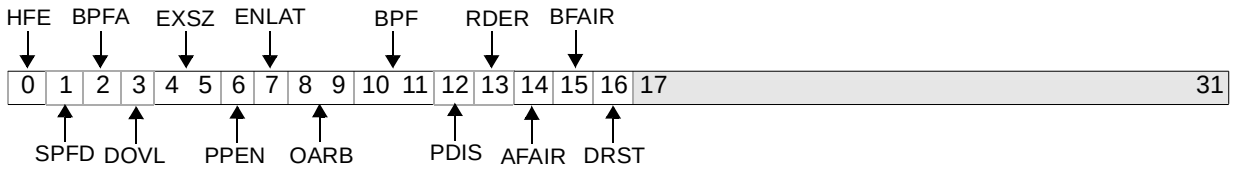


Figure 0-3. EBMI Control Register (EBM0_CTL)

0	HFE	Hold First Error 0 EBM0_BEAR and EBM0_BESR contain information from first error detected. 1 EBM0_BEAR and EBM0_BESR contain information from last error detected.	
1	SPFD	Single Beat Word Prefetch Disable 0 4-byte read on OPB for any single-beat EXPB read request. 1 1 to 4- byte read on OPB for any single-beat EXPB read request, according to EXPB PerWBE0:3 signals.	This bit must be set if the EBMI is to read only the bytes that are requested for a single beat read.

2	BPFA	Burst Prefetch Ahead 0 Data is read from OPB into the data buffer when the data is required. 1 Data is read from OPB into the data buffer before the data is required, while the 1st buffer is being emptied.	This bit should be set by software if the external master routinely does long or many sequential burst read operations longer than the burst prefetch size of the buffer. The EBMI will prefetch 1 buffer ahead to maintain EXPB bus bandwidth. This bit is ignored if the Burst Prefetch mode below is set to '10' to disable the prefetch.
3	DOVL	Disable HoldA Arbitration Overlap 0 HoldAck asserted in response to an active HoldReq asserted by external master immediately. There is a possible 2-cycle delay if a Special Cycle operation is in progress from the previous tenure. 1 HoldAck is not asserted in response to an active HoldReq from the external master until all pending OPB operations and Special Cycle operations from a previous bus tenure are complete.	This bit should be left as 0 and the EBM0_FAIR register used to tune arbitration.
4:5	EXSZ	External Master Data Bus Size 00 8-bit data bus, PerData0:7 01 16-bit data bus, PerData0:15 10 32-bit data bus, PerData0:31 11 No external master	Software must initialize this register to indicate the size of the external master data bus.
6	PPEN	Enable second 32-byte buffer 0 Second buffer disabled. Use only 1 buffer. 1 Second buffer enabled. Ping/Pong enabled.	
7	ENLAT	Enable OPB Latency Counter 0 OPB Latency counter is disabled 1 OPB Latency counter is enabled	This bit enables the EBM0_LCNT register function.
8:9	OARB	OPB Arbitration Control Must be0b01	
10:11	BPF	Burst Prefetch 00 8-beat (32-byte) read 01 4-beat (16-byte) read 10 1-beat (4/2/1-byte) read (no prefetch, read byte size of master) 11 - Reserved, Unused.	For most applications, this should remain at '00.' If BPF=2'b10, the request will be forwarded exactly as received if SPFD=1. These bits control the number of bytes prefetched on a burst read from the external master.
12	PDIS	EXPB Parity Checking Disabled 0 Write data parity is checked on EXPB 1 Write data parity is not checked on EXPB	Field enabled parity checking is only for external master transfers. This bit must be set by software if the external master does not support parity on PerData.

13	RDER	Enable Read Early Indication 0 Burst read data returned to external master when burst read from OPB is complete. 1 Burst read data returned to external master beginning when first beat from OPB is complete.	This mode bit is used to reduce latency for read burst operations.
14	AFAIR	Arbitration Fairness Counter Disable	This bit disables the AFCNT in the EBM0_FAIR register.
15	BFAIR	BReq Fairness Counter Disable	This bit disables the BFCNT in the EBM0_FAIR register.
16	DRST	Disable Reset Valid for New Tenure	This bit disables the clearing of the read buffer valid bits at the start of a new external master bus tenure. This bit should be set if the external master does long burst reads but often has to give up the EXPB before receiving all the data because BusReq is asserted.
17:31		Reserved	

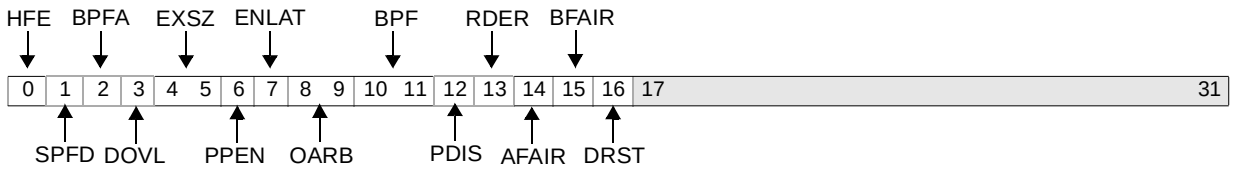


Figure 26-11. EBMI Control Register (EBM0_CTL)

0	HFE	Hold First Error 0 EBM0_BEAR and EBM0_BESR contain information from first error detected. 1 EBM0_BEAR and EBM0_BESR contain information from last error detected.	
1	SPFD	Single Beat Word Prefetch Disable 0 4-byte read on OPB for any single-beat EXPB read request. 1 1 to 4- byte read on OPB for any single-beat EXPB read request, according to EXPB PerWBE0:3 signals.	This bit must be set if the EBMI is to read only the bytes that are requested for a single beat read.
2	BPFA	Burst Prefetch Ahead 0 Data is read from OPB into the data buffer when the data is required. 1 Data is read from OPB into the data buffer before the data is required, while the 1st buffer is being emptied.	This bit should be set by software if the external master routinely does long or many sequential burst read operations longer than the burst prefetch size of the buffer. The EBMI will prefetch 1 buffer ahead to maintain EXPB bus bandwidth. This bit is ignored if the Burst Prefetch mode below is set to '10' to disable the prefetch.

3	DOVL	Disable HoldA Arbitration Overlap 0 HoldAck asserted in response to an active HoldReq asserted by external master immediately. There is a possible 2-cycle delay if a Special Cycle operation is in progress from the previous tenure. 1 HoldAck is not asserted in response to an active HoldReq from the external master until all pending OPB operations and Special Cycle operations from a previous bus tenure are complete.	This bit should be left as 0 and the EBM0_FAIR register used to tune arbitration.
4:5	EXSZ	External Master Data Bus Size 00 8-bit data bus, PerData0:7 01 16-bit data bus, PerData0:15 10 32-bit data bus, PerData0:31 11 No external master	Software must initialize this register to indicate the size of the external master data bus.
6	PPEN	Enable second 32-byte buffer 0 Second buffer disabled. Use only 1 buffer. 1 Second buffer enabled. Ping/Pong enabled.	
7	ENLAT	Enable OPB Latency Counter 0 OPB Latency counter is disabled 1 OPB Latency counter is enabled	This bit enables the EBM0_LCNT register function.
8:9	OARB	OPB Arbitration Control Must be0b01	
10:11	BPF	Burst Prefetch 00 8-beat (32-byte) read 01 4-beat (16-byte) read 10 1-beat (4/2/1-byte) read (no prefetch, read byte size of master) 11 - Reserved, Unused.	For most applications, this should remain at '00.' If BPF=2'b10, the request will be forwarded exactly as received if SPFD=1. These bits control the number of bytes prefetched on a burst read from the external master.
12	PDIS	EXPB Parity Checking Disabled 0 Write data parity is checked on EXPB 1 Write data parity is not checked on EXPB	Field enabled parity checking is only for external master transfers. This bit must be set by software if the external master does not support parity on PerData.
13	RDER	Enable Read Early Indication 0 Burst read data returned to external master when burst read from OPB is complete. 1 Burst read data returned to external master beginning when first beat from OPB is complete.	This mode bit is used to reduce latency for read burst operations.
14	AFAIR	Arbitration Fairness Counter Disable	This bit disables the AFCNT in the EBM0_FAIR register.
15	BFAIR	BReq Fairness Counter Disable	This bit disables the BFCNT in the EBM0_FAIR register.
16	DRST	Disable Reset Valid for New Tenure	This bit disables the clearing of the read buffer valid bits at the start of a new external master bus tenure. This bit should be set if the external master does long burst reads but often has to give up the EXPB before receiving all the data because BusReq is asserted.
17:31		Reserved	

26.4.1.4 EBMI OPB Latency Count Register (EBM0_LCNT)

The latency counter is the upper 6 bits of the initial load of a 10-bit OPB bus latency counter. The counter is a programmable timer that limits the EBMI latency on the OPB bus. The lowest four low-order bits of the latency count are hard-wired to zero, so the minimum latency count is 16.

When the latency counter is enabled in EBM0_CTL[ENLAT], the EBMI is transferring data on the OPB, and another OPB master requests ownership of the bus, the latency counter decrements with each data transfer. When the counter decrements to zero, the EBMI must relinquish control of the OPB as soon as possible.

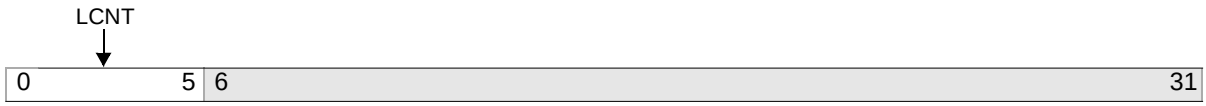


Figure 0-4. EBMI OPB Latency Count Register (EBM0_LCNT)

0:5	LCNT	Latency Counter This is the upper 6 bits of the initial load of a 10-bit OPB bus latency counter.
6:31		Reserved

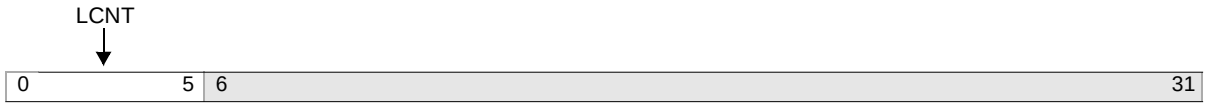


Figure 26-12. EBMI OPB Latency Count Register (EBM0_LCNT)

0:5	LCNT	Latency Counter This is the upper 6 bits of the initial load of a 10-bit OPB bus latency counter.
6:31		Reserved

26.4.1.5 EBMI Bus Error Address Register (EBM0_BEAR)

~~Figure 26-13~~*Figure 26-13* shows EBM0_BEAR bit definitions This register contains either the OPB or EXPB address of the access where a data bus error occurred. EBM0_BEAR is written when a data access error occurs and its contents are locked if the Hold First Error (HFE) bit in EBM0_CTL is set. If the HFE bit is not

set, then EBM0_BEAR continues to be updated with the address of any subsequent cycle in which an error occurs. If an OPB bus error and an EXPB error occur in the same cycle, this register is updated with the EXPB bus address.

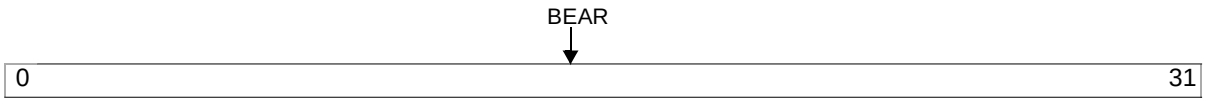


Figure 0-5. Peripheral Bus Error Address Register (EBM0_BEAR)

0:31	BEAR	Bus Error Address	The contents of this register are valid when any bit is set in the EBM0_BESR.

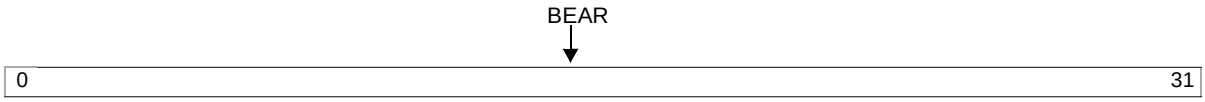


Figure 26-13. Peripheral Bus Error Address Register (EBM0_BEAR)

0:31	BEAR	Bus Error Address	The contents of this register are valid when any bit is set in the EBM0_BESR.

26.4.1.6 EBMI Bus Error Status Register (EBM0_BESR)

EBM0_BESR records the occurrence and type of errors for transactions attempted on behalf of the external master. It is possible to have multiple errors on a single transfer. If multiple errors occur on a transfer, multiple error bits may be set if the HFE bit in EBM0_CTL is not set. If the HFE bit is set, then only the first error detected is reported.

If a read operation is in progress when an error occurs, any data in the read buffer is returned to the external master with PerErr asserted when $\overline{\text{ExtAck}}$ is asserted until the transfer is terminated by the external master.

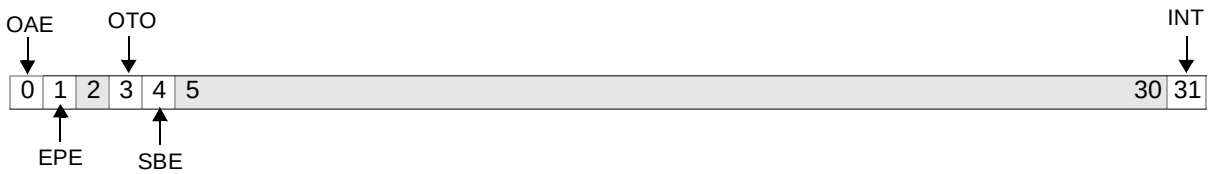


Figure 0-6. EBMI Bus Error Status Register (EBM0_BESR)

0	OAE	OPB Bus Ack Error 0 No ack error detected 1 Ack error detected	This bit is set when the EBMI detects an OPB error indication asserted during a data transfer on the OPB.
1	EPE	EXPB Bus Parity Error 0 No parity error detected 1 Parity error detected	This bit is set when the EBMI detects a parity error on incoming write data from the external master.0 The transfer with the parity error is not sent onto the OPB. Burst write operations must fill the buffer/terminate before they are sent onto the OPB. Consequently, none of the write data in the write buffer is forwarded to the OPB if a parity error is detected. All burst write data after the parity error is also discarded.
2		Reserved	
3	OTO	OPB Bus Timeout Error 0 No timeout detected 1 Timeout indication received when EBMI is reading or writing data on the OPB.	The current transfer on the OPB is terminated.
4	SBE	EXPB Special Cycle Error 0 No Special Cycle error detected. 1 External master has issued a non-Special Cycle operation when a Special Cycle operation was expected.	This error can only occur when using 16 or 8-bit external masters since the Special Cycle operation takes multiple transfers.
5:30			
31	INT	Interrupt Asserted 0 Interrupt line is not asserted. 1 Interrupt line is asserted to the chip internal interrupt controller.	This bit is a logical OR of all the other bits in the EBM0_BESR that are not masked in the EBM0_BEMR. This bit is not writable by software.

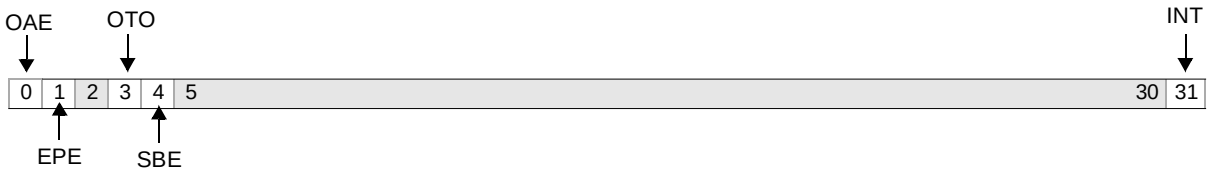


Figure 26-14. EBMI Bus Error Status Register (EBM0_BESR)

0	OAE	OPB Bus Ack Error 0 No ack error detected 1 Ack error detected	This bit is set when the EBMI detects an OPB error indication asserted during a data transfer on the OPB.

1	EPE	EXPB Bus Parity Error 0 No parity error detected 1 Parity error detected	This bit is set when the EBMI detects a parity error on incoming write data from the external master.0 The transfer with the parity error is not sent onto the OPB. Burst write operations must fill the buffer/terminate before they are sent onto the OPB. Consequently, none of the write data in the write buffer is forwarded to the OPB if a parity error is detected. All burst write data after the parity error is also discarded.
2		Reserved	
3	OTO	OPB Bus Timeout Error 0 No timeout detected 1 Timeout indication received when EBMI is reading or writing data on the OPB.	The current transfer on the OPB is terminated.
4	SBE	EXPB Special Cycle Error 0 No Special Cycle error detected. 1 External master has issued a non-Special Cycle operation when a Special Cycle operation was expected.	This error can only occur when using 16 or 8-bit external masters since the Special Cycle operation takes multiple transfers.
5:30			
31	INT	Interrupt Asserted 0 Interrupt line is not asserted. 1 Interrupt line is asserted to the chip internal interrupt controller.	This bit is a logical OR of all the other bits in the EBM0_BESR that are not masked in the EBM0_BEMR. This bit is not writable by software.

26.4.1.7 EBMI Bus Error Mask Register (EBM0_BEMR)

EBM0_BEMR has the same bit definitions as EBM0_BESR. This register is used to mask the reporting of each error condition as an interrupt. The appropriate bit in EBM0_BESR is still set, but an interrupt is not asserted for that error if the corresponding bit is a 1 in EBM0_BEMR.

Figure 0-7. EBMI Bus Error Mask Register (EBM0_BEMR)

0:5	EMSK	Bus Error Mask 0 Reporting of this error is not masked 1 Reporting of this error is masked.
6:31		Reserved

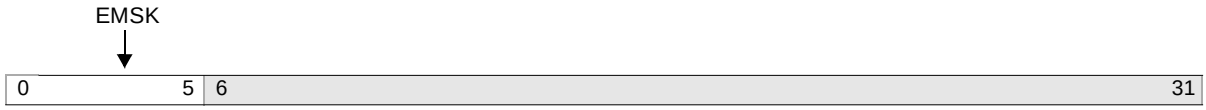


Figure 26-15. EBMI Bus Error Mask Register (EBM0_BEMR)

0:5	EMSK	Bus Error Mask 0 Reporting of this error is not masked 1 Reporting of this error is masked.
6:31		Reserved

26.4.1.8 EBMI OPB Upper Address Register (EBM0_UAR)

This register contains the upper bits[0:15] of the OPB address that are appended to the EXPB address to create a 36-bit OPB address. Bits [4:15] of the upper address correspond to PerAddr[0:11]. This register is used with EBM0_BEMR.

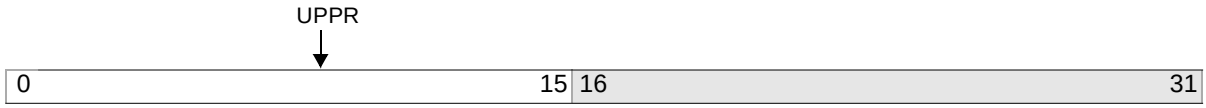


Figure 0-8. EBMI Upper Address Register (EBM0_UAR)

0:15	UPPR	OPB Upper Address	This register must be initialized by software.
16:31		Reserved	

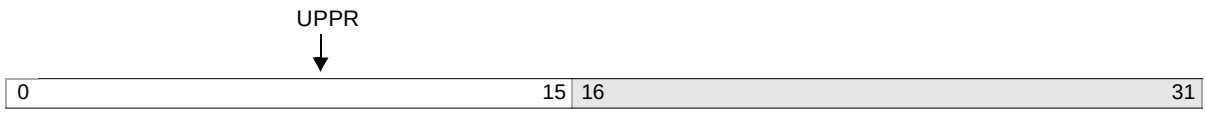


Figure 26-16. EBMI Upper Address Register (EBM0_UAR)

0:15	UPPR	OPB Upper Address	This register must be initialized by software.
16:31		Reserved	

26.4.1.9 EBMI OPB Upper Address Mask Register (EBM0_UAM)

~~Figure 26-17~~*Figure 26-17* shows EBM0_UAM bit definitions This register is used to indicate the bits from EBM0_UAR that should be used to pre-pend to the EXPB address to create the 36-bit OPB address. Note that the upper 4 bits of EBM0_UAM are always ignored, since the EXPB interface supports a maximum of 32 bits. See Table 26-6 for valid values.

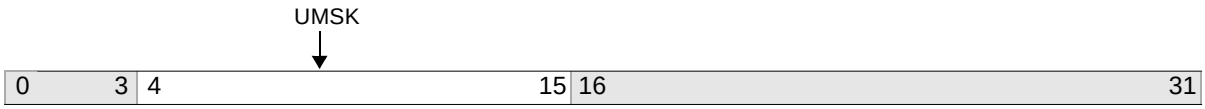


Figure 0-9. EBMI Upper Address Mask (EBM0_UAM)

0:3		Reserved	
4:15	UMSK	OPB Upper Address Mask	This register must be initialized by software iif the external master provides less than a complete 32-bit address.
16:31		Reserved	

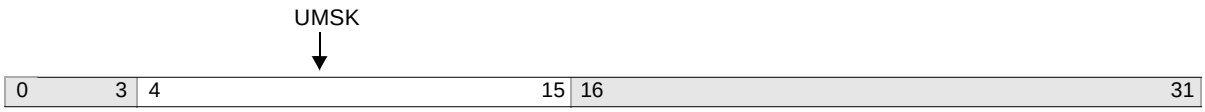


Figure 26-17. EBMI Upper Address Mask (EBM0_UAM)

0:3		Reserved	
4:15	UMSK	OPB Upper Address Mask	This register must be initialized by software iif the external master provides less than a complete 32-bit address.
16:31		Reserved	

Table 26-6. Valid EBM0_UAM Contents

UMSK(4:15)	OPB Address(0:35)	Number of EXPB Address Bits
000000000000	EBM0_UAR[0:3] || PerAddr0:31	32
100000000000	EBM0_UAR[0:4] || PerAddr1:31	31
110000000000	EBM0_UAR[0:5] || PerAddr2:31	30
111000000000	EBM0_UAR[0:6] || PerAddr3:31	29
111100000000	EBM0_UAR[0:7] || PerAddr4:31	28
111110000000	EBM0_UAR[0:8] || PerAddr5:31	27
111111000000	EBM0_UAR[0:9] || PerAddr6:31	26
111111100000	EBM0_UAR[0:10] || PerAddr7:31	25
111111110000	EBM0_UAR[0:11] || PerAddr8:31	24
111111111000	EBM0_UAR[0:12] || PerAddr9:31	23
111111111100	EBM0_UAR[0:13] || PerAddr10:31	22
111111111110	EBM0_UAR[0:14] || PerAddr11:31	21
111111111111	EBM0_UAR[0:15] || PerAddr12:31	20

26.4.1.10 _EBMI Sleep Mode Register (EBM0_SLPMD)

This register is used to enable the sleep mode and to program the number of OPB clock cycles to wait for the sleep request to be asserted after all of the requirements to go to sleep have been met.

EBMI sleep requirements are EBM0_SLPMD[SLEN]=1, HoldReq=0, pending MFDCR and MTDCR operations have completed, and all pending special cycle operations and OPB operations have completed. EBMI deasserts its sleep request immediately when the sleep requirements are no longer met.

Note: The minimum granularity of the idle timer is 32 clock cycles.

A value of 0 in the sleep counter (if bits 0:4 are left 0) waits 32 clocks from the time that the requirements to go to sleep have been met before asserting the sleep request signal.

A value of 1 in the sleep counter (after programming the register bits 0:4 to 1) waits 64 clocks from the time that the requirements to go to sleep have been met before making a sleep request to the clock and power management unit.

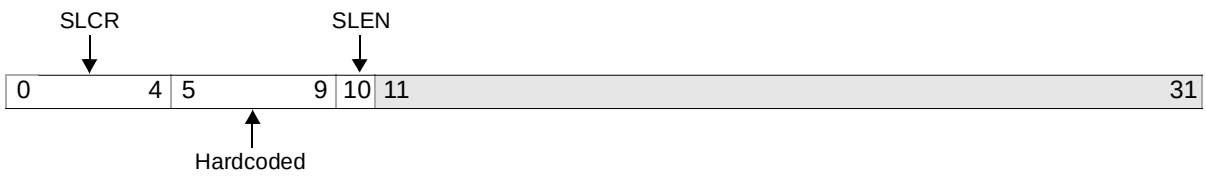


Figure 0-10. EBMI Sleep Mode Register (EBM0_SLPMD)

0:4	SLCR	Programmable timer values

5:9		Hardcoded to 1s	
10	SLEN	Sleep mode enable bit 0 Sleep request not asserted 1 Sleep request asserted when sleep requirements are met	Software must set this bit for the EBMI to assert its sleep request and be put to sleep.
11:31		Reserved	

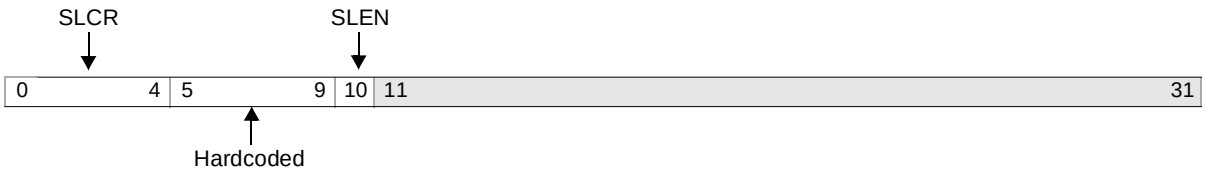


Figure 26-18. EBMI Sleep Mode Register (EBM0_SLPMD)

0:4	SLCR	Programmable timer values	
5:9		Hardcoded to 1s	
10	SLEN	Sleep mode enable bit 0 Sleep request not asserted 1 Sleep request asserted when sleep requirements are met	Software must set this bit for the EBMI to assert its sleep request and be put to sleep.
11:31		Reserved	

26.4.1.11 EBMI Core ID Register (EBM0_CID)

This register indicates the core ID. The core ID includes the technology, core number, and library revision number for this core. Writes to this register are ignored.

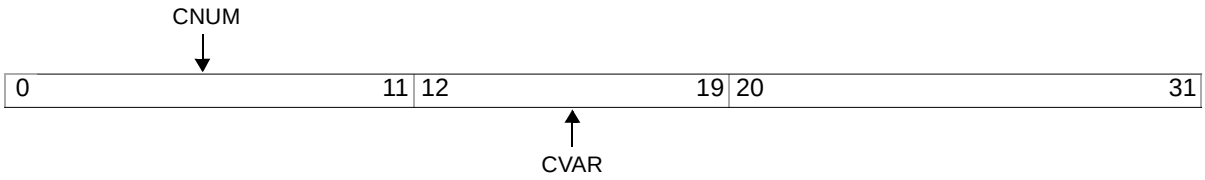


Figure 0-11. EBMI Core ID Register 0 (EBM0_CID)

0:11	CNUM	Core Number	'325'
12:19	CVAR	Core Variation	'01'

20:31	REV	Revision Number	This value is the IBM SCCS revision number tracked and modified by the core designer.

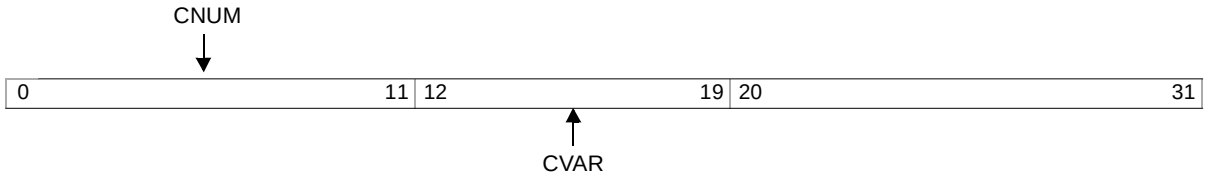


Figure 26-19. EBMI Core ID Register 0 (EBM0_CID)

0:11	CNUM	Core Number	'325'
12:19	CVAR	Core Variation	'01'
20:31	REV	Revision Number	This value is the IBM SCCS revision number tracked and modified by the core designer.

26.4.1.12 EBMI Fairness Control Register (EBM0_FAIR)

This register is used to tune the arbitration protocol between the EBMI and other internal SOC cores when trying to gain ownership of the EXPB.

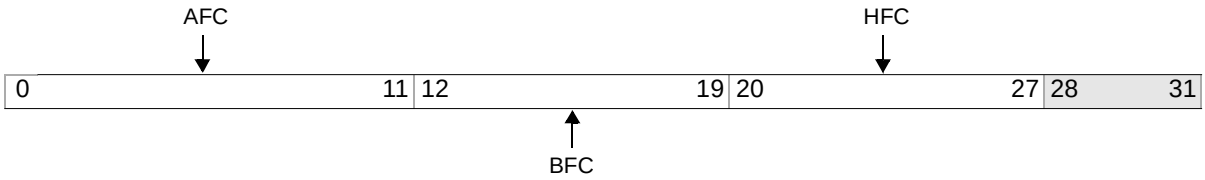


Figure 0-12. EBMI Fairness Control Register 0 (EBM0_FAIR)

0:11	AFC	Arbitration Fairness Count	These bits indicate the number of OPB clocks EBMI will wait before asserting a signal to stop PLB-to-OPB traffic when an external master request is pending. Under most conditions, these bits should be set to a high value to lower the priority of the external master. These bits are used to tune the priority of requests from the PLB to the OPB vs the priority of requests from the external master. If PLB-to-OPB traffic needs a relatively high priority, then a high value (for example x'F00') is written into this register by software.
12:19	BFC	BusReq Fairness Count	These bits indicate the number of PerClk cycles EBMI will wait before asserting BusReq to the external master. The BusReq fairness counter will begin decrementing when the EBCO has relinquished EXPB bus ownership. If BFC=0, BusReq may be asserted the same cycle as HoldAck. Under some conditions, the EBC may request ownership of the external bus and assert BusReq before the external master has asserted ExtReq. Setting the BFC bits to a non-zero value may allow the external master to make forward progress (if it is sensitive to this condition) by delaying BusReq until after the external master has asserted ExtReq.

20:27	HFC	HoldReq Fairness Count	These bits indicate the number of PerClk cycles EBMI will wait before forwarding HoldReq to the EBCO for the external master to gain bus ownership. This register is set to a high value to prioritize DMA transfers to and from the EBCO since DMA transfers are not affected by AFC.
28:31		Reserved	

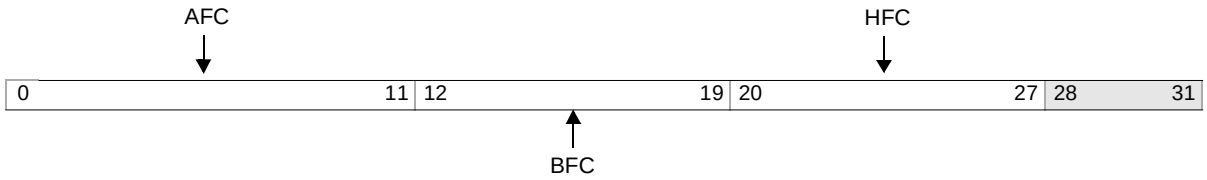


Figure 26-20. EBMI Fairness Control Register 0 (EBM0_FAIR)

0:11	AFC	Arbitration Fairness Count	These bits indicate the number of OPB clocks EBMI will wait before asserting a signal to stop PLB-to-OPB traffic when an external master request is pending. Under most conditions, these bits should be set to a high value to lower the priority of the external master. These bits are used to tune the priority of requests from the PLB to the OPB vs the priority of requests from the external master. If PLB-to-OPB traffic needs a relatively high priority, then a high value (for example x'F00') is written into this register by software.
12:19	BFC	BusReq Fairness Count	These bits indicate the number of PerClk cycles EBMI will wait before asserting BusReq to the external master. The BusReq fairness counter will begin decrementing when the EBCO has relinquished EXPB bus ownership. If BFC=0, BusReq may be asserted the same cycle as HoldAck. Under some conditions, the EBC may request ownership of the external bus and assert BusReq before the external master has asserted ExtReq. Setting the BFC bits to a non-zero value may allow the external master to make forward progress (if it is sensitive to this condition) by delaying BusReq until after the external master has asserted ExtReq.
20:27	HFC	HoldReq Fairness Count	These bits indicate the number of PerClk cycles EBMI will wait before forwarding HoldReq to the EBCO for the external master to gain bus ownership. This register is set to a high value to prioritize DMA transfers to and from the EBCO since DMA transfers are not affected by AFC.
28:31		Reserved	

27. External Bus Controller

The PPC440GP External Bus Controller attached to the OPB (EBC) provides direct attachment for most SRAM/Flash type memory and peripheral devices. The interface minimizes the amount of external glue logic needed to communicate with memory and peripheral devices. This reduces the embedded system device count, circuit board area, and cost.

To eliminate off-chip address decoding, the EBC provides eight programmable chip selects that enable system designers to locate memory and peripherals within the PPC440GP memory map. Chip select, data bus, and associated control signal timings are programmable for both single and burst transfers. For peripherals with variable timing requirements, the EBC supports device-paced transfers with optional bus-timeout. System design is further simplified through dynamic bus sizing, which supports seamlessly attaching 8-, 16-, and 32-bit wide memories and peripherals. Whenever a size mismatch exists between a read or write operation and the externally attached device, the EBC automatically packs or unpacks data as appropriate.

27.1 Interface Signals

Figure 27-1 illustrates the signal I/O between the EBC and the external peripheral bus.

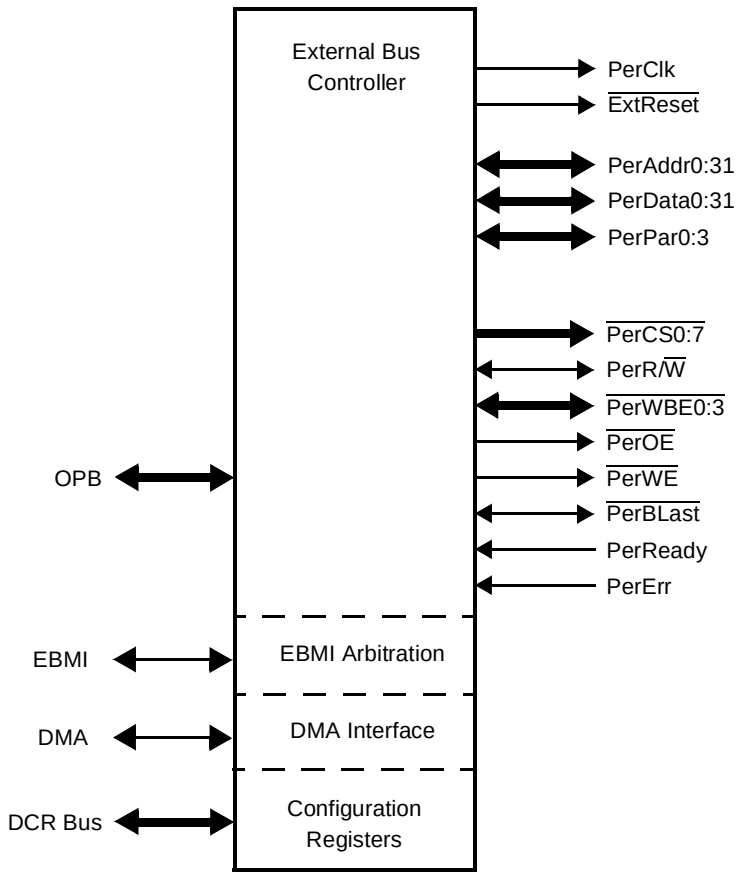


Figure 27-1. External Bus Controller Signals

The usage along with the state of these signals during and after a reset is as follows:

Table 27-1. EBC Signal Usage and State During/Following a Chip or System Reset

Signal	$\overline{\text{EBCOReset}}$ = 0	$\overline{\text{EBCOReset}}$ = 1	Usage
PerClk	See Note 1	Toggling	Peripheral bus clock. During an EBC transfer, all EBC signal transitions and data sampling occurs synchronous to PerClk.
$\overline{\text{ExtReset}}$	0	1	Peripheral reset for use by slaves and external bus masters.
PerAddr0:31	High impedance	Last Address	Peripheral address bus. PerAddr0 is the most significant bit.
PerData0:31	High impedance	00000000	Peripheral data bus. PerData0 is the most significant bit.
PerPar0:3	High impedance	0000	Peripheral parity bus. The EBC implements odd parity.
$\overline{\text{PerCS0:7}}$	High impedance	1	Chip selects. Note: Use CS0 to select the device when attaching boot device to the peripheral bus.
PerR/$\overline{\text{W}}$	High impedance	1	Read not write.
$\overline{\text{PerWBE0:3}}$	High impedance	1	Write byte enables or read/write byte enables.
$\overline{\text{PerOE}}$	High impedance	1	Output enable.
$\overline{\text{PerWE}}$	High impedance	1	Write enable. $\overline{\text{PerWE}}$ is low whenever any bit in $\overline{\text{PerWBE0:3}}$ is low and PerR/$\overline{\text{W}}$ = 0.
$\overline{\text{PerBLast}}$	High impedance	1	Burst Last. Active during non-burst operations and the last transfer of a burst access.
PerReady	Input	Input	An input to allow external peripherals to perform device-paced transfers.
PerErr	Input	Input	Peripheral data error input. Sampled during the data transfer only.

Note 1: PerClk is at frequency and stable from a minimum of 20 μS prior to the deassertion of ExtReset, assuming that SysClk is 66 MHz or lower.

Note 2: PerClk toggles at various frequencies during most of the system and chip reset process. Refer to Power-On Reset section of the Reset and Initialization section.

27.1.1 Interfacing to Byte, Halfword, and Word Devices

Figure 27-2 illustrates how to interface from byte, halfword, and word devices to the peripheral data bus. When devices are connected in this way, the EBC supports burst transfers and automatically converts read and write operations to the data width of the external device. As shown in Figure 27-2, halfword devices should not connect to PerAddr31. Similarly, a 32-bit device does not require either PerAddr30 or PerAddr31. Instead, the active byte lanes should be inferred from $\overline{\text{PerWBE0:3}}$, the read/write byte enables.

When a large number of byte and halfword devices are attached to the peripheral data bus, the capacitive loading on byte lane 0 (and byte lane 1, if many halfword devices are used) will be much larger than the loading on byte 3, possibly resulting in unacceptable timing performance on byte 0 or byte 1.

If a bank register is configured as word-wide, then byte-wide devices may be attached to the bus in any byte lane (and accessed using byte loads and stores). Similarly, if a bank register is configured as word-wide, then halfword-wide devices may be attached to the bus in the byte 0/byte 1 lane, or in the byte 2/byte 3 lane, and accessed using halfword loads and stores. External logic may be required to develop additional control signals if the data bus is utilized in this manner.

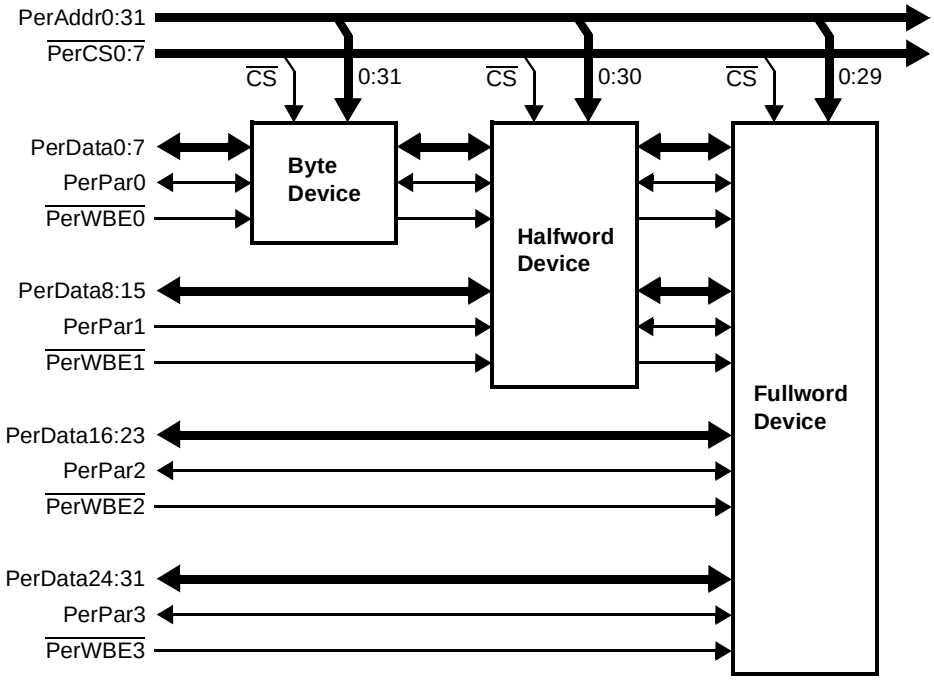


Figure 27-2. Attachment of Devices of Various Widths to the Peripheral Data Bus

27.1.2 Driver Enables

As shown in Table 27-2, the output enables for the peripheral address, data, and most of the EBC control signals are configurable. For systems that do not use an external master or where the external master does not directly control devices on the peripheral bus, setting EBC0_CFG[EMC] = 1 eliminates the need for pull-up resistors on PerCS0:7. If the EBCO shares the external bus with an external master it will high impedance all outputs when the external master wins the arbitration for the bus. This bit allows the EBCO to continue driving the Chip Selects only while the external master owns the external bus.

Pullups are also unnecessary on the remainder of the EBC control signals when EBC0_CFG[ATC] = 1, EBC0_CFG[DTC] = 1, and EBC0_CFG[CTC] = 1.

Pullups are required if EBC0_CFG[ATC] = 0, EBC0_CFG[DTC] = 0, and EBC0_CFG[CTC] = 0 because if the EBCO stops driving PerAddr for a sufficiently long time, PerAddr can charge/discharge to the threshold voltage of the receivers on the bus. This has the potential to cause the receivers to oscillate, creating excessive noise and/or low current draw.

Both chip and system resets set the output enable control bits EBC0_CFG[ATC] = 1, EBC0_CFG[DTC] = 1, and EBC0_CFG[CTC] = 1. In most applications, clearing ATC, DTC, or CTC is not recommended. If EBC0_CFG[CTC] = 0, EBC control signals can transition from the active state to high impedance without first being driven inactive. To prevent this, all peripheral banks must be configured with at least one hold cycle, EBC0_BnAP[TH] > 0.

Table 27-2. Effect of Driver Enable Programming on EBC Signal States

EBC Operation	$\overline{\text{ExtReset}}$ PerClk	$\overline{\text{PerCS0:7}}$	PerAddr0:31	PerR/$\overline{\text{W}}$ $\overline{\text{PerWBE0:3}}$ $\overline{\text{PerOE}}$ $\overline{\text{PerBLast}}$ $\overline{\text{PerWE}}$	PerData0:31 PerPar0:3
During Reset	High impedance	High impedance	High impedance	High impedance	High impedance
After Reset[1]	Driven	Driven	Driven	Driven	Driven
Idle	Driven	EBC0_CFG[CTC][2]	EBC0_CFG[ATC][2]	EBC0_CFG[CTC][2]	EBC0_CFG[DTC][2]
Read	Driven	EBC0_BnAP[OEO][3]	EBC0_BnAP[OEO][3]	EBC0_BnAP[OEO][3]	High impedance
Write	Driven	EBC0_BnAP[OEO][3]	EBC0_BnAP[OEO][3]	EBC0_BnAP[OEO][3]	EBC0_BnAP[OEO, $\overline{\text{OEN}}$][3]
External Master Ownership	Driven	EBC0_CFG[EMC]	High impedance	High impedance	High impedance
DMA Peripheral Transfer	Driven	Driven	High impedance[4]	Driven	Driven by DMA Controller

Note 1: SysReset active.

Note 2: If the EBC0_CFG bit is set, the signal is driven to the appropriate state during the indicated EBC operation. Otherwise, the I/O is high impedance.

Note 3: The OEO has an effect on driver enables as follows:

- For PerData0:31 and PerPar0:3,

  If EBC0_CFG[OEO] = 1, data is driven coincident with address for writes.

  If EBC0_CFG[OEO] = 0, data is driven EBC0_BnAP[OEN] cycles after $\overline{\text{PerCSn}}$ is asserted.

  - For PerCS0:7, PerAddr0:31, PerR/$\overline{\text{W}}$, $\overline{\text{PerWBE0:3}}$, $\overline{\text{PerOE}}$, $\overline{\text{PerBLast}}$, and $\overline{\text{PerWE}}$,

  If EBC0_CFG[OEO] = 1, they are driven beginning late in the cycle prior to the address.

  If EBC0_CFG[OEO] = 0, they are driven coincident with the address.

27.1.2.1 Effect of ATC and OEO On Address Bus Driver

If ATC = 1 and OEO = 0,1, then the external bus is always driven when the EBCO is idle, except when an external master owns the bus (see ~~Figure 27-3~~*Figure 27-3*).

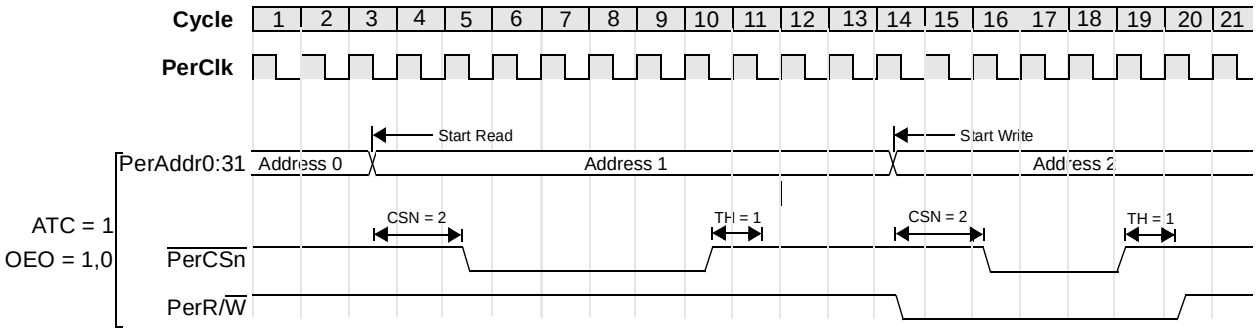


Figure 27-3. ATC = 1, OEO = 0,1

If ATC = 0 and OEO = 0, then the external bus address driver enables goes active when the cycle starts (off clock edge of first cycle) on the external bus (see ~~Figure 27-4~~*Figure 27-4*).

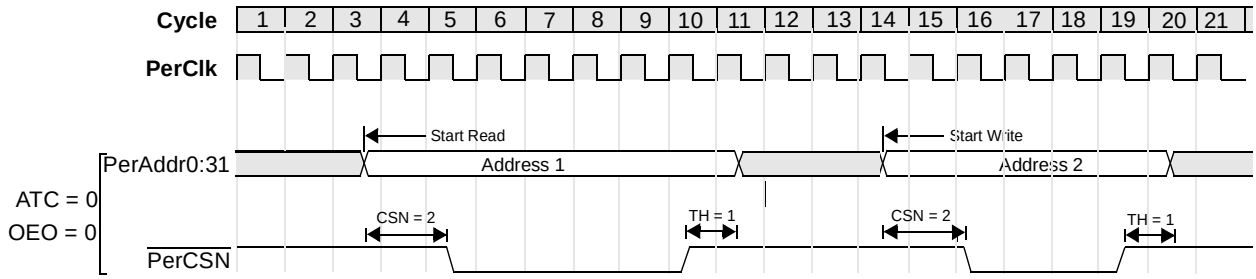


Figure 27-4. ATC = 0, OEO = 0

If ATC = 0 and OEO = 1, the external bus address driver enable goes active prior to the first clock of the cycle on the external bus (see ~~Figure 27-5~~*Figure 27-5*).

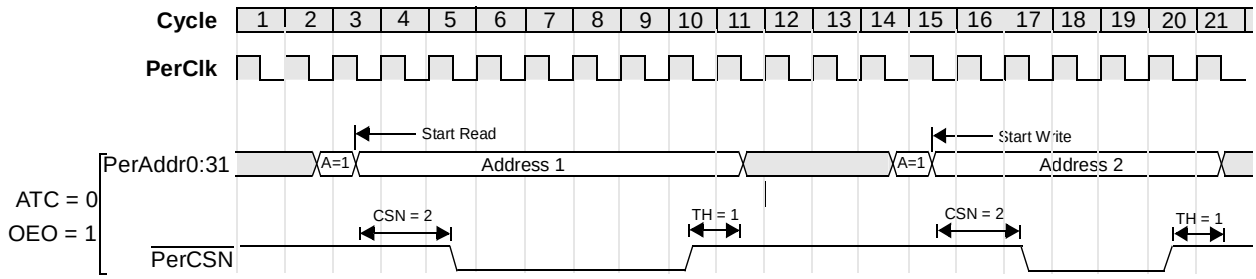


Figure 27-5. ATC = 0, OEO = 1

27.1.2.2 CTC, OEO, and EMC Effect on Control Signal Drivers

The control signals include $\overline{\text{PerCSN}}$, $\overline{\text{PerOE}}$, $\overline{\text{PerBlast}}$, PerR/$\overline{\text{W}}$, and $\overline{\text{PerWBE}}$.

If CTC = 1, then the control signals are always driven when the EBCO is idle except when the external master owns the bus. If the external master owns the bus, all control signals except $\overline{\text{PerCSN}}$ are high impedance. The $\overline{\text{PerCSN}}$ driver depends on the EMC value.

If EMC = 1, the $\overline{\text{PerCSN}}$ driver continues to be driven; otherwise, the $\overline{\text{PerCSN}}$ is high impedance. The OEO has no effect when CTC = 1 (see ~~Figure 27-6~~*Figure 27-6*).

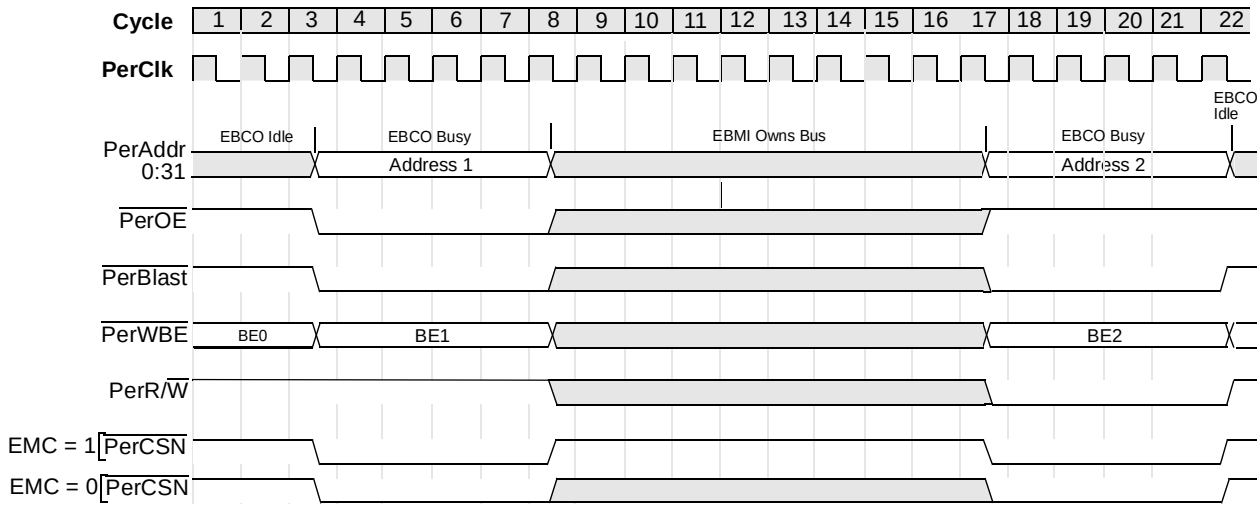


Figure 27-6. CTC = 1

If CTC = 0 and OEO = 0, then the control signal driver goes active when the cycle starts on the external bus (off clock edge of first cycle on the external bus) (see ~~Figure 27-7~~*Figure 27-7*).

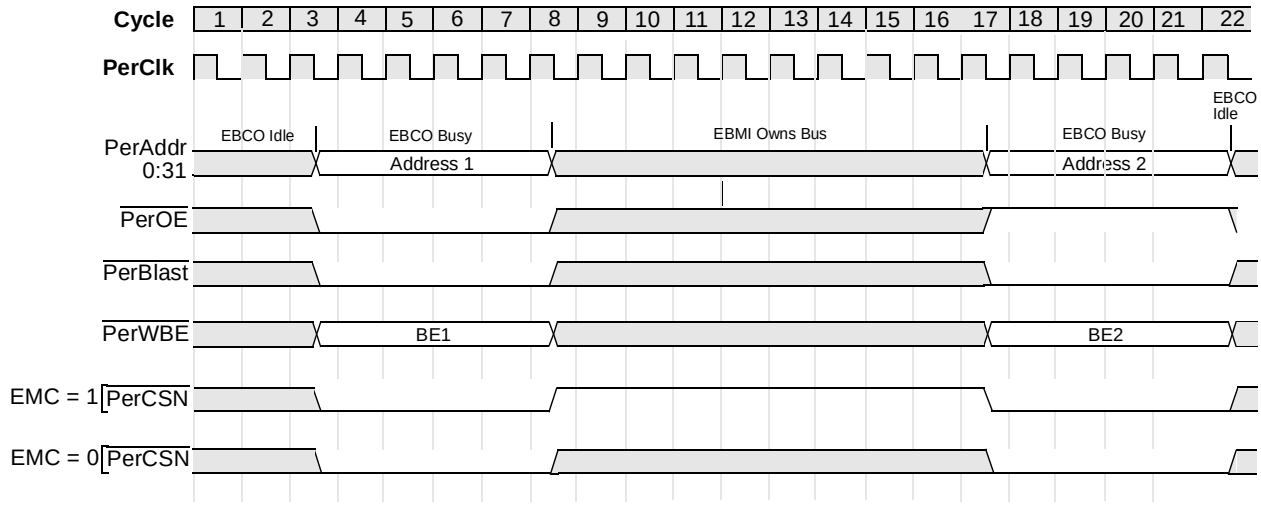


Figure 27-7. CTC = 0, OEO = 0

If CTC = 0 and OEO = 1, then the control signal drive enable goes active prior to the first clock of the cycle on the external bus (see ~~Figure 27-8~~*Figure 27-8*).

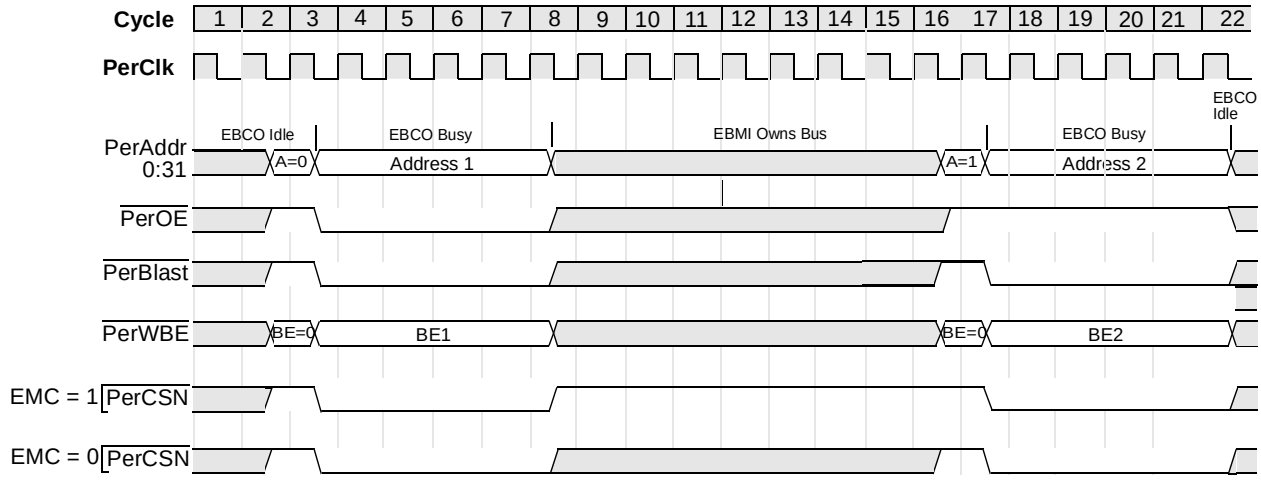


Figure 27-8. CTC = 0, OEO = 1

27.1.2.3 OEN, OEO, and DTC Effect on Bus Enable

If DTC = 1, then PerData and PerPar are always driven when the EBCO is idle except when an external master owns the bus. OEO has no effect when DTC = 1 (see ~~Figure 27-9~~*Figure 27-9*).

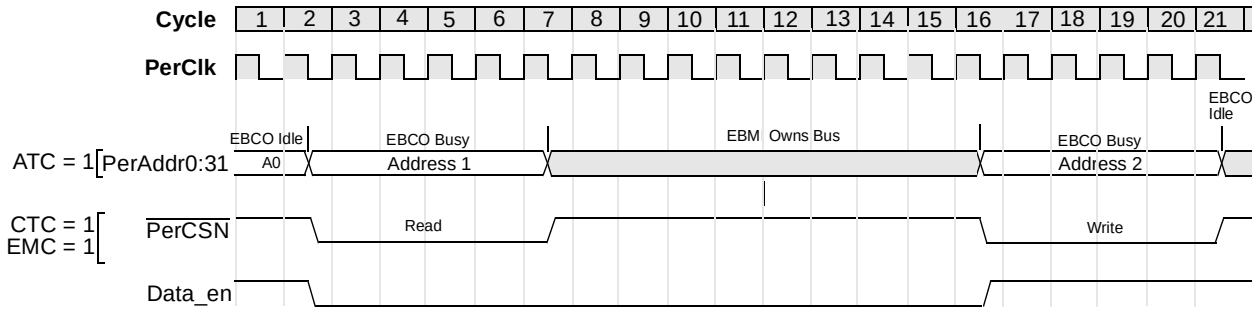


Figure 27-9. DTC = 1

If DTC = 0 and OEO = 1, then the external data bus driver enable goes active at the first clock of the cycle on the external bus (see ~~Figure 27-10~~*Figure 27-10*).

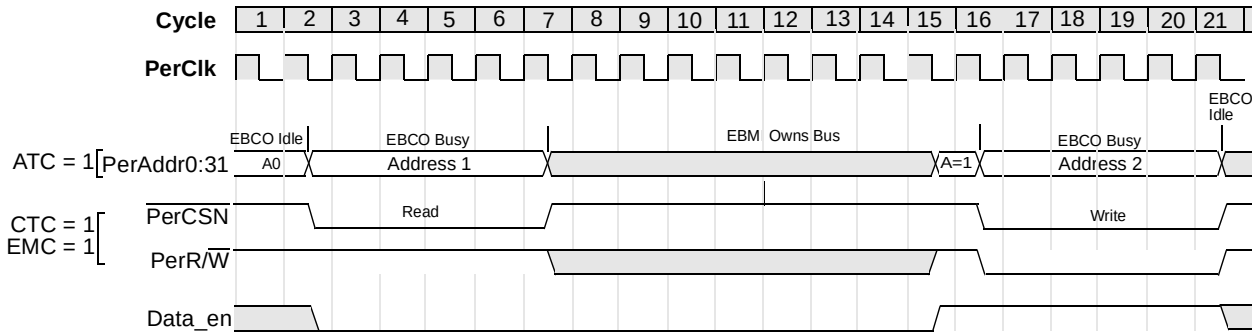


Figure 27-10. DTC = 0, OEO = 1

If DTC = 0 and OEO = 0, then the external data bus driver enable is controlled on a bank-by-bank basis by the EBC0_BnAP[OEN] parameter (see ~~Figure 27-11~~*Figure 27-11*).

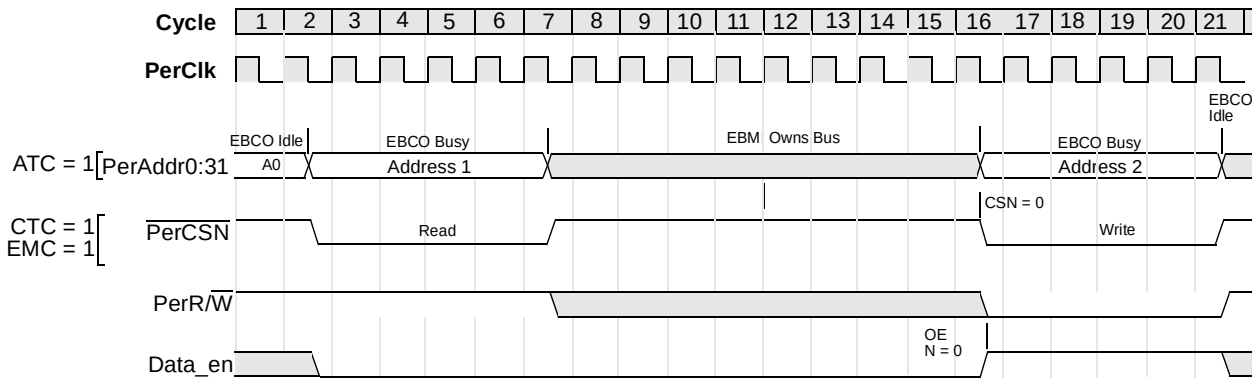


Figure 27-11. DTC = 0, OEO = 0

27.2 Non-Burst Peripheral Bus Transactions

The timing of the PerCSn, PerOE, and PerWBE0:3 signals is programmable via the Peripheral Bank Access Parameter (EBC0_BnAP) registers. For non-burst transfers, the access parameter registers control the peripheral bus timing as follows:

- PerCSn becomes active 0-3 PerClk cycles (EBC0_BnAP[CSN]) after the address is driven.
- PerOE is driven low 0-3 PerClk cycles (EBC0_BnAP[OEN]) after PerCSn is active.
- PerBLast is active throughout the entire transfer and is driven high during the programmed hold time (EBC0_BnAP[TH]).
- PerWBE0:3 can be either write byte enables or read and write enables.

  If EBC0_BnAP[BEM] = 0, PerWBE0:3 are write byte enables and:

  - PerWBE0:3 goes active 0-3 (EBC0_BnAP[WBN]) PerClk cycles after PerCSn becomes active.
  - PerWBE0:3 becomes inactive 0-3 (EBC0_BnAP[WBF]) PerClk cycles before PerCSn becomes inactive.

  If EBC0_BnAP[BEM] = 1, PerWBE0:3 are read/write byte enables and have timing identical to the peripheral address bus. In this case the EBC0_BnAP[WBN] and EBC0_BnAP[WBF] parameters are ignored.
- 1–256 PerClk cycles (EBC0_BnAP[TWT] + 1) after the address became valid:
  - If EBC0_CFG[CTC] = 1 or EBC0_BnAP[TH] > 0, PerCSn is driven high.
  - If EBC0_CFG[CTC] = 0 and EBC0_BnAP[TH] = 0, PerCSn transitions directly from logic 0 to the high-impedance state.
- The fields EBC0_BnAP[TWT, CSN, OEN, WBN, WBF] in are not independent. For non-burst configured banks (EBC0_BnAP[BME] = 0) that are also non-device-paced (EBC0_BnAP[RE] = 0), it is required that TWT ≥ CSN + MAX(EBC0_BnAP[OEN,WBN]) + EBC0_BnAP[WBF].
- The hold time, EBC0_BnAP[TH], is programmable from 0 to 7 PerClk cycles. During the hold time, the peripheral address bus remains driven with the last address, and all control signals are actively driven high. If the operation was a write, the peripheral data bus continues driving the last data value.
- There is no guarantee of dead cycles between transfers on the peripheral interface for one OPB transfer. If the OPB request size is greater than the bank size or the OPB request size is the same or less than the bank size but with the OPB seqaddr signal asserted, if the EBCO does packing data for read and unpacking data for write, and if the number of hold cycles is programmed to 0 (EBC0_BnAP[TH] = 0 and EBC0_BnAP[CSN] = 0), then:
  - PerCSn may not go inactive between the back-to-back transfers.
  - If EBC0_BnAP[OEN] = 0, PerOE may not become inactive between the back-to-back transfers.
  - If EBC0_BnAP[WBN] = 0 and EBC0_BnAP[WBF] = 0, PerWBE0:3 may not go inactive between the back-to-back transfers.
- There will always be a dead cycle between transfers on the peripheral interface that are initiated for separate OPB transfer.

27.2.1 Single Read Transfer

Figure 27-12 shows the peripheral interface timing for a single read transfer from a non-burst enabled (EBC0_BnAP[BME] = 0) bank. The transaction begins with the address being driven. Since this is a single transfer, PerBLast is also driven active along with the address. If byte enable mode is enabled for the bank (EBC0_BnAP[BEM] = 1), the byte enables are also output concurrently on PerWBE0:3. PerCSn then becomes active EBC0_BnAP[CSN] cycles after the address, while PerOE goes low EBC0_BnAP[OEN] cycles after PerCSn. The EBC then waits until EBC0_BnAP[TWT] + 1 cycles have elapsed since the start of the transaction and then reads the data bus and the peripheral error input, PerErr. If parity checking is enabled (EBC0_BnAP[PAR] = 1), the parity is also read at this time. The EBC then drives PerCSn, PerOE, and PerBLast high and waits EBC0_BnAP[TH] cycles.

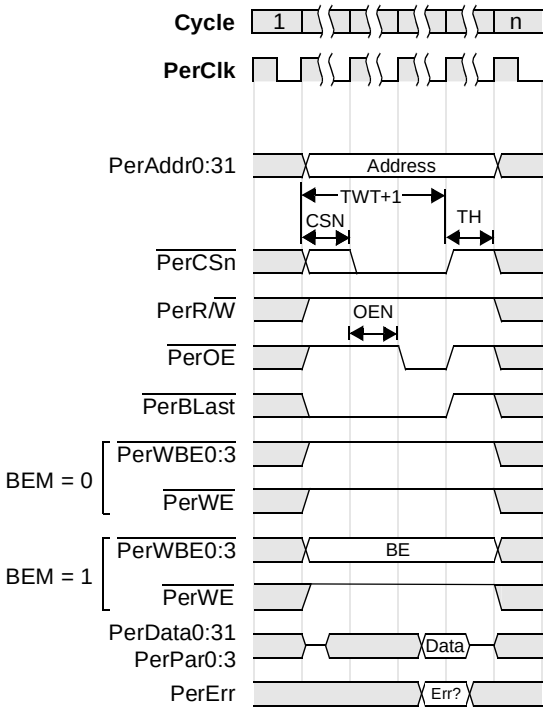


Figure 27-12. Single Read Transfer

27.2.2 Single Write Transfer

Figure 27-13 shows the peripheral interface timing for a single write transfer to a non-burst enabled (EBC0_BnAP[BME] = 0) bank. The transaction begins with the address being driven. Since this is a single transfer, PerBLast is also driven active along with the address. PerCSn then becomes active EBC0_BnAP[CSN] cycles after the address. At this point the signalling sequence depends on whether or not byte enable mode is enabled for the bank.

- If EBC0_BnAP[BEM] = 0, byte enable mode is disabled and the PerWBE0:3 are write byte enables. The appropriate write byte enables go low EBC0_BnAP[WBN] cycles after PerCSn. The EBC then waits until (EBC0_BnAP[TWT] - EBC0_BnAP[WBF] + 1) cycles have elapsed since the start of the transaction, then drives all the PerWBE0:3 inactive.
- If EBC0_BnAP[BEM] = 1, the PerWBE0:3 lines are byte enables and have the same timing as the peripheral address bus.

OEO also has an effect on PerData and PerPar.

- IF EBC0_CFG[OEO] = 1, PerData and PerPar become actively driven the same time as the address.
- IF EBC0_CFG[OEO] = 0, PerData and PerPar become actively driven EBC0_BnAP[OEN] after PerCSn falls.

After EBC0_BnAP[TWT+1] cycles elapse from the start of transfer, PerCSn and PerBLast are driven high. The EBC then waits EBC0_BnAP[TH] cycles before allowing any pending transfers to occur.

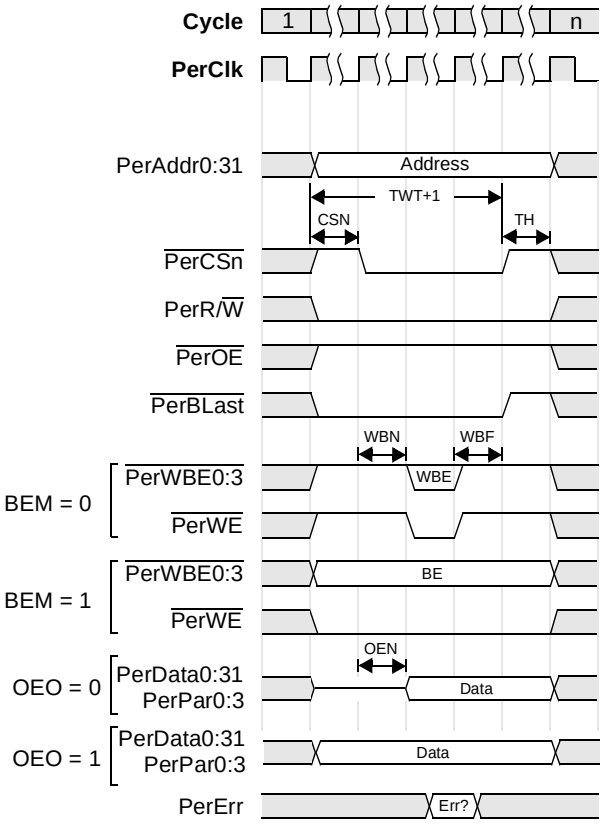


Figure 27-13. Single Write Transfer, EBC0_CFG[OEO] = 0/1

27.3 Burst Transactions

Bursting is controlled on a per-bank basis by the burst mode enable bit in the EBC0_BnAP registers. When enabled (EBC0_BnAP[BME] = 1) this mode activates bursting for all OPB sequential transfers to the EBC, and all packing and unpacking operations.

OPB sequential transfers to the OPB occur when the CPU performs cache line transfers to the EBC or when the CPU or any other PLB master performs fixed length burst transfers to the EBC.

Under many conditions, the EBC will execute extra read transfers on the peripheral interface for a sequential OPB burst operation. For example, if the transfer size on the OPB is less than or equal to the bank size, the EBC will do extra read(s) on the peripheral interface before the OPB termination can be detected. If extra read transfers are not allowed for a specific peripheral device (a FIFO for example), the EBC must be programmed to support fixed length bursts in EBC0_BnAP[BCE]. If the fixed length burst count is complete and the OPB operation has not terminated, the EBC will execute another fixed length burst read. The fixed length burst will terminate early if an OPB termination is detected.

Note: Access to FIFO devices must start and end on a word boundary.

When bursting is enabled:

- $\overline{\text{PerCSn}}$ becomes active 0-3 (EBC0_BnAP[CSN]) PerClk cycles after the address becomes valid.
- $\overline{\text{PerCSn}}$ is no longer actively driven:
  - 1-32 (EBC0_BnAP[FWT] + 1) PerClk cycles after the address becomes valid when a single transfer occurs to a burst-enabled bank.
  - 1-8 (EBC0_BnAP[BWT] + 1) PerClk cycles after the last address becomes valid during a burst:
    - If EBC0_CFG[CTC] = 1 or EBC0_BnAP[TH]>0, $\overline{\text{PerCSn}}$ is driven high.
    - If EBC0_CFG[CTC] = 0 and EBC0_BnAP[TH] = 0, $\overline{\text{PerCSn}}$ transitions directly from logic 0 to the high-impedance state.
- During read operations, $\overline{\text{PerOE}}$ is driven low 0-3 (EBC0_BnAP[OEN]) PerClk cycles after $\overline{\text{PerCSn}}$ is active. $\overline{\text{PerOE}}$ goes inactive when $\overline{\text{PerCSn}}$ goes inactive.
- For bursts, the EBC drives a new address (EBC0_BnAP[FWT] + 1) + $n \times$ (EBC0_BnAP[BWT] + 1) cycles after the start of the transaction, where n = 0, 1, 2, ...
- During write operations, the write data is driven concurrent with each address except the first one.
- For the first address:
  - IF EBC0_CFG[OEO] = 1, PerData and PerPar become actively driven the same time as the address.
  - IF EBC0_CFG[OEO] = 0, PerData and PerPar become actively driven EBC0_BnAP[OEN] after $\overline{\text{PerCSn}}$ falls.
- $\overline{\text{PerWBE0:3}}$ can be either write byte enables or read and write enables.

  If EBC0_BnAP[BEM] = 0, $\overline{\text{PerWBE0:3}}$ are write byte enables and:
  - For the first transfer of a burst or a single transfer to a burst enabled bank, the appropriate write byte enables go low 0–3 (EBC0_BnAP[WBN]) cycles after $\overline{\text{PerCSn}}$ becomes active. The EBC then waits until EBC0_BnAP[FWT – EBC0_BnAP[WBF] + 1 cycles have elapsed since the start of the transaction and drives $\overline{\text{PerWBE0:3}}$ inactive.

- The remaining transfers of the burst are similar, except that PerWBE0:3 becomes active when each new address is driven on the interface. The PerWBE0:3 signals remain low for (EBC0_BnAP[BWT] + 1) – EBC0_BnAP[WBN] – EBC0_BnAP[WBF] cycles.

If EBC0_BnAP[BEM] = 1, PerWBE0:3 are byte enables that have timing identical to the peripheral address bus. In this case, EBC0_BnAP[WBN] and EBC0_BnAP[WBF] are ignored.

- PerBLast is active throughout the entire last (or only) transfer of a burst operation and is deactivated during the programmed hold time (EBC0_BnAP[TH]).
- EBC0_BnAP[CSN, WBN, OEN] apply to the first (or only) transfer of a burst, while EBC0_BnAP[WBF] applies to all transfers. It is required that
  FWT ≥ CSN + MAX(EBC0_BnAP[OEN,WBN]) + EBC0_BnAP[WBF], and EBC0_BnAP[BWT] ≥ WBF.
- Hold time (EBC0_BnAP[TH]) is programmable from 0 to 7 cycles. During the hold time, the peripheral address bus remains driven, and all control signals are driven inactive. If the operation was a write, the peripheral data bus continues driving the write data.
- There is no guarantee of dead cycles between transfers on the peripheral interface for one OPB transfer. If the OPB request size is greater than the bank size or the OPB request size is the same or less than the bank size but with the OPB seqaddr signal asserted, if the EBCO does packing data for read and unpacking data for write, and if the number of hold cycles is programmed to 0 (EBC0_BnAP[TH] = 0 and EBC0_BnAP[CSN] = 0), then:
  - PerCSn may not go inactive between the back-to-back transfers.
  - If EBC0_BnAP[OEN] = 0, PerOE may not become inactive between the back-to-back transfers.
  - If EBC0_BnAP[WBN] = 0 and EBC0_BnAP[WBF] = 0, PerWBE0:3 may not go inactive between the back-to-back transfers.
- There will always be a dead cycle between transfers on the peripheral interface that are initiated for separate OPB transfers.

27.3.1 Burst Read Transfer

Figure 27-14 shows the peripheral interface timing for a burst read transfer from a burst enabled (EBC0_BnAP[BME] = 1) bank. The transaction begins with the address being driven. If byte enable mode is enabled for the bank (EBC0_BnAP[BEM] = 1) the byte enables are also output concurrently on PerWBE0:3. PerCSn then becomes active EBC0_BnAP[CSN] cycles after the address, while PerOE goes low EBC0_BnAP[OEN] cycles after PerCSn. The EBC waits until EBC0_BnAP[FWT]+1 cycles have elapsed since the start of the transaction and then reads the data bus and the peripheral error input (PerErr). If parity checking is enabled (EBC0_BnAP[PEN] = 1), the parity is read at the same time.

The next address of the burst is then driven, and after EBC0_BnAP[BWT]+1 cycles, the EBC performs the next read. The remaining items in the burst are read in the same manner, except that PerBLast is active during the last data element. The EBC then drives PerCSn, PerOE and PerBLast high and waits EBC0_BnAP[TH] cycles before allowing any pending transfers to occur.

.

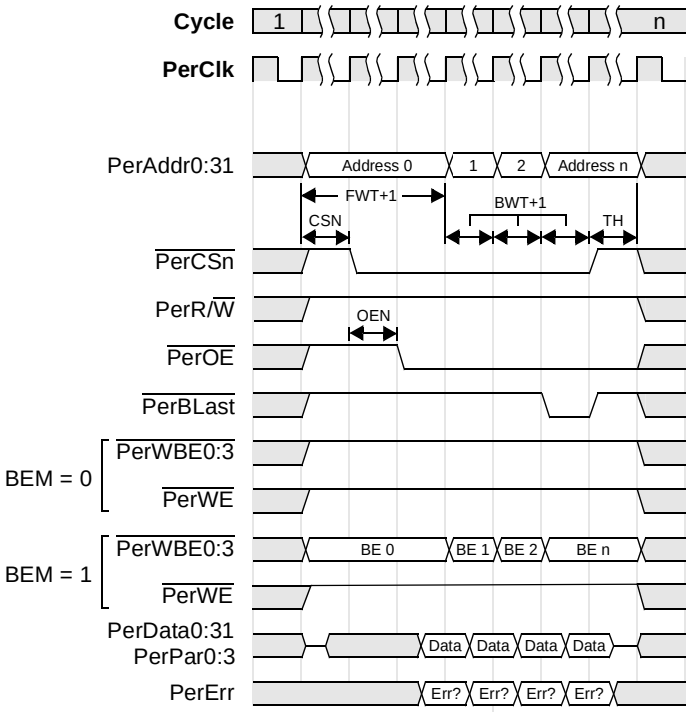


Figure 27-14. Burst Read Transfer

27.3.2 Burst Write Transfer

Figure 27-15 shows the peripheral interface timing for a burst write transfer to burst enabled (EBC0_BnAP[BME] = 1) bank. The transaction begins with the address being driven. At this point, the signalling sequence depends on whether byte enable mode is enabled for the bank.

- If EBC0_BnAP[BEM] = 0, byte enable mode is disabled, and PerWBE0:3 are write byte enables. In this case, the appropriate write byte enables go low EBC0_BnAP[WBN] cycles after PerCSn. The EBC then waits until (EBC0_BnAP[FWT] + 1) – EBC0_BnAP[WBF] cycles have elapsed since the start of the transaction and drives PerWBE0:3 inactive. EBC0_BnAP[WBF] cycles are then allowed to elapse, after which, the address and data are output for the second element in the burst. As shown in Figure 27-15, the EBC transfers the subsequent data items in a similar manner.
- If EBC0_BnAP[BEM] = 1, the PerWBE0:3 lines are byte enables and have the same timing as the peripheral address bus. In this configuration, external logic may be necessary to latch write data at the appropriate times. The driving of write data and data parity is controlled by EBC0_BnAP[OEN] when EBC0_CFG[OEO] = 0.

PerBLast goes low at the beginning of the last transfer to indicate that the burst is ending. The EBC then drives PerCSn and PerBLast high and waits EBC0_BnAP[TH] cycles before allowing any pending transfers to occur.

PerData0:31 and PerPar0:3 depend on OEO and OEN for their first data transfers as follows:

- If EBC0_CFG[OEO] = 1, PerData0:31 and PerPar0:3 become actively driven at the same time as the address.
- If EBC0_CFG[OEO] = 0, PerData0:31 and PerPar0:3 become actively driven EBC0_BnAP[OEN] after PerCSn falls.

For following transfers, PerData0:31 and PerPar0:3 are asserted at the same time as the address.

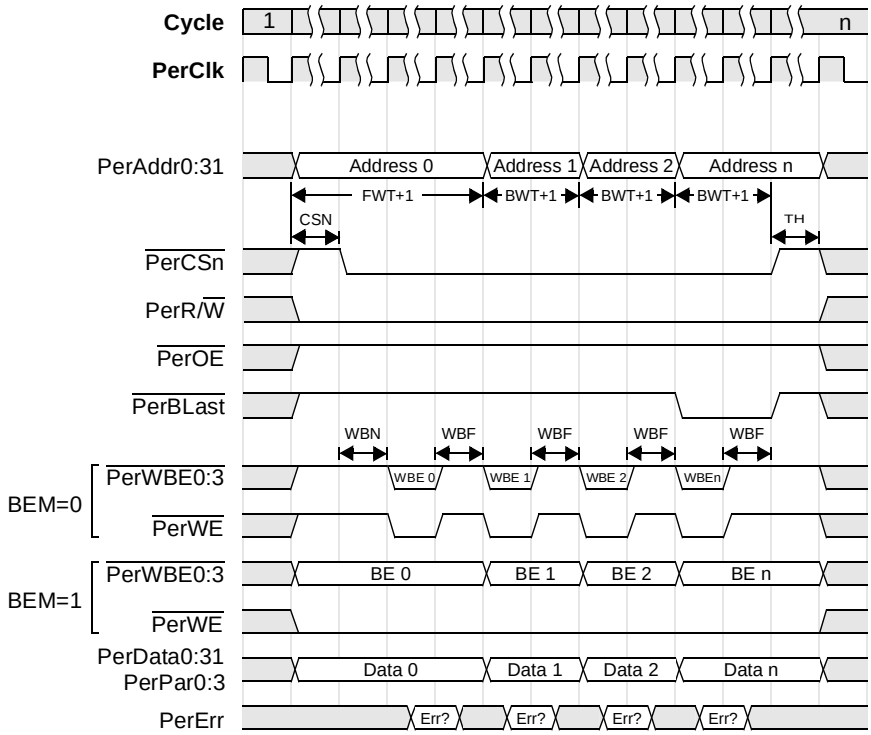
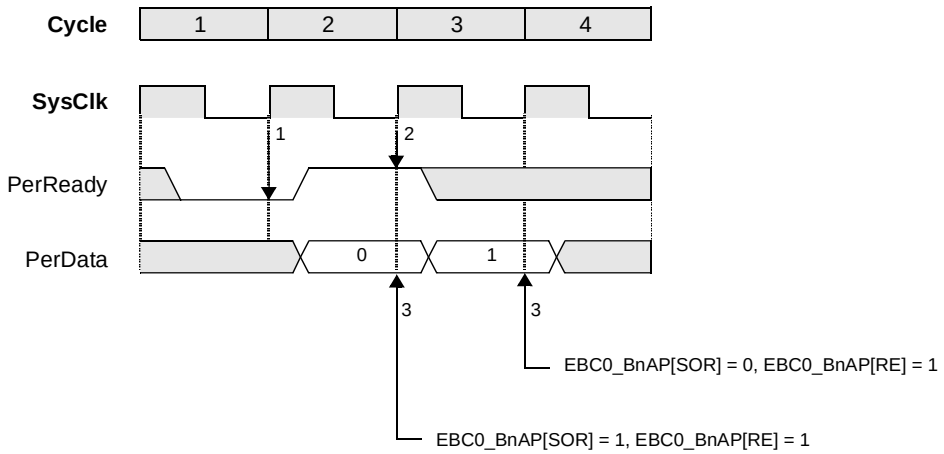


Figure 27-15. Burst Write Transfer

27.4 Device-Paced Transfers

The EBCO supports interfacing to peripheral and memory devices with variable timings via device-paced transfers. An EBCO bank is configured for device-paced transfers by programming EBC0_BnAP[RE]=1. During a device-paced transfer, the EBCO monitors the ready input (PerReady) and inserts wait states until the external device is ready or a timeout occurs. When using the ready input modes: Sample On Ready enabled and Sample On Ready disabled. The selection of these modes is controlled on a per-bank basis by EBC0_BnAP[SOR]. When Sample On Ready is enabled, (EBC0_BnAP[SOR] = 1) data is transferred on the PerClk rising edge where PerReady is sampled active. When Sampling On Ready is disabled (EBC0_BnAP[SOR] = 0), PerReady sampled active causes the data transfer to occur in the next cycle, which results in an additional cycle of wait time.



Notes:
(1) The above figure assumes that wait time expired in cycle 1 and Ready is being sampled.
(2) Arrow indicates Ready is sampled.
(3) Arrows indicate data being latched.

Figure 27-16. Available Ready Modes and Latch Times

- For burst disabled banks (EBC0_BnAP[BME] = 0), sampling of the PerReady input starts EBC0_BnAP[TWT] cycles after the beginning of the transfer. Wait states are inserted and sampling continues once per cycle until either PerReady is high when sampled or a timeout occurs.
- For burst enabled banks (EBC0_BnAP[BME] = 1), sampling of the PerReady input starts EBC0_BnAP[FWT] PerClk cycles after the beginning of the first transfer of a burst and EBC0_BnAP[BWT] cycles after the beginning of subsequent transfers of the burst.
  - If EBC0_BnAP[SOR] = 0, sampling continues every other cycle after the address goes active and the wait period (either TWT for non-burst, or FWT or BWT for burst bank) expires until either PerReady is sampled high or a timeout occurs. If EBC0_BnAP[FWT] = EBC0_BnAP[BWT] = 0, two-cycle transfers occur.
  - If EBC0_BnAP[SOR] = 1, sampling continues once per cycle after the address goes active and the wait period (either TWT for non-burst, or FWT or BWT for burst bank) expires until either PerReady is sampled high or a timeout occurs. If EBC0_BnAP[FWT] = EBC0_BnAP[BWT] = 0, single-cycle transfers occur.

- When EBC0_BnAP[SOR] = 1, data transfer occurs at the same PerClk edge where PerReady is sampled active. In contrast, if EBC0_BnAP[SOR] = 0, the data transfer occurs in the next cycle.
- When EBC0_BnAP[SOR] = 1, if the hold time is set to zero (EBC0_BnAP[TH] = 0), the programmed hold time is ignored, and the EBC performs the transaction with one hold cycle.
- When EBC0_BnAP[RE] = 1, the write byte enable off parameter must be programmed to zero (EBC0_BnAP[WBF] = 0). The Write Byte Enables (/PerWBE0:3) are not deasserted between data transfers but change as needed with the address.

The EBC may be programmed to wait only a limited time for PerReady to become active, or it may be programmed for unlimited wait. If EBC0_CFG[PTD] = 1, timeouts are disabled and the EBC waits indefinitely for an active PerReady.

If EBC0_CFG[PTD] = 0, device-paced timeouts are enabled, and the EBC only waits for the number of PerClk cycles programmed in EBC0_CFG[RTC]. The timeout counter is reset whenever the peripheral address changes. In this manner each data element within a device-paced burst transaction is treated separately for the purposes of determining whether a timeout error occurs. If PerReady does not become active before the timeout counter reaches the value programmed into EBC0_CFG[RTC], the transfer is aborted and an error is signalled. See ~~Section 27.5.5, Error Reporting~~*Error Reporting* on page 885 for details on how timeout errors are logged.

The required relationship between the access bank parameters changes for device-paced banks, which require that the bank timing wait parameters meet the following relationships:

For non-burst configured banks:

EBC0_BnAP[TWT] > EBC0_BnAP[CSN] + MAX(EBC0_BnAP[OEN, WBN]).

For burst configured bank:

For first transfer:

EBC0_BnAP[FWT] > EBC0_BnAP[CSN] + MAX(EBC0_BnAP[OEN, WBN]).

For following transfers:

EBC0_BnAP[BWT] Š EBC0_BnAP[WBF]

27.4.1 Device-Paced Single Read Transfer

Figure 27-17 shows the peripheral interface timing for a device-paced single read transfer to a burst disabled (EBC0_CFG[BME] = 0) bank. The transaction begins with the address being driven. Since this is a single transfer, PerBLast is driven active along with the address. If byte enable mode is enabled for the bank (EBC0_BnAP[BEM] = 1), the byte enables are also output concurrently on PerWBE0:3. PerCSn then becomes active EBC0_BnAP[CSN] cycles after the address, while PerOE goes low EBC0_BnAP[OEN] cycles after PerCSn.

The EBC then waits until EBC0_BnAP[TWT] cycles have elapsed since the start of the transaction and then begins sampling PerReady. If device-paced timeouts are disabled (EBC0_CFG[PTD] = 1), the EBC waits indefinitely for PerReady to become active; otherwise, the EBC waits only EBC0_CFG[RTC] cycles from the start of the transaction until logging a timeout error.

Once PerReady is sampled active and if Sample On Ready is disabled (EBC0_BnAP[SOR] = 0), the EBC waits one more cycle. The EBC then samples the data bus and the peripheral error input (PerErr). If parity checking is enabled (EBC0_BnAP[PEN] = 1), the parity is also read at this time. The EBC then drives PerCSn, PerOE, and PerBLast high and waits EBC0_BnAP[TH] cycles before allowing any pending EBC transfers to occur.

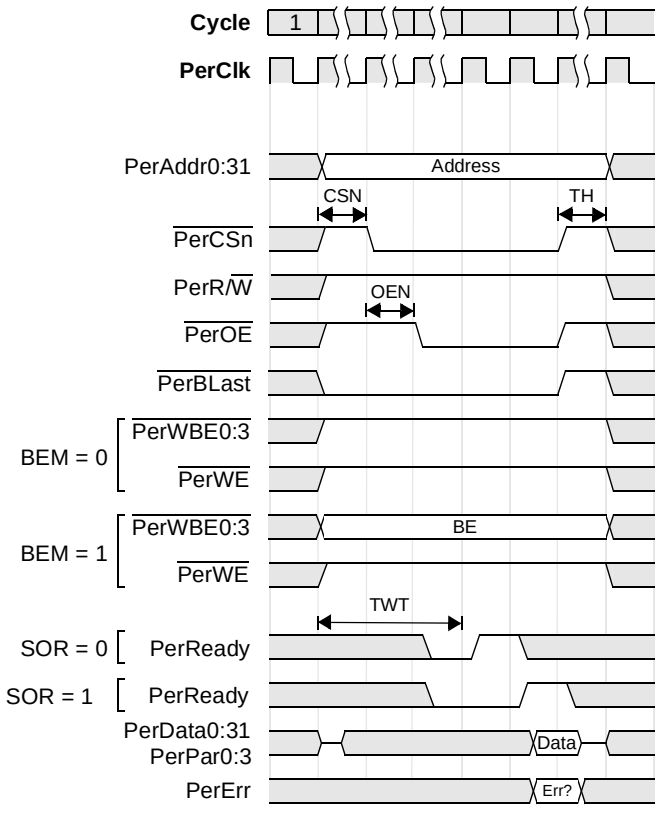


Figure 27-17. Device-Paced Single Read Transfer

27.4.2 Device-Paced Single Write Transfer

Figure 27-18 shows the peripheral interface timing for a device-paced single write transfer to a burst disabled (EBC0_BnAP[BME] = 0) bank. The transaction begins with the address being driven. Since this is a single transfer, PerBLast is also driven active along with the address. At this point, the signalling sequence depends on whether byte enable mode is enabled for the particular bank.

- If EBC0_BnAP[BEM] = 0, byte enable mode is disabled and the PerWBE0:3 are write byte enables. The appropriate write byte enables go low EBC0_BnAP[WBN] cycles after PerCSn went low. PerWBE0:3 return high on the same PerClk edge that the write data is transferred (see below).
- If EBC0_BnAP[BEM] = 1, the PerWBE0:3 lines are byte enables and have the same timing as the peripheral address bus.

EBC0_CFG[OEO] also has an effect on PerData and PerPar as follows:

- If EBC0_CFG[OEO] = 1, PerData and PerPar become actively driven at the same time as the address.
- If EBC0_CFG[OEO] = 0, PerData and PerPar become actively driven EBC0_BnAP[OEN] cycles after PerCSn falls.

The EBC then waits until EBC0_BnAP[TWT] cycles have elapsed since the start of the transaction and then begins sample PerReady. If device-paced timeouts are disabled (EBC0_CFG[PTD] = 1), the EBC waits indefinitely for PerReady to become active. Otherwise, the EBC waits only EBC0_CFG[RTC] cycles from the start of the transaction until logging a timeout error.

If PerReady is sampled active and Sample On Ready is disabled (EBC0_BnAP[SOR] = 0), the EBC waits one more cycle. At this point, the write transfer occurs, and the EBC reads the peripheral error input (PerErr). The EBC then drives PerCSn, PerOE, and PerBLast high and waits EBC0_BnAP[TH] cycles before allowing any pending EBC transfers to occur.

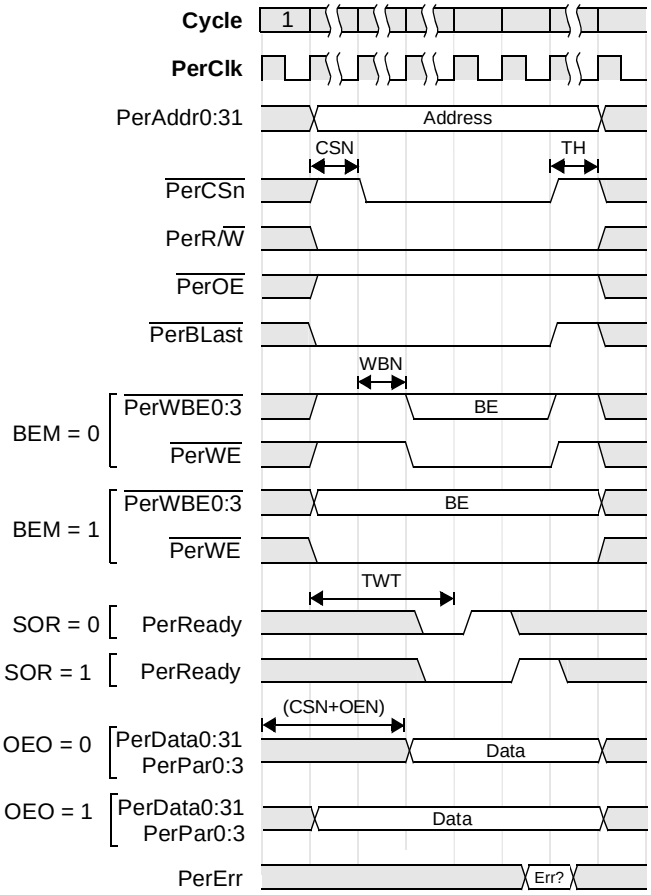


Figure 27-18. Device-Paced Single Write Transfer

27.4.3 Device-Paced Burst Read Transfer

Figure 27-19 shows the peripheral interface timing for a device-paced burst read transfer from a burst enabled (EBC0_BnAP[BME] = 1) bank. The transaction begins with the address being driven. If byte enable mode is enabled for the bank (EBC0_BnAP[BEM] = 1), the byte enables are also output concurrently on PerWBE0:3. PerCSn then becomes active EBC0_BnAP[CSN] cycles after the address, while PerOE goes low EBC0_BnAP[OEN] cycles after PerCSn. The EBC then waits until EBC0_BnAP[FWT] cycles have elapsed since the start of the transaction and begins sampling PerReady.

If device-paced timeouts are disabled (EBC0_CFG[PTD] = 1), the EBC waits indefinitely for PerReady to become active; otherwise, the EBC waits only EBC0_CFG[RTC] cycles from the start of each new address until logging a timeout error.

If PerReady is sampled active and Sample On Ready is disabled (EBC0_BnAP[SOR] = 0), the EBC waits one more cycle before sampling read data. The EBC then reads the data bus and the peripheral error input (PerErr). If parity checking is enabled (EBC0_BnAP[PEN] = 1), the parity is also read. See Figure 27.3for information on extra reads during burst operations.

The next address of the burst is then driven, and after EBC0_BnAP[BWT] cycles, the EBC waits for PerReady as described above. The remaining items in the burst are read in this same manner, except that PerBLast is active during the last data element. The EBC then drives PerCSn, PerOE, and PerBLast high and waits EBC0_BnAP[TH] cycles before allowing any pending transfers to occur.

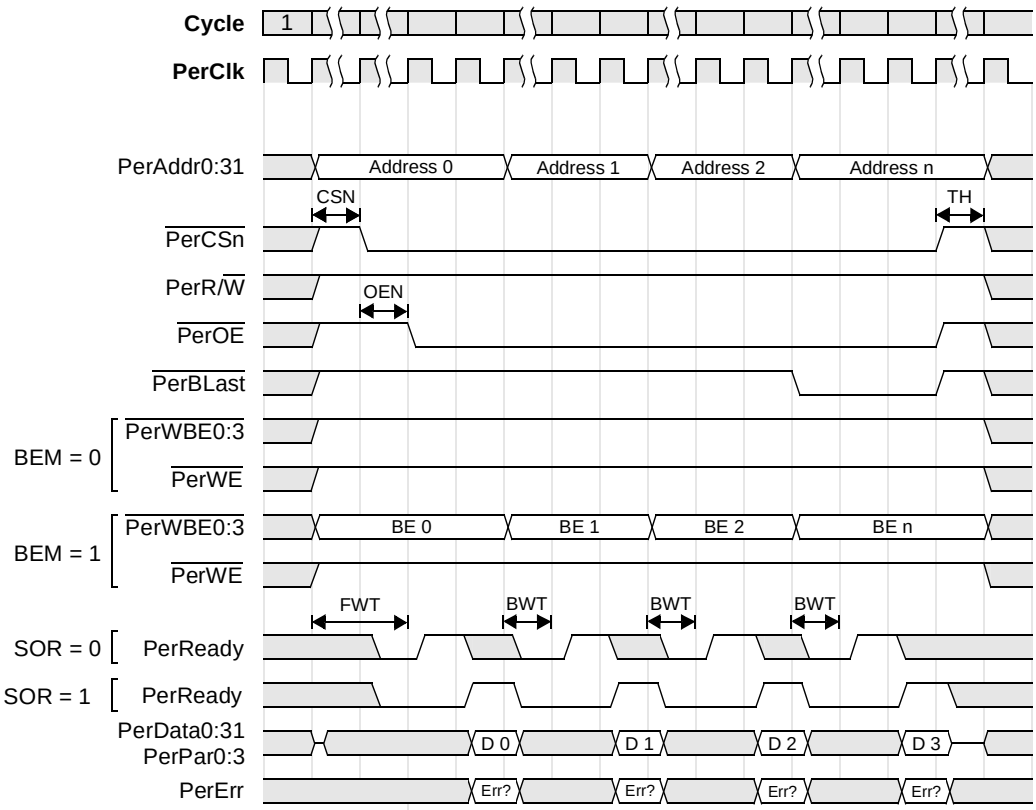


Figure 27-19. Device-Paced Burst Read Transfer

27.4.4 Device-Paced Burst Write Transfer

Figure 27-20 shows the peripheral interface timing for a device-paced burst write transfer to a burst enabled (EBC0_BnAP[BME] = 1) bank. The transaction begins with the address being driven. PerCSn then becomes active EBC0_BnAP[CSN] cycles after the address. At this point, the signalling sequence depends on whether or not byte enable mode is enabled for the bank.

- If byte enable mode is disabled (EBC0_BnAP[BEM] = 0), PerWBE0:3 are write byte enables. In this case, the appropriate write byte enables go low EBC0_BnAP[WBN] cycles after PerCSn becomes active for the first element in a burst and at the same time as each new address for the remainder of the burst. If EBC0_BnAP[WBN] <> 0, PerWBE0:3 is driven inactive on the same PerClk edge that write data is transferred (see Figure 27-20); otherwise, PerWBE0:3 remains low for all data elements in the burst.
- If EBC0_BnAP[BEM] = 1, PerWBE0:3 are byte enables and have the same timing as PerAddr0:31.

The EBC then waits until EBC0_BnAP[FWT] cycles have elapsed since the start of the transaction and begins sampling PerReady. If device-paced timeouts are disabled (EBC0_CFG[PTD] = 1), the EBC waits indefinitely for PerReady; otherwise, the EBC waits only EBC0_CFG[RTC] cycles from the start of the transaction until logging a timeout error.

If PerReady is sampled active and Sample On Ready is disabled (EBC0_BnAP[SOR] = 0), the EBC waits one more cycle. At this point, the write transfer occurs, and the EBC reads the peripheral error input (PerErr).

The next address of the burst is then driven, and after EBC0_BnAP[BWT] cycles, the EBC waits for PerReady as described above. The remaining items in the burst are transferred in this same manner, except that PerBLast is active for the last data element. The EBC then drives PerCSn, PerOE, and PerBLast high and waits EBC0_BnAP[TH] cycles before allowing any pending transfers to occur.

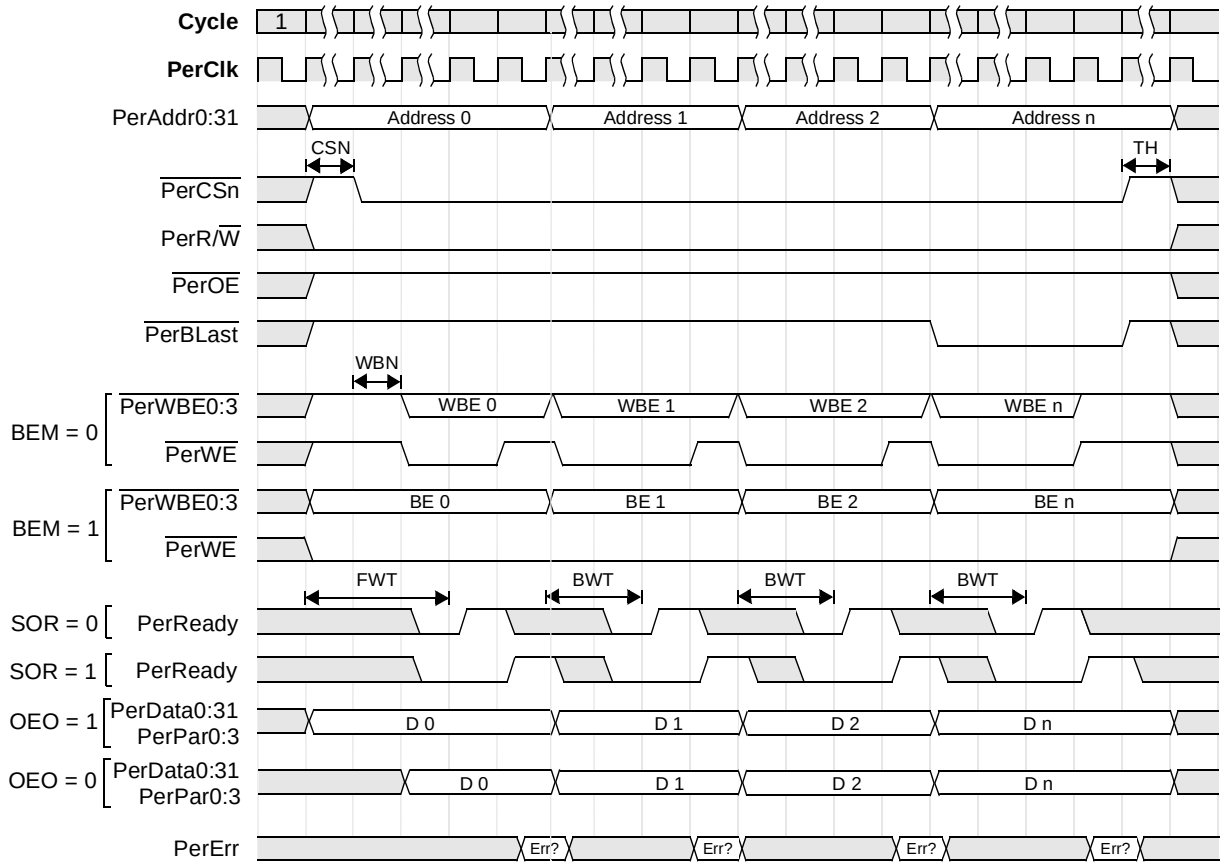


Figure 27-20. Device-Paced Burst Write Transfer

27.5 EBC Registers

All EBC configuration and status registers are accessed using the **mtdcr** and **mfdcr** instructions. Access to these registers is performed using an indirect addressing method through the EBC0_CFGADDR and EBC0_CFGDATA registers (see Table 27-3).

Table 27-3. EBC DCR Addresses

Mnemonic	Name	DCR Number	Access	Page
EBC0_CFGADDR	EBC Address Register	0x012	R/W	27-28880
EBC0_CFGDATA	EBC Data Register	0x013	R/W	27-28880

Table 27-4 lists the indirectly accessed EBC configuration and status registers.

Table 27-4. External Bus Configuration and Status Registers

Mnemonic	Name	Offset	Access	Page
EBC0_B0CR	Peripheral Bank 0 Configuration Register	0x00	R/W	27-28880
EBC0_B1CR	Peripheral Bank 1 Configuration Register	0x01	R/W	27-28880
EBC0_B2CR	Peripheral Bank 2 Configuration Register	0x02	R/W	27-28880
EBC0_B3CR	Peripheral Bank 3 Configuration Register	0x03	R/W	27-28880
EBC0_B4CR	Peripheral Bank 4 Configuration Register	0x04	R/W	27-28880
EBC0_B5CR	Peripheral Bank 5 Configuration Register	0x05	R/W	27-28880
EBC0_B6CR	Peripheral Bank 6 Configuration Register	0x06	R/W	27-28880
EBC0_B7CR	Peripheral Bank 7 Configuration Register	0x07	R/W	27-28880
EBC0_B0AP	Peripheral Bank 0 Access Parameter Registers	0x10	R/W	27-30882
EBC0_B1AP	Peripheral Bank 1 Access Parameter Registers	0x11	R/W	27-30882
EBC0_B2AP	Peripheral Bank 2 Access Parameter Registers	0x12	R/W	27-30882
EBC0_B3AP	Peripheral Bank 3 Access Parameter Registers	0x13	R/W	27-30882
EBC0_B4AP	Peripheral Bank 4 Access Parameter Registers	0x14	R/W	27-30882
EBC0_B5AP	Peripheral Bank 5 Access Parameter Registers	0x15	R/W	27-30882
EBC0_B6AP	Peripheral Bank 6 Access Parameter Registers	0x16	R/W	27-30882
EBC0_B7AP	Peripheral Bank 7 Access Parameter Registers	0x17	R/W	27-30882
EBC0_BEAR	Bus Error Address Register	0x20	R	27-34886
EBC0_BESR	Peripheral Bus Error Status Register	0x21	R/W	27-34886
EBC0_CFG	External Bus Configuration Register	0x23	R/W	27-36887
EBC0_CID	EBC ID Register	0x24	R	27-37889

To access one of these registers, software must first write the address offset into the EBC0_CFGADDR register. The target register can then be read or written through the EBC0_CFGDATA DCR address. The following PowerPC code illustrates this procedure by writing the EBC0_BnCR register and then reading back the written value.

```
li      r3,EBC0_B0CR            ! address offset
lis     r4,<config upper>       ! upper 16-bits of configuration data
ori     r4,r4,<config lower>    ! lower 16-bits of configuration data
mtdcr   EBC0_CFGADDR,r3         ! set offset addr
mtdcr   EBC0_CFGDATA,r4         ! write config data
mfdcr   r5,EBC0_CFGDATA         ! read back config data
```

Register bits marked as reserved must only be written to their defined reset value.

~~0.0.1 EBC Address Register~~

27.5.1 EBC Address Register (EBC0_CFGADDR)

~~Figure 27-21~~*Figure 27-21* shows the EBC0_CFGADDR bit definitions. This register is written by software to indirectly address the EBC registers.

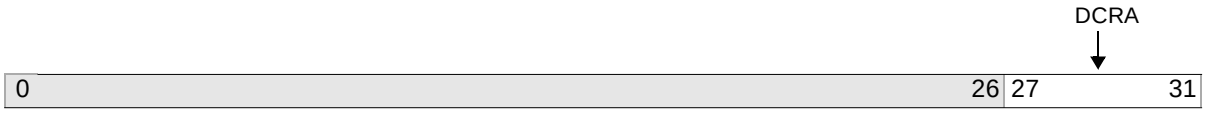


Figure 0-1. EBC Address Register (EBC0_CFGADDR)

0:26		Reserved
27:31	DCRA	DCR Address Offset

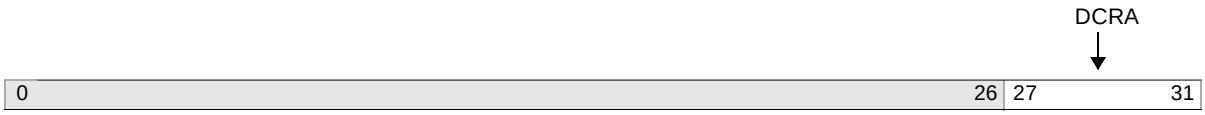


Figure 27-21. EBC Address Register (EBC0_CFGADDR)

0:26		Reserved
27:31	DCRA	DCR Address Offset

~~0.0.2 EBC Data Register~~

27.5.2 EBC Data Register (EBC0_DATA)

This register is a data port written by software after it writes the EBC0_CFGADDR to read or write the other EBC registers.

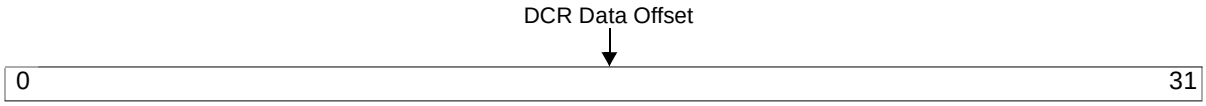


Figure 0-2. EBC Data Register (EBC0_CFGDATA)

0:31	DCRD	DCR Data Port

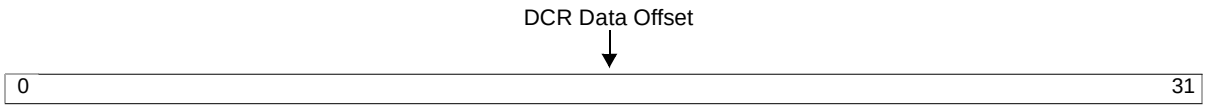


Figure 27-22. EBC Data Register (EBC0_CFGDATA)

0:31	DCRD	DCR Data Port

27.5.3 Peripheral Bank Configuration Registers (EBC0_B0CR-EBC0_B7CR)

EBC0_BnCR must be configured to enable memory in each respective bank. Boot ROM, if installed, must be attached to bank ~~0~~0 and must use CS0.

If a boot ROM is present, Bank 0 is automatically configured using the core strapping pins. After system reset, the EBC0_B0CR[BAS] is loaded with a value ~~from strapping pin I_ebco_dcr_bootrom_bus[0:11]~~of 0xFFE, EBC0_B0CR[BS] is loaded with a value ~~from I_ebco_dcr_bootrom_bsize~~of 0b001, and EBC0_B0CR[BU] is loaded with a value ~~from I_ebco_dcr_bootrom_bu[0:1]~~of 0b01. The EBC0_B0CR[BW] is loaded with a value ~~from I_ebco_dcr_bootrom_width~~of CCR_CPU_bROM_Size[0:~~1~~2]. Software may reconfigure EBC0_B0CR later if necessary.

EBC0_BnCR[BAS] sets the base address for a peripheral device. The bank starting address must be a multiple of the bank size programmed in EBC0_BnCR[BS]. EBC0_BnCR[BAS] is compared to bits 0:11 of the address. If the address is within the range of an EBC0_BnCR[BAS] field, the associated bank is enabled for the transaction.

Multiple bank registers must not be programmed with the same base address or as overlapping banks. An attempt to use such overlapping banks is not recorded as an error and may cause contention on the external bus.

EBC0_BnCR[BS] sets the number of bytes which the bank may access, beginning with the base address set in EBC0_BnCR[BAS].

EBC0_BnCR[BU] protects banks of physical devices from read or write accesses.

When a write access is attempted to an address within the range of EBC0_BnCR[BAS], and the bank is designated as read-only, a protection error occurs. Also, when a read access is attempted to an address within the range of EBC0_BnCR[BAS] field, and the bank is designated as write-only, a protection error occurs. The address of the attempted access is logged in EBC0_BEAR and type of error is logged in EBC0_BESR.

EBC0_BnCR[BW] controls the width of region accesses. If devices are attached to the data bus, the EBC automatically packs read data and unpacks write data when a data transfer size mismatch exists.

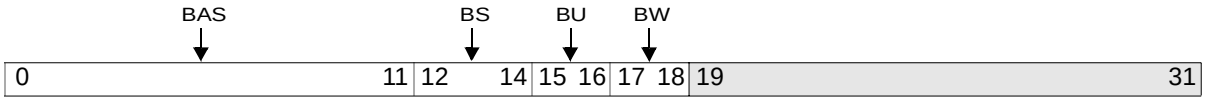


Figure 0-3. Peripheral Bank Configuration Registers (EBC0_B0CR-EBC0_B7CR)

0:11	BAS	Base Address Select	Specifies the bank starting address, which must be a multiple of the bank size.
12:14	BS	Bank Size 000 1 MB bank 001 2 MB bank 010 4 MB bank 011 8 MB bank 100 16 MB bank 101 32 MB bank 110 64 MB bank 111 128 MB bank	
15:16	BU	Bank Usage 00 Disabled 01 Read-only 10 Write-only 11 Read/Write	Specifies the type of accesses allowed for the bank. A protect error occurs if a write is attempted to a read-only bank or a read is attempted from a write-only bank.
17:18	BW	Bus Width 00 8-bit bus 01 16-bit bus 10 Reserved 11 32-bit bus	
19:31		Reserved	

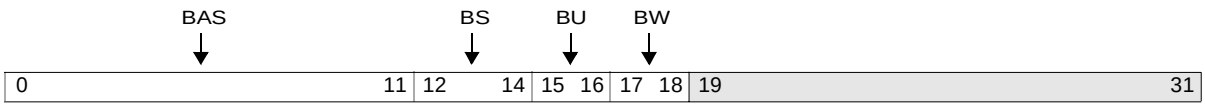


Figure 27-23. Peripheral Bank Configuration Registers (EBC0_B0CR-EBC0_B7CR)

0:11	BAS	Base Address Select	Specifies the bank starting address, which must be a multiple of the bank size.

12:14	BS	Bank Size 000 1 MB bank 001 2 MB bank 010 4 MB bank 011 8 MB bank 100 16 MB bank 101 32 MB bank 110 64 MB bank 111 128 MB bank	
15:16	BU	Bank Usage 00 Disabled 01 Read-only 10 Write-only 11 Read/Write	Specifies the type of accesses allowed for the bank. A protect error occurs if a write is attempted to a read-only bank or a read is attempted from a write-only bank.
17:18	BW	Bus Width 00 8-bit bus 01 16-bit bus 10 Reserved 11 32-bit bus	
19:31		Reserved	

27.5.4 Peripheral Bank 0–7 Access Parameters (EBC0_B0AP-EBC0_B7AP)

The EBC0_BnAP registers contain access parameters for their associated banks.

EBC0_BnAP[BME] controls bursting for external devices and all packing and unpacking operations. If ~~this transaction shows up on the OPB, then /???/. If~~ EBC0_BnAP[BME] = 1, bursting is enabled. When bursting is enabled, EBC0_BnAP[CSN, OEN, and FWT] apply only to the first transfer, while EBC0_BnAP[BWT and WBN] apply during all remaining transfers of the burst.

EBC0_BnAP[TWT] specifies the number of wait states taken by each transfer to the bank. The number of cycles from address valid to the deassertion of $\overline{\text{PerCSn}}$ is (1 + EBC0_BnAP[TWT]), where $0 \leq TWT \leq 255$. This field is used for non-burst configured banks (EBC0_BnAP[BME] = 0).

EBC0_BnAP[FWT] specifies the number of wait states to be taken by the first access to the bank during a burst configured bank (EBC0_BnAP[BME] = 1). During a burst, the number of cycles from the first address valid to the second address is (1 + EBC0_BnAP[FWT]), where $0 \leq FWT \leq 31$.

EBC0_BnAP[BWT] specifies the number of wait states to be taken by accesses beyond the first during a burst transfer to a burst enabled bank (EBC0_BnAP[BME] = 1). On burst accesses, except for the last, the number of cycles from address valid to the next valid address on each burst access is (1 + EBC0_BnAP[BWT]), where $0 \leq BWT \leq 7$. On the last burst access, the number of cycles from address valid to the deassertion of $\overline{\text{PerCSn}}$ is (1 + EBC0_BnAP[BWT]), where $0 \leq BWT \leq 7$.

EBC0_BnAP[CSN] specifies the chip select turn on delay relative to the address. $\overline{\text{PerCSn}}$ may turn on coincident with the address or be delayed by 1, 2, or 3 PerClk cycles.

EBC0_BnAP[BCE] controls burst read from a peripheral device so extra reads do not occur. If a bank is enabled for burst and EBC0_BnAP[BCE] = 1, EBC can burst for fixed lengths bursts of the programmed value. If a bank is enabled for burst and EBC0_BnAP[BCE] = 1, EBC runs burst cycles on the external bus until the OPB_SeqAddr signal goes away and may do extra read operations.

EBC0_BnAP[BCT] specifies the number of transfers from the peripheral device for a decoded OPB sequential read when EBC0_BnAP[BCE] = 1. If the OPB sequential transfer is longer than the number of transfers programmed, the EBC has enough time to pack the data and continues transferring until the OPB sequential transfer is deasserted. If the OPB sequential transfer is shorter than the number of transfers programmed, the EBCO divides the transfer into several fixed length transfers. It then continues to do fixed length transfers until the OPB sequential transfer is deasserted. Figure 27-5 shows the actual number of bytes transferred, depending upon the size of the peripheral device and EBC0_BnAP[BCT].

Table 27-5. Fixed Length Burst Count Transfer Size

EBC0_BnAP[BCT10:11]	EBC0_BnCR[BW17:18]	Number of Bytes Read
0b00	8-bit	2
0b00 0b01	16-bit 8-bit	4
0b00 0b01 0b10	32-bit 16-bit 8-bit	8
0b01 0b10 0b11	32-bit 16-bit 8-bit	16
0b10 0b11	32-bit 16-bit	32
0b11	32-bit	64

EBC0_BnAP[OEN], for read operations, specifies when the output enable signal, $\overline{\text{PerOE}}$, is asserted for read operations relative to the chip select signal. If EBC0_BnAP[OEN] = 0, $\overline{\text{PerOE}}$ is asserted coincident with the chip select. If EBC0_BnAP[OEN] = 1, 2, or 3, $\overline{\text{PerOE}}$ is delayed by 1, 2, or 3 PerClk cycles.

EBC0_BnAP[OEN], for write operations, specifies when the peripheral data bus is driven relative to the chip select signal when EBC0_CFC [OEO] = 0.

EBC0_BnAP[WBN], when EBC0_BnAP[BEM]=0, specifies when the write byte enables, $\overline{\text{PerWBE0:3}}$, are asserted relative to the chip select signal. If EBC0_BnAP[WBN] = 0, $\overline{\text{PerWBE0:3}}$ turns on coincident with the chip select. If EBC0_BnAP[WBN] = 1, 2, or 3, $\overline{\text{PerWBE0:3}}$ is delayed 1, 2, or 3 PerClk cycles from the chip select.

EBC0_BnAP[WBF], when EBC0_BnAP[BEM]=0, specifies when the write byte enables are deasserted relative to the deassertion of the chip select signal. If EBC0_BnAP[WBF] = 0, $\overline{\text{PerWBE0:3}}$ goes high coincident with the chip select signal. If EBC0_BnAP[WBF] = 1, 2, or 3, then $\overline{\text{PerWBE0:3}}$ turns off 1, 2, or 3 PerClk cycles before the turn-off of the chip select signal.

> **Programming Note:** It is an error to set EBC0_BnAP[WBF] > EBC0_BnAP[BWT]; moreover, for device-paced transfers (EBC0_BnAP[RE] = 1), EBC0_BnAP[WBF] must be 0.

EBC0_BnAP[TH] specifies the number of PerClk cycles (0–7) that the peripheral bus is held idle after the deassertion of $\overline{\text{PerCSn}}$. During these cycles, the address bus and data bus are active and PerR/$\overline{\text{W}}$ is valid. During the hold time, chip select, output enable, and write byte enables are inactive. If Ready Mode is used (EBC0_BnAP[RE] = 1) with sample on ready (EBC0_BnAP[SOR] = 1), EBC0_BnAP[TH] must be set to at least 1.

EBC0_BnAP[RE] controls the use of the PerReady input signal. If EBC0_BnAP[RE] = 0, the PerReady input is ignored and no additional wait states are inserted into bus transactions. If EBC0_BnAP[RE] = 1, the PerReady input is examined after the wait period expires; additional wait states are inserted if the PerReady input is 0. The maximum number of wait states in each new address is determined by the settings of EBC0_CFG[PTD, RTC]. If EBC0_CFG[PTD] = 1, the ready timeout function is disabled, and the PPC440GP waits indefinitely until PerReady = 1. If EBC0_CFG[PTD] = 1, the PPC440GP waits the number of cycle indicated by EBC0_CFG[RTC] for PerReady to become active. If PerReady does not become active in the allotted time, the address of the error is logged in EBC0_BEAR and the type of error is captured in EBC0_BESR.

EBC0_BnAP[SOR] controls the location of the data transfer cycle with respect to the PerReady input. If EBC0_BnAP[SOR] = 1 the data transfer occurs on the same PerClk edge that PerReady is sampled active, whereas if EBC0_BnAP[SOR] = 0 the data transfer occurs one cycle later.

EBC0_BnAP[BEM] controls whether $\overline{\text{PerWBE0:3}}$ is active during writes or for both reads and writes.

EBC0_BnAP[PEN] enables odd parity generation and checking.

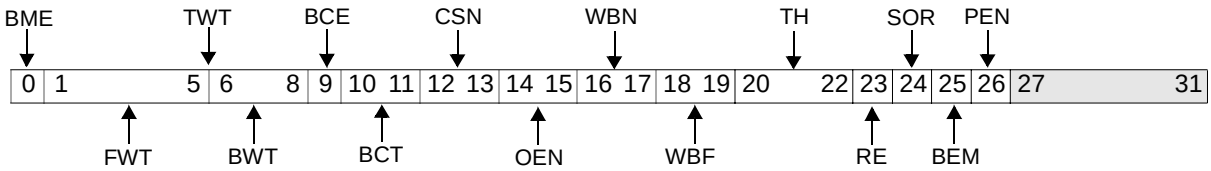


Figure 0-4. Peripheral Bank Access Parameters (EBC0_B0AP-EBC0_B7AP)

0	BME	Burst Mode Enable 0 Bursting is disabled 1 Bursting is enabled	
1:8	TWT	Transfer Wait 0 to 255 PerClk cycles	If bursting is disabled, BME=0, wait states on all non-burst transfers.
1:5	FWT	First Wait 0 to 31 PerClk cycles	If bursting is enabled, BME=1, number of wait states on the first transfer of a burst.
6:8	BWT	Burst Wait 0 to 7 PerClk cycles	If bursting is enabled, BME=1, number of wait states on non-first transfers of a burst.
9	BCE	Fixed Length Burst Reads 0 Disabled 1 Enabled	If BCE=1, all burst read operations consist of exactly BCT transfers. When BCE=0, burst read operations may read more than the requested amount of data.
10:11	BCT	Fixed Length Burst Count 00 2 transfers 01 4 transfers 10 8 transfers 11 16 transfers	The amount of data transferred depends upon the programmed bank width, EBC0_BnCR[BW].
12:13	CSN	Chip Select On Timing 0 to 3 PerClk cycles.	Number of cycles from peripheral address driven to $\overline{\text{PerCSn}}$ low.
14:15	OEN	Output Enable On TIming 0 to 3 PerClk cycles	Number of cycles from $\overline{\text{PerCSn}}$ low to $\overline{\text{PerOE}}$ low.

16:17	WBN	Write Byte Enable On Timing 0 to 3 PerClk cycles	If BEM=0, number of cycles from $\overline{\text{PerCSn}}$ low to $\overline{\text{PerWBE0:3}}$ active.
18:19	WBF	Write Byte Enable Off Timing (If bursting and ready pin input are disabled) 0 to 3 PerClk cycles	If BEM=0 and RE=0, number of cycles $\overline{\text{PerWBE0:3}}$ becomes inactive prior to $\overline{\text{PerCSn}}$ inactive.
20:22	TH	Transfer Hold 0 to 7 PerClk cycles	Contains the number of hold cycles inserted at the end of a transfer. Hold cycles insert idle bus cycles between transfers to enable slow peripherals to remove data from the data bus before the next transfer begins.
23	RE	Ready Enable 0 PerReady input is disabled 1 PerReady input is enabled	Set for Device Paced Transfers
24	SOR	Sampling of Ready for Device Paced 0 Ready is asserted by the device one clock before data is ready on the external bus 1 Ready is asserted by the device on the same clock that data is ready on the external bus	
25	BEM	Byte Enable Mode 0 $\overline{\text{PerWBE0:3}}$ pins are only active on write cycles 1 $\overline{\text{PerWBE0:3}}$ pins are active for both read and write cycles	
26	PEN	Parity Enable 0 Disable Parity checking 1 Enable Parity checking	The EBCO implements odd parity.
27:31		Reserved	

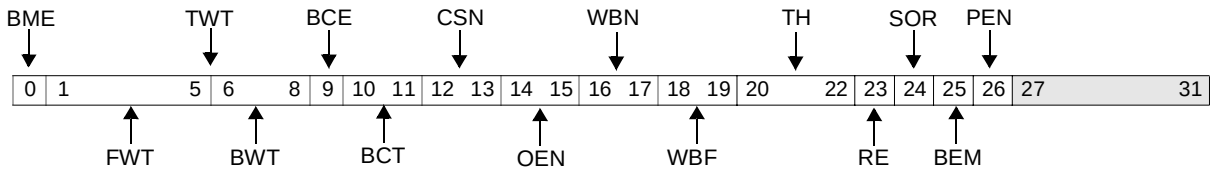


Figure 27-24. Peripheral Bank Access Parameters (EBC0_B0AP-EBC0_B7AP)

0	BME	Burst Mode Enable Bursting is disabled Bursting is enabled	
1:8	TWT	Transfer Wait 0 to 255 PerClk cycles	If bursting is disabled, BME=0, wait states on all non-burst transfers.
1:5	FWT	First Wait 0 to 31 PerClk cycles	If bursting is enabled, BME=1, number of wait states on the first transfer of a burst.

6:8	BWT	Burst Wait 0 to 7 PerClk cycles	If bursting is enabled, BME=1, number of wait states on non-first transfers of a burst.
9	BCE	Fixed Length Burst Reads Disabled Enabled	If BCE=1, all burst read operations consist of exactly BCT transfers. When BCE=0, burst read operations may read more than the requested amount of data.
10:11	BCT	Fixed Length Burst Count 2 transfers 4 transfers 8 transfers 16 transfers	The amount of data transferred depends upon the programmed bank width, EBC0_BnCR[BW].
12:13	CSN	Chip Select On Timing 0 to 3 PerClk cycles.	Number of cycles from peripheral address driven to PerCSn low.
14:15	OEN	Output Enable On TIming 0 to 3 PerClk cycles	Number of cycles from PerCSn low to PerOE low.
16:17	WBN	Write Byte Enable On Timing 0 to 3 PerClk cycles	If BEM=0, number of cycles from PerCSn low to PerWBE0:3 active.
18:19	WBF	Write Byte Enable Off Timing (If bursting and ready pin input are disabled) 0 to 3 PerClk cycles	If BEM=0 and RE=0, number of cycles PerWBE0:3 becomes inactive prior to PerCSn inactive.
20:22	TH	Transfer Hold 0 to 7 PerClk cycles	Contains the number of hold cycles inserted at the end of a transfer. Hold cycles insert idle bus cycles between transfers to enable slow peripherals to remove data from the data bus before the next transfer begins.
23	RE	Ready Enable PerReady input is disabled PerReady input is enabled	Set for Device Paced Transfers
24	SOR	Sampling of Ready for Device Paced Ready is asserted by the device one clock before data is ready on the external bus Ready is asserted by the device on the same clock that data is ready on the external bus	
25	BEM	Byte Enable Mode PerWBE0:3 pins are only active on write cycles PerWBE0:3 pins are active for both read and write cycles	
26	PEN	Parity Enable Disable Parity checking Enable Parity checking	The EBCO implements odd parity.
27:31		Reserved	

27.5.5 Error Reporting

The EBC monitors four kinds of errors when performing read and write transfers. Of these four, bank protect and external bus errors are always checked, while timeout and read parity error checking must be enabled via DCR-mapped configuration registers. When an enabled error is detected by the EBC, the error condition is logged into the EBC0_BESR, the address is logged into EBC0_BEAR, and an interrupt is asserted to the

system interrupt controller for one PerClk cycle. See ~~"Peripheral Bus Error Status Register (EBC0_BESR)" on page 34~~*Peripheral Bus Error Status Register (EBC0_BESR)* on page 886 for more detailed error handling information.

27.5.5.1 Protect Error

The Requested read or write operation violates the bank usage programmed in EBC0_BnCR[BU]. For example, in all cases of a write attempt to read-only bank, no external bus activity occurs.

27.5.5.2 External Bus Error

The PerErr input was sampled active during the data transfer cycle of a read or write operation. The associated data is read or written as usual.

27.5.5.3 Timeout Error

This error is possible during memory operations when both PerReady sampling is enabled (EBC0_BnAP[RE] = 1) and device paced timeouts are enabled (EBC0_CFG[PTD] = 0). Whenever the peripheral address bus changes, the EBC begins counting PerClk cycles. If the count reaches the value represented by EBC0_CFG[RTC], a timeout error occurs. Note that timeout errors are not possible during the peripheral portion of DMA transfers.

27.5.5.4 Parity Error

A parity error indicates that the parity calculated for the read data did not match the parity read. Parity generation and checking is enabled for memory operations by setting EBC0_BnAP[PEN] = 1 and for DMA peripheral transfers by programming DMA0_CRn[PCE] = 1.

27.5.5.5 Error Locking

EBC0_CFG[HFE]controls the locking of error information in the EBC0_BESR and EBC0_BEAR.

- If EBC0_CFG[HFE] = 0, the contents of the EBC0_BEAR and the EBC0_BESR reflect the most recent error.
- If EBC0_CFG[HFE] = 1, only the first error detected is reported in the EBC0_BESR and EBC0_BEAR. Subsequent errors are not permitted to overwrite the information detailing the first error. When software processes an error, it must clear all EBC0_BESR bits by writing 0 to allow another error to be logged.

27.5.5.6 Peripheral Bus Error Address Register (EBC0_BEAR)

The Peripheral Bus Error Address Register (EBC0_BEAR) is a 32-bit register containing the address of the access where a data bus error occurred. If EBC0_CFG[HFE] = 1, enabling error locking, and the EBC0_BEAR is not already locked, the contents of EBC0_BEAR are locked until the Peripheral Bus Error Status Register (EBC0_BESR) is cleared. The contents of the EBC0_BEAR are accessed indirectly through EBC0_CFGADDR and EBC0_CFGDATA registers using the **mfdcr** and **mtdcr** instructions.

Precise address capture in the EBC0_BEAR when a parity error occurs only applies to banks with at least one cycle of hold time (EBC0_BnAP[TH] > 0). This is because parity errors are not calculated until the cycle after the data was valid on the external bus, and the EBC0_BEAR is loaded with the address on the bus during that cycle. If the device timings do not include at least one hold cycle, the EBC0_BEAR may capture the next address in a burst or the address of the next transaction.

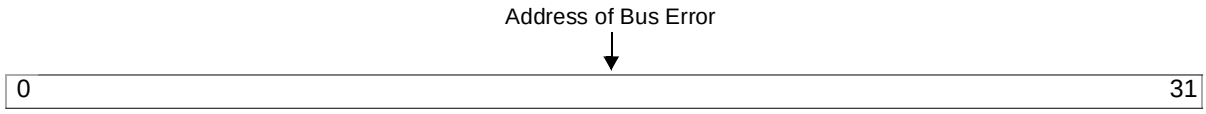


Figure 0-5. Peripheral Bus Error Address Register (EBC0_BEAR)

0:31		Address of Bus Error

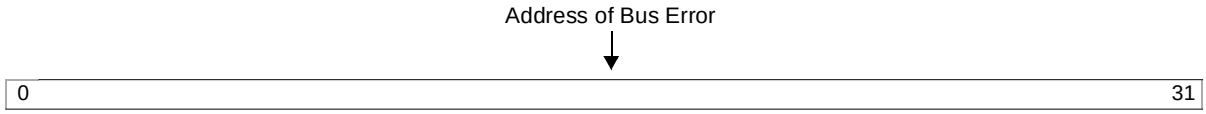


Figure 27-25. Peripheral Bus Error Address Register (EBC0_BEAR)

0:31		Address of Bus Error

27.5.5.7 Peripheral Bus Error Status Register (EBC0_BESR)

The Peripheral Bus Error Status Register (EBC0_BESR) records the occurrence and type of errors for transactions attempted on behalf of the OPB master. The OPB master may be the DMA controller or the PLB-to-OPB Bridge on behalf of the CPU, PCIX, or any other master. The contents of EBC0_BESR are accessed indirectly through EBC0_CFGADDR and EBC0_CFGDATA registers using the **mfdcr** and **mtdcr** instructions.

It is possible to have both a parity error and a bus error during the same data transfer. If this occurs, the bus error is detected first and EBC0_BESR and EBC0_BEAR are updated. In the next cycle after the parity error is detected, and, if error locking is not enabled, the error is logged in EBC0_BESR (Refer to ~~Section 27.5.5.5, Error Locking~~*Error Locking* on page 885).

When an external bus input error is detected during an EBC read operation, the EBC0_BESR[EBE] bit is set. If the error is detected during a DMA read operation, the DMA controller logs the error in its status register and generates an interrupt, if interrupts are enabled. If the error is detected during a PLB master read operation, the error is recorded in the appropriate master's POB0_GESR0 field on the PLB-to-OPB Bridge, and the master is signalled that an error occurred.

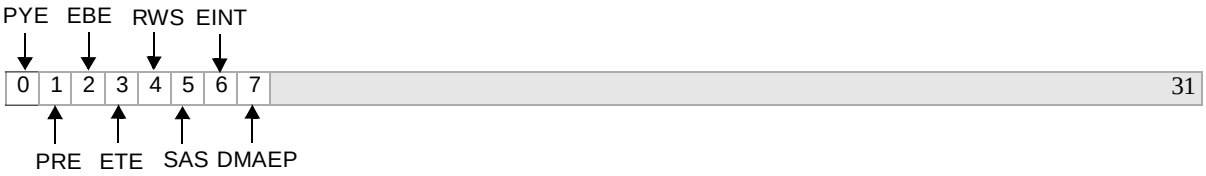


Figure 0-6. Bus Error Status Register (EBC0_BESR)

0	PYE	Parity Error Detected on Read 0 No Parity Error Detected 1 Parity Error Detected	O_ebco_opb_errAck is asserted.
1	PRE	Bank Protection Error 0 No Protection Error 1 Protection Error	O_ebco_opb_errAck is asserted for read or write.
2	EBE	PerErr 0 No PerErr Detected during EBCO transaction. 1 PerErr Detected during EBCO transaction.	If PerErr is detected at any other time during a read, errAck is not asserted. ErrAck is not asserted for write operations.
3	ETE	External Bus Timeout Error 0 No External Bus Timeout Error Detected 1 External Bus Timeout Error Detected	See PDT bit in Figure 27.5.6 on page 36 for timeout enable information.
4	RWS	Read/write Error Status for 0 Errant operation was a write operation 1 Errant operation was a read operation	
5	SAS	Sequential Address status 0 Sequential Address Not Active during error detection 1 Sequential Address Active during error detection	
6	EINT	Error Interrupt This bit is a logical "OR" of the four different errors that may be reported and indicates an interrupt has been asserted to the Interrupt Controller	
7	DMAEP	DMA External Peripheral Error 0 No error 1 Error	DMA external peripheral error detected during EBCO transaction.
8:31		Reserved	

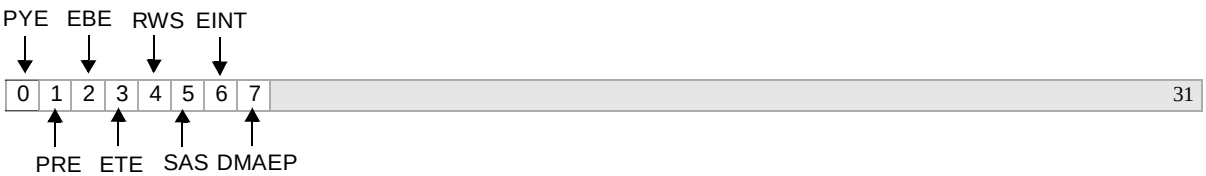


Figure 27-26. Bus Error Status Register (EBC0_BESR)

0	PYE	Parity Error Detected on Read 0 No Parity Error Detected 1 Parity Error Detected	O_ebco_opb_errAck is asserted.
1	PRE	Bank Protection Error 0 No Protection Error 1 Protection Error	O_ebco_opb_errAck is asserted for read or write.
2	EBE	PerErr 0 No PerErr Detected during EBCO transaction. 1 PerErr Detected during EBCO transaction.	If PerErr is detected at any other time during a read, errAck is not asserted. ErrAck is not asserted for write operations.
3	ETE	External Bus Timeout Error 0 No External Bus Timeout Error Detected 1 External Bus Timeout Error Detected	See PDT bit in *Figure 27.5.6* on page 887 for time-out enable information.
4	RWS	Read/write Error Status for 0 Errant operation was a write operation 1 Errant operation was a read operation	
5	SAS	Sequential Address status 0 Sequential Address Not Active during error detection 1 Sequential Address Active during error detection	
6	EINT	Error Interrupt This bit is a logical "OR" of the four different errors that may be reported and indicates an interrupt has been asserted to the Interrupt Controller	
7	DMAEP	DMA External Peripheral Error 0 No error 1 Error	DMA external peripheral error detected during EBCO transaction.
8:31		Reserved	

27.5.6 EBC Configuration Register (EBC0_CFG)

The contents of EBC0_CFG are accessed indirectly through the EBC0_CFGADDR and EBC0_CFGDATA registers using the **mfdcr** and **mtdcr** instructions.

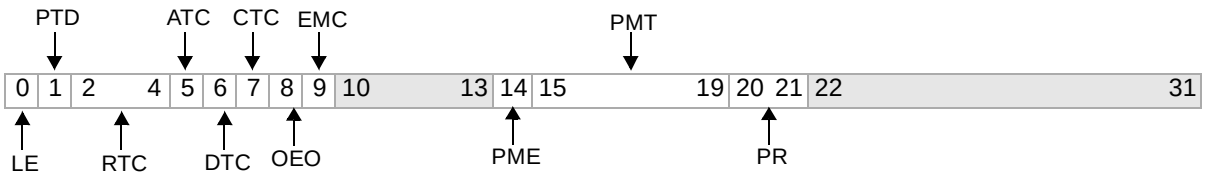


Figure 0-7. EBC Configuration Register (EBC0_CFG)

0	LE	Lock Error 0 Do not lock error 1 Lock error	This bit is used for error locking in the EBC. If this bit is set, then the EBC will latch the first error detected and save it with associated parameters and address. If this bit is not set, then the EBC will continue to latch new errors and associated parameters and addresses that will overwrite previous errors.
1	PTD	Device-Paced Time-out Disable 0 Enabled time-outs 1 Disable time-outs	This bit is valid only when EBC0_BnAP[RE]=1. If PTD=1, the EBC waits indefinitely for assertion of PerReady during device-paced accesses.
2:4	RTC	Ready Timeout Count 000 16 PerClk cycles 001 32 PerClk cycles 010 64 PerClk cycles 011 128 PerClk cycles 100 256 PerClk cycles 101 512 PerClk cycles 110 1024 PerClk cycles 111 2048 PerClk cycles	When PTD=0, the number of clock cycles from the assertion of PerAddr0:31 until a time-out error occurs.
5	ATC	Address Bus High Impedance Control 0 External address bus is high impedance when EBC is idle 1 External address bus drive previous value when EBC is idle and has ownership of the peripheral interface	Default after reset is ATC=1. For details on how the ATC affects driver enables, See "Driver Enables" on page 27-4.
6	DTC	Data Bus High Impedance Control 0 External data bus is high impedance when EBC is idle 1 External data bus drive previous write data value when EBC is idle and has ownership of the peripheral interface	Default after reset is DTC=1. For details on how the DTC affects driver enables, See "Driver Enables" on page 27-4.
7	CTC	Control Signal High Impedance Control 0 External bus Control Signals are high impedance when EBC is idle 1 External bus Control Signals are driven inactive and held when EBC is idle and has ownership of the peripheral interface	Default after reset is CTC=1. For details on how the CTC affects driver enables, See "Driver Enables" on page 27-4.

8	OEO	External Bus Override High Impedance Control 0 External Bus Output Enable Override Disabled 1 External Bus Output Enable Override Enabled	Default after reset is OEO=1. For details on how the OEO affects driver enables, See "Driver Enables" on page 27-4.
9	EMC	External Master High Impedance Control 0 High impedance all EBC outputs when HOLD_ACK = 1. 1 High impedance all EBC outputs when HOLD_ACK = 1 except PerCS0:7 are always driven.	Default after reset is EMC=1. For details, See "Driver Enables" on page 27-4.
10:13		Reserved	
14	PME	Power Management Enable 0 Disabled 1 Enabled	
15:19	PMT	Power Management Timer 0-31	The EBC makes a sleep request to the Clock and Power Management unit when PME=1 and the EBC has been idle for 32*PMT PerClk cycles.
20:21	PR	Pending Request Timer 00 -16 01 - 32 10 - 64 11 - 128	Number of PerClk cycles to wait for a retried OPB operation[1] to return before relinquishing external bus ownership back to the external master.
22:31		Reserved	

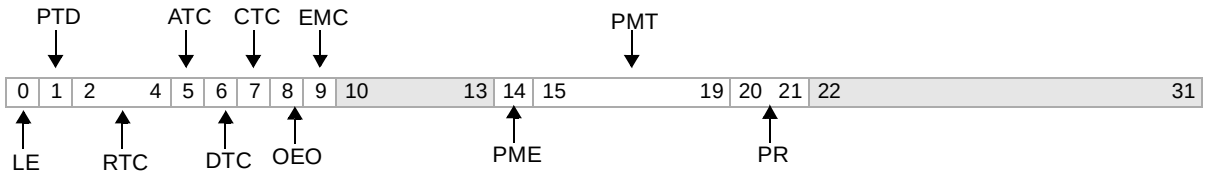


Figure 27-27. EBC Configuration Register (EBC0_CFG)

0	LE	Lock Error 0 Do not lock error 1 Lock error	This bit is used for error locking in the EBC. If this bit is set, then the EBC will latch the first error detected and save it with associated parameters and address. If this bit is not set, then the EBC will continue to latch new errors and associated parameters and addresses that will overwrite previous errors.
1	PTD	Device-Paced Time-out Disable 0 Enabled time-outs 1 Disable time-outs	This bit is valid only when EBC0_BnAP[RE]=1. If PTD=1, the EBC waits indefinitely for assertion of PerReady during device-paced accesses.

2:4	RTC	Ready Timeout Count 000 16 PerClk cycles 001 32 PerClk cycles 010 64 PerClk cycles 011 128 PerClk cycles 100 256 PerClk cycles 101 512 PerClk cycles 110 1024 PerClk cycles 111 2048 PerClk cycles	When PTD=0, the number of clock cycles from the assertion of PerAddr0:31 until a time-out error occurs.
5	ATC	Address Bus High Impedance Control 0 External address bus is high impedance when EBC is idle 1 External address bus drive previous value when EBC is idle and has ownership of the peripheral interface	Default after reset is ATC=1. For details on how the ATC affects driver enables, See *Driver Enables* on page 857.
6	DTC	Data Bus High Impedance Control 0 External data bus is high impedance when EBC is idle 1 External data bus drive previous write data value when EBC is idle and has ownership of the peripheral interface	Default after reset is DTC=1. For details on how the DTC affects driver enables, See *Driver Enables* on page 857.
7	CTC	Control Signal High Impedance Control 0 External bus Control Signals are high impedance when EBC is idle 1 External bus Control Signals are driven inactive and held when EBC is idle and has ownership of the peripheral interface	Default after reset is CTC=1. For details on how the CTC affects driver enables, See *Driver Enables* on page 857.
8	OEO	External Bus Override High Impedance Control 0 External Bus Output Enable Override Disabled 1 External Bus Output Enable Override Enabled	Default after reset is OEO=1. For details on how the OEO affects driver enables, See *Driver Enables* on page 857.
9	EMC	External Master High Impedance Control 0 High impedance all EBC outputs when HOLD_ACK = 1. 1 High impedance all EBC outputs when HOLD_ACK = 1 except PerCS0:7 are always driven.	Default after reset is EMC=1. For details, See *Driver Enables* on page 857.
10:13		Reserved	
14	PME	Power Management Enable 0 Disabled 1 Enabled	
15:19	PMT	Power Management Timer 0-31	The EBC makes a sleep request to the Clock and Power Management unit when PME=1 and the EBC has been idle for 32*PMT PerClk cycles.
20:21	PR	Pending Request Timer 00 -16 01 - 32 10 - 64 11 - 128	Number of PerClk cycles to wait for a retried OPB operation[1] to return before relinquishing external bus ownership back to the external master.
22:31		Reserved	

1. A retried OPB operation occurs as follows:

The EBMI owns the external bus and there is an OPB requested EBCO transfer. The EBCO retries that request. When the EBMI releases the external bus, the EBCO owns it again. The EBCO has several previously retried OPB operations to complete; however, some of the retried OPB operations have not returned and, at the same time, no other OPB requests target the EBCO.

The EBCO enters the "Pending request" state. To detect this condition, the EBCO has a programmable 8-bit timer (PR). The PR loads with its programmed initialized value (16, 32, 64, 128) and begin decrementing each OPB cycle. When the PR reaches zero, the EBCO exits the "Pending Request" state and stops waiting for OPB commands to return.

27.5.7 EBC Core ID Register (EBC0_CID)

This register indicates the core ID. The core ID includes the technology, core number, and library revision number for this core. Writes to this register will be ignored.

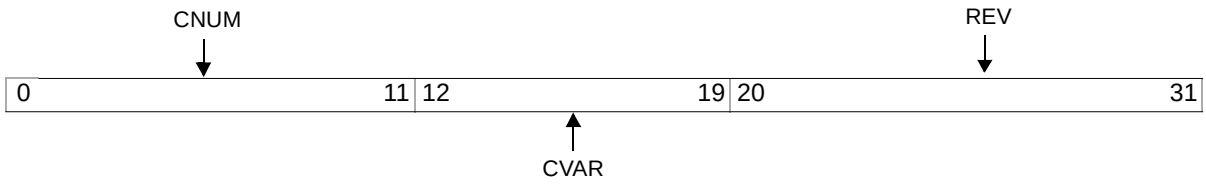


Figure 0-8. EBCO Core ID Register 0 (EBC0_CIDn)

0:11	CNUM	Core Number	'324
12:19	CVAR	Core Variation	'01'
20:31	REV	Revision Number	This value is the IBM SCCS revision number tracked and modified by the core designer.

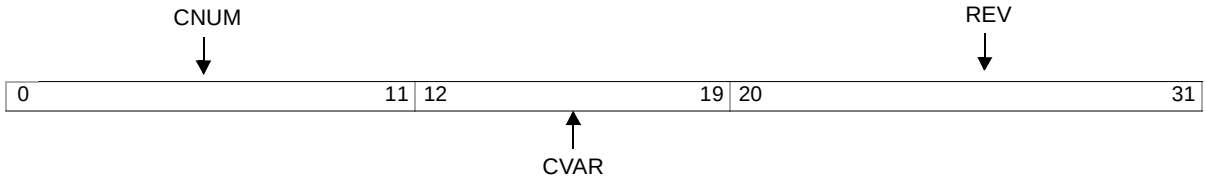


Figure 27-28. EBCO Core ID Register 0 (EBC0_CIDn)

0:11	CNUM	Core Number	'324
12:19	CVAR	Core Variation	'01'
20:31	REV	Revision Number	This value is the IBM SCCS revision number tracked and modified by the core designer.

Part V. Reference

28. Instruction Set

Descriptions of the PPC440GP instructions follow. Each description contains the following elements:

- Instruction names (mnemonic and full)
- Instruction syntax
- Instruction format diagram
- Pseudocode description
- Prose description
- Registers altered

Where appropriate, instruction descriptions list invalid instruction forms and exceptions, and provide programming notes.

~~Table 28-1~~*Table 28-1* summarizes the PPC440GP instruction set by category.

Table 0-1. Instruction Categories

Category	Sub-Category	Instruction Types
Integer	Integer Storage Access	load, store
	Integer Arithmetic	add, subtract, multiply, divide, negate
	Integer Logical	and, andc, or, orc, xor, nand, nor, xnor, extend sign, count leading zeros
	Integer Compare	compare, compare logical
	Integer Trap	trap
	Integer Rotate	rotate and insert, rotate and mask
	Integer Shift	shift left, shift right, shift right algebraic
Branch		branch, branch conditional, branch to link, branch to count
Processor Control	Condition Register Logical	crand, crandc, cror, crorc, crnand, crnor, crxor, crxnor
	Register Management	move to/from SPR, move to/from DCR, move to/from MSR, write to external interrupt enable bit, move to/from CR
	System Linkage	system call, return from interrupt, return from critical interrupt
	Processor Synchronization	instruction synchronize
Storage Control	Cache Management	data allocate, data invalidate, data touch, data zero, data flush, data store, instruction invalidate, instruction touch
	TLB Management	read, write, search, synchronize
	Storage Synchronization	memory synchronize, memory barrier

Table 0-1. Instruction Categories (continued)

Category	Sub-Category	Instruction Types
Allocated	Allocated Arithmetic	multiply-accumulate, negative multiply-accumulate, multiply halfword
	Allocated Logical	detect left-most zero byte
	Allocated Cache Management	data congruence-class invalidate, instruction congruence-class invalidate
	Allocated Cache Debug	data read, instruction read

Table 28-1. Instruction Categories

Category	Sub-Category	Instruction Types
Integer	Integer Storage Access	load, store
	Integer Arithmetic	add, subtract, multiply, divide, negate
	Integer Logical	and, andc, or, orc, xor, nand, nor, xnor, extend sign, count leading zeros
	Integer Compare	compare, compare logical
	Integer Trap	trap
	Integer Rotate	rotate and insert, rotate and mask
	Integer Shift	shift left, shift right, shift right algebraic
Branch		branch, branch conditional, branch to link, branch to count
Processor Control	Condition Register Logical	crand, crandc, cror, crorc, crnand, crnor, crxor, crxnor
	Register Management	move to/from SPR, move to/from DCR, move to/from MSR, write to external interrupt enable bit, move to/from CR
	System Linkage	system call, return from interrupt, return from critical interrupt
	Processor Synchronization	instruction synchronize
Storage Control	Cache Management	data allocate, data invalidate, data touch, data zero, data flush, data store, instruction invalidate, instruction touch
	TLB Management	read, write, search, synchronize
	Storage Synchronization	memory synchronize, memory barrier
Allocated	Allocated Arithmetic	multiply-accumulate, negative multiply-accumulate, multiply halfword
	Allocated Logical	detect left-most zero byte
	Allocated Cache Management	data congruence-class invalidate, instruction congruence-class invalidate
	Allocated Cache Debug	data read, instruction read

28.1 Instruction Set Portability

To support embedded real-time applications, the PPC440GP implements the defined instruction set of the Book-E Enhanced PowerPC Architecture, with the exception of those operations which are defined for 64-bit implementations only, and those which are defined as floating-point operations. Support for the floating-point

operations is provided via the auxiliary processor interface, while the 64-bit operations are not supported at all. See ~~"Instruction Classes" on page 4-35~~*Instruction Classes* on page 175 for more information on the support for defined instructions within the PPC440GP.

The PPC440GP also implements a number of instructions that are not part of PowerPC Book-E architecture, but are included as part of the PPC440GP. Architecturally, they are considered allocated instructions, as they use opcodes which are within the allocated class of instructions, which the PowerPC Book-E architecture identifies as being available for implementation-dependent and/or application-specific purposes. However, all of the allocated instructions which are implemented within the PPC440GP are "standard" for IBM PowerPC 400 Series family of embedded controllers, and are not unique to the PPC440GP.

The allocated instructions implemented within the PPC440GP are divided into four sub-categories, and are shown in ~~Table 28-2~~*Table 28-2*. Programs using these instructions may not be portable to other PowerPC Book-E implementations.

Table 28-2. Allocated Instructions

Arithmetic			Logical	Cache Management	Cache Debug
Multiply-Accumulate	Negative Multiply-Accumulate	Multiply Halfword			
macchw[**o**][**.**] **macchws**[**o**][**.**] **macchwsu**[**o**][**.**] **macchwu**[**o**][**.**] **machhw**[**o**][**.**] **machhws**[**o**][**.**] **machhwsu**[**o**][**.**] **machhwu**[**o**][**.**] **maclhw**[**o**][**.**] **maclhws**[**o**][**.**] **maclhwsu**[**o**][**.**] **maclhwu**[**o**][**.**]	**nmacchw**[**o**][**.**] **nmacchws**[**o**][**.**] **nmachhw**[**o**][**.**] **nmachhws**[**o**][**.**] **nmaclhw**[**o**][**.**] **nmaclhws**[**o**][**.**]	**mulchw**[**.**] **mulchwu**[**.**] **mulhhw**[**.**] **mulhhwu**[**.**] **mullhw**[**.**] **mullhwu**[**.**]	**dlmzb**[**.**]	**dccci** **iccci**	**dcread** **icread**

28.2 Instruction Formats

For more detailed information about instruction formats, including a summary of instruction field usage and instruction format diagrams for the PPC440GP, see ~~"Instruction Formats" on page A-1~~*Instruction Formats* on page 1539.

Instructions are four bytes long. Instruction addresses are always word-aligned.

Instruction bits 0 through 5 always contain the primary opcode. Many instructions have an extended opcode field as well. The remaining instruction bits contain additional fields. All instruction fields belong to one of the following categories:

- Defined

  These instruction fields contain values, such as opcodes, that cannot be altered. The instruction format diagrams specify the values of defined fields.

- Variable

These fields contain operands, such as general purpose register specifiers and immediate values, each of which may contain any one of a number of values. The instruction format diagrams specify the field names of variable fields.

- Reserved

  Bits in a reserved field should be set to 0. In the instruction format diagrams, reserved fields are shaded.

If any bit in a defined field does not contain the specified value, the instruction is illegal and an Illegal Instruction exception type Program interrupt occurs. If any bit in a reserved field does not contain 0, the instruction form is invalid and its result is architecturally undefined. Unless otherwise noted, the PPC440GP will execute all invalid instruction forms without causing an Illegal Instruction exception.

28.3 Pseudocode

The pseudocode that appears in the instruction descriptions provides a semi-formal language for describing instruction operations.

The pseudocode uses the following notation:

+	Twos complement addition
%	Remainder of an integer division; (33 % 32) = 1.
$<^u$, $>^u$	Unsigned comparison relations
(GPR(r))	The contents of GPR r, where $0 \le r \le 31$.
(RA|0)	The contents of the register RA or 0, if the RA field is 0.
(Rx)	The contents of a GPR, where *x* is A, B, S, or T
0bn	A binary number
0xn	A hexadecimal number
<, >	Signed comparison relations
=	Assignment
=, ≠	Equal, not equal relations
CEIL(x)	Least integer ≥ x.
CIA	Current instruction address; the 32-bit address of the instruction being described by a sequence of pseudocode. This address is used to set the next instruction address (NIA). Does not correspond to any architected register.
DCR(DCRN)	A Device Control Register (DCR) specified by the DCRF field in an **mfdcr** or **mtdcr** instruction
EA	Effective address; the 32-bit address, derived by applying indexing or indirect addressing rules to the specified operand, that specifies an location in main storage.
EXTS(x)	The result of extending *x* on the left with sign bits.

FLD	An instruction or register field
FLD_b	A bit in a named instruction or register field
$FLD_{b,b,\ldots}$	A list of bits, by number or name, in a named instruction or register field
$FLD_{b:b}$	A range of bits in a named instruction or register field
GPR(r)	General Purpose Register (GPR) r, where $0 \le r \le 31$.
GPRs	RA, RB, . . .
MASK(MB,ME)	Mask having 1s in positions MB through ME (wrapping if MB > ME) and 0s elsewhere.
MS(addr, n)	The number of bytes represented by *n* at the location in main storage represented by *addr*.
NIA	Next instruction address; the 32-bit address of the next instruction to be executed. In pseudocode, a successful branch is indicated by assigning a value to NIA. For instructions that do not branch, the NIA is CIA +4.
PC	Program counter.
REG[FLD, FLD . . .]	A list of fields in a named register
REG[FLD:FLD]	A range of fields in a named register
REG[FLD]	A field in a named register
REG_b	A bit in a named register
$REG_{b,b,\ldots}$	A list of bits, by number or name, in a named register
$REG_{b:b}$	A range of bits in a named register
RESERVE	Reserve bit; indicates whether a process has reserved a block of storage.
ROTL((RS),n)	Rotate left; the contents of RS are shifted left the number of bits specified by *n*.
SPR(SPRN)	A Special Purpose Register (SPR) specified by the SPRF field in an **mfspr** or **mtspr** instruction
$c_{0:3}$	A four-bit object used to store condition results in compare instructions.
do	Do loop. "to" and "by" clauses specify incrementing an iteration variable; "while" and "until" clauses specify terminating conditions. Indenting indicates the scope of a loop.
if...then...else...	Conditional execution; if *condition* then *a* else *b*, where *a* and *b* represent one or more pseudocode statements. Indenting indicates the ranges of *a* and *b*. If *b* is null, the else does not appear.
instruction(EA)	An instruction operating on a data or instruction cache block associated with an EA.
leave	Leave innermost do loop or do loop specified in a leave statement.
n	A decimal number

^{n}b	The bit or bit value *b* is replicated *n* times.
xx	Bit positions which are don't-cares.
||	Concatenation
×	Multiplication
÷	Division yielding a quotient
⊕	Exclusive-OR (XOR) logical operator
–	Twos complement subtraction, unary minus
¬	NOT logical operator
∧	AND logical operator
∨	OR logical operator

28.3.1 Operator Precedence

Table 28-3 lists the pseudocode operators and their associativity in descending order of precedence:

Table 28-3. Operator Precedence

Operators	Associativity
REG_b, REG[FLD], function evaluation	Left to right
^{n}b	Right to left
¬, – (unary minus)	Right to left
×, ÷	Left to right
+, –	Left to right
||	Left to right
=, ≠, <, >, $\overset{u}{<}$, $\overset{u}{>}$	Left to right
∧, ⊕	Left to right
∨	Left to right
←	None

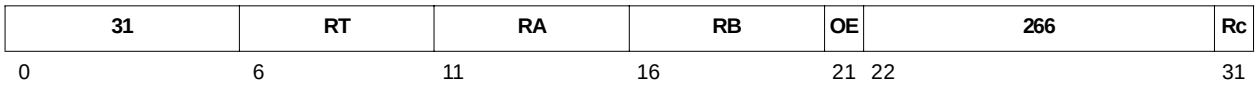
28.4 Register Usage

Each instruction description lists the registers altered by the instruction. Some register changes are explicitly detailed in the instruction description (for example, the target register of a load instruction). Some instructions also change other registers, but the details of the changes are not included in the instruction descriptions. Common examples of these kinds of register changes include the Condition Register (CR) and the Integer Exception Register (XER). For discussion of the CR, see ~~"Condition Register (CR)" on page 4-50~~*Condition Register (CR)* on page 189. For discussion of the XER, see ~~"Integer Exception Register (XER)" on page 4-53~~*Integer Exception Register (XER)* on page 192.

28.5 Alphabetical Instruction Listing

The following pages list the instructions, both defined and allocated, which are implemented within the PPC440GP.

add	RT, RA, RB	OE = 0, Rc = 0
add.	RT, RA, RB	OE = 0, Rc = 1
addo	RT, RA, RB	OE = 1, Rc = 0
addo.	RT, RA, RB	OE = 1, Rc = 1



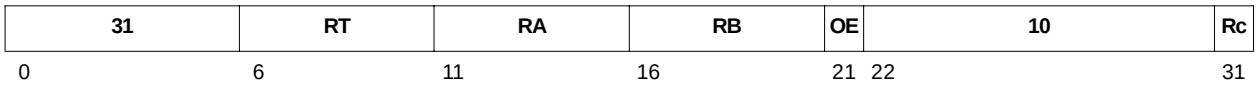
(RT) ← (RA) + (RB)

The sum of the contents of register RA and the contents of register RB is placed into register RT.

Registers Altered

- RT
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE contains 1

addc	RT, RA, RB	OE = 0, Rc = 0
addc.	RT, RA, RB	OE = 0, Rc = 1
addco	RT, RA, RB	OE = 1, Rc = 0
addco.	RT, RA, RB	OE = 1, Rc = 1



(RT) $\leftarrow$ (RA) + (RB)
if (RA) + (RB) $\overset{u}{\geq} 2^{32} - 1$ then
  XER[CA] $\leftarrow$ 1
else
  XER[CA] $\leftarrow$ 0

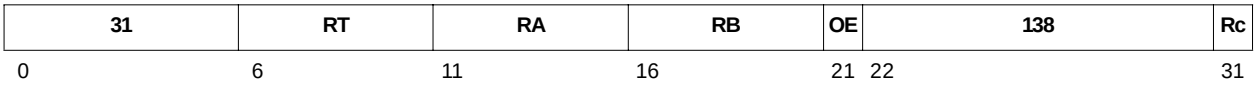
The sum of the contents of register RA and register RB is placed into register RT.

XER[CA] is set to a value determined by the unsigned magnitude of the result of the add operation.

Registers Altered

- RT
- XER[CA]
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE contains 1

adde	RT, RA, RB	OE = 0, Rc = 0
adde.	RT, RA, RB	OE = 0, Rc = 1
addeo	RT, RA, RB	OE = 1, Rc = 0
addeo.	RT, RA, RB	OE = 1, Rc = 1



(RT) ← (RA) + (RB) + XER[CA]
if (RA) + (RB) + XER[CA] $\overset{u}{\geq} 2^{32} - 1$ then
    XER[CA] ← 1
else
    XER[CA] ← 0

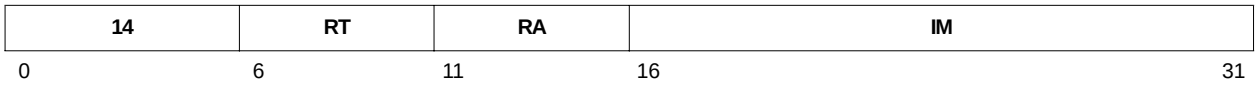
The sum of the contents of register RA, register RB, and XER[CA] is placed into register RT.

XER[CA] is set to a value determined by the unsigned magnitude of the result of the add operation.

Registers Altered

- RT
- XER[CA]
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE contains 1

addi RT, RA, IM



(RT) ← (RA|0) + EXTS(IM)

If the RA field is 0, the IM field, sign-extended to 32 bits, is placed into register RT.

If the RA field is nonzero, the sum of the contents of register RA and the contents of the IM field, sign-extended to 32 bits, is placed into register RT.

Registers Altered

- RT

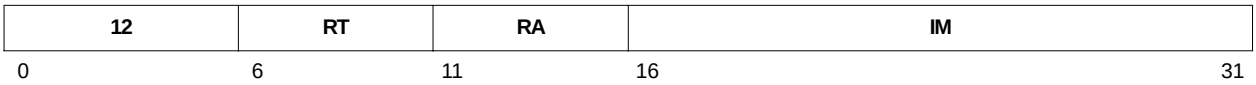
Programming Note

To place an immediate, sign-extended value into the GPR specified by RT, set RA = 0.

Table 28-4. Extended Mnemonics for addi

Mnemonic	Operands	Function	Other Registers Altered
la	RT, D(RA)	Load address (RA ≠ 0); D is an offset from a base address that is assumed to be (RA). (RT) ← (RA) + EXTS(D) *Extended mnemonic for* **addi RT,RA,D**	
li	RT, IM	Load immediate. (RT) ← EXTS(IM) *Extended mnemonic for* **addi RT,0,IM**	
subi	RT, RA, IM	Subtract EXTS(IM) from (RA|0). Place result in RT. *Extended mnemonic for* **addi RT,RA,–IM**	

addic RT, RA, IM



(RT) ← (RA) + EXTS(IM)
if (RA) + EXTS(IM) $\overset{u}{\geq} 2^{32} - 1$ then
  XER[CA] ← 1
else
  XER[CA] ← 0

The sum of the contents of register RA and the contents of the IM field, sign-extended to 32 bits, is placed into register RT.

XER[CA] is set to a value determined by the unsigned magnitude of the result of the add operation.

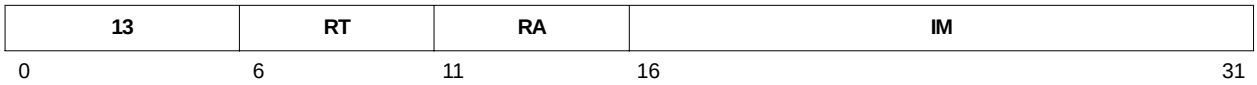
Registers Altered

- RT
- XER[CA]

Table 28-5. Extended Mnemonics for addic

Mnemonic	Operands	Function	Other Registers Altered
subic	RT, RA, IM	Subtract EXTS(IM) from (RA) Place result in RT; place carry-out in XER[CA]. *Extended mnemonic for* **addic RT,RA,–IM**	

addic. RT, RA, IM



(RT) ← (RA) + EXTS(IM)
if (RA) + EXTS(IM) $\geq^{u} 2^{32} - 1$ then
  XER[CA] ← 1
else
  XER[CA] ← 0

The sum of the contents of register RA and the contents of the IM field, sign-extended to 32 bits, is placed into register RT.

XER[CA] is set to a value determined by the unsigned magnitude of the result of the add operation.

Registers Altered

- RT
- XER[CA]
- CR[CR0]

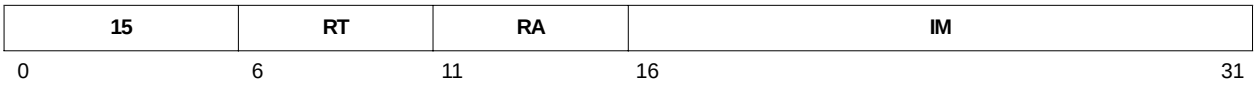
Programming Note

addic. is one of three instructions that implicitly update CR[CR0] without having an RC field. The other instructions are **andi.** and **andis.**.

Table 28-6. Extended Mnemonics for addic.

Mnemonic	Operands	Function	Other Registers Altered
subic.	RT, RA, IM	Subtract EXTS(IM) from (RA). Place result in RT; place carry-out in XER[CA]. *Extended mnemonic for* **addic. RT,RA,–IM**	CR[CR0]

addis RT, RA, IM



(RT) ← (RA|0) + (IM || $^{16}0$)

If the RA field is 0, the IM field is concatenated on its right with sixteen 0-bits and placed into register RT.

If the RA field is nonzero, the contents of register RA are added to the contents of the extended IM field. The sum is stored into register RT.

Registers Altered

- RT

Programming Note

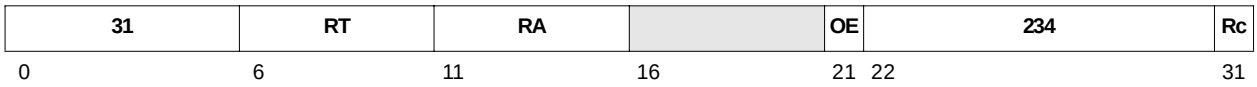
An **addi** instruction stores a sign-extended 16-bit value in a GPR. An **addis** instruction followed by an **ori** instruction stores an arbitrary 32-bit value in a GPR, as shown in the following example:

```
addis    RT, 0, high 16 bits of value
ori      RT, RT, low 16 bits of value
```

Table 28-7. Extended Mnemonics for addis

Mnemonic	Operands	Function	Other Registers Altered
lis	RT, IM	Load immediate shifted. (RT) ← (IM || $^{16}0$) *Extended mnemonic for* **addis RT,0,IM**	
subis	RT, RA, IM	Subtract (IM || $^{16}0$) from (RA|0). Place result in RT. *Extended mnemonic for* **addis RT,RA,–IM**	

addme	RT, RA	OE = 0, Rc = 0
addme.	RT, RA	OE = 0, Rc = 1
addmeo	RT, RA	OE = 1, Rc = 0
addmeo.	RT, RA	OE = 1, Rc = 1



(RT) ← (RA) + XER[CA] + (–1)
if (RA) + XER[CA] + 0xFFFF FFFF $\overset{u}{\geq}$ $2^{32} - 1$ then
  XER[CA] ← 1
else
  XER[CA] ← 0

The sum of the contents of register RA, XER[CA], and –1 is placed into register RT.

XER[CA] is set to a value determined by the unsigned magnitude of the result of the add operation.

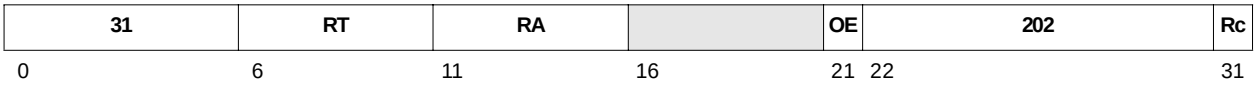
Registers Altered

- RT
- XER[CA]
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE contains 1

Invalid Instruction Forms

- Reserved fields

addze	RT, RA	OE=0, Rc=0
addze.	RT, RA	OE=0, Rc=1
addzeo	RT, RA	OE=1, Rc=0
addzeo.	RT, RA	OE=1, Rc=1



(RT) $\leftarrow$ (RA) + XER[CA]
if (RA) + XER[CA] $\geq^{u}$ $2^{32} - 1$ then
    XER[CA] $\leftarrow$ 1
else
    XER[CA] $\leftarrow$ 0

The sum of the contents of register RA and XER[CA] is placed into register RT.

XER[CA] is set to a value determined by the unsigned magnitude of the result of the add operation.

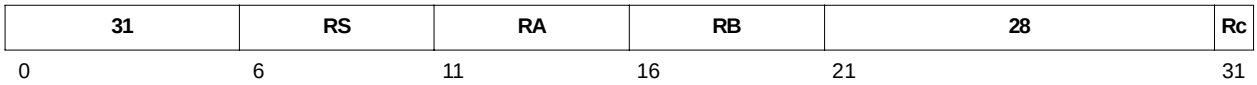
Registers Altered

- RT
- XER[CA]
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE contains 1

Invalid Instruction Forms

- Reserved fields

and	RA, RS, RB	Rc=0
and.	RA, RS, RB	Rc=1



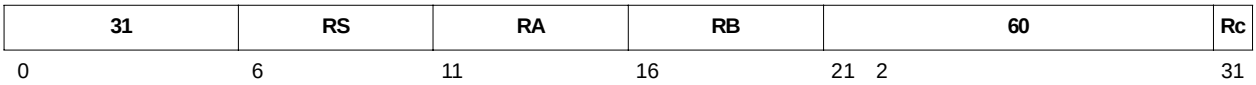
(RA) $\leftarrow$ (RS) $\wedge$ (RB)

The contents of register RS are ANDed with the contents of register RB; the result is placed into register RA.

Registers Altered

- RA
- CR[CR0] if Rc contains 1

andc	RA,RS,RB	Rc=0
andc.	RA,RS,RB	Rc=1



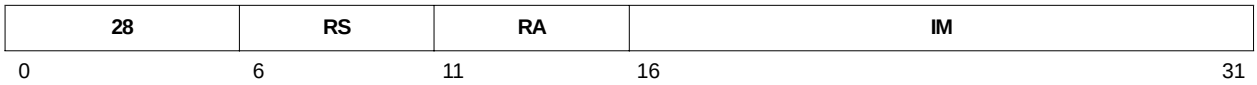
(RA) $\leftarrow$ (RS) $\wedge$ $\neg$(RB)

The contents of register RS are ANDed with the ones complement of the contents of register RB; the result is placed into register RA.

Registers Altered

- RA
- CR[CR0] if Rc contains 1

andi. RA, RS, IM



(RA) $\leftarrow$ (RS) $\wedge$ ($^{16}0$ || IM)

The IM field is extended to 32 bits by concatenating 16 0-bits on its left. The contents of register RS is ANDed with the extended IM field; the result is placed into register RA.

Registers Altered

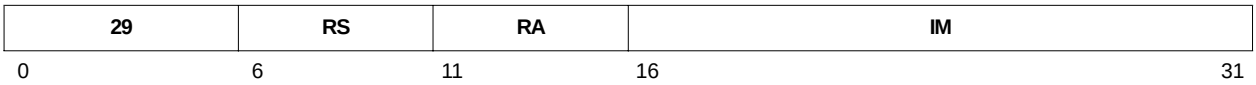
- RA
- CR[CR0]

Programming Note

The **andi.** instruction can test whether any of the 16 least-significant bits in a GPR are 1-bits.

andi. is one of three instructions that implicitly update CR[CR0] without having an Rc field. The other instructions are **addic.** and **andis.**.

andis. RA, RS, IM



(RA) $\leftarrow$ (RS) $\wedge$ (IM || $^{16}0$)

The IM field is extended to 32 bits by concatenating 16 0-bits on its right. The contents of register RS are ANDed with the extended IM field; the result is placed into register RA.

Registers Altered

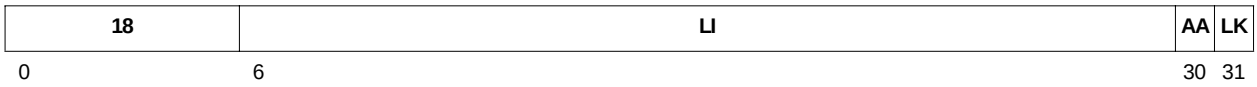
- RA
- CR[CR0]

Programming Note

The **andis.** instruction can test whether any of the 16 most-significant bits in a GPR are 1-bits.

andis. is one of three instructions that implicitly update CR[CR0] without having an Rc field. The other instructions are **addic.** and **andi.**.

b	target	AA=0, LK=0
ba	target	AA=1, LK=0
bl	target	AA=0, LK=1
bla	target	AA=1, LK=1




```
If AA = 1 then
   LI ← target6:29
   NIA ← EXTS(LI || 20)
else
   LI ← (target – CIA)6:29
   NIA ← CIA + EXTS(LI || 20)
if LK = 1 then
   (LR) ← CIA + 4
PC ← NIA
```

The next instruction address (NIA) is the effective address of the branch target. The NIA is formed by adding a displacement to a base address. The displacement is obtained by concatenating two 0-bits to the right of the LI field and sign-extending the result to 32 bits.

If the AA field contains 0, the base address is the address of the branch instruction, which is the current instruction address (CIA). If the AA field contains 1, the base address is 0.

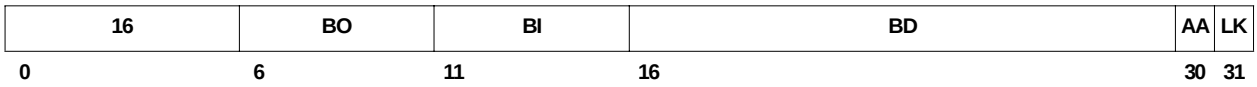
Instruction execution resumes with the instruction at the NIA.

If the LK field contains 1, then (CIA + 4) is placed into the LR.

Registers Altered

- LR if LK contains 1

bc	BO, BI, target	AA=0, LK=0
bca	BO, BI, target	AA=1, LK=0
bcl	BO, BI, target	AA=0, LK=1
bcla	BO, BI, target	AA=1, LK=1




```
if BO2 = 0 then
   CTR ← CTR – 1
if (BO2 = 1 ∨ ((CTR = 0) = BO3)) ∧ (BO0 = 1 ∨ (CRBI = BO1))  then
   if AA = 1 then
      BD ← target16:29
      NIA ← EXTS(BD || 20)
   else
      BD ← (target – CIA)16:29
      NIA ← CIA + EXTS(BD || 20)
else
   NIA ← CIA + 4
if LK = 1 then
   (LR) ← CIA + 4
PC ← NIA
```

If BO_2 contains 0, the CTR decrements, and the decremented value is tested for 0 as part of the branch condition. In this case, BO_3 indicates whether the test for 0 must be true or false in order for the branch to be taken. If BO_2 contains 1, then the CTR is neither decremented nor tested as part of the branch condition.

If BO_0 contains 0, then the CR bit specified by the BI field is compared to BO_1 as part of the branch condition. If BO_0 contains 1, then the CR is not tested as part of the branch condition, and the BI field is ignored.

The next instruction address (NIA) is either the effective address of the branch target, or the address of the instruction after the branch, depending on whether the branch is taken or not. The branch target address is formed by adding a displacement to a base address. The displacement is obtained by concatenating two 0-bits to the right of the BD field and sign-extending the result to 32 bits.

If the AA field contains 0, the base address is the address of the branch instruction, which is the current instruction address (CIA). If the AA field contains 1, the base address is 0.

BO_4 affects branch prediction, a performance-improvement feature. See ~~"Branch Prediction" on page 4-47~~*Branch Prediction* on page 187 for a complete discussion.

Instruction execution resumes with the instruction at the NIA.

If the LK field contains 1, then (CIA + 4) is placed into the LR.

Registers Altered

- CTR if BO_2 contains 0
- LR if LK contains 1

Table 28-8. Extended Mnemonics for bc, bca, bcl, bcla

Mnemonic	Operands	Function	Other Registers Altered
bdnz	target	Decrement CTR; branch if CTR ≠ 0. *Extended mnemonic for* **bc 16,0,target**	
bdnza		*Extended mnemonic for* **bca 16,0,target**	
bdnzl		*Extended mnemonic for* **bcl 16,0,target**	(LR) ← CIA + 4.
bdnzla		*Extended mnemonic for* **bcla 16,0,target**	(LR) ← CIA + 4.
bdnzf	cr_bit, target	Decrement CTR. Branch if CTR ≠ 0 AND CR_{cr_bit} = 0. *Extended mnemonic for* **bc 0,cr_bit,target**	
bdnzfa		*Extended mnemonic for* **bca 0,cr_bit,target**	
bdnzfl		*Extended mnemonic for* **bcl 0,cr_bit,target**	(LR) ← CIA + 4.
bdnzfla		*Extended mnemonic for* **bcla 0,cr_bit,target**	(LR) ← CIA + 4.
bdnzt	cr_bit, target	Decrement CTR. Branch if CTR ≠ 0 AND CR_{cr_bit} = 1. *Extended mnemonic for* **bc 8,cr_bit,target**	
bdnzta		*Extended mnemonic for* **bca 8,cr_bit,target**	
bdnztl		*Extended mnemonic for* **bcl 8,cr_bit,target**	(LR) ← CIA + 4.
bdnztla		*Extended mnemonic for* **bcla 8,cr_bit,target**	(LR) ← CIA + 4.
bdz	target	Decrement CTR; branch if CTR = 0. *Extended mnemonic for* **bc 18,0,target**	
bdza		*Extended mnemonic for* **bca 18,0,target**	
bdzl		*Extended mnemonic for* **bcl 18,0,target**	(LR) ← CIA + 4.
bdzla		*Extended mnemonic for* **bcla 18,0,target**	(LR) ← CIA + 4.

Table 28-8. Extended Mnemonics for bc, bca, bcl, bcla (continued)

Mnemonic	Operands	Function	Other Registers Altered
bdzf	cr_bit, target	Decrement CTR Branch if CTR = 0 AND CR_{cr_bit} = 0. *Extended mnemonic for* **bc 2,cr_bit,target**	
bdzfa		*Extended mnemonic for* **bca 2,cr_bit,target**	
bdzfl		*Extended mnemonic for* **bcl 2,cr_bit,target**	(LR) ← CIA + 4.
bdzfla		*Extended mnemonic for* **bcla 2,cr_bit,target**	(LR) ← CIA + 4.
bdzt	cr_bit, target	Decrement CTR. Branch if CTR = 0 AND CR_{cr_bit} = 1. *Extended mnemonic for* **bc 10,cr_bit,target**	
bdzta		*Extended mnemonic for* **bca 10,cr_bit,target**	
bdztl		*Extended mnemonic for* **bcl 10,cr_bit,target**	(LR) ← CIA + 4.
bdztla		*Extended mnemonic for* **bcla 10,cr_bit,target**	(LR) ← CIA + 4.
beq	[cr_field,] target	Branch if equal. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bc 12,4∗cr_field+2,target**	
beqa		*Extended mnemonic for* **bca 12,4∗cr_field+2,target**	
beql		*Extended mnemonic for* **bcl 12,4∗cr_field+2,target**	(LR) ← CIA + 4.
beqla		*Extended mnemonic for* **bcla 12,4∗cr_field+2,target**	(LR) ← CIA + 4.
bf	cr_bit, target	Branch if CR_{cr_bit} = 0. *Extended mnemonic for* **bc 4,cr_bit,target**	
bfa		*Extended mnemonic for* **bca 4,cr_bit,target**	
bfl		*Extended mnemonic for* **bcl 4,cr_bit,target**	LR
bfla		*Extended mnemonic for* **bcla 4,cr_bit,target**	LR

Table 28-8. Extended Mnemonics for bc, bca, bcl, bcla (continued)

Mnemonic	Operands	Function	Other Registers Altered
bge	[cr_field,] target	Branch if greater than or equal. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bc 4,4∗cr_field+0,target**	
bgea		*Extended mnemonic for* **bca 4,4∗cr_field+0,target**	
bgel		*Extended mnemonic for* **bcl 4,4∗cr_field+0,target**	LR
bgela		*Extended mnemonic for* **bcla 4,4∗cr_field+0,target**	LR
bgt	[cr_field,] target	Branch if greater than. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bc 12,4∗cr_field+1,target**	
bgta		*Extended mnemonic for* **bca 12,4∗cr_field+1,target**	
bgtl		*Extended mnemonic for* **bcl 12,4∗cr_field+1,target**	LR
bgtla		*Extended mnemonic for* **bcla 12,4∗cr_field+1,target**	LR
ble	[cr_field,] target	Branch if less than or equal. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bc 4,4∗cr_field+1,target**	
blea		*Extended mnemonic for* **bca 4,4∗cr_field+1,target**	
blel		*Extended mnemonic for* **bcl 4,4∗cr_field+1,target**	LR
blela		*Extended mnemonic for* **bcla 4,4∗cr_field+1,target**	LR
blt	[cr_field,] target	Branch if less than Use CR0 if cr_field is omitted. *Extended mnemonic for* **bc 12,4∗cr_field+0,target**	
blta		*Extended mnemonic for* **bca 12,4∗cr_field+0,target**	
bltl		*Extended mnemonic for* **bcl 12,4∗cr_field+0,target**	(LR) ← CIA + 4.
bltla		*Extended mnemonic for* **bcla 12,4∗cr_field+0,target**	(LR) ← CIA + 4.

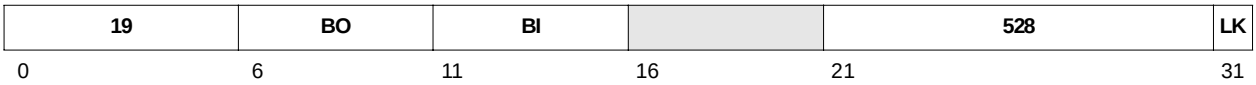
Table 28-8. Extended Mnemonics for bc, bca, bcl, bcla (continued)

Mnemonic	Operands	Function	Other Registers Altered
bne	[cr_field,] target	Branch if not equal. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bc 4,4∗cr_field+2,target**	
bnea		*Extended mnemonic for* **bca 4,4∗cr_field+2,target**	
bnel		**Extended mnemonic for** **bcl 4,4∗cr_field+2,target**	(LR) ← CIA + 4.
bnela		*Extended mnemonic for* **bcla 4,4∗cr_field+2,target**	(LR) ← CIA + 4.
bng	[cr_field,] target	Branch if not greater than. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bc 4,4∗cr_field+1,target**	
bnga		*Extended mnemonic for* **bca 4,4∗cr_field+1,target**	
bngl		*Extended mnemonic for* **bcl 4,4∗cr_field+1,target**	(LR) ← CIA + 4.
bngla		*Extended mnemonic for* **bcla 4,4∗cr_field+1,target**	(LR) ← CIA + 4.
bnl	[cr_field,] target	Branch if not less than; use CR0 if cr_field is omitted. *Extended mnemonic for* **bc 4,4∗cr_field+0,target**	
bnla		*Extended mnemonic for* **bca 4,4∗cr_field+0,target**	
bnll		*Extended mnemonic for* **bcl 4,4∗cr_field+0,target**	(LR) ← CIA + 4.
bnlla		*Extended mnemonic for* **bcla 4,4∗cr_field+0,target**	(LR) ← CIA + 4.
bns	[cr_field,] target	Branch if not summary overflow. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bc 4,4∗cr_field+3,target**	
bnsa		*Extended mnemonic for* **bca 4,4∗cr_field+3,target**	
bnsl		*Extended mnemonic for* **bcl 4,4∗cr_field+3,target**	(LR) ← CIA + 4.
bnsla		*Extended mnemonic for* **bcla 4,4∗cr_field+3,target**	(LR) ← CIA + 4.

Table 28-8. Extended Mnemonics for bc, bca, bcl, bcla (continued)

Mnemonic	Operands	Function	Other Registers Altered
bnu	[cr_field,] target	Branch if not unordered. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bc 4,4∗cr_field+3,target**	
bnua		*Extended mnemonic for* **bca 4,4∗cr_field+3,target**	
bnul		*Extended mnemonic for* **bcl 4,4∗cr_field+3,target**	(LR) ← CIA + 4.
bnula		*Extended mnemonic for* **bcla 4,4∗cr_field+3,target**	(LR) ← CIA + 4.
bso	[cr_field,] target	Branch if summary overflow. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bc 12,4∗cr_field+3,target**	
bsoa		*Extended mnemonic for* **bca 12,4∗cr_field+3,target**	
bsol		*Extended mnemonic for* **bcl 12,4∗cr_field+3,target**	(LR) ← CIA + 4.
bsola		*Extended mnemonic for* **bcla 12,4∗cr_field+3,target**	(LR) ← CIA + 4.
bt	cr_bit, target	Branch if CR_{cr_bit} = 1. *Extended mnemonic for* **bc 12,cr_bit,target**	
bta		*Extended mnemonic for* **bca 12,cr_bit,target**	
btl		*Extended mnemonic for* **bcl 12,cr_bit,target**	(LR) ← CIA + 4.
btla		*Extended mnemonic for* **bcla 12,cr_bit,target**	(LR) ← CIA + 4.
bun	[cr_field], target	Branch if unordered. Use CR0 if *cr_field* is omitted. *Extended mnemonic for* **bc 12,4∗cr_field+3,target**	
buna		*Extended mnemonic for* **bca 12,4∗cr_field+3,target**	
bunl		*Extended mnemonic for* **bcl 12,4∗cr_field+3,target**	(LR) ← CIA + 4.
bunla		*Extended mnemonic for* **bcla 12,4∗cr_field+3,target**	(LR) ← CIA + 4.

bcctr	BO, BI	LK = 0
bcctrl	BO, BI	LK = 1




```
if (BO0 = 1 ∨ (CRBI = BO1))  then
   NIA ← CTR0:29 || 2 0
else
   NIA ← CIA + 4
if LK = 1 then
   (LR) ← CIA + 4
PC ← NIA
```

If BO_0 contains 0, then the CR bit specified by the BI field is compared to BO_1 as part of the branch condition. If BO_0 contains 1, then the CR is not tested as part of the branch condition, and the BI field is ignored.

The next instruction address (NIA) is either the effective address of the branch target, or the address of the instruction after the branch, depending on whether the branch is taken or not. The branch target address is formed by concatenating two 0-bits to the right of the 30 most significant bits of the CTR.

BO_4 affects branch prediction, a performance-improvement feature. See ~~"Branch Prediction" on page 4-47~~*Branch Prediction* on page 187 for a complete discussion.

Instruction execution resumes with the instruction at the NIA.

If the LK field contains 1, then (CIA + 4) is placed into the LR.

Registers Altered

- LR if LK contains 1

Invalid Instruction Forms

- Reserved fields
- If BO_2 contains 0, the instruction form is invalid, and the result of the instruction (in particular, the branch target address and whether or not the branch is taken) is undefined. The architecture does not permit the combination of decrementing the CTR as part of the branch condition, together with using the CTR as the branch target address.

Table 28-9. Extended Mnemonics for bcctr, bcctrl

Mnemonic	Operands	Function	Other Registers Altered
bctr		Branch unconditionally to address in CTR. *Extended mnemonic for* **bcctr 20,0**	
bctrl		*Extended mnemonic for* **bcctrl 20,0**	(LR) ← CIA + 4.

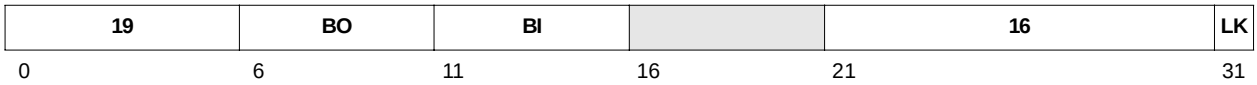
Table 28-9. Extended Mnemonics for bcctr, bcctrl (continued)

Mnemonic	Operands	Function	Other Registers Altered
beqctr	[cr_field]	Branch, if equal, to address in CTR Use CR0 if cr_field is omitted. *Extended mnemonic for* **bcctr 12,4∗cr_field+2**	
beqctrl		*Extended mnemonic for* **bcctrl 12,4∗cr_field+2**	(LR) ← CIA + 4.
bfctr	cr_bit	Branch, if CR_{cr_bit} = 0, to address in CTR. *Extended mnemonic for* **bcctr 4,cr_bit**	
bfctrl		*Extended mnemonic for* **bcctrl 4,cr_bit**	(LR) ← CIA + 4.
bgectr	[cr_field]	Branch, if greater than or equal, to address in CTR. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bcctr 4,4∗cr_field+0**	
bgectrl		*Extended mnemonic for* **bcctrl 4,4∗cr_field+0**	(LR) ← CIA + 4.
bgtctr	[cr_field]	Branch, if greater than, to address in CTR. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bcctr 12,4∗cr_field+1**	
bgtctrl		*Extended mnemonic for* **bcctrl 12,4∗cr_field+1**	(LR) ← CIA + 4.
blectr	[cr_field]	Branch, if less than or equal, to address in CTR. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bcctr 4,4∗cr_field+1**	
blectrl		*Extended mnemonic for* **bcctrl 4,4∗cr_field+1**	(LR) ← CIA + 4.
bltctr	[cr_field]	Branch, if less than, to address in CTR. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bcctr 12,4∗cr_field+0**	
bltctrl		*Extended mnemonic for* **bcctrl 12,4∗cr_field+0**	(LR) ← CIA + 4.
bnectr	[cr_field]	Branch, if not equal, to address in CTR. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bcctr 4,4∗cr_field+2**	
bnectrl		*Extended mnemonic for* **bcctrl 4,4∗cr_field+2**	(LR) ← CIA + 4.

Table 28-9. Extended Mnemonics for bcctr, bcctrl (continued)

Mnemonic	Operands	Function	Other Registers Altered
bngctr	[cr_field]	Branch, if not greater than, to address in CTR. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bcctr 4,4∗cr_field+1**	
bngctrl		*Extended mnemonic for* **bcctrl 4,4∗cr_field+1**	(LR) ← CIA + 4.
bnlctr	[cr_field]	Branch, if not less than, to address in CTR. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bcctr 4,4∗cr_field+0**	
bnlctrl		*Extended mnemonic for* **bcctrl 4,4∗cr_field+0**	(LR) ← CIA + 4.
bnsctr	[cr_field]	Branch, if not summary overflow, to address in CTR. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bcctr 4,4∗cr_field+3**	
bnsctrl		*Extended mnemonic for* **bcctrl 4,4∗cr_field+3**	(LR) ← CIA + 4.
bnuctr	[cr_field]	Branch, if not unordered, to address in CTR; use CR0 if cr_field is omitted. *Extended mnemonic for* **bcctr 4,4∗cr_field+3**	
bnuctrl		*Extended mnemonic for* **bcctrl 4,4∗cr_field+3**	(LR) ← CIA + 4.
bsoctr	[cr_field]	Branch, if summary overflow, to address in CTR. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bcctr 12,4∗cr_field+3**	
bsoctrl		*Extended mnemonic for* **bcctrl 12,4∗cr_field+3**	(LR) ← CIA + 4.
btctr	cr_bit	Branch if CR_{cr_bit} = 1 to address in CTR. *Extended mnemonic for* **bcctr 12,cr_bit**	
btctrl		*Extended mnemonic for* **bcctrl 12,cr_bit**	(LR) ← CIA + 4.
bunctr	[cr_field]	Branch if unordered to address in CTR. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bcctr 12,4∗cr_field+3**	
bunctrl		*Extended mnemonic for* **bcctrl 12,4∗cr_field+3**	(LR) ← CIA + 4.

bclr	BO, BI	LK = 0
bclrl	BO, BI	LK = 1




```
if BO2 = 0 then
   CTR ← CTR – 1
if (BO2 = 1 ∨ ((CTR = 0) = BO3)) ∧ (BO0 = 1 ∨ (CRBI = BO1))  then
   NIA ← LR0:29 || 20
else
   NIA ← CIA + 4
if LK = 1 then
   (LR) ← CIA + 4
PC ← NIA
```

If BO_2 contains 0, the CTR decrements, and the decremented value is tested for 0 as part of the branch condition. In this case, BO_3 indicates whether the test for 0 must be true or false in order for the branch to be taken. If BO_2 contains 1, then the CTR is neither decremented nor tested as part of the branch condition.

If BO_0 contains 0, then the CR bit specified by the BI field is compared to BO_1 as part of the branch condition. If BO_0 contains 1, then the CR is not tested as part of the branch condition, and the BI field is ignored.

The next instruction address (NIA) is either the effective address of the branch target, or the address of the instruction after the branch, depending on whether the branch is taken or not. The branch target address is formed by concatenating two 0-bits to the right of the 30 most significant bits of the LR.

BO_4 affects branch prediction, a performance-improvement feature. See ~~"Branch Prediction" on page 4-47~~*Branch Prediction* on page 187 for a complete discussion.

Instruction execution resumes with the instruction at the NIA.

If the LK field contains 1, then (CIA + 4) is placed into the LR.

Registers Altered

- CTR if BO_2 contains 0
- LR if LK contains 1

Invalid Instruction Forms

- Reserved fields

Table 28-10. Extended Mnemonics for bclr, bclrl

Mnemonic	Operands	Function	Other Registers Altered
blr		Branch unconditionally to address in LR. *Extended mnemonic for* **bclr 20,0**	
blrl		*Extended mnemonic for* **bclrl 20,0**	(LR) ← CIA + 4.
bdnzlr		Decrement CTR. Branch if CTR ≠ 0 to address in LR. *Extended mnemonic for* **bclr 16,0**	
bdnzlrl		*Extended mnemonic for* **bclrl 16,0**	(LR) ← CIA + 4.
bdnzflr	cr_bit	Decrement CTR. Branch if CTR ≠ 0 AND CR_{cr_bit} = 0 to address in LR. *Extended mnemonic for* **bclr 0,cr_bit**	
bdnzflrl		*Extended mnemonic for* **bclrl 0,cr_bit**	(LR) ← CIA + 4.
bdnztlr	cr_bit	Decrement CTR. Branch if CTR ≠ 0 AND CR_{cr_bit} = 1 to address in LR. *Extended mnemonic for* **bclr 8,cr_bit**	
bdnztlrl		*Extended mnemonic for* **bclrl 8,cr_bit**	(LR) ← CIA + 4.
bdzlr		Decrement CTR. Branch if CTR = 0 to address in LR. *Extended mnemonic for* **bclr 18,0**	
bdzlrl		*Extended mnemonic for* **bclrl 18,0**	(LR) ← CIA + 4.
bdzflr	cr_bit	Decrement CTR. Branch if CTR = 0 AND CR_{cr_bit} = 0 to address in LR. *Extended mnemonic for* **bclr 2,cr_bit**	
bdzflrl		*Extended mnemonic for* **bclrl 2,cr_bit**	(LR) ← CIA + 4.
bdztlr	cr_bit	Decrement CTR. Branch if CTR = 0 AND CR_{cr_bit} = 1 to address in LR. *Extended mnemonic for* **bclr 10,cr_bit**	
bdztlrl		*Extended mnemonic for* **bclrl 10,cr_bit**	(LR) ← CIA + 4.

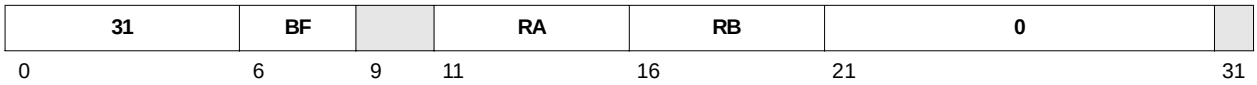
Table 28-10. Extended Mnemonics for bclr, bclrl (continued)

Mnemonic	Operands	Function	Other Registers Altered
beqlr	[cr_field]	Branch if equal to address in LR. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bclr 12,4∗cr_field+2**	
beqlrl		*Extended mnemonic for* **bclrl 12,4∗cr_field+2**	(LR) ← CIA + 4.
bflr	cr_bit	Branch if CR_{cr_bit} = 0 to address in LR. *Extended mnemonic for* **bclr 4,cr_bit**	
bflrl		*Extended mnemonic for* **bclrl 4,cr_bit**	(LR) ← CIA + 4.
bgelr	[cr_field]	Branch, if greater than or equal, to address in LR. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bclr 4,4∗cr_field+0**	
bgelrl		*Extended mnemonic for* **bclrl 4,4∗cr_field+0**	(LR) ← CIA + 4.
bgtlr	[cr_field]	Branch, if greater than, to address in LR. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bclr 12,4∗cr_field+1**	
bgtlrl		*Extended mnemonic for* **bclrl 12,4∗cr_field+1**	(LR) ← CIA + 4.
blelr	[cr_field]	Branch, if less than or equal, to address in LR. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bclr 4,4∗cr_field+1**	
blelrl		*Extended mnemonic for* **bclrl 4,4∗cr_field+1**	(LR) ← CIA + 4.
bltlr	[cr_field]	Branch, if less than, to address in LR. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bclr 12,4∗cr_field+0**	
bltlrl		*Extended mnemonic for* **bclrl 12,4∗cr_field+0**	(LR) ← CIA + 4.
bnelr	[cr_field]	Branch, if not equal, to address in LR. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bclr 4,4∗cr_field+2**	
bnelrl		*Extended mnemonic for* **bclrl 4,4∗cr_field+2**	(LR) ← CIA + 4.

Table 28-10. Extended Mnemonics for bclr, bclrl (continued)

Mnemonic	Operands	Function	Other Registers Altered
bnglr	[cr_field]	Branch, if not greater than, to address in LR. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bclr 4,4∗cr_field+1**	
bnglrl		*Extended mnemonic for* **bclrl 4,4∗cr_field+1**	(LR) ← CIA + 4.
bnllr	[cr_field]	Branch, if not less than, to address in LR. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bclr 4,4∗cr_field+0**	
bnllrl		*Extended mnemonic for* **bclrl 4,4∗cr_field+0**	(LR) ← CIA + 4.
bnslr	[cr_field]	Branch if not summary overflow to address in LR. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bclr 4,4∗cr_field+3**	
bnslrl		*Extended mnemonic for* **bclrl 4,4∗cr_field+3**	(LR) ← CIA + 4.
bnulr	[cr_field]	Branch if not unordered to address in LR. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bclr 4,4∗cr_field+3**	
bnulrl		*Extended mnemonic for* **bclrl 4,4∗cr_field+3**	(LR) ← CIA + 4.
bsolr	[cr_field]	Branch if summary overflow to address in LR. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bclr 12,4∗cr_field+3**	
bsolrl		*Extended mnemonic for* **bclrl 12,4∗cr_field+3**	(LR) ← CIA + 4.
btlr	cr_bit	Branch if CR_{cr_bit} = 1 to address in LR. *Extended mnemonic for* **bclr 12,cr_bit**	
btlrl		*Extended mnemonic for* **bclrl 12,cr_bit**	(LR) ← CIA + 4.
bunlr	[cr_field]	Branch if unordered to address in LR. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bclr 12,4∗cr_field+3**	
bunlrl		*Extended mnemonic for* **bclrl 12,4∗cr_field+3**	(LR) ← CIA + 4.

cmp BF, 0, RA, RB



$c_{0:3} \leftarrow {}^{4}0$
if (RA) < (RB) then $c_0 \leftarrow 1$
if (RA) > (RB) then $c_1 \leftarrow 1$
if (RA) = (RB) then $c_2 \leftarrow 1$
$c_3 \leftarrow$ XER[SO]
n ← BF
CR[CRn] $\leftarrow c_{0:3}$

The contents of register RA are compared with the contents of register RB using a 32-bit signed compare.

The CR field specified by the BF field is updated to reflect the results of the compare and the value of XER[SO] is placed into the same CR field.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- CR[CR*n*] where *n* is specified by the BF field

Invalid Instruction Forms

- Reserved fields

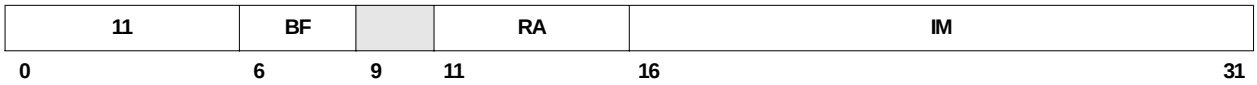
Programming Note

PowerPC Book-E architecture defines this instruction as **cmp BF,L,RA,RB**, where L selects operand size for 64-bit implementations. For all 32-bit implementations, L = 0 is required (L = 1 is an invalid form); hence for the PPC440GP, use of the extended mnemonic **cmpw BF,RA,RB** is recommended.

Table 28-11. Extended Mnemonics for cmp

Mnemonic	Operands	Function	Other Registers Altered
cmpw	[BF,] RA, RB	Compare Word; use CR0 if BF is omitted. *Extended mnemonic for* **cmp BF,0,RA,RB**	

cmpi BF, 0, RA, IM



$c_{0:3} \leftarrow {}^{4}0$
if (RA) < EXTS(IM) then $c_0 \leftarrow 1$
if (RA) > EXTS(IM) then $c_1 \leftarrow 1$
if (RA) = EXTS(IM) then $c_2 \leftarrow 1$
$c_3 \leftarrow$ XER[SO]
n ← BF
CR[CRn] $\leftarrow c_{0:3}$

The IM field is sign-extended to 32 bits. The contents of register RA are compared with the extended IM field, using a 32-bit signed compare.

The CR field specified by the BF field is updated to reflect the results of the compare and the value of XER[SO] is placed into the same CR field.

Registers Altered

- CR[CR*n*] where *n* is specified by the BF field

Invalid Instruction Forms

- Reserved fields

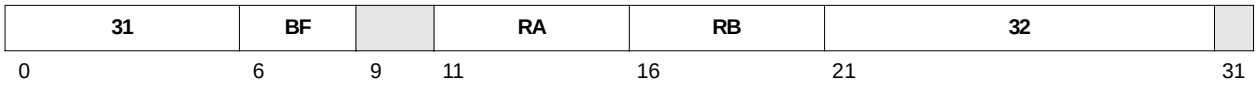
Programming Note

PowerPC Book-E Architecture defines this instruction as **cmpi BF,L,RA,IM**, where L selects operand size for 64-bit implementations. For all 32-bit implementations, L = 0 is required (L = 1 is an invalid form); hence for the PPC440GP, use of the extended mnemonic **cmpwi BF,RA,IM** is recommended.

Table 28-12. Extended Mnemonics for cmpi

Mnemonic	Operands	Function	Other Registers Altered
cmpwi	[BF,] RA, IM	Compare Word Immediate. Use CR0 if BF is omitted. *Extended mnemonic for* **cmpi BF,0,RA,IM**	

cmpl BF, 0, RA, RB



$c_{0:3} \leftarrow {}^{4}0$
if (RA) $\overset{u}{<}$ (RB) then $c_0 \leftarrow 1$
if (RA) $\overset{u}{>}$ (RB) then $c_1 \leftarrow 1$
if (RA) = (RB) then $c_2 \leftarrow 1$
$c_3 \leftarrow$ XER[SO]
n $\leftarrow$ BF
CR[CRn] $\leftarrow$ $c_{0:3}$

The contents of register RA are compared with the contents of register RB, using a 32-bit unsigned compare.

The CR field specified by the BF field is updated to reflect the results of the compare and the value of XER[SO] is placed into the same CR field.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- CR[CR*n*] where *n* is specified by the BF field

Invalid Instruction Forms

- Reserved fields

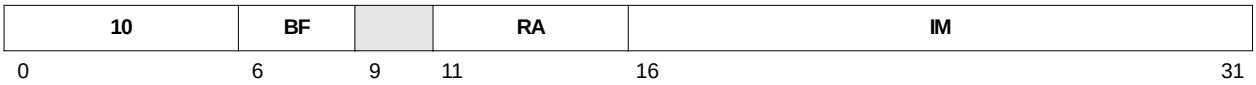
Programming Notes

PowerPC Book-E Architecture defines this instruction as **cmpl BF,L,RA,RB**, where L selects operand size for 64-bit implementations. For all 32-bit implementations, L = 0 is required (L = 1 is an invalid form); hence for PPC440GP, use of the extended mnemonic **cmplw BF,RA,RB** is recommended.

Table 28-13. Extended Mnemonics for cmpl

Mnemonic	Operands	Function	Other Registers Altered
cmplw	[BF,] RA, RB	Compare Logical Word. Use CR0 if BF is omitted. *Extended mnemonic for* **cmpl BF,0,RA,RB**	

cmpli BF, 0, RA, IM



$c_{0:3} \leftarrow {}^{4}0$
if (RA) $\overset{u}{<}$ ($^{16}0$ || IM) then $c_0 \leftarrow 1$
if (RA) $\overset{u}{>}$ ($^{16}0$ || IM) then $c_1 \leftarrow 1$
if (RA) = ($^{16}0$ || IM) then $c_2 \leftarrow 1$
$c_3 \leftarrow$ XER[SO]
n ← BF
CR[CRn] $\leftarrow$ $c_{0:3}$

The IM field is extended to 32 bits by concatenating 16 0-bits to its left. The contents of register RA are compared with IM using a 32-bit unsigned compare.

The CR field specified by the BF field is updated to reflect the results of the compare and the value of XER[SO] is placed into the same CR field.

Registers Altered

- CR[CR*n*] where *n* is specified by the BF field

Invalid Instruction Forms

- Reserved fields

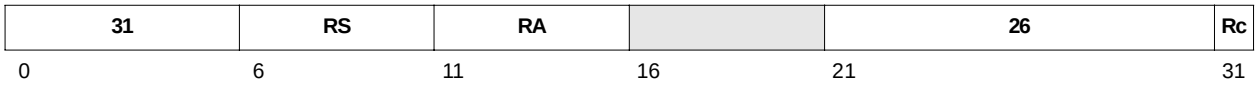
Programming Note

PowerPC Book-E Architecture defines this instruction as **cmpli BF,L,RA,IM**, where L selects operand size for 64-bit implementations. For all 32-bit implementations, L = 0 is required (L = 1 is an invalid form); hence for the PPC440GP, use of the extended mnemonic **cmplwi BF,RA,IM** is recommended.

Table 28-14. Extended Mnemonics for cmpli

Mnemonic	Operands	Function	Other Registers Changed
cmplwi	[BF,] RA, IM	Compare Logical Word Immediate. Use CR0 if BF is omitted. *Extended mnemonic for* **cmpli BF,0,RA,IM**	

cntlzw	RA, RS	Rc=0
cntlzw.	RA, RS	Rc=1




```
n ← 0
do while n < 32
   if (RS)n = 1 then leave
   n ← n + 1
(RA) ← n
```

The consecutive leading 0 bits in register RS are counted; the count is placed into register RA.

The count ranges from 0 through 32, inclusive.

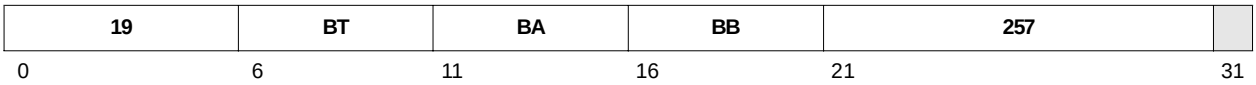
Registers Altered

- RA
- CR[CR0] if Rc contains 1

Invalid Instruction Forms

- Reserved fields

crand BT, BA, BB



$CR_{BT} \leftarrow CR_{BA} \wedge CR_{BB}$

The CR bit specified by the BA field is ANDed with the CR bit specified by the BB field; the result is placed into the CR bit specified by the BT field.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

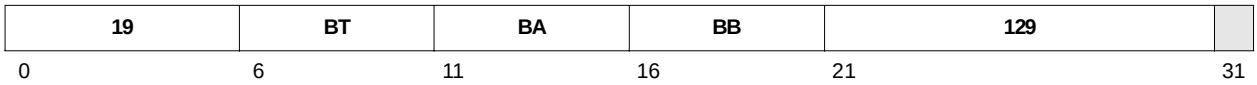
Registers Altered

- CR_{BT}

Invalid Instruction Forms

- Reserved fields

crandc BT, BA, BB



$CR_{BT} \leftarrow CR_{BA} \wedge \neg CR_{BB}$

The CR bit specified by the BA field is ANDed with the ones complement of the CR bit specified by the BB field; the result is placed into the CR bit specified by the BT field.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

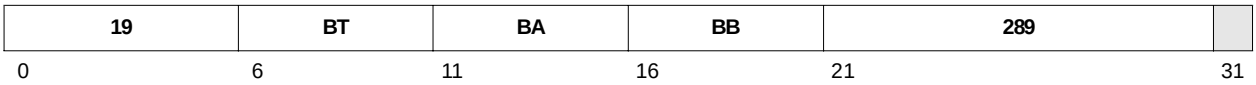
Registers Altered

- CR_{BT}

Invalid Instruction Forms

- Reserved fields

creqv BT, BA, BB



$CR_{BT} \leftarrow \neg(CR_{BA} \oplus CR_{BB})$

The CR bit specified by the BA field is XORed with the CR bit specified by the BB field; the ones complement of the result is placed into the CR bit specified by the BT field.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- CR_{BT}

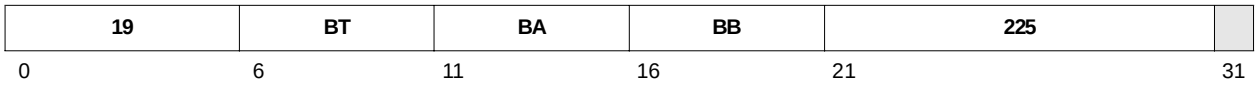
Invalid Instruction Forms

- Reserved fields

Table 28-15. Extended Mnemonics for creqv

Mnemonic	Operands	Function	Other Registers Altered
crset	bx	CR set. *Extended mnemonic for* **creqv bx,bx,bx**	

crnand BT, BA, BB



$CR_{BT} \leftarrow \neg(CR_{BA} \wedge CR_{BB})$

The CR bit specified by the BA field is ANDed with the CR bit specified by the BB field; the ones complement of the result is placed into the CR bit specified by the BT field.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

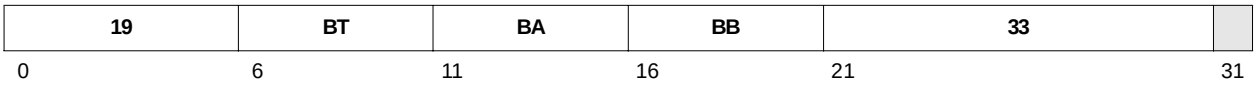
Registers Altered

- CR_{BT}

Invalid Instruction Forms

- Reserved fields

crnor BT, BA, BB



$CR_{BT} \leftarrow \neg(CR_{BA} \vee CR_{BB})$

The CR bit specified by the BA field is ORed with the CR bit specified by the BB field; the ones complement of the result is placed into the CR bit specified by the BT field.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- CR_{BT}

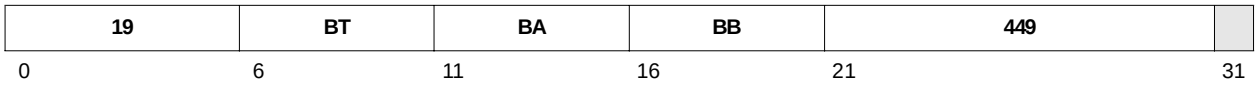
Invalid Instruction Forms

- Reserved fields

Table 28-16. Extended Mnemonics for crnor

Mnemonic	Operands	Function	Other Registers Altered
crnot	bx, by	CR not. *Extended mnemonic for* **crnor bx,by,by**	

cror BT, BA, BB



$CR_{BT} \leftarrow CR_{BA} \vee CR_{BB}$

The CR bit specified by the BA field is ORed with the CR bit specified by the BB field; the result is placed into the CR bit specified by the BT field.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- CR_{BT}

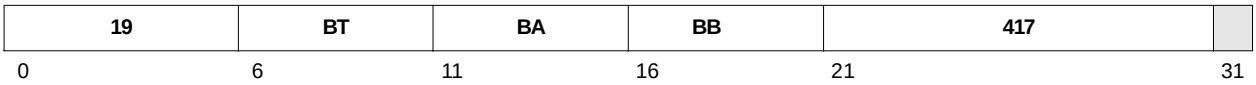
Invalid Instruction Forms

- Reserved fields

Table 28-17. Extended Mnemonics for cror

Mnemonic	Operands	Function	Other Registers Altered
crmove	bx, by	CR move. *Extended mnemonic for* **cror bx,by,by**	

crorc BT, BA, BB



$CR_{BT} \leftarrow CR_{BA} \vee \neg CR_{BB}$

The condition register (CR) bit specified by the BA field is ORed with the ones complement of the CR bit specified by the BB field; the result is placed into the CR bit specified by the BT field.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

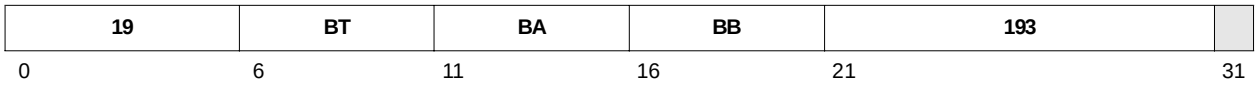
Registers Altered

- CR_{BT}

Invalid Instruction Forms

- Reserved fields

crxor BT, BA, BB



$CR_{BT} \leftarrow CR_{BA} \oplus CR_{BB}$

The CR bit specified by the BA field is XORed with the CR bit specified by the BB field; the result is placed into the CR bit specified by the BT field.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- CR_{BT}

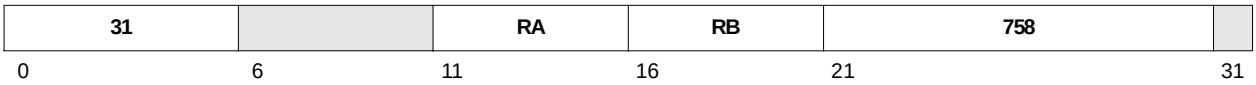
Invalid Instruction Forms

- Reserved fields

Table 28-18. Extended Mnemonics for crxor

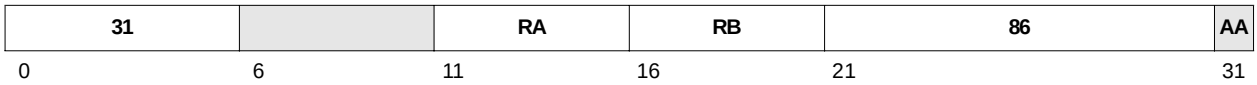
Mnemonic	Operands	Function	Other Registers Altered
crclr	bx	Condition register clear. *Extended mnemonic for* **crxor bx,bx,bx**	

dcba RA, RB



dcba is treated as a no-op by the PPC440GP.

dcbf RA, RB



EA ← (RA|0) + (RB)
DCBF(EA)

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise.

If the data block corresponding to the EA is in the data cache and marked as modified (stored into), the data block is copied back to main storage and then marked invalid in the data cache. If the data block is not marked as modified, it is simply marked invalid in the data cache. The operation is performed whether or not the memory page referenced by the EA is marked as cacheable.

If the data block at the EA is not in the data cache, no operation is performed.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- None

Invalid Instruction Forms

- Reserved fields

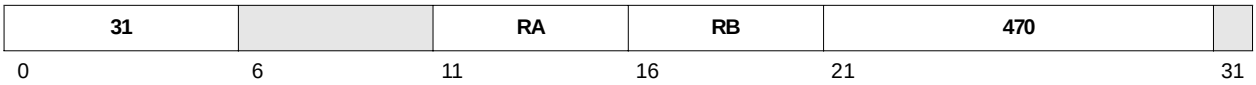
Exceptions

This instruction is considered a "load" with respect to Data Storage exceptions. See ~~"Data Storage Interrupt" on page 10-21~~*Data Storage Interrupt* on page 356 for more information.

This instruction is considered a "store" with respect to data address compare (DAC) Debug exceptions. See ~~"Debug Interrupt" on page 10-35~~*Debug Interrupt* on page 369 for more information.

This instruction may cause a Cache Locking type of Data Storage exception. See ~~"Data Storage Interrupt" on page 10-21~~*Data Storage Interrupt* on page 356 for more information.

dcbi RA, RB



EA ← (RA|0) + (RB)
DCBI(EA)

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise.

If the data block at the EA is in the data cache, the data block is marked invalid, regardless of whether or not the memory page referenced by the EA is marked as cacheable. If modified data existed in the data block prior to the operation of this instruction, that data is lost.

If the data block at the EA is not in the data cache, no operation is performed.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- None

Invalid Instruction Forms

- Reserved fields

Programming Notes

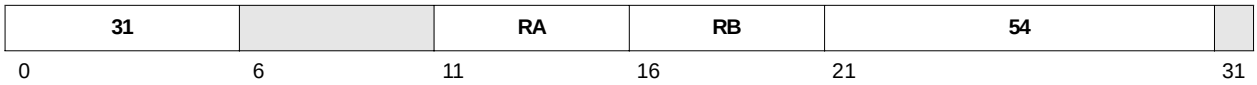
Execution of this instruction is privileged.

Exceptions

This instruction is considered a "store" with respect to Data Storage exceptions. See ~~"Data Storage Interrupt" on page 10-21~~*Data Storage Interrupt* on page 356 for more information.

This instruction is considered a "store" with respect to data address compare (DAC) Debug exceptions. See ~~"Debug Interrupt" on page 10-35~~*Debug Interrupt* on page 369 for more information.

dcbst RA, RB



EA ← (RA|0) + (RB)
DCBST(EA)

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 if the RA field is 0, and is the contents of register RA otherwise.

If the data block at the EA is in the data cache and marked as modified, the data block is copied back to main storage and marked as unmodified in the data cache.

If the data block at the EA is in the data cache, and is not marked as modified, or if the data block at the EA is not in the data cache, no operation is performed.

The operation specified by this instruction is performed whether or not the memory page referenced by the EA is marked as cacheable.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- None

Invalid Instruction Forms

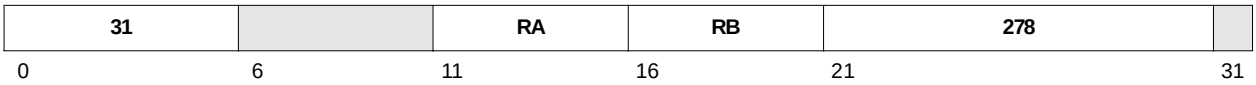
- Reserved fields

Exceptions

This instruction is considered a "load" with respect to Data Storage exceptions. See ~~"Data Storage Interrupt" on page 10-21~~*Data Storage Interrupt* on page 356 for more information.

This instruction is considered a "store" with respect to data address compare (DAC) Debug exceptions. See ~~"Debug Interrupt" on page 10-35~~*Debug Interrupt* on page 369 for more information.

dcbt RA, RB



EA ← (RA|0) + (RB)
DCBT(EA)

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 when the RA field is 0, and is the contents of register RA otherwise.

If the data block at the EA is not in the data cache and the memory page referenced by the EA is marked as cacheable, the block is read from main storage into the data cache.

If the data block at the EA is in the data cache, or if the memory page referenced by the EA is marked as caching inhibited, no operation is performed.

This instruction is not allowed to cause Data Storage interrupts nor Data TLB Error interrupts. If execution of the instruction causes either of these types of exception, then no operation is performed, and no interrupt occurs.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- None

Invalid Instruction Forms

- Reserved fields

Programming Notes

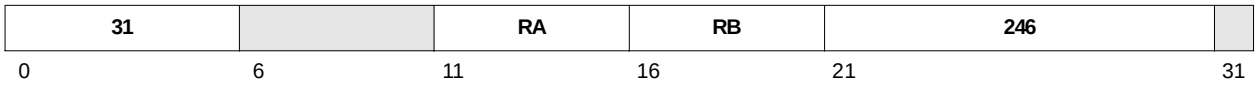
The **dcbt** instruction allows a program to begin a cache block fetch from main storage before the program needs the data. The program can later load data from the cache into registers without incurring the latency of a cache miss.

Exceptions

This instruction is considered a "load" with respect to Data Storage exceptions. See ~~"Data Storage Interrupt" on page 10-21~~*Data Storage Interrupt* on page 356 for more information.

This instruction is considered a "load" with respect to data address compare (DAC) Debug exceptions. See ~~"Debug Interrupt" on page 10-35~~*Debug Interrupt* on page 369 for more information.

dcbtst RA, RB



EA ← (RA|0) + (RB)
DCBTST(EA)

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise.

If the data block at the EA is not in the data cache and the memory page referenced by the EA address is marked as cacheable, the data block is loaded into the data cache.

If the data block at the EA is in the data cache, or if the memory page referenced by the EA is marked as caching inhibited, no operation is performed.

This instruction is not allowed to cause Data Storage interrupts nor Data TLB Error interrupts. If execution of the instruction causes either of these types of exception, then no operation is performed, and no interrupt occurs.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- None

Invalid Instruction Forms

- Reserved fields

Programming Notes

The **dcbtst** instruction allows a program to begin a cache block fetch from main storage before the program needs the data. The program can later store data from GPRs into the cache block, without incurring the latency of a cache miss.

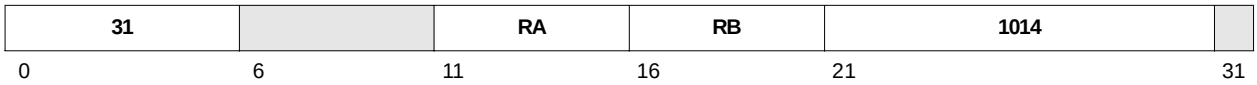
Architecturally, **dcbtst** is intended to bring a cache block into the data cache in a manner which will permit future instructions to store to that block efficiently. For example, in an implementation which supports the "MESI" cache coherency protocol, the block would be brought into the cache in "Exclusive" mode, allowing the block to be stored to without having to broadcast any coherency operations on the system bus. However, since the PPC440GP does not support hardware-enforcement of multiprocessor coherency, there is no distinction between a block being brought in for a read or a write, and hence the implementation of the **dcbtst** instruction is identical to the implementation of the **dcbt** instruction.

Exceptions

This instruction is considered a "load" with respect to Data Storage exceptions. See ~~"Data Storage Interrupt" on page 10-21~~*Data Storage Interrupt* on page 356 for more information.

This instruction is considered a "load" with respect to data address compare (DAC) Debug exceptions. See ~~"Debug Interrupt" on page 10-35~~*Debug Interrupt* on page 369 for more information.

dcbz RA, RB



EA ← (RA|0) + (RB)
DCBZ(EA)

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise.

If the data block at the EA is in the data cache and the memory page referenced by the EA is marked as cacheable and non-write-through, the data in the cache block is set to ~~0~~0 and marked as *dirty* (modified).

If the data block at the EA is not in the data cache and the memory page referenced by the EA is marked as cacheable and non-write-through, a cache block is established and set to ~~0~~0 and marked as dirty. Note that nothing is read from main storage, as described in the programming note.

If the memory page referenced by the EA is marked as either write-through or as caching inhibited, an Alignment exception occurs.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- None

Invalid Instruction Forms

- Reserved fields

Programming Notes

Because **dcbz** can establish an address in the data cache without copying the contents of that address from main storage, the address established may be invalid with respect to the storage subsystem. A subsequent operation may cause the address to be copied back to main storage, for example, to make room for a new cache block; a Data Machine Check exception could occur under these circumstances.

If **dcbz** is attempted to an EA in a memory page which is marked as caching inhibited or as write-through, the software alignment exception handler should emulate the instruction by storing zeros to the block referenced by the EA. The store instructions in the emulation software will cause main storage to be updated (and possibly the cache, if the EA is in a page marked as write-through).

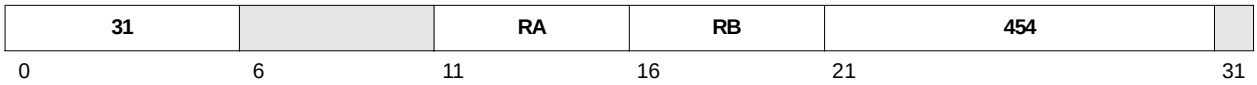
Exceptions

An alignment exception occurs if the EA is marked as caching inhibited or as write-through.

This instruction is considered a "store" with respect to Data Storage exceptions. See ~~"Data Storage Interrupt" on page 10-21~~*Data Storage Interrupt* on page 356 for more information.

This instruction is considered a "store" with respect to data address compare (DAC) Debug exceptions. See ~~"Debug Interrupt" on page 10-35~~*Debug Interrupt* on page 369 for more information.

dccci RA, RB



DCCCI

This instruction flash invalidates the entire data cache array. The RA and RB operands are not used; previous implementations used these operands to calculate an effective address (EA) which specified the particular block or blocks to be invalidated. The instruction form (including the specification of RA and RB operands) is maintained for software and tool compatibility.

Registers Altered

- None

Invalid Instruction Forms

- Reserved fields

Programming Notes

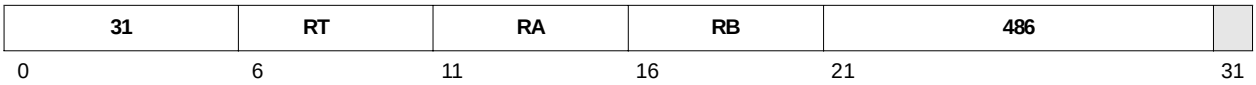
Execution of this instruction is privileged.

This instruction is intended for use in the power-on reset routine to invalidate the entire data cache array before ~~cacheing~~ caching is enabled.

Architecture Note

This instruction is implementation-specific and programs which use this instruction may not be portable to other PowerPC Book-E implementations. See ~~"Instruction Set Portability" on page 28-2~~ *Instruction Set Portability* on page 894.

dcread RT, RA, RB



EA ← (RA|0) + (RB)
INDEX ← $EA_{17:26}$
WORD ← $EA_{27:29}$
(RT) ← (data cache data)[INDEX,WORD]
DCDBTRH ← (data cache tag high)[INDEX]
DCDBTRL ← (data cache tag low)[INDEX]

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise.

$EA_{17:26}$ selects a line of tag and data from the data cache. $EA_{27:29}$ selects a word from the 8-word data portion of the selected cache line, and this word is read into register RT. $EA_{30:31}$ must be 0b00; if not, the value placed in register RT is undefined.

The tag portion of the selected cache line is read into the DCDBTRH and DCDBTRL registers, as follows:

Register[bit(s)]	Tag Field	Name	
DCDBTRH[0:23]	TRA	Tag Real Address	Bits 0:23 of the lower 32 bits of the 36-bit real address associated with this cache line
DCDBTRH[24]	V	Valid	The valid indicator for the cache line (1 indicates valid)
DCDBTRH[25:27]		reserved	Reserved fields are read as 0s
DCDBTRH[28:31]	TERA	Tag Extended Real Address	Upper 4 bits of the 36-bit real address associated with this cache line
DCDBTRL[0:23]		reserved	Reserved fields are read as 0s
DCDBTRL[24:27]	D	Dirty Indicators	The "dirty" (modified) indicators for each of the four doublewords in the cache line
DCDBTRL[28]	U0	U0 Storage Attribute	The U0 storage attribute for the memory page associated with this cache line
DCDBTRL[29]	U1	U1 Storage Attribute	The U0 storage attribute for the memory page associated with this cache line
DCDBTRL[30]	U2	U2 Storage Attribute	The U0 storage attribute for the memory page associated with this cache line
DCDBTRL[31]	U3	U3 Storage Attribute	The U0 storage attribute for the memory page associated with this cache line

This instruction can be used by a debug tool to determine the contents of the data cache, without knowing the specific addresses of the lines which are currently contained within the cache.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- RT
- DCDBTRH
- DCDBTRL

Invalid Instruction Forms

- Reserved fields

Programming Note

Execution of this instruction is privileged.

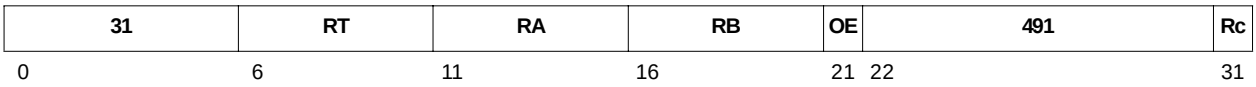
The PPC440GP does not support the use of the **dcread** instruction when the ~~DCC~~ data cache controller is still in the process of performing cache operations associated with previously executed instructions (such as line fills and line flushes). Also, the PPC440GP does not automatically synchronize context between a **dcread** instruction and the subsequent **mfspr** instructions that read the results of the **dcread** instruction into GPRs. In order to guarantee that the **dcread** instruction operates correctly, and that the **mfspr** instructions obtain the results of the **dcread** instruction, a sequence such as the following must be used:

```
msync                   # ensure that all previous cache operations have completed
dcread    regT,regA,regB# read cache information; the contents of GPR A and GPR B are
                        # added and the result used to specify a cache line index to be read;
                        # the data word is moved into GPR T and the tag information is read
                        # into DCDBTRH and DCDBTRL
isync                   # ensure dcread completes before attempting to read results
mfdcdbtrh regD          # move high portion of tag into GPR D
mfdcdbtrl regE          # move low portion of tag into GPR E
```

Architecture Note

This instruction is implementation-specific and programs which use this instruction may not be portable to other PowerPC Book-E implementations. See ~~"Instruction Set Portability" on page 28-2~~ *Instruction Set Portability* on page 894.

divw	RT, RA, RB	OE=0, Rc=0
divw.	RT, RA, RB	OE=0, Rc=1
divwo	RT, RA, RB	OE=1, Rc=0
divwo.	RT, RA, RB	OE=1, Rc=1



(RT) $\leftarrow$ (RA) $\div$ (RB)

The contents of register RA are divided by the contents of register RB. The quotient is placed into register RT.

Both the dividend and the divisor are interpreted as signed integers. The quotient is the unique signed integer that satisfies:

dividend = (quotient $\times$ divisor) + remainder

where the remainder has the same sign as the dividend and its magnitude is less than that of the divisor.

If an attempt is made to perform (0x8000 0000 $\div$ –1) or ($n \div 0$), the contents of register RT are undefined; if the Rc field also contains 1, the contents of $CR[CR0]_{0:2}$ are undefined. Either invalid division operation sets XER[OV, SO] (and $CR[CR0]_3$ if Rc contains 1) to 1 if the OE field contains 1.

Registers Altered

- RT
- CR[CR0] if Rc contains 1
- XER[OV, SO] if OE contains 1

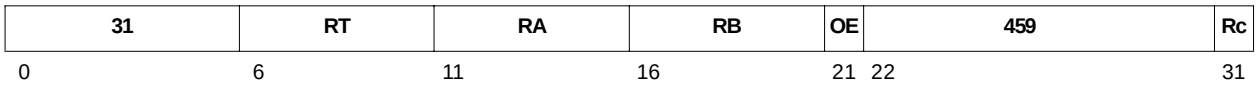
Programming Note

The 32-bit remainder can be calculated using the following sequence of instructions:

```
divw    RT,RA,RB        # RT = quotient
mullw   RT,RT,RB        # RT = quotient × divisor
subf    RT,RT,RA        # RT = remainder
```

The sequence does not calculate correct results for the invalid divide operations.

divwu	RT, RA, RB	OE=0, Rc=0
divwu.	RT, RA, RB	OE=0, Rc=1
divwuo	RT, RA, RB	OE=1, Rc=0
divwuo.	RT, RA, RB	OE=1, Rc=1



(RT) $\leftarrow$ (RA) $\div$ (RB)

The contents of register RA are divided by the contents of register RB. The quotient is placed into register RT.

The dividend and the divisor are interpreted as unsigned integers. The quotient is the unique unsigned integer that satisfies:

dividend = (quotient $\times$ divisor) + remainder

If an attempt is made to perform ($n \div 0$), the contents of register RT are undefined; if the Rc also contains 1, the contents of $CR[CR0]_{0:2}$ are also undefined. The invalid division operation also sets XER[OV, SO] (and $CR[CR0]_3$ if Rc contains 1) to 1 if the OE field contains 1.

Registers Altered

- RT
- CR[CR0] if Rc contains 1
- XER[OV, SO] if OE contains 1

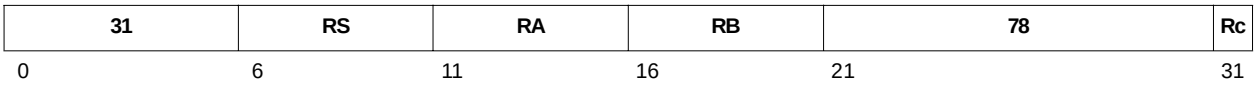
Programming Note

The 32-bit remainder can be calculated using the following sequence of instructions

```
divwu   RT,RA,RB    # RT = quotient
mullw   RT,RT,RB    # RT = quotient × divisor
subf    RT,RT,RA    # RT = remainder
```

This sequence does not calculate the correct result if the divisor is 0.

dlmzb	RA, RS, RB	Rc=0
dlmzb.	RA, RS, RB	Rc=1




```
d ← (RS) || (RB)
i, x, y ← 0
do while (x < 8) ∧ (y = 0)
   x ← x + 1
   if d_{i:i + 7} = 0 then
      y ← 1
   else
      i ← i + 8
(RA) ← x
XER[TBC] ← x
if Rc = 1 then
   CR[CR0]_3 ←XER[SO]
   if y = 1 then
      if x < 5 then
         CR[CR0]_{0:2} ← 0b010
      else
         CR[CR0]_{0:2} ← 0b100
   else
      CR[CR0]_{0:2} ← 0b001
```

The contents of registers RS and RB are concatenated to form an 8-byte operand. The operand is searched for the leftmost byte in which each bit is 0 (a 0-byte).

Bytes in the operand are numbered from left to right starting with 1. If a 0-byte is found, its byte number is placed into XER[TBC] and register RA. Otherwise, the number 8 is placed into XER[TBC] and register RA.

If the Rc field contains 1, XER[SO] is copied to $CR[CR0]_3$ and $CR[CR0]_{0:2}$ are updated as follows:

- If no 0-byte is found, $CR[CR0]_{0:2}$ is set to 0b001.
- If the leftmost 0-byte is in the first 4 bytes (in the RS register), $CR[CR0]_{0:2}$ is set to 0b010.
- If the leftmost 0-byte is in the last 4 bytes (in the RB register), $CR[CR0]_{0:2}$ is set to 0b100.

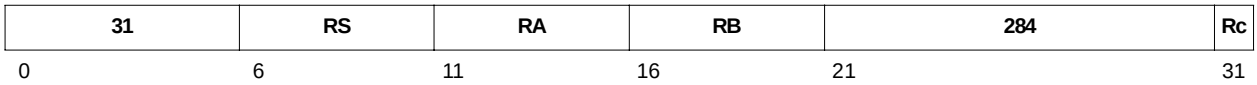
Registers Altered

- XER[TBC]
- RA
- CR[CR0] if Rc contains 1

Architecture Note

This instruction is implementation-specific and programs which use this instruction may not be portable to other PowerPC Book-E implementations. See ~~"Instruction Set Portability" on page 28-2~~*Instruction Set Portability* on page 894.

eqv	RA, RS, RB	Rc=0
eqv.	RA, RS, RB	Rc=1



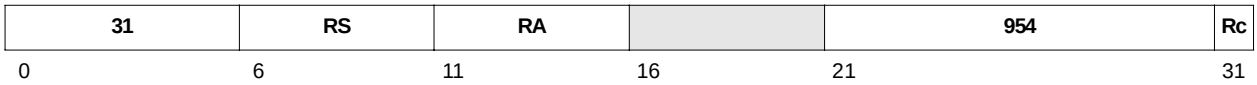
(RA) $\leftarrow$ $\neg$((RS) $\oplus$ (RB))

The contents of register RS are XORed with the contents of register RB; the ones complement of the result is placed into register RA.

Registers Altered

- RA
- CR[CR0] if Rc contains 1

extsb	RA, RS	Rc=0
extsb.	RA, RS	Rc=1



(RA) $\leftarrow$ EXTS(RS)$_{24:31}$

The least significant byte of register RS is sign-extended to 32 bits by replicating bit 24 of the register into bits 0 through 23 of the result. The result is placed into register RA.

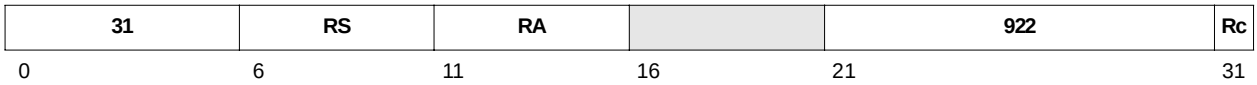
Registers Altered

- RA
- CR[CR0] if Rc contains 1

Invalid Instruction Forms

- Reserved fields

extsh	RA, RS	Rc=0
extsh.	RA, RS	Rc=1



(RA) $\leftarrow$ EXTS(RS)$_{16:31}$

The least significant halfword of register RS is sign-extended to 32 bits by replicating bit 16 of the register into bits 0 through 15 of the result. The result is placed into register RA.

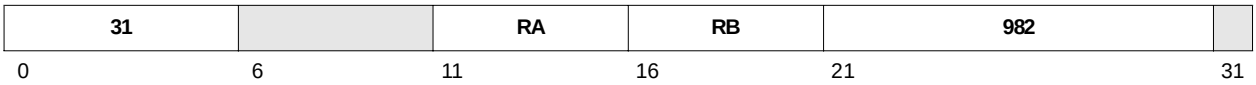
Registers Altered

- RA
- CR[CR0] if Rc contains 1

Invalid Instruction Forms

- Reserved fields

icbi RA, RB



EA ← (RA|0) + (RB)
ICBI(EA)

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise.

If the instruction block at the EA is in the instruction cache, the cache block is marked invalid.

If the instruction block at the EA is not in the instruction cache, no additional operation is performed.

The operation specified by this instruction is performed whether or not the EA is marked as cacheable.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- None

Invalid Instruction Forms

- Reserved fields

Programming Note

Instruction cache management instructions use MSR[DS], not MSR[IS], as part of the virtual address. Also, the instruction cache on the PPC440GP is "virtually-tagged", which means that the EA is converted to a virtual address (VA), and the VA is compared against the cache tag field. See ~~"Instruction Cache Synonyms" on page 5-12~~*Instruction Cache Synonyms* on page 215 for more information on the ramifications of virtual tagging on software.

Exceptions

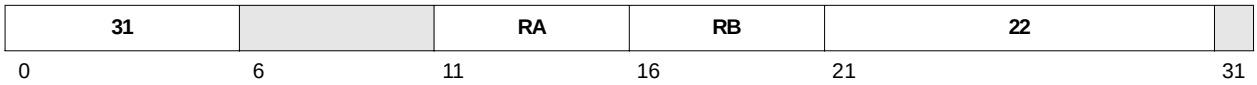
Instruction Storage interrupts and Instruction TLB Error interrupts are associated with exceptions which occur during instruction *fetching*, not during instruction *execution*. *Execution* of instruction cache management instructions may cause Data Storage or Data TLB Error exceptions.

This instruction is considered a "load" with respect to Data Storage exceptions. See ~~"Data Storage Interrupt" on page 10-21~~*Data Storage Interrupt* on page 356 for more information.

This instruction is considered a "load" with respect to data address compare (DAC) Debug exceptions. See ~~"Debug Interrupt" on page 10-35~~*Debug Interrupt* on page 369 for more information.

This instruction may cause a Cache Locking type of Data Storage exception. See ~~"Data Storage Interrupt" on page 10-21~~*Data Storage Interrupt* on page 356 for more information.

icbt RA, RB



EA← (RA|0) + (RB)
ICBT(EA)

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise.

If the instruction block at the EA is not in the instruction cache and the memory page referenced by the EA is marked as cacheable, the instruction block is fetched into the instruction cache.

If the instruction block at the EA is in the instruction cache, or if the memory page referenced by the EA is marked as caching inhibited, no operation is performed.

If the memory page referenced by the EA is marked as "no-execute" for the current operating mode (user mode or supervisor mode, as specified by MSR[PR]), no operation is performed.

This instruction is not allowed to cause Data Storage interrupts nor Data TLB Error interrupts. If execution of the instruction causes either of these types of exception, then no operation is performed, and no interrupt occurs.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- None

Invalid Instruction Forms

- Reserved fields

Programming Notes

This instruction allows a program to begin a cache block fetch from main storage before the program needs the instruction. The program can later branch to the instruction address and fetch the instruction from the cache without incurring the latency of a cache miss.

Instruction cache management instructions use MSR[DS], not MSR[IS], as part of the virtual address. Also, the instruction cache on the PPC440GP is "virtually-tagged", which means that the EA is converted to a virtual address (VA), and the VA is compared against the cache tag field. See ~~"Instruction Cache Synonyms" on page 5-12~~ *Instruction Cache Synonyms* on page 215 for more information on the ramifications of virtual tagging on software.

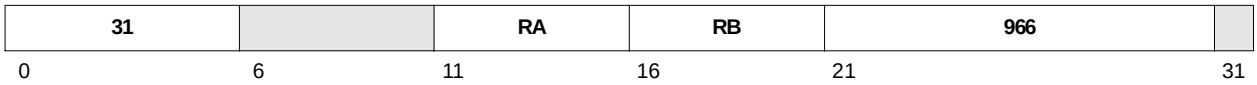
Exceptions

Instruction Storage interrupts and Instruction TLB Error interrupts are associated with exceptions which occur during instruction *fetching*, not during instruction *execution*. *Execution* of instruction cache management instructions may cause Data Storage or Data TLB Error exceptions, but are not allowed to cause the associated interrupt. Instead, if such an exception occurs, then no operation is performed.

This instruction is considered a "load" with respect to Data Storage exceptions. See ~~"Data Storage Interrupt" on page 10-21~~*Data Storage Interrupt* on page 356 for more information.

This instruction is considered a "load" with respect to data address compare (DAC) Debug exceptions. See ~~"Debug Interrupt" on page 10-35~~*Debug Interrupt* on page 369 for more information.

iccci RA, RB



ICCCI

This instruction flash invalidates the entire ~~data~~ instruction cache array. The RA and RB operands are not used; previous implementations used these operands to calculate an effective address (EA) which specified the particular block or blocks to be invalidated. The instruction form (including the specification of RA and RB operands) is maintained for software and tool compatibility.

Registers Altered

- None

Invalid Instruction Forms

- Reserved fields

Programming Notes

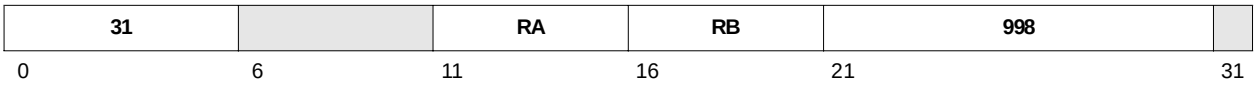
Execution of this instruction is privileged.

This instruction is intended for use in the power-on reset routine to invalidate the entire instruction cache array before ~~cacheing~~ caching is enabled.

Architecture Note

This instruction is implementation-specific and programs which use this instruction may not be portable to other PowerPC Book-E implementations. See ~~"Instruction Set Portability" on page 28-2~~ *Instruction Set Portability* on page 894.

icread RA, RB



EA ← (RA|0) + (RB)
INDEX ← $EA_{17:26}$
WORD ← $EA_{27:29}$
ICDBDR ← (instruction cache data)[INDEX,WORD]
ICDBTRH ← (instruction cache tag high)[INDEX]
ICDBTRL ← (instruction cache tag low)[INDEX]

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise.

$EA_{17:26}$ selects a line of tag and data (instructions) from the instruction cache. $EA_{27:29}$ selects a 32-bit instruction from the 8-instruction data portion of the selected cache line, and this instruction is read into the ICDBDR. $EA_{30:31}$ are ignored, as are $EA_{0:16}$.

The tag portion of the selected cache line is read into the ICDBTRH and ICDBTRL registers, as follows:

Register[bit(s)]	Tag Field	Name	
ICDBTRH[0:23]	TEA	Tag Effective Address	Bits 0:23 of the 32-bit effective address associated with this cache line
ICDBTRH[24]	V	Valid	The valid indicator for the cache line (1 indicates valid)
ICDBTRH[25:31]		reserved	Reserved fields are read as 0s
ICDBTRL[0:21]		reserved	Reserved fields are read as 0s
ICDBTRL[22]	TS	Translation Space	The address space portion of the virtual address associated with this cache line.
ICDBTRL[23]	TD	Translation ID (TID) Disable	TID Disable field for the memory page associated with this cache line
ICDBTRL[24:31]	TID	Translation ID	TID field portion of the virtual address associated with this cache line

The instruction cache on PPC440GP is "virtually-tagged", which means that the tag field contains the virtual address, which consists of the TEA, TS, and TID fields. See ~~Chapter 6, "Memory Management"~~ *Memory Management* on page 233 for more information on the function of the TS, TD, and TID fields.

This instruction can be used by a debug tool to determine the contents of the instruction cache, without knowing the specific addresses of the lines which are currently contained within the cache.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- ICDBDR
- ICDBTRH
- ICDBTRL

Invalid Instruction Forms

- Reserved fields

Programming Note

Execution of this instruction is privileged.

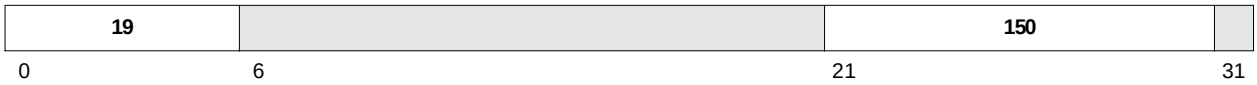
The PPC440GP does not automatically synchronize context between an **icread** instruction and the subsequent **mfspr** instructions which read the results of the **icread** instruction into GPRs. In order to guarantee that the **mfspr** instructions obtain the results of the **icread** instruction, a sequence such as the following must be used:

```
icread      regA,regB      # read cache information (the contents of GPR A and GPR B are
                           # added and the result used to specify a cache line index to be read)
isync                      # ensure icread completes before attempting to read results
mficdbdr    regC           # move instruction information into GPR C
mficdbtrh   regD           # move high portion of tag into GPR D
mficdbtrl   regE           # move low portion of tag into GPR E
```

Architecture Note

This instruction is implementation-specific and programs which use this instruction may not be portable to other PowerPC Book-E implementations. See ~~"Instruction Set Portability" on page 28-2~~*Instruction Set Portability* on page 894.

isync



The **isync** instruction is a context synchronizing instruction.

isync provides an ordering function for the effects of all instructions executed by the processor. Executing **isync** insures that all instructions preceding the **isync** instruction execute before **isync** completes, except that storage accesses caused by those instructions need not have completed. Furthermore, all instructions preceding the **isync** are guaranteed to be unaffected by any context changes initiated by instructions after the **isync**.

No subsequent instructions are initiated by the processor until **isync** completes. Finally, execution of **isync** causes the processor to discard any prefetched instructions (prefetched from the cache, not instructions that are in the cache or on their way into the cache), with the effect that subsequent instructions are fetched and executed in the context established by the instructions preceding **isync**.

isync causes any caching inhibited instruction fetches from memory to be aborted and any data associated with them to be discarded. Cacheable instruction fetches from memory are not aborted however, as these should be handled by the **icbi** instructions which must precede the **isync** if software wishes to invalidate any cached instructions.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- None

Invalid Instruction Forms

- Reserved fields

Programming Note

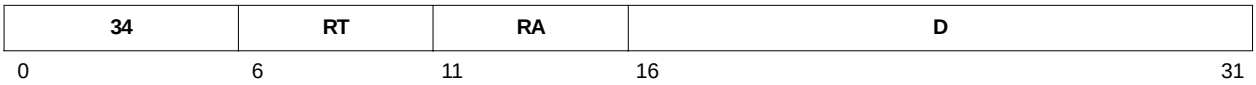
See the discussion of context synchronizing instructions in ~~"Synchronization" on page 4-62~~*Synchronization on page 201*.

The following code example illustrates the necessary steps for self-modifying code. This example assumes that addr1 is both data and instruction cacheable.

```
    stw     regN, addr1     # data in regN is to become an instruction at addr1
    dcbst   addr1           # forces data from the data cache to memory
    msync                   # wait until the data actually reaches the memory
    icbi    addr1           # invalidate the instruction if it is in the cache (or in the #
process of being fetched into the cache)
    msync                   # wait until the icbi completes
    isync                   # discard and refetch any instructions (including
                            # possibly the instruction at addr1) which may have
                            # already been fetched from the cache and be in the
```

```
# pipeline after the isync
```

lbz RT, D(RA)



EA ← (RA|0) + EXTS(D)
(RT) ← $^{24}0$ || MS(EA,1)

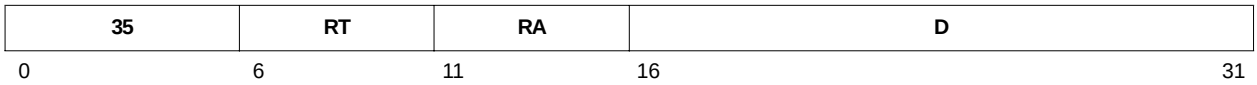
An effective address (EA) is formed by adding a displacement to a base address. The displacement is obtained by sign-extending the 16-bit D field to 32 bits. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise.

The byte at the EA is extended to 32 bits by concatenating 24 0-bits to its left. The result is placed into register RT.

Registers Altered

- RT

lbzu RT, D(RA)



EA ← (RA|0) + EXTS(D)
(RA) ← EA
(RT) ← $^{24}0$ || MS(EA,1)

An effective address (EA) is formed by adding a displacement to a base address. The displacement is obtained by sign-extending the 16-bit D field to 32 bits. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise. The EA is placed into register RA.

The byte at the EA is extended to 32 bits by concatenating 24 0-bits to its left. The result is placed into register RT.

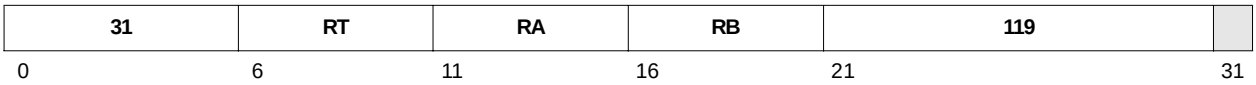
Registers Altered

- RA
- RT

Invalid Instruction Forms

- RA = RT
- RA = 0

lbzux RT, RA, RB



EA $\leftarrow$ (RA|0) + (RB)
(RA) $\leftarrow$ EA
(RT) $\leftarrow$ $^{24}0$ || MS(EA,1)

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise. The EA is placed into register RA.

The byte at the EA is extended to 32 bits by concatenating 24 0-bits to its left. The result is placed into register RT.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

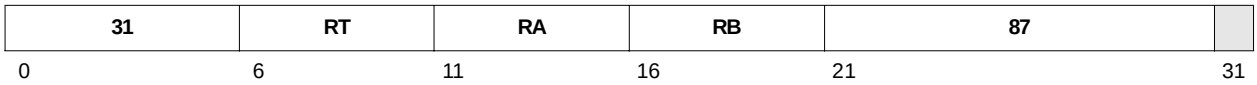
Registers Altered

- RA
- RT

Invalid Instruction Forms

- Reserved fields
- RA = RT
- RA = 0

lbzx RT,RA, RB



EA ← (RA|0) + (RB)
(RT) ← $^{24}0$ || MS(EA,1)

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise.

The byte at the EA is extended to 32 bits by concatenating 24 0-bits to its left. The result is placed into register RT.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

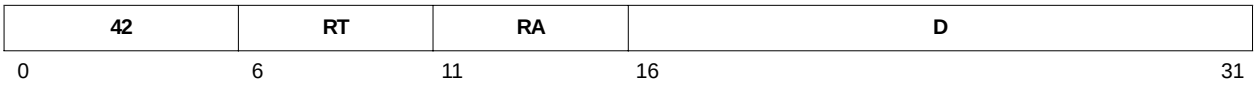
Registers Altered

- RT

Invalid Instruction Forms

- Reserved fields

lha RT, D(RA)



EA ← (RA|0) + EXTS(D)
(RT) ← EXTS(MS(EA,2))

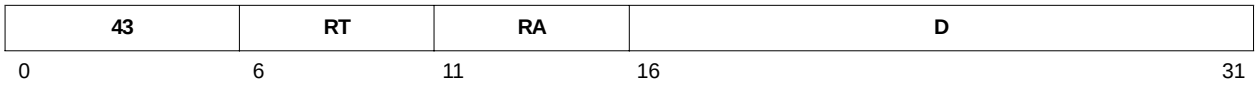
An effective address (EA) is formed by adding a displacement to a base address. The displacement is obtained by sign-extending the 16-bit D field to 32 bits. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise.

The halfword at the EA is sign-extended to 32 bits and placed into register RT.

Registers Altered

- RT

lhau RT, D(RA)



EA ← (RA|0) + EXTS(D)
(RA) ← EA
(RT) ← EXTS(MS(EA,2))

An effective address (EA) is formed by adding a displacement to a base address. The displacement is obtained by sign-extending the 16-bit D field to 32 bits. The base address is 0 when the RA field is 0 and is the contents of register RA otherwise. The EA is placed into register RA.

The halfword at the EA is sign-extended to 32 bits and placed into register RT.

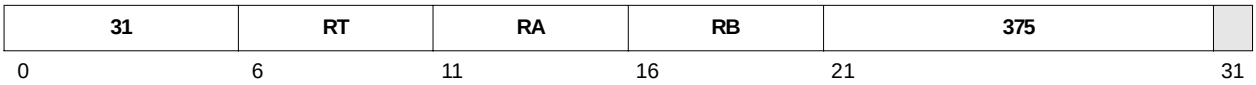
Registers Altered

- RA
- RT

Invalid Instruction Forms

- RA = RT
- RA = 0

lhaux RT, RA, RB



EA ← (RA|0) + (RB)
(RA) ← EA
(RT) ← EXTS(MS(EA,2))

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise. The EA is placed into register RA.

The halfword at the EA is sign-extended to 32 bits and placed into register RT.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

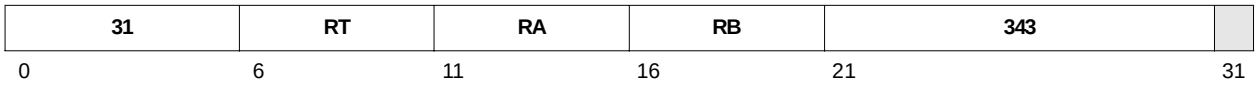
Registers Altered

- RA
- RT

Invalid Instruction Forms

- Reserved fields
- RA = RT
- RA = 0

lhax RT, RA, RB



EA ← (RA|0) + (RB)
(RT) ← EXTS(MS(EA,2))

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise.

The halfword at the EA is sign-extended to 32 bits and placed into register RT.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

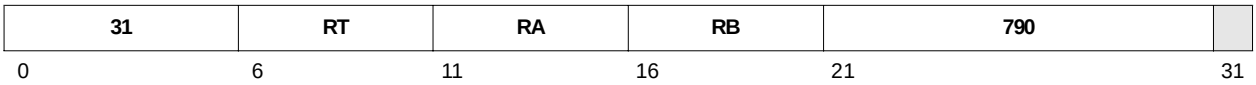
Registers Altered

- RT

Invalid Instruction Forms

- Reserved fields

lhbrx RT, RA, RB



EA ← (RA|0) + (RB)
(RT) ← $^{16}0$ || BYTE_REVERSE(MS(EA,2))

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise.

The halfword at the EA is byte-reversed from the default byte ordering for the memory page referenced by the EA. The resulting halfword is extended to 32 bits by concatenating 16 0-bits to its left. The result is placed into register RT.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- RT

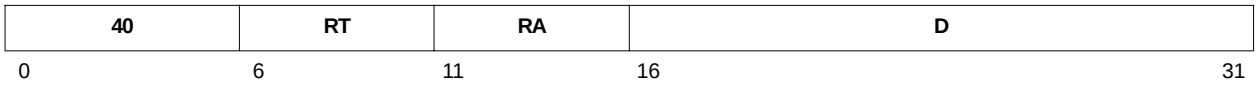
Invalid Instruction Forms

- Reserved fields

Programming Note

Byte ordering is generally controlled by the Endian (E) storage attribute (see ~~Chapter 6, "Memory Management"~~*Memory Management* on page 233). The load byte reverse instructions provide a mechanism for data to be loaded from a memory page using the opposite byte ordering from that specified by the Endian storage attribute.

lhz RT, D(RA)



EA ← (RA|0) + EXTS(D)
(RT) ← $^{16}0$ || MS(EA,2)

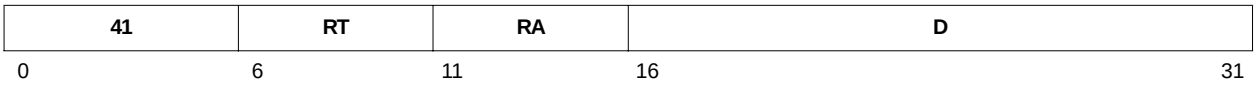
An effective address (EA) is formed by adding a displacement to a base address. The displacement is obtained by sign-extending the 16-bit D field to 32 bits. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise.

The halfword at the EA is extended to 32 bits by concatenating 16 0-bits to its left. The result is placed into register RT.

Registers Altered

- RT

lhzu RT, D(RA)



EA ← (RA|0) + EXTS(D)
(RA) ← EA
(RT) ← $^{16}0$ || MS(EA,2)

An effective address (EA) is formed by adding a displacement to a base address. The displacement is obtained by sign-extending the 16-bit D field to 32 bits. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise. The EA is placed into register RA.

The halfword at the EA is extended to 32 bits by concatenating 16 0-bits to its left. The result is placed into register RT.

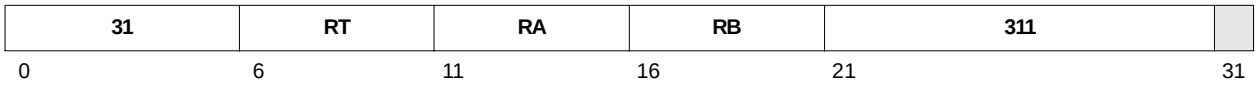
Registers Altered

- RA
- RT

Invalid Instruction Forms

- RA = RT
- RA = 0

lhzux RT, RA, RB



EA ← (RA|0) + (RB)
(RA) ← EA
(RT) ← $^{16}0$ || MS(EA,2)

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise. The EA is placed into register RA.

The halfword at the EA is extended to 32 bits by concatenating 16 0-bits to its left. The result is placed into register RT.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

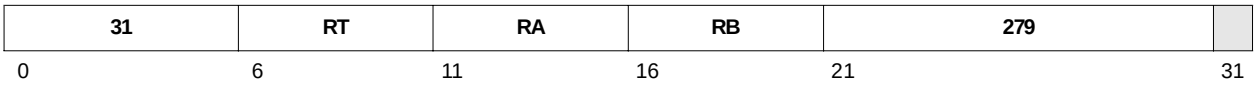
Registers Altered

- RA
- RT

Invalid Instruction Forms

- Reserved fields
- RA = RT
- RA = 0

lhzx RT, RA, RB



EA $\leftarrow$ (RA|0) + (RB)
(RT) $\leftarrow$ $^{16}0$ || MS(EA,2)

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise.

The halfword at the EA is extended to 32 bits by concatenating 16 0-bits to its left. The result is placed into register RT.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

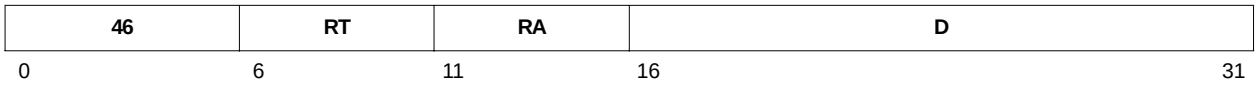
Registers Altered

- RT

Invalid Instruction Forms

- Reserved fields

lmw RT, D(RA)




```
EA ← (RA|0) + EXTS(D)
r ← RT
do while r ≤ 31
   GPR(r)) ← MS(EA,4)
   r ← r + 1
   EA ← EA + 4
```

An effective address (EA) is formed by adding a displacement to a base address. The displacement is obtained by sign-extending the 16-bit D field in the instruction to 32 bits. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise.

A series of consecutive words starting at the EA are loaded into a set of consecutive GPRs, starting with register RT and continuing to and including GPR(31).

Registers Altered

- RT through GPR(31).

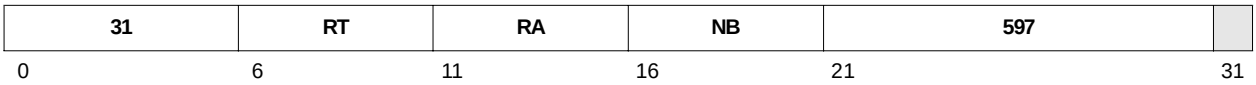
Invalid Instruction Forms

- RA is in the range of registers to be loaded, including the case RA = RT = 0.

Programming Note

This instruction can be restarted, meaning that it could be interrupted after having already updated some of the target registers, and then re-executed from the beginning (after returning from the interrupt), in which case the registers which had already been loaded prior to the interrupt will be loaded a second time. Note that if RA is in the range of registers to be loaded (an invalid form; see above) and is also one of the registers which is loaded prior to the interrupt, then when the instruction is restarted the re-calculated EA will be incorrect, since RA will no longer contain the original base address. Hence the definition of this as an invalid form which software must avoid.

lswi RT, RA, NB




```
EA ← (RA|0)
if NB = 0 then
   CNT ← 32
else
   CNT ← NB
n ← CNT
R_FINAL ← ((RT + CEIL(CNT/4) – 1) % 32)
r ← RT – 1
i ← 0
do while n > 0
   if i = 0 then
      r ← r + 1
      if r = 32 then
         r ← 0
      GPR(r)) ← 0
   GPR(r)_i:i+7) ← MS(EA,1)
   i ← i + 8
   if i = 32 then
      i ← 0
   EA ← EA + 1
   n ← n – 1
```

An effective address (EA) is determined by the RA field. If the RA field contains 0, the EA is 0. Otherwise, the EA is the contents of register RA.

The NB field specifies the byte count CNT. If the NB field contains 0, the byte count is CNT = 32. Otherwise, the byte count is CNT = NB.

A series of CNT consecutive bytes in main storage, starting at the EA, are loaded into CEIL(CNT/4) consecutive GPRs, four bytes per GPR, until the byte count is exhausted. Bytes are loaded into GPRs; the byte at the lowest address is loaded into the most significant byte. Bits to the right of the last byte loaded into the last GPR are set to 0.

The set of loaded GPRs starts at register RT, continues consecutively through GPR(31), and wraps to register 0, loading until the byte count is exhausted, which occurs in register R_{FINAL}.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- RT and subsequent GPRs as described above.

Invalid Instruction Forms

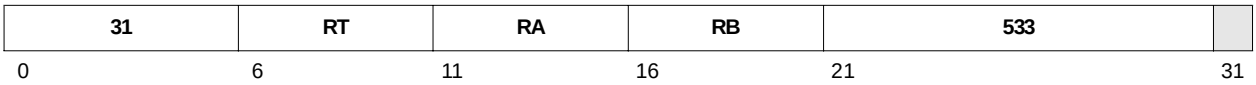
- Reserved fields

- RA is in the range of registers to be loaded
- RA = RT = 0

Programming Note

This instruction can be restarted, meaning that it could be interrupted after having already updated some of the target registers, and then re-executed from the beginning (after returning from the interrupt), in which case the registers which had already been loaded prior to the interrupt will be loaded a second time. Note that if RA is in the range of registers to be loaded (an invalid form; see above) and is also one of the registers which is loaded prior to the interrupt, then when the instruction is restarted the re-calculated EA will be incorrect, since RA will no longer contain the original base address. Hence the definition of this as an invalid form which software must avoid.

lswx RT, RA, RB




```
EA ← (RA|0) + (RB)
CNT ← XER[TBC]
n ← CNT
R_FINAL ← ((RT + CEIL(CNT/4) – 1) % 32)
r ← RT – 1
i ← 0
do while n > 0
   if i = 0 then
      r ← r + 1
      if r = 32 then
         r ← 0
      GPR(r)) ← 0
   GPR(r)_i:i+7) ← MS(EA,1)
   i ← i + 8
   if i = 32 then
      i ← 0
   EA ← EA + 1
   n ← n – 1
```

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise.

A byte count CNT is obtained from XER[TBC].

A series of CNT consecutive bytes in main storage, starting at the EA, are loaded into CEIL(CNT/4) consecutive GPRs, four bytes per GPR, until the byte count is exhausted. Bytes are loaded into GPRs; the byte having the lowest address is loaded into the most significant byte. Bits to the right of the last byte loaded in the last GPR used are set to 0.

The set of consecutive GPRs loaded starts at register RT, continues through GPR(31), and wraps to register 0, loading until the byte count is exhausted, which occurs in register R_{FINAL}.

If XER[TBC] is 0, the byte count is 0 and the contents of register RT are undefined.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- RT and subsequent GPRs as described above.

Invalid Instruction Forms

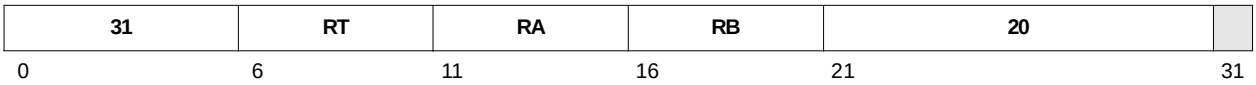
- Reserved fields
- RA or RB is in the range of registers to be loaded.
- RA = RT = 0

Programming Note

This instruction can be restarted, meaning that it could be interrupted after having already updated some of the target registers, and then re-executed from the beginning (after returning from the interrupt), in which case the registers which had already been loaded prior to the interrupt will be loaded a second time. Note that if RA or RB is in the range of registers to be loaded (an invalid form; see above) and is also one of the registers which is loaded prior to the interrupt, then when the instruction is restarted the re-calculated EA will be incorrect, since the affected register will no longer contain the original base address or index. Hence the definition of these as invalid forms which software must avoid.

If XER[TBC] = 0, the contents of register RT are undefined and **lswx** is treated as a no-op. Furthermore, if the EA is such that a Data Storage, Data TLB Error, or Data Address Compare Debug exception occurs, **lswx** is treated as a no-op and no interrupt occurs as a result of the exception.

lwarx RT, RA, RB



EA ← (RA|0) + (RB)
RESERVE ← 1
(RT) ← MS(EA,4)

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise.

The word at the EA is placed into register RT.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Execution of the **lwarx** instruction sets the reservation bit.

Registers Altered

- RT

Invalid Instruction Forms

- Reserved fields

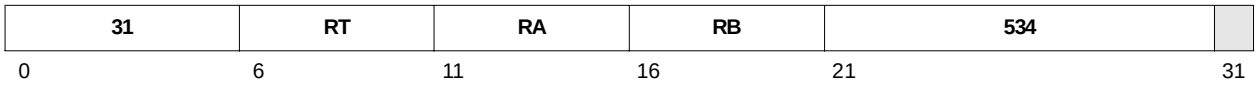
Programming Note

The **lwarx** and **stwcx.** instructions are typically paired in a loop, as shown in the following example, to create the effect of an atomic operation to a memory area used as a semaphore between multiple processes. Only **lwarx** can set the reservation bit to 1. **stwcx.** sets the reservation bit to 0 upon its completion, whether or not **stwcx.** actually stored (RS) to memory. $CR[CR0]_2$ must be examined to determine whether (RS) was sent to memory.

```
loop: lwarx     # read the semaphore from memory; set reservation
      "alter"   # change the semaphore bits in the register as required
      stwcx.    # attempt to store the semaphore; reset reservation
      bne loop  # some other process intervened and cleared the reservation prior to the above
                # stwcx.; try again
```

The PowerPC Book-E architecture specifies that the EA for the **lwarx** instruction must be word-aligned (that is, a multiple of 4 bytes); otherwise, the result is undefined. Although the PPC440GP will execute **lwarx** regardless of the EA alignment, in order for the operation of the pairing of **lwarx** and **stwcx.** to produce the desired result, software must ensure that the EA for both instructions is word-aligned. This requirement is due to the manner in which misaligned storage accesses may be broken up into separate, aligned accesses by the PPC440GP.

lwbrx RT, RA, RB



EA ← (RA|0) + (RB)
(RT) ← BYTE_REVERSE(MS(EA,4))

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise.

The word at the EA is byte-reversed from the default byte ordering for the memory page referenced by the EA. The result is placed into register RT.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- RT

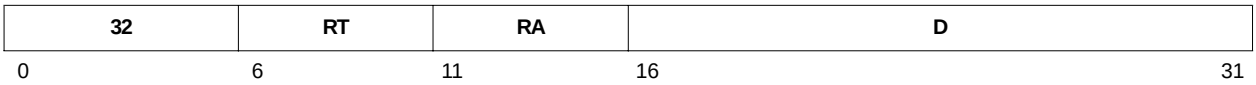
Invalid Instruction Forms

- Reserved fields

Programming Note

Byte ordering is generally controlled by the Endian (E) storage attribute (see ~~Chapter 6, "Memory Management"~~*Memory Management* on page 233). The load byte reverse instructions provide a mechanism for data to be loaded from a memory page using the opposite byte ordering from that specified by the Endian storage attribute.

lwz RT, D(RA)



EA ← (RA|0) + EXTS(D)
(RT) ← MS(EA,4)

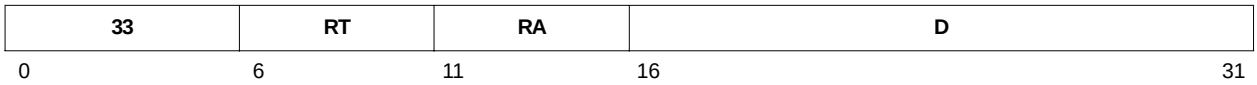
An effective address (EA) is formed by adding a displacement to a base address. The displacement is obtained by sign-extending the 16-bit D field to 32 bits. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise.

The word at the EA is placed into register RT.

Registers Altered

- RT

lwzu RT, D(RA)



EA ← (RA|0) + EXTS(D)
(RA) ← EA
(RT) ← MS(EA,4)

An effective address (EA) is formed by adding a displacement to a base address. The displacement is obtained by sign-extending the 16-bit D field to 32 bits. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise. The EA is placed into register RA.

The word at the EA is placed into register RT.

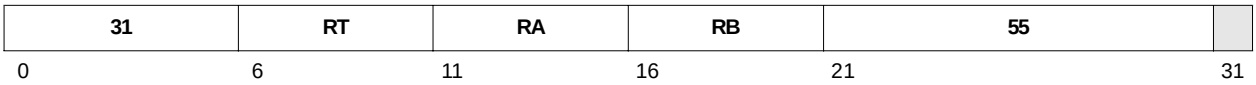
Registers Altered

- RA
- RT

Invalid Instruction Forms

- RA = RT
- RA = 0

lwzux RT, RA, RB



EA ← (RA|0) + (RB)
(RA) ← EA
(RT) ← MS(EA,4)

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise. The EA is placed into register RA.

The word at the EA is placed into register RT.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

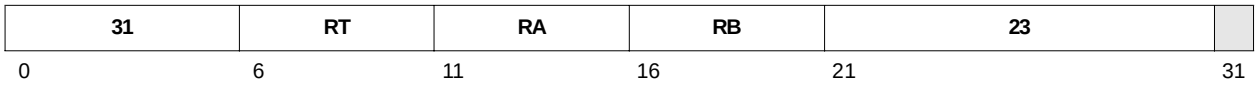
Registers Altered

- RA
- RT

Invalid Instruction Forms

- Reserved fields
- RA = RT
- RA = 0

lwzx RT, RA, RB



EA ← (RA|0) + (RB)
(RT) ← MS(EA,4)

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise.

The word at the EA is placed into register RT.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

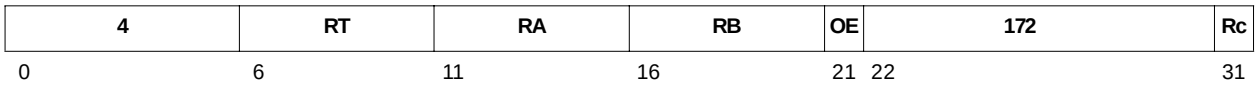
Registers Altered

- RT

Invalid Instruction Forms

- Reserved fields

macchw	RT, RA, RB	OE=0, Rc=0
macchw.	RT, RA, RB	OE=0, Rc=1
macchwo	RT, RA, RB	OE=1, Rc=0
macchwo.	RT, RA, RB	OE=1, Rc=1



$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{0:15}$ signed

$temp_{0:32} \leftarrow prod_{0:31} + (RT)$

$(RT) \leftarrow temp_{1:32}$

The low-order halfword of RA is multiplied by the high-order halfword of RB. The signed product is summed with the contents of RT and RT is updated with the low-order 32 bits of the signed sum.

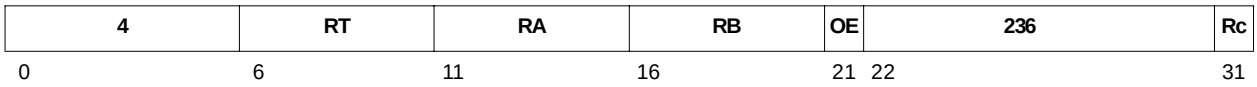
Registers Altered

- RT
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE contains 1

Architecture Note

This instruction is implementation-specific and programs which use this instruction may not be portable to other PowerPC Book-E implementations. See ~~"Instruction Set Portability" on page 28-2~~*Instruction Set Portability* on page 894.

macchws	RT, RA, RB	OE=0, Rc=0
macchws.	RT, RA, RB	OE=0, Rc=1
macchwso	RT, RA, RB	OE=1, Rc=0
macchwso.	RT, RA, RB	OE=1, Rc=1



$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{0:15}$ signed

$temp_{0:32} \leftarrow prod_{0:31} + (RT)$

if $((prod_0 = RT_0) \wedge (RT_0 \neq temp_1))$ then $(RT) \leftarrow (RT_0 \,||\, ^{31}(\neg RT_0))$

else $(RT) \leftarrow temp_{1:32}$

The low-order halfword of RA is multiplied by the high-order halfword of RB. The signed product is summed with the contents of RT.

If the signed sum can be represented in 32 bits, then RT is updated with the low-order 32 bits of the signed sum.

If the signed sum cannot be represented in 32 bits, then RT is updated with a value which is "saturated" to the nearest representable value. That is, if the signed sum is less than -2^{31}, then RT is updated with -2^{31}. Likewise, if the signed sum is greater than $2^{31} - 1$, then RT is updated with $2^{31} - 1$.

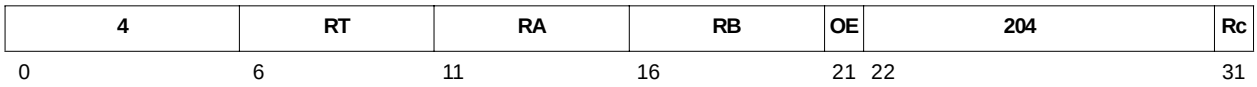
Registers Altered

- RT
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE contains 1

Architecture Note

This instruction is implementation-specific and programs which use this instruction may not be portable to other PowerPC Book-E implementations. See ~~"Instruction Set Portability" on page 28-2~~*Instruction Set Portability* on page 894.

macchwsu	RT, RA, RB	OE=0, Rc=0
macchwsu.	RT, RA, RB	OE=0, Rc=1
macchwsuo	RT, RA, RB	OE=1, Rc=0
macchwsuo.	RT, RA, RB	OE=1, Rc=1



$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{0:15}$ unsigned

$temp_{0:32} \leftarrow prod_{0:31} + (RT)$

$(RT) \leftarrow (temp_{1:32} \vee {}^{32}temp_{0})$

The low-order halfword of RA is multiplied by the high-order halfword of RB. The unsigned product is summed with the contents of RT.

If the unsigned sum can be represented in 32 bits, then RT is updated with the low-order 32 bits of the unsigned sum.

If the unsigned sum cannot be represented in 32 bits, then RT is updated with a value which is "saturated" to the maximum representable value of $2^{32} - 1$.

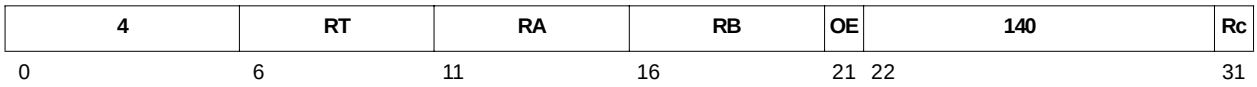
Registers Altered

- RT
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE contains 1

Architecture Note

This instruction is implementation-specific and programs which use this instruction may not be portable to other PowerPC Book-E implementations. See ~~"Instruction Set Portability" on page 28-2~~*Instruction Set Portability* on page 894.

macchwu	RT, RA, RB	OE=0, Rc=0
macchwu.	RT, RA, RB	OE=0, Rc=1
macchwuo	RT, RA, RB	OE=1, Rc=0
macchwuo.	RT, RA, RB	OE=1, Rc=1



$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{0:15}$ unsigned

$temp_{0:32} \leftarrow prod_{0:31} + (RT)$

$(RT) \leftarrow temp_{1:32}$

The low-order halfword of RA is multiplied by the high-order halfword of RB. The unsigned product is summed with the contents of RT and RT is updated with the low-order 32 bits of the unsigned sum.

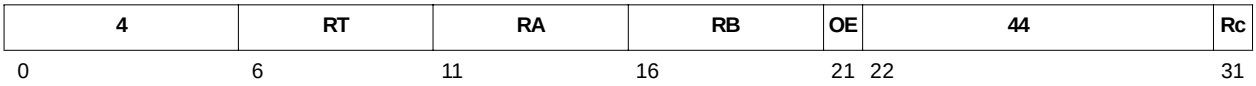
Registers Altered

- RT
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE contains 1

Architecture Note

This instruction is implementation-specific and programs which use this instruction may not be portable to other PowerPC Book-E implementations. See ~~"Instruction Set Portability" on page 28-2~~*Instruction Set Portability* on page 894.

machhw	RT, RA, RB	OE=0, Rc=0
machhw.	RT, RA, RB	OE=0, Rc=1
machhwo	RT, RA, RB	OE=1, Rc=0
machhwo.	RT, RA, RB	OE=1, Rc=1



$prod_{0:31} \leftarrow (RA)_{0:15} \times (RB)_{0:15}$ signed

$temp_{0:32} \leftarrow prod_{0:31} + (RT)$

$(RT) \leftarrow temp_{1:32}$

The high-order halfword of RA is multiplied by the high-order halfword of RB. The signed product is summed with the contents of RT and RT is updated with the low-order 32 bits of the signed sum.

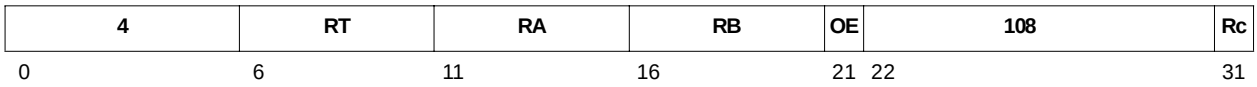
Registers Altered

- RT
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE contains 1

Architecture Note

This instruction is implementation-specific and programs which use this instruction may not be portable to other PowerPC Book-E implementations. See ~~"Instruction Set Portability" on page 28-2~~*Instruction Set Portability* on page 894.

machhws	RT, RA, RB	OE=0, Rc=0
machhws.	RT, RA, RB	OE=0, Rc=1
machhwso	RT, RA, RB	OE=1, Rc=0
machhwso.	RT, RA, RB	OE=1, Rc=1



$prod_{0:31} \leftarrow (RA)_{0:15} \times (RB)_{0:15}$ signed

$temp_{0:32} \leftarrow prod_{0:31} + (RT)$

if $((prod_0 = RT_0) \wedge (RT_0 \neq temp_1))$ then $(RT) \leftarrow (RT_0 \,||\, ^{31}(\neg RT_0))$

else $(RT) \leftarrow temp_{1:32}$

The high-order halfword of RA is multiplied by the high-order halfword of RB. The signed product is summed with the contents of RT.

If the signed sum can be represented in 32 bits, then RT is updated with the low-order 32 bits of the signed sum.

If the signed sum cannot be represented in 32 bits, then RT is updated with a value which is "saturated" to the nearest representable value. That is, if the signed sum is less than -2^{31}, then RT is updated with -2^{31}. Likewise, if the signed sum is greater than $2^{31} - 1$, then RT is updated with $2^{31} - 1$.

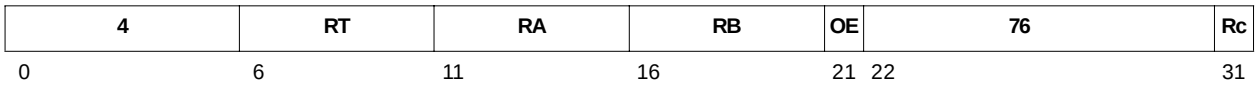
Registers Altered

- RT
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE contains 1

Architecture Note

This instruction is implementation-specific and programs which use this instruction may not be portable to other PowerPC Book-E implementations. See ~~"Instruction Set Portability" on page 28-2~~ *Instruction Set Portability* on page 894.

machhwsu	RT, RA, RB	OE=0, Rc=0
machhwsu.	RT, RA, RB	OE=0, Rc=1
machhwsuo	RT, RA, RB	OE=1, Rc=0
machhwsuo.	RT, RA, RB	OE=1, Rc=1



$prod_{0:31} \leftarrow (RA)_{0:15} \times (RB)_{0:15}$ unsigned

$temp_{0:32} \leftarrow prod_{0:31} + (RT)$

$(RT) \leftarrow (temp_{1:32} \vee {}^{32}temp_{0})$

The high-order halfword of RA is multiplied by the high-order halfword of RB. The unsigned product is summed with the contents of RT.

If the unsigned sum can be represented in 32 bits, then RT is updated with the low-order 32 bits of the unsigned sum.

If the unsigned sum cannot be represented in 32 bits, then RT is updated with a value which is "saturated" to the maximum representable value of $2^{32} - 1$.

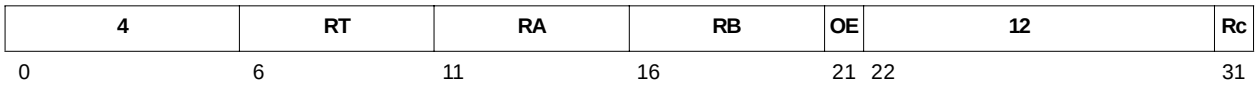
Registers Altered

- RT
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE contains 1

Architecture Note

This instruction is implementation-specific and programs which use this instruction may not be portable to other PowerPC Book-E implementations. See ~~"Instruction Set Portability" on page 28-2~~*Instruction Set Portability* on page 894.

machhwu	RT, RA, RB	OE=0, Rc=0
machhwu.	RT, RA, RB	OE=0, Rc=1
machhwuo	RT, RA, RB	OE=1, Rc=0
machhwuo.	RT, RA, RB	OE=1, Rc=1



$prod_{0:31} \leftarrow (RA)_{0:15} \times (RB)_{0:15}$ unsigned

$temp_{0:32} \leftarrow prod_{0:31} + (RT)$

$(RT) \leftarrow temp_{1:32}$

The high-order halfword of RA is multiplied by the high-order halfword of RB. The unsigned product is summed with the contents of RT and RT is updated with the low-order 32 bits of the unsigned sum.

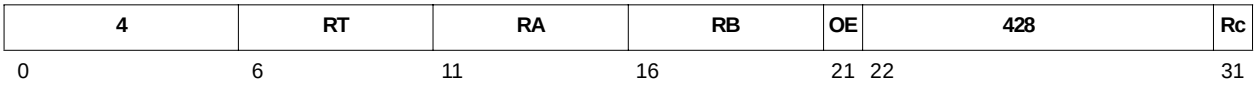
Registers Altered

- RT
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE contains 1

Architecture Note

This instruction is implementation-specific and programs which use this instruction may not be portable to other PowerPC Book-E implementations. See ~~"Instruction Set Portability" on page 28-2~~*Instruction Set Portability* on page 894.

maclhw	RT, RA, RB	OE=0, Rc=0
maclhw.	RT, RA, RB	OE=0, Rc=1
maclhwo	RT, RA, RB	OE=1, Rc=0
maclhwo.	RT, RA, RB	OE=1, Rc=1



$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{16:31}$ signed

$temp_{0:32} \leftarrow prod_{0:31} + (RT)$

$(RT) \leftarrow temp_{1:32}$

The low-order halfword of RA is multiplied by the low-order halfword of RB. The signed product is summed with the contents of RT and RT is updated with the low-order 32 bits of the signed sum.

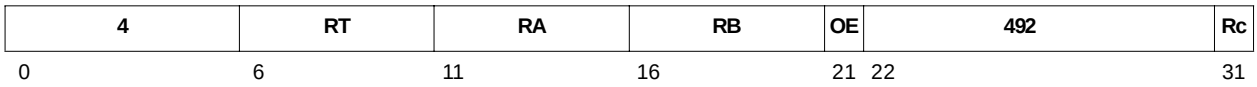
Registers Altered

- RT
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE contains 1

Architecture Note

This instruction is implementation-specific and programs which use this instruction may not be portable to other PowerPC Book-E implementations. See ~~"Instruction Set Portability" on page 28-2~~*Instruction Set Portability* on page 894.

maclhws	RT, RA, RB	OE=0, Rc=0
maclhws.	RT, RA, RB	OE=0, Rc=1
maclhwso	RT, RA, RB	OE=1, Rc=0
maclhwso.	RT, RA, RB	OE=1, Rc=1



$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{16:31}$ signed

$temp_{0:32} \leftarrow prod_{0:31} + (RT)$

if $((prod_0 = RT_0) \wedge (RT_0 \neq temp_1))$ then $(RT) \leftarrow (RT_0 \,||\, {}^{31}(\neg RT_0))$

else $(RT) \leftarrow temp_{1:32}$

The low-order halfword of RA is multiplied by the low-order halfword of RB. The signed product is summed with the contents of RT.

If the signed sum can be represented in 32 bits, then RT is updated with the low-order 32 bits of the signed sum.

If the signed sum cannot be represented in 32 bits, then RT is updated with a value which is "saturated" to the nearest representable value. That is, if the signed sum is less than -2^{31}, then RT is updated with -2^{31}. Likewise, if the signed sum is greater than $2^{31} - 1$, then RT is updated with $2^{31} - 1$.

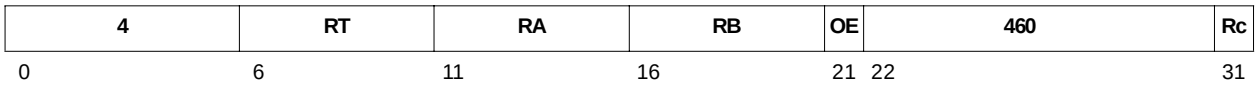
Registers Altered

- RT
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE contains 1

Architecture Note

This instruction is implementation-specific and programs which use this instruction may not be portable to other PowerPC Book-E implementations. See ~~"Instruction Set Portability" on page 28-2~~*Instruction Set Portability* on page 894.

maclhwsu	RT, RA, RB	OE=0, Rc=0
maclhwsu.	RT, RA, RB	OE=0, Rc=1
maclhwsuo	RT, RA, RB	OE=1, Rc=0
maclhwsuo.	RT, RA, RB	OE=1, Rc=1



$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{16:31}$ unsigned

$temp_{0:32} \leftarrow prod_{0:31} + (RT)$

$(RT) \leftarrow (temp_{1:32} \vee {}^{32}temp_{0})$

The low-order halfword of RA is multiplied by the low-order halfword of RB. The unsigned product is summed with the contents of RT.

If the unsigned sum can be represented in 32 bits, then RT is updated with the low-order 32 bits of the unsigned sum.

If the unsigned sum cannot be represented in 32 bits, then RT is updated with a value which is "saturated" to the maximum representable value of $2^{32} - 1$.

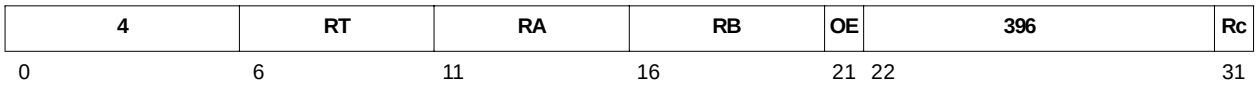
Registers Altered

- RT
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE contains 1

Architecture Note

This instruction is implementation-specific and programs which use this instruction may not be portable to other PowerPC Book-E implementations. See ~~"Instruction Set Portability" on page 28-2~~*Instruction Set Portability* on page 894.

maclhwu	RT, RA, RB	OE=0, Rc=0
maclhwu.	RT, RA, RB	OE=0, Rc=1
maclhwuo	RT, RA, RB	OE=1, Rc=0
maclhwuo.	RT, RA, RB	OE=1, Rc=1



$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{16:31}$ unsigned

$temp_{0:32} \leftarrow prod_{0:31} + (RT)$

$(RT) \leftarrow temp_{1:32}$

The low-order halfword of RA is multiplied by the low-order halfword of RB. The unsigned product is summed with the contents of RT and RT is updated with the low-order 32 bits of the unsigned sum.

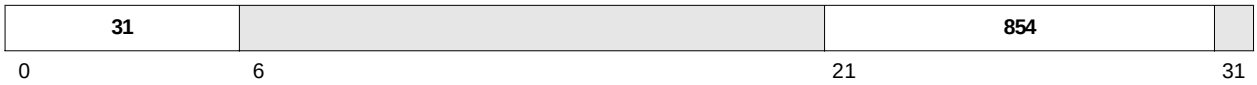
Registers Altered

- RT
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE contains 1

Architecture Note

This instruction is implementation-specific and programs which use this instruction may not be portable to other PowerPC Book-E implementations. See ~~"Instruction Set Portability" on page 28-2~~*Instruction Set Portability* on page 894.

mbar



The **mbar** instruction ensures that all loads and stores preceding **mbar** complete with respect to main storage before any loads and stores following **mbar** access main storage.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- None

Invalid Instruction Forms

- Reserved fields

Programming Note

Architecturally, **mbar** merely orders storage accesses, and does not perform execution nor context synchronization (see ~~"Synchronization" on page 4-62~~*Synchronization* on page 201). Therefore, non-storage access instructions after **mbar** could complete before the storage access instructions which were executed prior to **mbar** have actually completed their storage accesses. The **msync** instruction, on the other hand, *is* execution synchronizing, and *does* guarantee that all storage accesses initiated by instructions executed prior to the **msync** have completed before any instructions after the **msync** begin execution. However, the PPC440GP implements the **mbar** instruction identically to the **msync** instruction, and thus both are execution synchronizing.

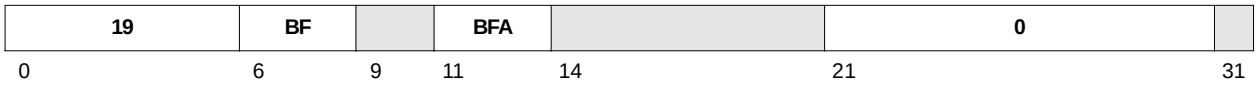
Software should nevertheless use the correct instruction (**mbar** or **msync**) as called for by the specific ordering and synchronizing requirements of the application, in order to guarantee portability to other implementations.

See ~~"Storage Ordering and Synchronization" on page 4-64~~*Storage Ordering and Synchronization* on page 203 for additional information on the use of the **msync** and **mbar** instructions.

Architecture Note

mbar replaces the PowerPC **eieio** instruction. **mbar** uses the same opcode as **eieio**; PowerPC applications which used **eieio** will get the function of **mbar** when executed on a PowerPC Book-E implementation. **mbar** is architecturally "stronger" than **eieio**, in that **eieio** forced separate ordering amongst different *categories* of storage accesses, while **mbar** forces such ordering amongst *all* storage accesses as a single category.

mcrf BF, BFA



m ← BFA
n ← BF
(CR[CRn]) ← (CR[CRm])

The contents of the CR field specified by the BFA field are placed into the CR field specified by the BF field.

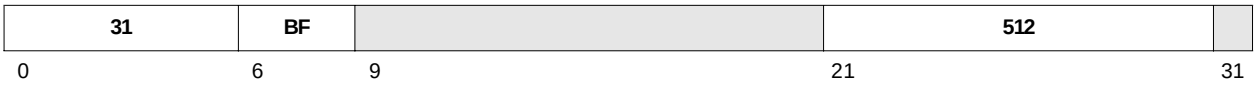
Registers Altered

- CR[CR*n*] where *n* is specified by the BF field.

Invalid Instruction Forms

- Reserved fields

mcrxr BF



n ← BF
(CR[CRn]) ← $XER_{0:3}$
$XER_{0:3}$ ← $^{4}0$

The contents of $XER_{0:3}$ are placed into the CR field specified by the BF field. $XER_{0:3}$ are then set to 0.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- CR[CR*n*] where *n* is specified by the BF field.
- XER[SO, OV, CA]

Invalid Instruction Forms

- Reserved fields

mfcr RT

(RT) ← (CR)

The contents of the CR are placed into register RT.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- RT

Invalid Instruction Forms

- Reserved fields

mfdcr RT, DCRN

$DCRN \leftarrow DCRF_{5:9} \parallel DCRF_{0:4}$
$(RT) \leftarrow (DCR(DCRN))$

The contents of the DCR specified by the DCRF field are placed into register RT.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- RT

Invalid Instruction Forms

- Reserved fields

Programming ~~Note~~Notes

Execution of this instruction is privileged.

The DCR number (DCRN) specified in the assembler language coding of the **mfdcr** instruction refers to a DCR number. The assembler handles the unusual register number encoding to generate the DCRF field.

Architecture Note

The specific numbers and definitions of any DCRs are outside the scope of both the PowerPC Book-E architecture and the PPC440GP. Any DCRs are defined as part of the chip-level product incorporating the PPC440GP.

mfmsr RT

(RT) ← (MSR)

The contents of the MSR are placed into register RT.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- RT

Invalid Instruction Forms

- Reserved fields

Programming Note

Execution of this instruction is privileged.

mfspr RT, SPRN

SPRN ← $SPRF_{5:9}$ || $SPRF_{0:4}$
(RT) ← (SPR(SPRN))

The contents of the SPR specified by the SPRF field are placed into register RT. See ~~"Special Purpose Registers Sorted by SPR Number" on page 29-4~~*Special Purpose Registers Sorted by SPR Number* on page 1095 for a listing of SPR mnemonics and corresponding SPRN values.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- RT

Invalid Instruction Forms

- Reserved fields
- Invalid SPRF values

Programming Note

Execution of this instruction is privileged if instruction bit 11 contains 1. See ~~"Privileged SPRs" on page 4-61~~*Privileged SPRs* on page 200 for a list of privileged SPRs.

The SPR number (SPRN) specified in the assembler language coding of the **mfspr** instruction refers to an SPR number. The assembler handles the unusual register number encoding to generate the SPRF field.

Table 28-19. Extended Mnemonics for mfspr

Mnemonic	Operands	Function
mfccr0 mfcsrr0 mfcsrr1 **mfctr** **mfdac1** mfdac2 **mfdbcr0** **mfdbcr1** **mfdbcr2** **mfdbdr** **mfdbsr** mfdcdbtrh mfdcdbtrl mfdear mfdec mfdnv0 mfdnv1 mfdnv2 mfdnv3 mfdtv0 mfdtv1 mfdtv2 mfdtv3 mfdvc1 mfdvc2 mfdvlim **mfesr** **mfiac1** mfiac2 mfiac3 mfiac4 **mficdbdr** mficdbtrh mficdbtrl mfinv0 mfinv1 mfinv2 mfinv3 mfitv0 mfitv1 mfitv2 mfitv3 **mfivlim** mfivor0 mfivor1 mfivor2 mfivor3 mfivor4	RT	Move from special purpose register SPRN. *Extended mnemonic for* **mfspr RT,SPRN** See ~~"Special Purpose Registers Sorted by SPR Number" on page 29-4~~*Special Purpose Registers Sorted by SPR Number* on page 1095 for a list of valid SPRN values.

Table 28-19. Extended Mnemonics for mfspr (continued)

Mnemonic	Operands	Function
mfivor5		
mfivor6		
mfivor7		
mfivor8		
mfivor9		
mfivor10		
mfivor11		
mfivor12		
mfivor13		
mfivor14		
mfivor15		
mfivpr		
mflr		
mfmmucr		
mfpid		
mfpir		
mfpvr		
mfsprg0		
mfsprg1		
mfsprg2		
mfsprg3		
mfsprg4		
mfsprg5		
mfsprg6		
mfsprg7		
mfsrr0		
mfsrr1		
mftbl		
mftbu		
mftcr		
mftsr		
mfusprg0		
mfxer		

msync

The **msync** instruction guarantees that all instructions initiated by the processor preceding **msync** will complete before **msync** completes, and that no subsequent instructions will be initiated by the processor until after **msync** completes. **msync** also will not complete until all storage accesses associated with instructions preceding **msync** have completed.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- None.

Invalid Instruction Forms

- Reserved fields

Programming Notes

The **msync** instruction is execution synchronizing (see ~~"Execution Synchronization" on page 4-64~~*Execution Synchronization* on page 202), and guarantees that all storage accesses initiated by instructions executed prior to the **msync** have completed before any instructions after the **msync** begin execution. On the other hand, architecturally the **mbar** instruction merely *orders* storage accesses, and does *not* perform execution synchronization. Therefore, non-storage access instructions after **mbar** *could* complete before the storage access instructions which were executed prior to **mbar** have actually completed their storage accesses. However, the PPC440GP implements the **mbar** instruction identically to the **msync** instruction, and thus both are execution synchronizing.

Software should nevertheless use the correct instruction (**mbar** or **msync**) as called for by the specific ordering and synchronizing requirements of the application, in order to guarantee portability to other implementations.

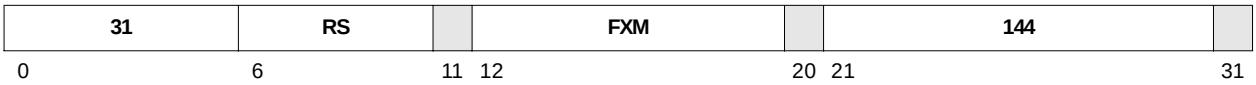
See ~~"Storage Ordering and Synchronization" on page 4-64~~*Storage Ordering and Synchronization* on page 203 for additional information on the use of the **msync** and **mbar** instructions.

Architecture Note

mbar replaces the PowerPC **eieio** instruction. **mbar** uses the same opcode as **eieio**; PowerPC applications which used **eieio** will get the function of **mbar** when executed on a PowerPC Book-E implementation. **mbar** is architecturally "stronger" than **eieio**, in that **eieio** forced separate ordering amongst different *categories* of storage accesses, while **mbar** forces such ordering amongst *all* storage accesses as a single category.

msync replaces the PowerPC **sync** instruction. **msync** uses the same opcode as **sync**; PowerPC applications which used **sync** get the function of **msync** when executed on a PowerPC Book-E implementation. **msync** is architecturally identical to the version of **sync** specified by an earlier version of the PowerPC architecture.

mtcrf FXM, RS



mask $\leftarrow$ $^{4}(FXM_0)$ || $^{4}(FXM_1)$ || ... || $^{4}(FXM_6)$ || $^{4}(FXM_7)$
(CR) $\leftarrow$ ((RS) $\wedge$ mask) $\vee$ ((CR) $\wedge$ $\neg$mask)

Some or all of the contents of register RS are placed into the CR as specified by the FXM field.

Each bit in the FXM field controls the copying of 4 bits in register RS into the corresponding bits in the CR. The correspondence between the bits in the FXM field and the bit copying operation is shown in the following table:

FXM Bit Number	CR Bits Affected
0	0:3
1	4:7
2	8:11
3	12:15
4	16:19
5	20:23
6	24:27
7	28:31

Table 28-20. FXM and CR Bits

FXM Bit Number	CR Bits Affected
0	0:3
1	4:7
2	8:11
3	12:15
4	16:19
5	20:23
6	24:27
7	28:31

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- CR

Invalid Instruction Forms

- Reserved fields

Table 28-21. Extended Mnemonics for mtcrf

Mnemonic	Operands	Function
mtcr	RS	Move to CR. *Extended mnemonic for* **mtcrf 0xFF,RS**

mtdcr DCRN, RS

$DCRN \leftarrow DCRF_{5:9} \,||\, DCRF_{0:4}$
$(DCR(DCRN)) \leftarrow (RS)$

The contents of register RS are placed into the DCR specified by the DCRF field.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- DCR(DCRN)

Invalid Instruction Forms

- Reserved fields

Programming Note

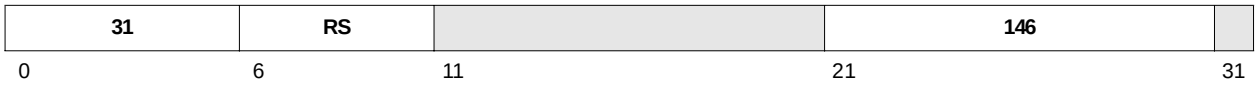
Execution of this instruction is privileged.

The DCR number (DCRN) specified in the assembler language coding of the **mtdcr** instruction refers to a DCR number. The assembler handles the unusual register number encoding to generate the DCRF field.

Architecture Note

The specific numbers and definitions of any DCRs are outside the scope of both the PowerPC Book-E architecture and the PPC440GP. Any DCRs are defined as part of the chip-level product incorporating the PPC440GP.

mtmsr RS



(MSR) ← (RS)

The contents of register RS are placed into the MSR.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- MSR

Invalid Instruction Forms

- Reserved fields

Programming Note

The **mtmsr** instruction is privileged and execution synchronizing (see ~~"Execution Synchronization" on page 4-64~~*Execution Synchronization* on page 202).

mtspr SPRN, RS

SPRN ← $SPRF_{5:9}$ || $SPRF_{0:4}$
(SPR(SPRN)) ← (RS)

The contents of register RS are placed into register RT. See ~~"Special Purpose Registers Sorted by SPR Number" on page 29-4~~*Special Purpose Registers Sorted by SPR Number* on page 1095 for a listing of SPR mnemonics and corresponding SPRN values.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- SPR(SPRN)

Invalid Instruction Forms

- Reserved fields
- Invalid SPRF values

Programming Note

Execution of this instruction is privileged if instruction bit 11 contains 1. See ~~"Privileged SPRs" on page 4-61~~*Privileged SPRs* on page 200 for a list of privileged SPRs.

The SPR number (SPRN) specified in the assembler language coding of the **mtspr** instruction refers to an SPR number. The assembler handles the unusual register number encoding to generate the SPRF field.

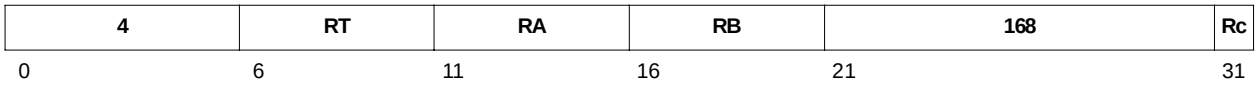
Table 28-22. Extended Mnemonics for mtspr

Mnemonic	Operands	Function
mtccr0 mtcsrr0 mtcsrr1 **mtctr** **mtdac1** mtdac2 **mtdbcr0** **mtdbcr1** **mtdbcr2** **mtdbdr** **mtdbsr** mtdear mtdec mtdecar mtdnv0 mtdnv1 mtdnv2 mtdnv3 mtdtv0 mtdtv1 mtdtv2 mtdtv3 mtdvc1 mtdvc2 mtdvlim **mtesr** **mtiac1** mtiac2 mtiac3 mtiac4 mtinv0 mtinv1 mtinv2 mtinv3 mtitv0 mtitv1 mtitv2 mtitv3 **mtivlim** mtivor0 mtivor1 mtivor2 mtivor3 mtivor4	RT	Move to special purpose register SPRN. *Extended mnemonic for* **mtspr RT,SPRN** See ~~"Special Purpose Registers Sorted by SPR Number" on page 29-4~~*Special Purpose Registers Sorted by SPR Number* on page 1095 for a list of valid SPRN values.

Table 28-22. Extended Mnemonics for mtspr (continued)

Mnemonic	Operands	Function
mtivor5		
mtivor6		
mtivor7		
mtivor8		
mtivor9		
mtivor10		
mtivor11		
mtivor12		
mtivor13		
mtivor14		
mtivor15		
mtivpr		
mtlr		
mtmmucr		
mtpid		
mtsprg0		
mtsprg1		
mtsprg2		
mtsprg3		
mtsprg4		
mtsprg5		
mtsprg6		
mtsprg7		
mtsrr0		
mtsrr1		
mttbl		
mttbu		
mttcr		
mttsr		
mtusprg0		
mtxer		

mulchw	RT, RA, RB	Rc=0
mulchw.	RT, RA, RB	Rc=1



$(RT)_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{0:15}$ signed

The low-order halfword of RA is multiplied by the high-order halfword of RB, considering both source operands as signed integers. The 32-bit result is placed into register RT.

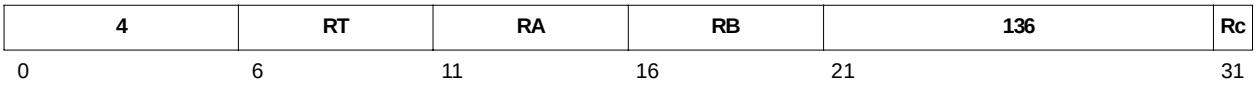
Registers Altered

- RT
- CR[CR0] if Rc contains 1

Architecture Note

This instruction is implementation-specific and programs which use this instruction may not be portable to other PowerPC Book-E implementations. See ~~"Instruction Set Portability" on page 28-2~~ *Instruction Set Portability* on page 894.

mulchwu	RT, RA, RB	Rc=0
mulchwu.	RT, RA, RB	Rc=1



$(RT)_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{0:15}$ unsigned

The low-order halfword of RA is multiplied by the high-order halfword of RB, considering both source operands as unsigned integers. The 32-bit result is placed into register RT.

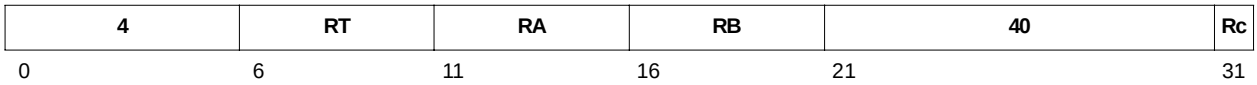
Registers Altered

- RT
- CR[CR0] if Rc contains 1

Architecture Note

This instruction is implementation-specific and programs which use this instruction may not be portable to other PowerPC Book-E implementations. See ~~"Instruction Set Portability" on page 28-2~~*Instruction Set Portability* on page 894.

mulhhw RT, RA, RB Rc=0
mulhhw. RT, RA, RB Rc=1



$(RT)_{0:31} \leftarrow (RA)_{0:15} \times (RB)_{0:15}$ signed

The high-order halfword of RA is multiplied by the high-order halfword of RB, considering both source operands as signed integers. The 32-bit result is placed into register RT.

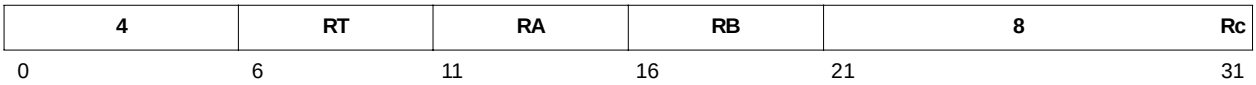
Registers Altered

- RT
- CR[CR0] if Rc contains 1

Architecture Note

This instruction is implementation-specific and programs which use this instruction may not be portable to other PowerPC Book-E implementations. See ~~"Instruction Set Portability" on page 28-2~~ *Instruction Set Portability* on page 894.

mulhhwu RT, RA, RB Rc=0
mulhhwu. RT, RA, RB Rc=1



$(RT)_{0:31} \leftarrow (RA)_{0:15} \times (RB)_{0:15}$ unsigned

The high-order halfword of RA is multiplied by the high-order halfword of RB, considering both source operands as unsigned integers. The 32-bit result is placed into register RT.

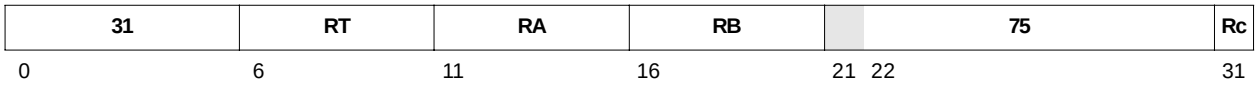
Registers Altered

- RT
- CR[CR0] if Rc contains 1

Architecture Note

This instruction is implementation-specific and programs which use this instruction may not be portable to other PowerPC Book-E implementations. See ~~"Instruction Set Portability" on page 28-2~~*Instruction Set Portability* on page 894.

mulhw	RT, RA, RB	Rc=0
mulhw.	RT, RA, RB	Rc=1



$prod_{0:63} \leftarrow$ (RA) $\times$ (RB) signed
(RT) $\leftarrow prod_{0:31}$

The 64-bit signed product of registers RA and RB is formed. The most significant 32 bits of the result is placed into register RT.

Registers Altered

- RT
- CR[CR0] if Rc contains 1

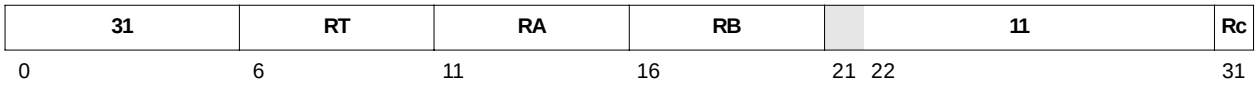
Programming Note

The most significant 32 bits of the product, unlike the least significant 32 bits, may differ depending on whether the registers RA and RB are interpreted as signed or unsigned quantities. **mulhw** generates the correct result when these operands are interpreted as signed quantities. **mulhwu** generates the correct result when these operands are interpreted as unsigned quantities.

Invalid Instruction Forms

- Reserved fields

mulhwu	RT, RA, RB	Rc=0
mulhwu.	RT, RA, RB	Rc=1



$prod_{0:63} \leftarrow (RA) \times (RB)$ unsigned
$(RT) \leftarrow prod_{0:31}$

The 64-bit unsigned product of registers RA and RB is formed. The most significant 32 bits of the result are placed into register RT.

Registers Altered

- RT
- CR[CR0] if Rc contains 1

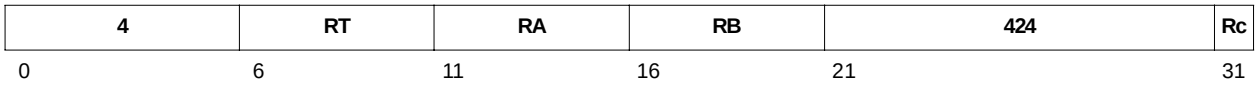
Programming Note

The most significant 32 bits of the product, unlike the least significant 32 bits, may differ depending on whether the registers RA and RB are interpreted as signed or unsigned quantities. The **mulhw** instruction generates the correct result when these operands are interpreted as signed quantities. The **mulhwu** instruction generates the correct result when these operands are interpreted as unsigned quantities.

Invalid Instruction Forms

- Reserved fields

mullhw	RT, RA, RB	Rc=0
mullhw.	RT, RA, RB	Rc=1



$(RT)_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{16:31}$ signed

The low-order halfword of RA is multiplied by the low-order halfword of RB, considering both source operands as signed integers. The 32-bit result is placed into register RT.

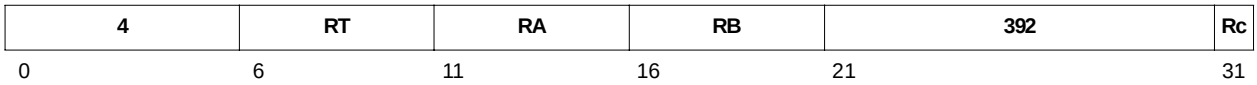
Registers Altered

- RT
- CR[CR0] if Rc contains 1

Architecture Note

This instruction is implementation-specific and programs which use this instruction may not be portable to other PowerPC Book-E implementations. See ~~"Instruction Set Portability" on page 28-2~~ *Instruction Set Portability* on page 894.

mullhwu	RT, RA, RB	~~OE=0,~~ Rc=0
mullhwu.	RT, RA, RB	~~OE=0,~~ Rc=1



$(RT)_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{16:31}$ unsigned

The low-order halfword of RA is multiplied by the low-order halfword of RB, considering both source operands as unsigned integers. The 32-bit result is placed into register RT.

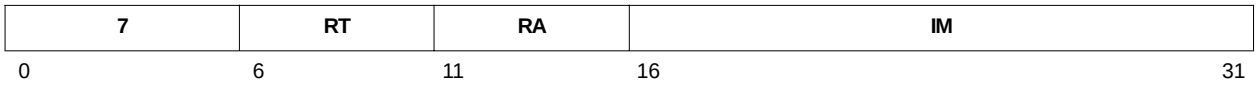
Registers Altered

- RT
- CR[CR0] if Rc contains 1

Architecture Note

This instruction is implementation-specific and programs which use this instruction may not be portable to other PowerPC Book-E implementations. See ~~"Instruction Set Portability" on page 28-2~~*Instruction Set Portability* on page 894.

mulli RT, RA, IM



$prod_{0:47} \leftarrow (RA) \times IM$
$(RT) \leftarrow prod_{16:47}$

The 48-bit product of register RA and the 16-bit IM field is formed. The least significant 32 bits of the product are placed into register RT.

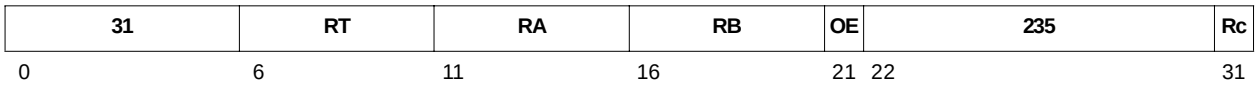
Registers Altered

- RT

Programming Note

The least significant 32 bits of the product are correct, regardless of whether register RA and field IM are interpreted as signed or unsigned numbers.

mullw	RT, RA, RB	OE=0, Rc=0
mullw.	RT, RA, RB	OE=0, Rc=1
mullwo	RT, RA, RB	OE=1, Rc=0
mullwo.	RT, RA, RB	OE=1, Rc=1



$prod_{0:63} \leftarrow$ (RA) $\times$ (RB) signed
(RT) $\leftarrow prod_{32:63}$

The 64-bit signed product of register RA and register RB is formed. The least significant 32 bits of the result is placed into register RT.

If the signed product cannot be represented in 32 bits and OE=1, XER[SO, OV] are set to 1.

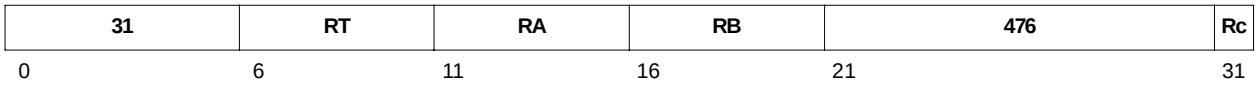
Registers Altered

- RT
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE=1

Programming Note

The least significant 32 bits of the product are correct, regardless of whether register RA and register RB are interpreted as signed or unsigned numbers. The overflow indication, however, is calculated specifically for a 64-bit signed product, and is dependent upon interpretation of the source operands as signed numbers.

nand	RA, RS, RB	Rc=0
nand.	RA, RS, RB	Rc=1



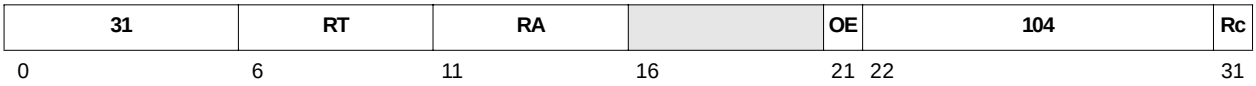
$(RA) \leftarrow \neg((RS) \wedge (RB))$

The contents of register RS is ANDed with the contents of register RB; the ones complement of the result is placed into register RA.

Registers Altered

- RA
- CR[CR0] if Rc contains 1

neg	RT, RA	OE=0, Rc=0
neg.	RT, RA	OE=0, Rc=1
nego	RT, RA	OE=1, Rc=0
nego.	RT, RA	OE=1, Rc=1



(RT) $\leftarrow$ $\neg$(RA) + 1

The twos complement of the contents of register RA are placed into register RT.

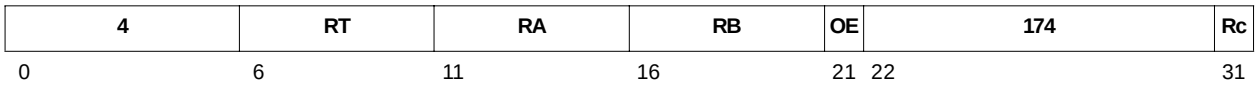
Registers Altered

- RT
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE=1

Invalid Instruction Forms

- Reserved fields

nmacchw	RT, RA, RB	OE=0, Rc=0
nmacchw.	RT, RA, RB	OE=0, Rc=1
nmacchwo	RT, RA, RB	OE=1, Rc=0
nmacchwo.	RT, RA, RB	OE=1, Rc=1



$nprod_{0:31} \leftarrow -((RA)_{16:31} \times (RB)_{0:15})$ signed

$temp_{0:32} \leftarrow nprod_{0:31} + (RT)$

$(RT) \leftarrow temp_{1:32}$

The low-order halfword of RA is multiplied by the high-order halfword of RB. The signed product is subtracted from the contents of RT and RT is updated with the low-order 32 bits of the result.

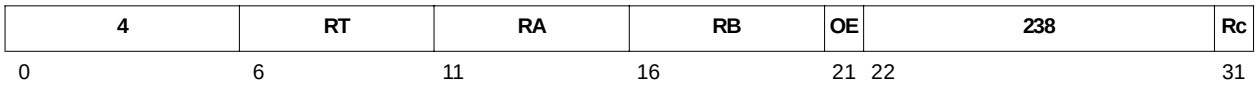
Registers Altered

- RT
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE contains 1

Architecture Note

This instruction is implementation-specific and programs which use this instruction may not be portable to other PowerPC Book-E implementations. See ~~"Instruction Set Portability" on page 28-2~~*Instruction Set Portability* on page 894.

nmacchws	RT, RA, RB	OE=0, Rc=0
nmacchws.	RT, RA, RB	OE=0, Rc=1
nmacchwso	RT, RA, RB	OE=1, Rc=0
nmacchwso.	RT, RA, RB	OE=1, Rc=1



$nprod_{0:31} \leftarrow -((RA)_{16:31} \times (RB)_{0:15}$ signed

$temp_{0:32} \leftarrow nprod_{0:31} + (RT)$

if $((nprod_0 = RT_0) \wedge (RT_0 \neq temp_1))$ then $(RT) \leftarrow (RT_0 \,||\, ^{31}(\neg RT_0))$

else $(RT) \leftarrow temp_{1:32}$

The low-order halfword of RA is multiplied by the high-order halfword of RB. The signed product is subtracted from the contents of RT.

If the result of the subtraction can be represented in 32 bits, then RT is updated with the low-order 32 bits of the result.

If the result of the subtraction cannot be represented in 32 bits, then RT is updated with a value which is "saturated" to the nearest representable value. That is, if the result is less than -2^{31}, then RT is updated with -2^{31}. Likewise, if the result is greater than $2^{31} - 1$, then RT is updated with $2^{31} - 1$.

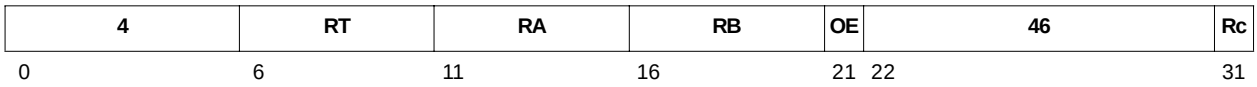
Registers Altered

- RT
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE contains 1

Architecture Note

This instruction is implementation-specific and programs which use this instruction may not be portable to other PowerPC Book-E implementations. See ~~"Instruction Set Portability" on page 28-2~~*Instruction Set Portability* on page 894.

nmachhw	RT, RA, RB	OE=0, Rc=0
nmachhw.	RT, RA, RB	OE=0, Rc=1
nmachhwo	RT, RA, RB	OE=1, Rc=0
nmachhwo.	RT, RA, RB	OE=1, Rc=1



$nprod_{0:31} \leftarrow -((RA)_{0:15} \times (RB)_{0:15})$ signed

$temp_{0:32} \leftarrow nprod_{0:31} + (RT)$

$(RT) \leftarrow temp_{1:32}$

The high-order halfword of RA is multiplied by the high-order halfword of RB. The signed product is subtracted from the contents of RT and RT is updated with the low-order 32 bits of the result.

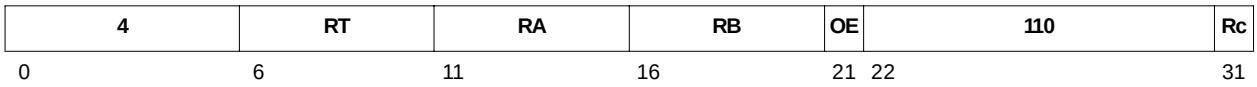
Registers Altered

- RT
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE contains 1

Architecture Note

This instruction is implementation-specific and programs which use this instruction may not be portable to other PowerPC Book-E implementations. See ~~"Instruction Set Portability" on page 28-2~~*Instruction Set Portability* on page 894.

nmachhws	RT, RA, RB	OE=0, Rc=0
nmachhws.	RT, RA, RB	OE=0, Rc=1
nmachhwso	RT, RA, RB	OE=1, Rc=0
nmachhwso.	RT, RA, RB	OE=1, Rc=1



$nprod_{0:31} \leftarrow -((RA)_{0:15} \times (RB)_{0:15})$ signed

$temp_{0:32} \leftarrow nprod_{0:31} + (RT)$

if $((nprod_0 = RT_0) \wedge (RT_0 \neq temp_1))$ then $(RT) \leftarrow (RT_0 \,||\, ^{31}(\neg RT_0))$

else $(RT) \leftarrow temp_{1:32}$

The high-order halfword of RA is multiplied by the high-order halfword of RB. The signed product is subtracted from the contents of RT.

If the result of the subtraction can be represented in 32 bits, then RT is updated with the low-order 32 bits of the result.

If the result of the subtraction cannot be represented in 32 bits, then RT is updated with a value which is "saturated" to the nearest representable value. That is, if the result is less than -2^{31}, then RT is updated with -2^{31}. Likewise, if the result is greater than $2^{31} - 1$, then RT is updated with $2^{31} - 1$.

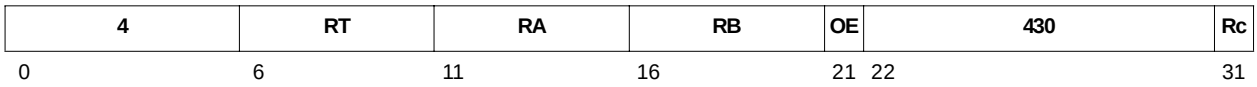
Registers Altered

- RT
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE contains 1

Architecture Note

This instruction is implementation-specific and programs which use this instruction may not be portable to other PowerPC Book-E implementations. See ~~"Instruction Set Portability" on page 28-2~~*Instruction Set Portability* on page 894.

nmaclhw	RT, RA, RB	OE=0, Rc=0
nmaclhw.	RT, RA, RB	OE=0, Rc=1
nmaclhwo	RT, RA, RB	OE=1, Rc=0
nmaclhwo.	RT, RA, RB	OE=1, Rc=1



$nprod_{0:31} \leftarrow -((RA)_{16:31} \times (RB)_{16:31})$ signed

$temp_{0:32} \leftarrow nprod_{0:31} + (RT)$

$(RT) \leftarrow temp_{1:32}$

The low-order halfword of RA is multiplied by the low-order halfword of RB. The signed product is subtracted from the contents of RT and RT is updated with the low-order 32 bits of the result.

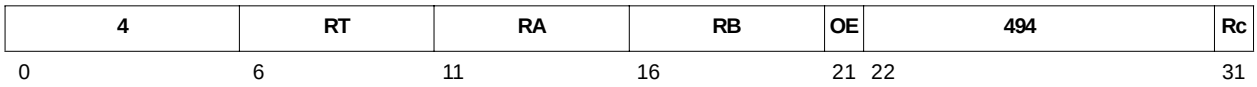
Registers Altered

- RT
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE contains 1

Architecture Note

This instruction is implementation-specific and programs which use this instruction may not be portable to other PowerPC Book-E implementations. See ~~"Instruction Set Portability" on page 28-2~~*Instruction Set Portability* on page 894.

nmaclhws	RT, RA, RB	OE=0, Rc=0
nmaclhws.	RT, RA, RB	OE=0, Rc=1
nmaclhwso	RT, RA, RB	OE=1, Rc=0
nmaclhwso.	RT, RA, RB	OE=1, Rc=1



$nprod_{0:31} \leftarrow -((RA)_{16:31} \times (RB)_{16:31})$ signed

$temp_{0:32} \leftarrow nprod_{0:31} + (RT)$

if $((nprod_0 = RT_0) \wedge (RT_0 \neq temp_1))$ then $(RT) \leftarrow (RT_0 \,||\, ^{31}(\neg RT_0))$

else $(RT) \leftarrow temp_{1:32}$

The low-order halfword of RA is multiplied by the low-order halfword of RB. The signed product is subtracted from the contents of RT.

If the result of the subtraction can be represented in 32 bits, then RT is updated with the low-order 32 bits of the result.

If the result of the subtraction cannot be represented in 32 bits, then RT is updated with a value which is "saturated" to the nearest representable value. That is, if the result is less than -2^{31}, then RT is updated with -2^{31}. Likewise, if the result is greater than $2^{31} - 1$, then RT is updated with $2^{31} - 1$.

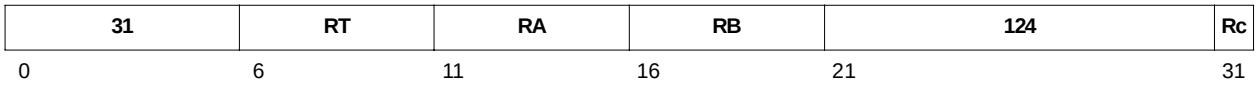
Registers Altered

- RT
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE contains 1

Architecture Note

This instruction is implementation-specific and programs which use this instruction may not be portable to other PowerPC Book-E implementations. See ~~"Instruction Set Portability" on page 28-2~~*Instruction Set Portability* on page 894.

nor	RA, RS, RB	Rc=0
nor.	RA, RS, RB	Rc=1



(RA) $\leftarrow$ $\neg$((RS) $\vee$ (RB))

The contents of register RS is ORed with the contents of register RB; the ones complement of the result is placed into register RA.

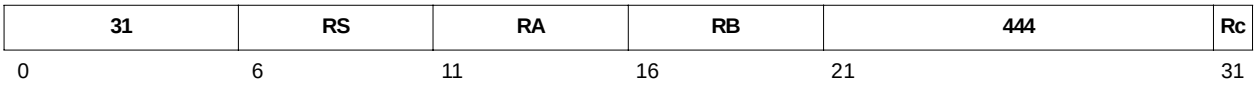
Registers Altered

- RA
- CR[CR0] if Rc contains 1

Table 28-23. Extended Mnemonics for nor, nor.

Mnemonic	Operands	Function	Other Registers Altered
not	RA, RS	Complement register. (RA) $\leftarrow$ $\neg$(RS) *Extended mnemonic for* **nor RA,RS,RS**	
not.		*Extended mnemonic for* **nor. RA,RS,RS**	CR[CR0]

or	RA, RS, RB	Rc=0
or.	RA, RS, RB	Rc=1



(RA) ← (RS) ∨ (RB)

The contents of register RS is ORed with the contents of register RB; the result is placed into register RA.

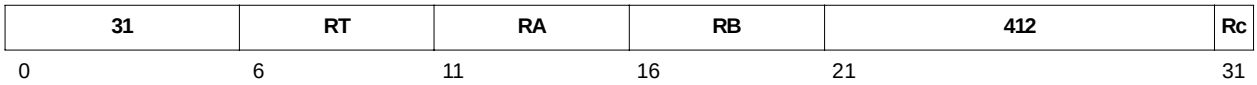
Registers Altered

- RA
- CR[CR0] if Rc contains 1

Table 28-24. Extended Mnemonics for or, or.

Mnemonic	Operands	Function	Other Registers Altered
mr	RT, RS	Move register. (RT) ← (RS) *Extended mnemonic for* **or RT,RS,RS**	
mr.		*Extended mnemonic for* **or. RT,RS,RS**	CR[CR0]

orc	RA, RS, RB	Rc=0
orc.	RA, RS, RB	Rc=1



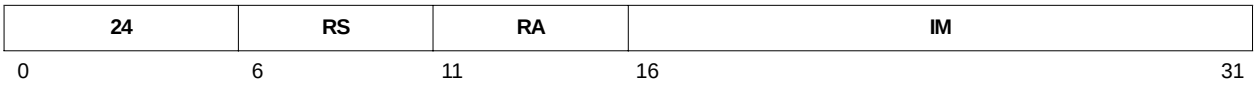
(RA) $\leftarrow$ (RS) $\vee$ $\neg$(RB)

The contents of register RS is ORed with the ones complement of the contents of register RB; the result is placed into register RA.

Registers Altered

- RA
- CR[CR0] if Rc contains 1

ori RA, RS, IM



(RA) $\leftarrow$ (RS) $\vee$ ($^{16}0$ || IM)

The IM field is extended to 32 bits by concatenating 16 0-bits on the left. Register RS is ORed with the extended IM field; the result is placed into register RA.

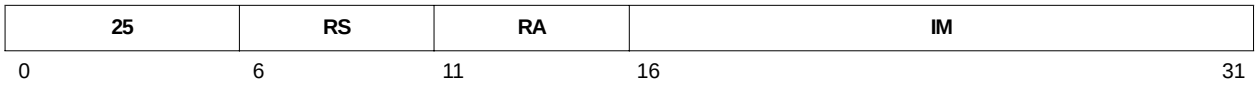
Registers Altered

- RA

Table 28-25. Extended Mnemonics for ori

Mnemonic	Operands	Function	Other Registers Changed
nop		Preferred no-op; triggers optimizations based on no-ops. *Extended mnemonic for* **ori 0,0,0**	

oris RA, RS, IM



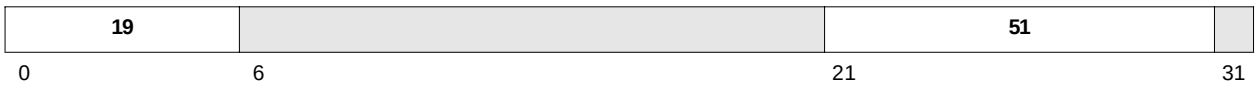
(RA) $\leftarrow$ (RS) $\vee$ (IM $\|$ $^{16}0$)

The IM Field is extended to 32 bits by concatenating 16 0-bits on the right. Register RS is ORed with the extended IM field and the result is placed into register RA.

Registers Altered

- RA

rfci



(PC) ← (~~CSSR0~~CSRR0)
(MSR) ← (~~CSSR1~~CSRR1)

This instruction is used to return from a critical interrupt.

The program counter (PC) is restored with the contents of ~~CSSR0~~ CSRR0 and the MSR is restored with the contents of ~~CSSR1~~CSRR1.

Instruction execution returns to the address contained in the PC.

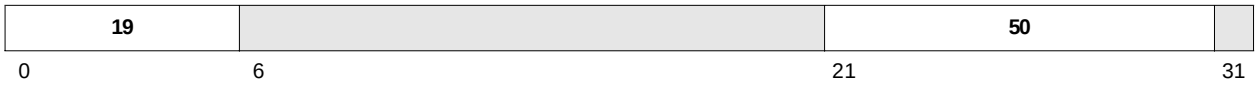
Registers Altered

- MSR

Programming Note

Execution of this instruction is privileged and context-synchronizing (see ~~"Context Synchronization" on page 4-62~~*Context Synchronization* on page 201).

rfi



(PC) ← (SRR0)
(MSR) ← (SRR1)

This instruction is used to return from a non-critical interrupt.

The program counter (PC) is restored with the contents of SRR0 and the MSR is restored with the contents of SRR1.

Instruction execution returns to the address contained in the PC.

Registers Altered

- MSR

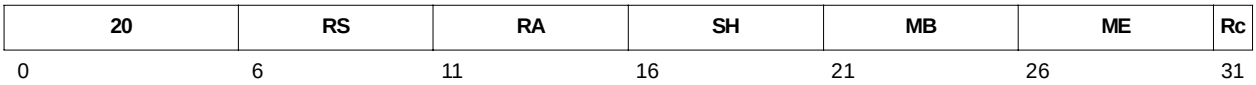
Invalid Instruction Forms

- Reserved fields

Programming Note

Execution of this instruction is privileged and context-synchronizing (see ~~"Context Synchronization" on page 4-62~~*Context Synchronization* on page 201).

rlwimi	RA, RS, SH, MB, ME	Rc=0
rlwimi.	RA, RS, SH, MB, ME	Rc=1



r ← ROTL((RS), SH)
m ← MASK(MB, ME)
(RA) ← (r ∧ m) ∨ ((RA) ∧ ¬m)

The contents of register RS are rotated left by the number of bit positions specified in the SH field. A mask is generated, having 1-bits starting at the bit position specified in the MB field and ending in the bit position specified by the ME field, with 0-bits elsewhere.

If the starting point of the mask is at a higher bit position than the ending point, the 1-bits portion of the mask wraps from the highest bit position back around to the lowest. The rotated data is inserted into register RA, in positions corresponding to the bit positions in the mask that contain a 1-bit.

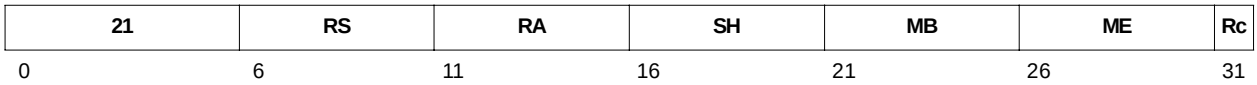
Registers Altered

- RA
- CR[CR0] if Rc contains 1

Table 28-26. Extended Mnemonics for rlwimi, rlwimi.

Mnemonic	Operands	Function	Other Registers Altered
inslwi	RA, RS, n, b	Insert from left immediate (n > 0). $(RA)_{b:b+n-1} \leftarrow (RS)_{0:n-1}$ *Extended mnemonic for* **rlwimi RA,RS,32–b,b,b+n–1**	
inslwi.		*Extended mnemonic for* **rlwimi. RA,RS,32–b,b,b+n–1**	CR[CR0]
insrwi	RA, RS, n, b	Insert from right immediate. (n > 0) $(RA)_{b:b+n-1} \leftarrow (RS)_{32-n:31}$ *Extended mnemonic for* **rlwimi RA,RS,32–b–n,b,b+n–1**	
insrwi.		*Extended mnemonic for* **rlwimi. RA,RS,32–b–n,b,b+n–1**	CR[CR0]

rlwinm	RA, RS, SH, MB, ME	Rc=0
rlwinm.	RA, RS, SH, MB, ME	Rc=1



r ← ROTL((RS), SH)
m ← MASK(MB, ME)
(RA) ← r ∧ m

The contents of register RS are rotated left by the number of bit positions specified in the SH field. A mask is generated, having 1-bits starting at the bit position specified in the MB field and ending in the bit position specified by the ME field with 0-bits elsewhere.

If the starting point of the mask is at a higher bit position than the ending point, the 1-bits portion of the mask wraps from the highest bit position back around to the lowest. The rotated data is ANDed with the generated mask; the result is placed into register RA.

Registers Altered

- RA
- CR[CR0] if Rc contains 1

Table 28-27. Extended Mnemonics for rlwinm, rlwinm.

Mnemonic	Operands	Function	Other Registers Altered
clrlwi	RA, RS, n	Clear left immediate. ($n < 32$) $(RA)_{0:n-1} \leftarrow {}^{n}0$ *Extended mnemonic for* **rlwinm RA,RS,0,n,31**	
clrlwi.		*Extended mnemonic for* **rlwinm. RA,RS,0,n,31**	CR[CR0]
clrlslwi	RA, RS, b, n	Clear left and shift left immediate. ($n \leq b < 32$) $(RA)_{b-n:31-n} \leftarrow (RS)_{b:31}$ $(RA)_{32-n:31} \leftarrow {}^{n}0$ $(RA)_{0:b-n-1} \leftarrow {}^{b-n}0$ *Extended mnemonic for* **rlwinm RA,RS,n,b–n,31–n**	
clrlslwi.		*Extended mnemonic for* **rlwinm. RA,RS,n,b–n,31–n**	CR[CR0]

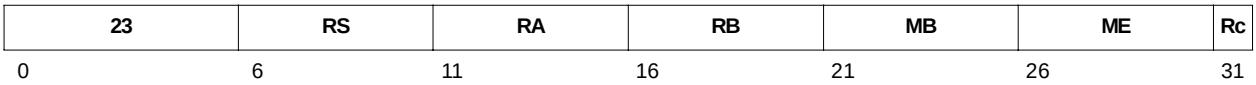
Table 28-27. Extended Mnemonics for rlwinm, rlwinm. (continued)

Mnemonic	Operands	Function	Other Registers Altered
clrrwi	RA, RS, n	Clear right immediate. ($n < 32$) $(RA)_{32-n:31} \leftarrow {}^{n}0$ *Extended mnemonic* for **rlwinm RA,RS,0,0,31–n**	
clrrwi.		*Extended mnemonic for* **rlwinm. RA,RS,0,0,31–n**	CR[CR0]
extlwi	RA, RS, n, b	Extract and left justify immediate. ($n > 0$) $(RA)_{0:n-1} \leftarrow (RS)_{b:b+n-1}$ $(RA)_{n:31} \leftarrow {}^{32-n}0$ *Extended mnemonic for* **rlwinm RA,RS,b,0,n–1**	
extlwi.		*Extended mnemonic for* **rlwinm. RA,RS,b,0,n–1**	CR[CR0]
extrwi	RA, RS, n, b	Extract and right justify immediate. ($n > 0$) $(RA)_{32-n:31} \leftarrow (RS)_{b:b+n-1}$ $(RA)_{0:31-n} \leftarrow {}^{32-n}0$ *Extended mnemonic for* **rlwinm RA,RS,b+n,32–n,31**	
extrwi.		*Extended mnemonic for* **rlwinm. RA,RS,b+n,32–n,31**	CR[CR0]
rotlwi	RA, RS, n	Rotate left immediate. (RA) ← ROTL((RS), n) *Extended mnemonic for* **rlwinm RA,RS,n,0,31**	
rotlwi.		*Extended mnemonic for* **rlwinm. RA,RS,n,0,31**	CR[CR0]
rotrwi	RA, RS, n	Rotate right immediate. (RA) ← ROTL((RS), 32–n) *Extended mnemonic for* **rlwinm RA,RS,32–n,0,31**	
rotrwi.		*Extended mnemonic for* **rlwinm. RA,RS,32–n,0,31**	CR[CR0]
slwi	RA, RS, n	Shift left immediate. ($n < 32$) $(RA)_{0:31-n} \leftarrow (RS)_{n:31}$ $(RA)_{32-n:31} \leftarrow {}^{n}0$ *Extended mnemonic for* **rlwinm RA,RS,n,0,31–n**	
slwi.		*Extended mnemonic for* **rlwinm. RA,RS,n,0,31–n**	CR[CR0]

Table 28-27. Extended Mnemonics for rlwinm, rlwinm. (continued)

Mnemonic	Operands	Function	Other Registers Altered
srwi	RA, RS, n	Shift right immediate. (n < 32) $(RA)_{n:31} \leftarrow (RS)_{0:31-n}$ $(RA)_{0:n-1} \leftarrow {}^{n}0$ *Extended mnemonic for* **rlwinm RA,RS,32−n,n,31**	
srwi.		*Extended mnemonic for* **rlwinm. RA,RS,32−n,n,31**	CR[CR0]

rlwnm	RA, RS, RB, MB, ME	Rc=0
rlwnm.	RA, RS, RB, MB, ME	Rc=1



$r \leftarrow ROTL((RS), (RB)_{27:31})$
$m \leftarrow MASK(MB, ME)$
$(RA) \leftarrow r \wedge m$

The contents of register RS are rotated left by the number of bit positions specified by the contents of register $RB_{27:31}$. A mask is generated, having 1-bits starting at the bit position specified in the MB field and ending in the bit position specified by the ME field with 0-bits elsewhere.

If the starting point of the mask is at a higher bit position than the ending point, the ones portion of the mask wraps from the highest bit position back to the lowest. The rotated data is ANDed with the generated mask and the result is placed into register RA.

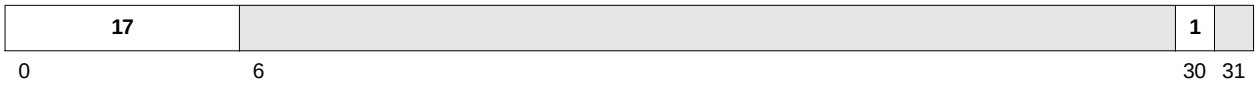
Registers Altered

- RA
- CR[CR0] if Rc contains 1

Table 28-28. Extended Mnemonics for rlwnm, rlwnm.

Mnemonic	Operands	Function	Other Registers Altered
rotlw	RA, RS, RB	Rotate left. $(RA) \leftarrow ROTL((RS), (RB)_{27:31})$ *Extended mnemonic for* **rlwnm RA,RS,RB,0,31**	
rotlw.		*Extended mnemonic for* **rlwnm. RA,RS,RB,0,31**	CR[CR0]

sc



SRR1 ← MSR
SRR0 ← 4 + address of **sc** instruction
PC ← $IVPR_{0:15}$ || $IVOR8_{16:27}$ || $^{4}0$
MSR[WE, EE, PR, FP, FE0, FE1, DWE, DS, IS] ← $^{9}0$

A System Call exception is generated, and a System Call interrupt occurs (see ~~"System Call Interrupt" on page 10-30~~*System Call Interrupt* on page 365 for more information on System Call interrupts). The contents of the MSR are copied into SRR1 and (4 + address of **sc** instruction) is placed into SRR0.

The program counter (PC) is then loaded with the interrupt vector address. The interrupt vector address is formed by concatenating the high halfword of the Interrupt Vector Prefix Register (IVPR), bits 16:27 of the Interrupt Vector Offset Register 8 (IVOR8), and 0b0000.

The MSR[WE, EE, PR, FP, FE0, FE1, DWE, DS, IS] bits are set to 0.

Program execution continues at the new address in the PC.

Registers Altered

- SRR0
- SRR1
- MSR[WE, EE, PR, FP, FE0, FE1, DWE, DS, IS]

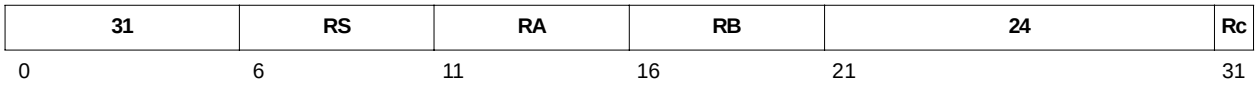
Invalid Instruction Forms

- Reserved fields

Programming Note

Execution of this instruction is context-synchronizing (see ~~"Context Synchronization" on page 4-62~~*Context Synchronization* on page 201).

slw	RA, RS, RB	Rc=0
slw.	RA, RS, RB	Rc=1



$n \leftarrow (RB)_{26:31}$
$r \leftarrow ROTL((RS), n)$
if n < 32 then
  $m \leftarrow MASK(0, 31 - n)$
else
  $m \leftarrow {}^{32}0$
$(RA) \leftarrow r \wedge m$

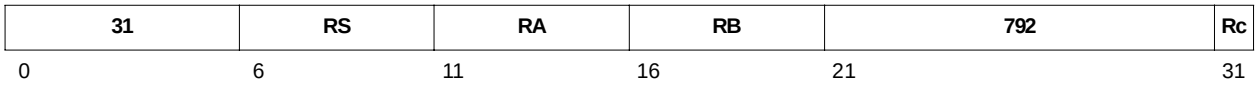
The contents of register RS are shifted left by the number of bits specified by the contents of register $RB_{26:31}$. Bits shifted left out of the most significant bit are lost, and 0-bits fill vacated bit positions on the right. The result is placed into register RA.

Note that if $RB_{26} = 1$, then the shift amount is 32 bits or more, and thus all bits are shifted out such that register RA is set to zero.

Registers Altered

- RA
- CR[CR0] if Rc contains 1

sraw	RA, RS, RB	Rc=0
sraw.	RA, RS, RB	Rc=1



$n \leftarrow (RB)_{26:31}$
$r \leftarrow ROTL((RS), 32 - n)$
if $n < 32$ then
  $m \leftarrow MASK(n, 31)$
else
  $m \leftarrow {}^{32}0$
$s \leftarrow (RS)_0$
$(RA) \leftarrow (r \wedge m) \vee ({}^{32}s \wedge \neg m)$
$XER[CA] \leftarrow s \wedge ((r \wedge \neg m) \neq 0)$

The contents of register RS are shifted right by the number of bits specified the contents of register $RB_{26:31}$. Bits shifted out of the least significant bit are lost. Bit 0 of Register RS is replicated to fill the vacated positions on the left. The result is placed into register RA.

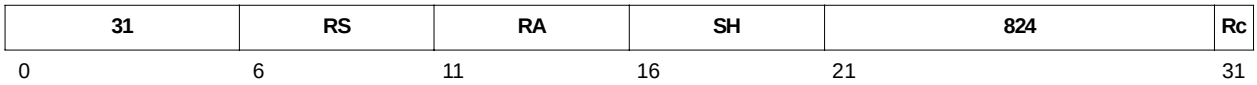
If register RS contains a negative number and any 1-bits were shifted out of the least significant bit position, XER[CA] is set to 1; otherwise, it is set to 0.

Note that if $RB_{26} = 1$, then the shift amount is 32 bits or more, and thus all bits are shifted out such that register RA and XER[CA] are set to bit 0 of register RS.

Registers Altered

- RA
- XER[CA]
- CR[CR0] if Rc contains 1

srawi	RA, RS, SH	Rc=0
srawi.	RA, RS, SH	Rc=1



n ← SH
r ← ROTL((RS), 32 – n)
m ← MASK(n, 31)
s ← $(RS)_0$
(RA) ← (r ∧ m) ∨ (^{32}s ∧ ¬m)
XER[CA] ← s ∧ ((r ∧ ¬m)≠0)

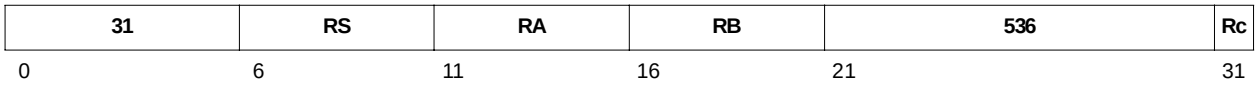
The contents of register RS are shifted right by the number of bits specified in the SH field. Bits shifted out of the least significant bit are lost. Bit RS_0 is replicated to fill the vacated positions on the left. The result is placed into register RA.

If register RS contains a negative number and any 1-bits were shifted out of the least significant bit position, XER[CA] is set to 1; otherwise, it is set to 0.

Registers Altered

- RA
- XER[CA]
- CR[CR0] if Rc contains 1

srw	RA, RS, RB	Rc=0
srw.	RA, RS, RB	Rc=1



$n \leftarrow (RB)_{26:31}$
$r \leftarrow ROTL((RS), 32 - n)$
if n < 32 then
  $m \leftarrow MASK(n, 31)$
else
  $m \leftarrow {}^{32}0$
$(RA) \leftarrow r \wedge m$

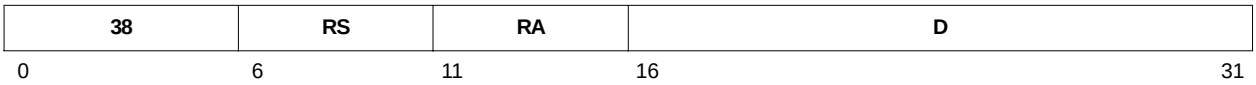
The contents of register RS are shifted right by the number of bits specified the contents of register $RB_{26:31}$. Bits shifted right out of the least significant bit are lost, and 0-bits fill the vacated bit positions on the left. The result is placed into register RA.

Note that if $RB_{26} = 1$, then the shift amount is 32 bits or more, and thus all bits are shifted out such that register RA is set to zero.

Registers Altered

- RA
- CR[CR0] if Rc contains 1

stb RS, D(RA)



EA ← (RA|0) + EXTS(D)
MS(EA, 1) ← $(RS)_{24:31}$

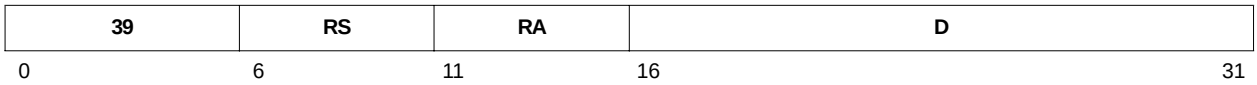
An effective address (EA) is formed by adding a displacement to a base address. The displacement is obtained by sign-extending the 16-bit D field to 32 bits. The base address is 0 when the RA field is 0, and is the contents of register RA otherwise.

The least significant byte of register RS is stored into the byte at the EA.

Registers Altered

- None

stbu RS, D(RA)



EA $\leftarrow$ (RA|0) + EXTS(D)
MS(EA, 1) $\leftarrow$ $(RS)_{24:31}$
(RA) $\leftarrow$ EA

An effective address (EA) is formed by adding a displacement to a base address. The displacement is obtained by sign-extending the 16-bit D field to 32 bits. The base address is 0 when the RA field is 0, and is the contents of register RA otherwise.

The least significant byte of register RS is stored into the byte at the EA.

The EA is placed into register RA.

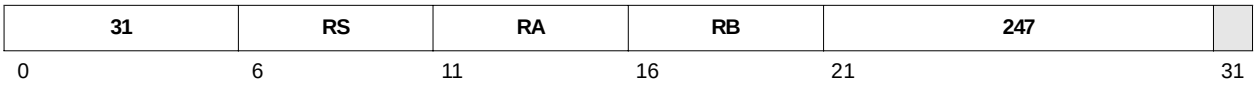
Registers Altered

- RA

Invalid Instruction Forms

RA = 0

stbux RS, RA, RB



EA ← (RA|0) + (RB)
MS(EA, 1) ← $(RS)_{24:31}$
(RA) ← EA

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 when the RA field is 0, and is the contents of register RA otherwise.

The least significant byte of register RS is stored into the byte at the EA.

The EA is placed into register RA.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

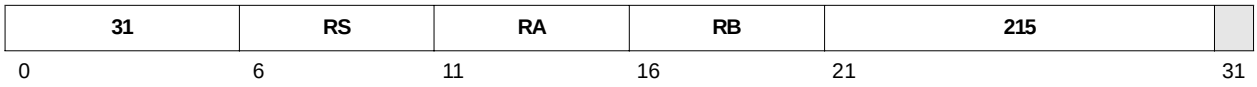
- RA

Invalid Instruction Forms

- Reserved fields

RA = 0

stbx RS, RA, RB



EA $\leftarrow$ (RA|0) + (RB)
MS(EA, 1) $\leftarrow$ $(RS)_{24:31}$

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 when the RA field is 0, and is the contents of register RA otherwise.

The least significant byte of register RS is stored into the byte at the EA.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

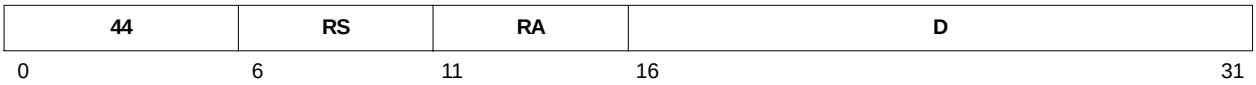
Registers Altered

- None

Invalid Instruction Forms

- Reserved fields

sth RS, D(RA)



EA ← (RA|0) + EXTS(D)
MS(EA, 2) ← $(RS)_{16:31}$

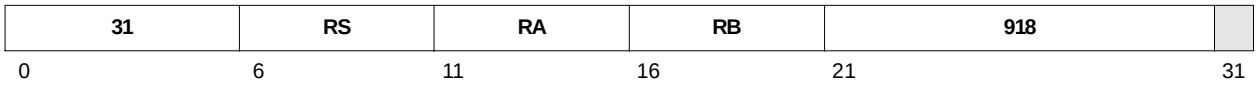
An effective address (EA) is formed by adding a displacement to a base address. The displacement is obtained by sign-extending the 16-bit D field to 32 bits. The base address is 0 when the RA field is 0 and is the contents of register RA otherwise.

The least significant halfword of register RS is stored into the halfword at the EA in main storage.

Registers Altered

- None

sthbrx RS, RA, RB



EA ← (RA|0) + (RB)
MS(EA, 2) ← BYTE_REVERSE($(RS)_{16:31}$)

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 when the RA field is 0 and is the contents of register RA otherwise.

The least significant halfword of register RS is byte-reversed from the default byte ordering for the memory page referenced by the EA. The resulting halfword is stored at the EA.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- None

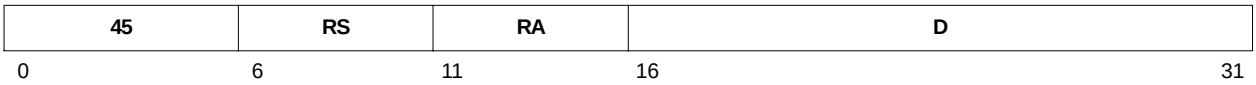
Invalid Instruction Forms

- Reserved fields

Programming Note

Byte ordering is generally controlled by the Endian (E) storage attribute (see ~~Chapter 6, "Memory Management"~~ *Memory Management* on page 233). The store byte reverse instructions provide a mechanism for data to be stored to a memory page using the opposite byte ordering from that specified by the Endian storage attribute.

sthu RS, D(RA)



EA $\leftarrow$ (RA|0) + EXTS(D)
MS(EA, 2) $\leftarrow$ $(RS)_{16:31}$
(RA) $\leftarrow$ EA

An effective address (EA) is formed by adding a displacement to a base address. The displacement is obtained by sign-extending the 16-bit D field to 32 bits. The base address is 0 when the RA field is 0, and is the contents of register RA otherwise.

The least significant halfword of register RS is stored into the halfword at the EA.

The EA is placed into register RA.

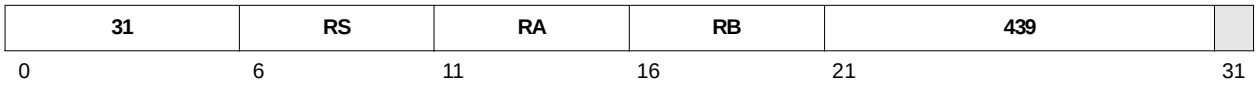
Registers Altered

- RA

Invalid Instruction Forms

RA = 0

sthux RS, RA, RB



EA $\leftarrow$ (RA|0) + (RB)
MS(EA, 2) $\leftarrow$ $(RS)_{16:31}$
(RA) $\leftarrow$ EA

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 when the RA field is 0, and is the contents of register RA otherwise.

The least significant halfword of register RS is stored into the halfword at the EA.

The EA is placed into register RA.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

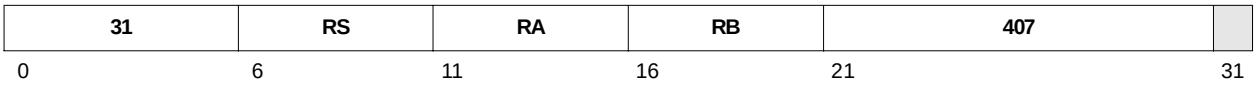
Registers Altered

- RA

Invalid Instruction Forms

- Reserved fields
- RA = 0

sthx RS, RA, RB



EA $\leftarrow$ (RA|0) + (RB)
MS(EA, 2) $\leftarrow$ $(RS)_{16:31}$

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 when the RA field is 0, and is the contents of register RA otherwise.

The least significant halfword of register RS is stored into the halfword at the EA.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

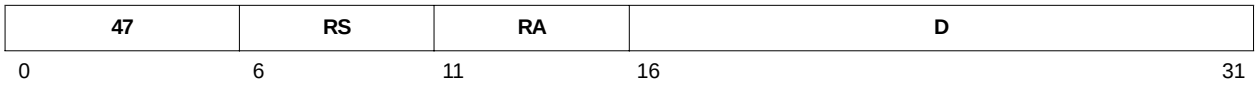
Registers Altered

- None

Invalid Instruction Forms

- Reserved fields

stmw RS, D(RA)




```
EA ← (RA|0) + EXTS(D)
r ← RS
do while r ≤ 31
   MS(EA, 4) ← (GPR(r))
   r ← r + 1
   EA ← EA + 4
```

An effective address (EA) is formed by adding a displacement to a base address. The displacement is obtained by sign-extending the 16-bit D field to 32 bits. The base address is 0 when the RA field is 0, and is the contents of register RA otherwise.

The contents of a series of consecutive registers, starting with register RS and continuing through GPR(31), are stored into consecutive words starting at the EA.

Registers Altered

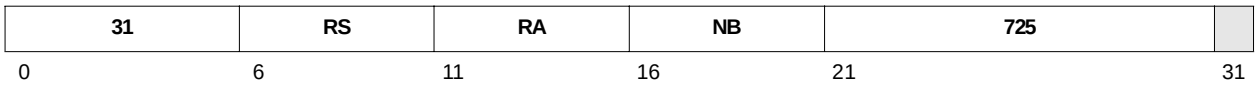
- None

Programming Note

This instruction can be restarted, meaning that it could be interrupted after having already stored some of the register values to memory, and then re-executed from the beginning (after returning from the interrupt), in which case the registers which had already been stored prior to the interrupt will be stored a second time.

Store String Word Immediate

stswi RS, RA, NB




```
EA ← (RA|0)
if NB = 0 then
   n ← 32
else
   n ← NB
r ← RS – 1
i ← 0
do while n > 0
   if i = 0 then
      r ← r + 1
   if r = 32 then
      r ← 0
   MS(EA,1) ← (GPR(r)i:i+7)
   i ← i + 8
   if i = 32 then
      i ← 0
   EA ← EA + 1
   n ← n – 1
```

An effective address (EA) is determined by the RA field. If the RA field contains 0, the EA is 0; otherwise, the EA is the contents of register RA.

A byte count is determined by the NB field. If the NB field contains 0, the byte count is 32; otherwise, the byte count is the contents of the NB field.

The contents of a series of consecutive GPRs (starting with register RS, continuing through GPR(31) and wrapping to GPR(0) as necessary, and continuing to the final byte count) are stored, starting at the EA. The bytes in each GPR are accessed starting with the most significant byte. The byte count determines the number of transferred bytes.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- None

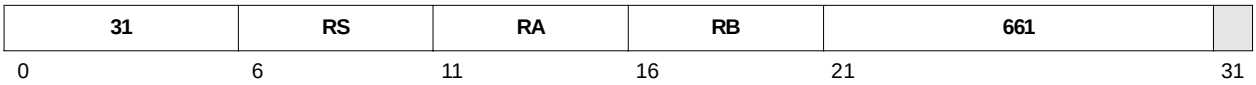
Invalid Instruction Forms

- Reserved fields

Programming Note

This instruction can be restarted, meaning that it could be interrupted after having already stored some of the register values to memory, and then re-executed from the beginning (after returning from the interrupt), in which case the registers which had already been stored prior to the interrupt will be stored a second time.

stswx RS, RA, RB




```
EA ← (RA|0) + (RB)
n ← XER[TBC]
r ← RS – 1
i ← 0
do while n > 0
   if i = 0 then
      r ← r + 1
   if r = 32 then
      r ← 0
   MS(EA, 1) ← (GPR(r)i:i+7)
   i ← i + 8
   if i = 32 then
      i ← 0
   EA ← EA + 1
   n ← n – 1
```

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 when the RA field is 0, and is the contents of register RA otherwise.

A byte count is contained in XER[TBC].

The contents of a series of consecutive GPRs (starting with register RS, continuing through GPR(31) and wrapping to GPR(0) as necessary, and continuing to the final byte count) are stored, starting at the EA. The bytes in each GPR are accessed starting with the most significant byte. The byte count determines the number of transferred bytes.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- None

Invalid Instruction Forms

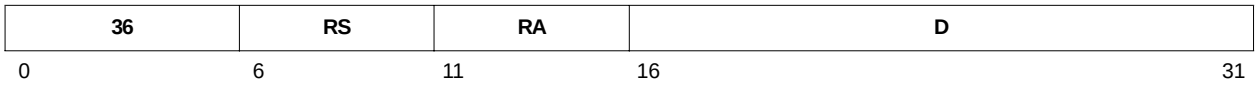
- Reserved fields

Programming Note

This instruction can be restarted, meaning that it could be interrupted after having already stored some of the register values to memory, and then re-executed from the beginning (after returning from the interrupt), in which case the registers which had already been stored prior to the interrupt will be stored a second time.

If XER[TBC] = 0, no GPRs are stored to memory, and **stswx** is treated as a no-op. Furthermore, if the EA is such that a Data Storage, Data TLB Error, or Data Address Compare Debug exception occurs, **stswx** is treated as a no-op and no interrupt occurs as a result of the exception.

stw RS, D(RA)



EA ← (RA|0) + EXTS(D)
MS(EA, 4) ← (RS)

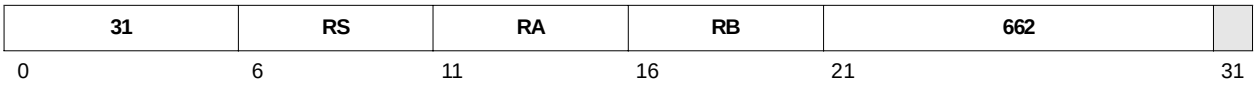
An effective address (EA) is formed by adding a displacement to a base address. The displacement is obtained by sign-extending the 16-bit D field to 32 bits. The base address is 0 when the RA field is 0, and is the contents of register RA otherwise.

The contents of register RS are stored at the EA.

Registers Altered

- None

stwbrx RS, RA, RB



EA $\leftarrow$ (RA|0) + (RB)
MS(EA, 4) $\leftarrow$ BYTE_REVERSE($(RS)_{0:31}$)

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 when the RA field is 0 and is the contents of register RA otherwise.

The word in register RS is byte-reversed from the default byte ordering for the memory page referenced by the EA. The resulting word is stored at the EA.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- None

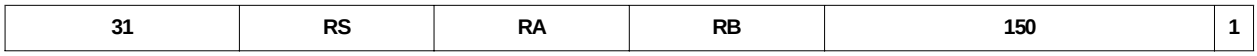
Invalid Instruction Forms

- Reserved fields

Programming Note

Byte ordering is generally controlled by the Endian (E) storage attribute (see ~~Chapter 6, "Memory Management"~~*Memory Management* on page 233). The store byte reverse instructions provide a mechanism for data to be stored to a memory page using the opposite byte ordering from that specified by the Endian storage attribute.

stwcx. RS, RA, RB




```
EA ← (RA|0) + (RB)
if RESERVE = 1 then
   MS(EA, 4) ← (RS)
   RESERVE ← 0
   (CR[CR0]) ← 20 || 1 || XER[SO]
else
   (CR[CR0]) ← 20 || 0 || XER[SO]
```

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 when the RA field is 0, and is the contents of register RA otherwise.

If the reservation bit contains 1 when the instruction is executed, the contents of register RS are stored into the word at the EA and the reservation bit is cleared. If the reservation bit contains 0 when the instruction is executed, no store operation is performed.

CR[CR0] is set as follows:

- $CR[CR0]_{0:1}$ are cleared
- $CR[CR0]_2$ is set to indicate whether or not the store was performed (1 indicates that it was)
- $CR[CR0]_3$ is set to the contents of the XER[SO] bit

Registers Altered

- CR[CR0]

Programming Notes

The **lwarx** and **stwcx.** instructions are typically paired in a loop, as shown in the following example, to create the effect of an atomic operation to a memory area used as a semaphore between multiple processes. Only **lwarx** can set the reservation bit to 1. **stwcx.** sets the reservation bit to 0 upon its completion, whether or not **stwcx.** actually stored (RS) to memory. $CR[CR0]_2$ must be examined to determine whether (RS) was sent to memory.

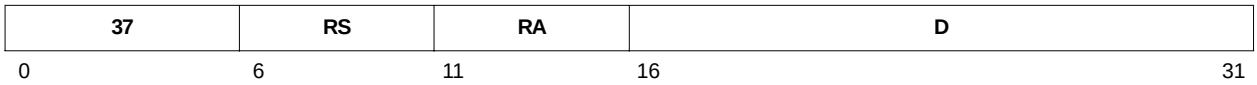
```
loop: lwarx     # read the semaphore from memory; set reservation
      "alter"   # change the semaphore bits in the register as required
      stwcx.    # attempt to store the semaphore; reset reservation
      bne loop  # some other process intervened and cleared the reservation prior to the above
                # stwcx.; try again
```

The PowerPC Book-E architecture specifies that the EA for the **lwarx** instruction must be word-aligned (that is, a multiple of 4 bytes); otherwise, the result is undefined. Although the PPC440GP will execute **stwcx.** regardless of the EA alignment, in order for the operation of the pairing of **lwarx** and **stwcx.** to produce the desired result, software must ensure that the EA for both instructions is word-aligned. This requirement is due to the manner in which misaligned storage accesses may be broken up into separate, aligned accesses by the PPC440GP.

The PowerPC Book-E architecture also specifies that it is implementation-dependent as to whether a Data Storage, Data TLB Error, Alignment, or Debug interrupt occurs when the reservation bit is off at the time of execution of an **stwcx.** instruction, and when the conditions are such that a non-**stwcx.** store-type storage access instruction would have resulted in such an interrupt. The PPC440GP implements **stwcx.** such that Data Storage and Debug (DAC and/or DVC exception type) interrupts do not occur when the reservation bit is off at the time of execution of the **stwcx.** Instead, the **stwcx.** instruction completes without causing the interrupt and without storing to memory, and CR[CR0] is updated to indicate the failure of the **stwcx.**

On the other hand, the PPC440GP causes a Data TLB Error interrupt if a Data TLB Miss exception occurs during due to the execution of a **stwcx.** instruction, regardless of the state of the reservation. Similarly, the PPC440GP causes an Alignment interrupt if the EA of the **stwcx.** operand is not word-aligned when CCR0[FLSTA] is 1, regardless of the state of the reservation (see ~~"Core Configuration Register 0 (CCR0)" on page 5-13~~*Core Configuration Register 0 (CCR0)* on page 216 for more information on the Force Load/Store Alignment function).

stwu RS, D(RA)



EA ← (RA|0) + EXTS(D)
MS(EA, 4) ← (RS)
(RA) ← EA

An effective address (EA) is formed by adding a displacement to a base address. The displacement is obtained by sign-extending the 16-bit D field to 32 bits. The base address is 0 when the RA field is 0, and is the contents of register RA otherwise.

The contents of register RS are stored into the word at the EA.

The EA is placed into register RA.

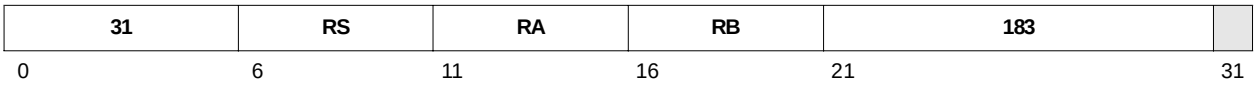
Registers Altered

- RA

Invalid Instruction Forms

RA = 0

stwux RS, RA, RB



EA ← (RA|0) + (RB)
MS(EA, 4) ← (RS)
(RA) ← EA

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 when the RA field is 0, and is the contents of register RA otherwise.

The contents of register RS are stored into the word at the EA.

The EA is placed into register RA.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

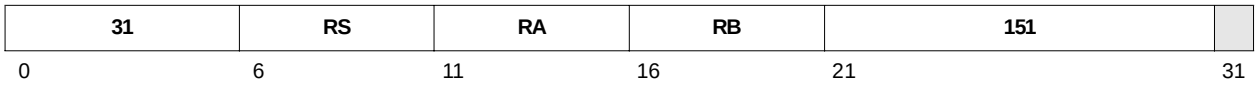
Registers Altered

- RA

Invalid Instruction Forms

- Reserved fields
- RA = 0

stwx RS, RA, RB



EA ← (RA|0) + (RB)
MS(EA,4) ← (RS)

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 when the RA field is 0, and is the contents of register RA otherwise.

The contents of register RS are stored into the word at the EA.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

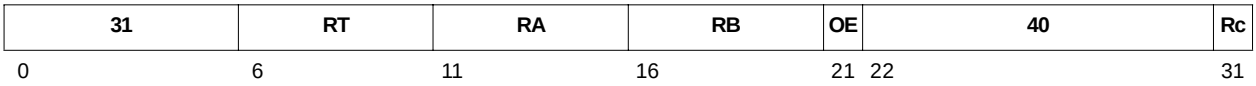
Registers Altered

- None

Invalid Instruction Forms

- Reserved fields

subf	RT, RA, RB	OE=0, Rc=0
subf.	RT, RA, RB	OE=0, Rc=1
subfo	RT, RA, RB	OE=1, Rc=0
subfo.	RT, RA, RB	OE=1, Rc=1



(RT) $\leftarrow$ $\neg$(RA) + (RB) + 1

The sum of the ones complement of register RA, register RB, and 1 is stored into register RT.

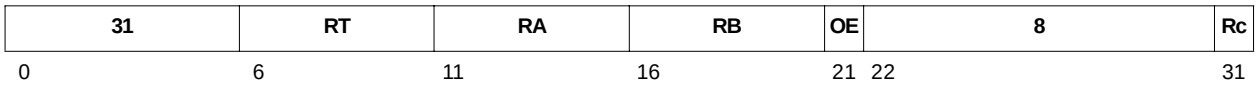
Registers Altered

- RT
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE contains 1

Table 28-29. Extended Mnemonics for subf, subf., subfo, subfo.

Mnemonic	Operands	Function	Other Registers Altered
sub	RT, RA, RB	Subtract (RB) from (RA). (RT) $\leftarrow$ $\neg$(RB) + (RA) + 1. *Extended mnemonic for* **subf RT,RB,RA**	
sub.		*Extended mnemonic for* **subf. RT,RB,RA**	CR[CR0]
subo		*Extended mnemonic for* **subfo RT,RB,RA**	XER[SO, OV]
subo.		*Extended mnemonic for* **subfo. RT,RB,RA**	CR[CR0] XER[SO, OV]

subfc	RT, RA, RB	OE=0, Rc=0
subfc.	RT, RA, RB	OE=0, Rc=1
subfco	RT, RA, RB	OE=1, Rc=0
subfco.	RT, RA, RB	OE=1, Rc=1



(RT) $\leftarrow$ $\neg$(RA) + (RB) + 1
if $\neg$(RA) + (RB) + 1 $\overset{u}{>}$ $2^{32} - 1$ then
  XER[CA] $\leftarrow$ 1
else
  XER[CA] $\leftarrow$ 0

The sum of the ones complement of register RA, register RB, and 1 is stored into register RT.

XER[CA] is set to a value determined by the unsigned magnitude of the result of the subtract operation.

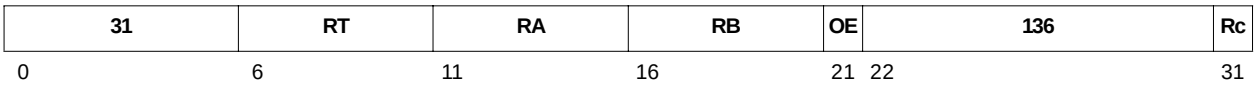
Registers Altered

- RT
- XER[CA]
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE contains 1

Table 28-30. Extended Mnemonics for subfc, subfc., subfco, subfco.

Mnemonic	Operands	Function	Other Registers Altered
subc	RT, RA, RB	Subtract (RB) from (RA). (RT) $\leftarrow$ $\neg$(RB) + (RA) + 1. Place carry-out in XER[CA]. *Extended mnemonic for* **subfc RT,RB,RA**	
subc.		*Extended mnemonic for* **subfc. RT,RB,RA**	CR[CR0]
subco		*Extended mnemonic for* **subfco RT,RB,RA**	XER[SO, OV]
subco.		*Extended mnemonic for* **subfco. RT,RB,RA**	CR[CR0] XER[SO, OV]

subfe	RT, RA, RB	OE=0, Rc=0
subfe.	RT, RA, RB	OE=0, Rc=1
subfeo	RT, RA, RB	OE=1, Rc=0
subfeo.	RT, RA, RB	OE=1, Rc=1



(RT) $\leftarrow$ $\neg$(RA) + (RB) + XER[CA]
if $\neg$(RA) + (RB) + XER[CA] $\overset{u}{>}$ $2^{32} - 1$ then
  XER[CA] $\leftarrow$ 1
else
  XER[CA] $\leftarrow$ 0

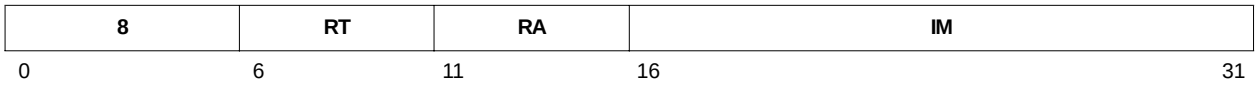
The sum of the ones complement of register RA, register RB, and XER[CA] is placed into register RT.

XER[CA] is set to a value determined by the unsigned magnitude of the result of the subtract operation.

Registers Altered

- RT
- XER[CA]
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE contains 1

subfic RT, RA, IM



(RT) ← ¬(RA) + EXTS(IM) + 1
if ¬(RA) + EXTS(IM) + 1 $\overset{u}{>}$ $2^{32} - 1$ then
  XER[CA] ← 1
else
  XER[CA] ← 0

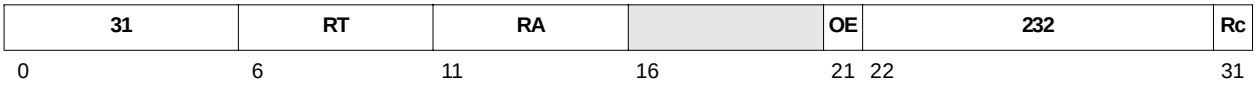
The sum of the ones complement of RA, the IM field sign-extended to 32 bits, and 1 is placed into register RT.

XER[CA] is set to a value determined by the unsigned magnitude of the result of the subtract operation.

Registers Altered

- RT
- XER[CA]

subfme	RT, RA	OE=0, Rc=0
subfme.	RT, RA	OE=0, Rc=1
subfmeo	RT, RA	OE=1, Rc=0
subfmeo.	RT, RA	OE=1, Rc=1



(RT) $\leftarrow$ $\neg$(RA) – 1 + XER[CA]
if $\neg$(RA) + 0xFFFF FFFF + XER[CA] $\gtrsim^{u}$ 2^{32} – 1 then
  XER[CA] $\leftarrow$ 1
else
  XER[CA] $\leftarrow$ 0

The sum of the ones complement of register RA, –1, and XER[CA] is placed into register RT.

XER[CA] is set to a value determined by the unsigned magnitude of the result of the subtract operation.

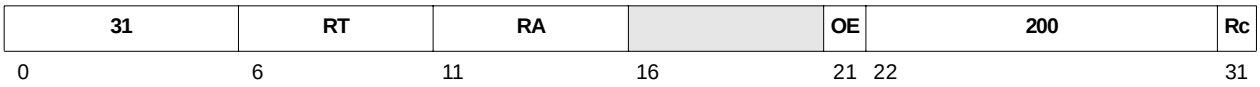
Registers Altered

- RT
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE contains 1
- XER[CA]

Invalid Instruction Forms

- Reserved fields

subfze	RT, RA	OE=0, Rc=0
subfze.	RT, RA	OE=0, Rc=1
subfzeo	RT, RA	OE=1, Rc=0
subfzeo.	RT, RA	OE=1, Rc=1



(RT) $\leftarrow$ $\neg$(RA) + XER[CA]
if $\neg$(RA) + XER[CA] $\gtrdot^{u}$ $2^{32} - 1$ then
  XER[CA] $\leftarrow$ 1
else
  XER[CA] $\leftarrow$ 0

The sum of the ones complement of register RA and XER[CA] is stored into register RT.

XER[CA] is set to a value determined by the unsigned magnitude of the result of the subtract operation.

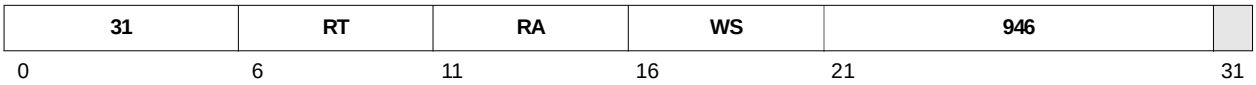
Registers Altered

- RT
- XER[CA]
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE contains 1

Invalid Instruction Forms

- Reserved fields

tlbre RT, RA, WS




```
tlbentry ← TLB[(RA)26:31]
if WS =0
   (RT)0:27 ← tlbentry[EPN,V,TS,SIZE]
   (RT)28:31 ← 40
   MMUCR[STID] ← tlbentry[TID]
else if WS = 1
   (RT)0:21 ← tlbentry[RPN]
   (RT)22:27 ← 60
   (RT)28:31 ← tlbentry[ERPN]
else if WS = 2
   (RT)0:15 ← 160
   (RT)16:24 ← tlbentry[U0,U1,U2,U3,W,I,M,G,E]
   (RT)25 ← 0
   (RT)26:31 ← tlbentry[UX,UW,UR,SX,SW,SR]
else (RT), MMUCR[STID] ← undefined
```

The contents of the specified portion of the selected TLB entry are placed into register RT (and also MMUCR[STID] if WS = 0).

The contents of RA are used as an index into the TLB. If this value is greater than the ~~number~~ index of the highest numbered TLB ~~entries~~ entry (~~64~~63), the results are undefined.

The WS field specifies which portion of the TLB entry is placed into RT. If WS = 0, the TID field of the selected TLB entry is read into MMUCR[STID]. See ~~Chapter 6, "Memory Management"~~*Memory Management* on page 233 for descriptions of the TLB entry fields.

If the value of the WS field is greater than 2, the instruction form is invalid and the result is undefined.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- RT
- MMUCR[STID] (if WS = 0)

Invalid Instruction Forms

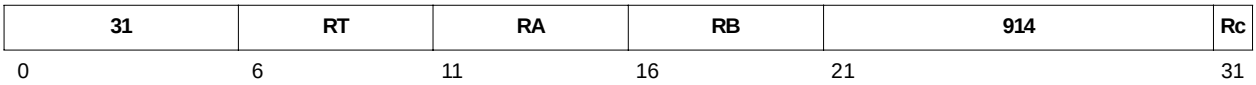
- Reserved fields
- Invalid WS value

Programming ~~Note~~Notes

Execution of this instruction is privileged.

The PPC440GP does not automatically synchronize the context of the MMUCR[STID] field between a **tlbre** instruction which updates the field, and a **tlbsx[.]** instruction which uses it as a source operand. Therefore, software must execute an **isync** instruction between the execution of a **tlbre** instruction and a subsequent **tlbsx[.]** instruction to ensure that the **tlbsx[.]** instruction will use the new value of MMUCR[STID]. On the other hand, the PPC440GP *does* automatically synchronize the context of MMUCR[STID] between **tlbre** and **tlbwe**, as well as between **tlbre** and **mfspr** which specifies the MMUCR as the source SPR, so no **isync** is required in these cases.

tlbsx	RT, RA, RB	Rc=0
tlbsx.	RT, RA, RB	Rc=1




```
EA ← (RA|0) + (RB)
if Rc = 1
   CR[CR0]0 ← 0
   CR[CR0]1 ← 0
   CR[CR0]3 ← XER[SO}
if  Valid TLB entry matching EA and MMUCR[STID,STS] is in the TLB then
   (RT) ← Index of matching TLB Entry
   if Rc = 1
      CR[CR0]2 ← 1
else
   (RT) ← Undefined
   if Rc = 1
      CR[CR0]2 ← 0
```

An effective address is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise.

The TLB is searched for a valid entry which translates EA and MMUCR[STID,STS]. See ~~Chapter 6, "Memory Management"~~*Memory Management* on page 233 for descriptions of the TLB fields and how they participate in the determination of a match. If a matching entry is found, its index (0 - 63) is placed into bits 26:31 of RT, and bits 0:25 are set to 0. If no match is found, the contents of RT are undefined.

The record bit (Rc) specifies whether the results of the search will affect CR[CR0] as shown above, such that $CR[CR0]_2$ can be tested if there is a possibility that the search may fail.

Registers Altered

- CR[CR0] if Rc contains 1

Invalid Instruction Forms

- None

Programming ~~Note~~Notes

Execution of this instruction is privileged.

The PPC440GP does not automatically synchronize the context of the MMUCR[STID] field between a **tlbre** instruction which updates the field, and a **tlbsx[.]** instruction which uses it as a source operand. Therefore, software must execute an **isync** instruction between the execution of a **tlbre** instruction and a subsequent **tlbsx[.]** instruction to ensure that the **tlbsx[.]** instruction will use the new value of MMUCR[STID]. On the other hand, the PPC440GP *does* automatically synchronize the context of MMUCR[STID] between **tlbre** and **tlbwe**, as well as between **tlbre** and **mfspr** which specifies the MMUCR as the source SPR, so no **isync** is required in these cases.

tlbsync

The **tlbsync** instruction is provided by the PowerPC Book-E architecture to support synchronization of TLB operations between processors in a coherent multi-processor system. Since the PPC440GP does not support coherent multi-processing, this instruction performs no operation, and is provided only to facilitate code portability.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- None

Invalid Instruction Forms

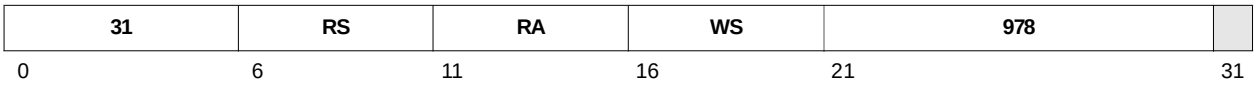
- Reserved fields

Programming Note

This instruction is privileged. Translation is not required to be active during the execution of this instruction.

Since the PPC440GP does not support tightly-coupled multiprocessor systems, **tlbsync** performs no operation.

tlbwe RS, RA, WS




```
tlbentry ← TLB[(RA)26:31]
if WS = 0
   tlbentry[EPN,V,TS,SIZE] ← (RS)0:27
   tlbentry[TID] ← MMUCR[STID]
else if WS = 1
   tlbentry[RPN] ← (RS)0:21
   tlbentry[ERPN] ← (RS)28:31
else if WS = 2
   tlbentry[U0,U1,U2,U3,W,I,M,G,E] ← (RS)16:24
   tlbentry[UX,UW,UR,SX,SW,SR] ← (RS)26:31
else tlbentry ← undefined
```

The contents of the specified portion of the selected TLB entry are replaced with the contents of register RS (and also MMUCR[STID] if WS = 0).

The contents of RA are used as an index into the TLB. If this value is greater than the ~~number~~ index of the highest numbered TLB ~~entries~~ entry (~~64~~63), the results are undefined.

The WS field specifies which portion of the TLB entry is replaced by the contents of RS. If WS = 0, the TID field of the selected TLB entry is replaced by the value in MMUCR[STID]. See ~~Chapter 6, "Memory Management"~~ *Memory Management* on page 233 for descriptions of the TLB entry fields.

If the value of the WS field is greater than 2, the instruction form is invalid and the result is undefined.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- None

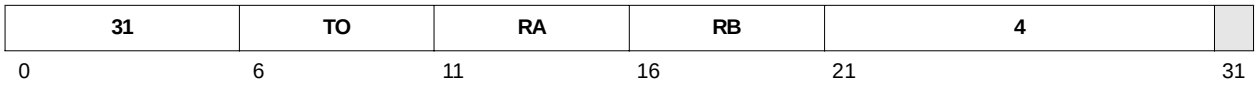
Invalid Instruction Forms

- Reserved fields
- Invalid WS value

Programming Note

Execution of this instruction is privileged.

tw TO, RA, RB




```
if (   ((RA) <  (RB) ∧ TO0 = 1)   ∨
       ((RA) >  (RB) ∧ TO1 = 1)   ∨
       ((RA) =  (RB) ∧ TO2 = 1)   ∨
       ((RA) <u (RB) ∧ TO3 = 1)   ∨
       ((RA) >u (RB) ∧ TO4 = 1)   )
   SRR0 ← address of tw instruction
   SRR1 ← MSR
   ESR[PTR] ← 1 (other bits cleared)
   MSR[WE, EE, PR, FP, FE0, FE1, DWE, DS, IS]) ← 9 0
   PC ← IVPR0:15 || IVOR616:27 || 4 0
else no operation
```

Register RA is compared with register RB. If any comparison condition enabled by the TO field is true, a Trap exception type Program interrupt occurs as follows (see ~~"Program Interrupt" on page 10-27~~*Program Interrupt* on page 362 for more information on Program interrupts). The contents of the MSR are copied into SRR1 and the address of the **tw** instruction) is placed into SRR0. ESR[PTR] is set to 1 and the other bits ESR bits cleared to indicate the type of exception causing the Program interrupt.

The program counter (PC) is then loaded with the interrupt vector address. The interrupt vector address is formed by concatenating the high halfword of the Interrupt Vector Prefix Register (IVPR), bits 16:27 of the Interrupt Vector Offset Register 6 (IVOR6), and 0b0000.

MSR[WE, EE, PR, FP, FE0, FE1, DWE, DS, IS] are set to 0.

Program execution continues at the new address in the PC.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- SRR0 (if trap condition is met)
- SRR1 (if trap condition is met)
- MSR[WE, EE, PR, FP, FE0, FE1, DWE, DS, IS] (if trap condition is met)
- ESR (if trap condition is met)

Invalid Instruction Forms

- Reserved fields

Programming Notes

This instruction can be inserted into the execution stream by a debugger to implement breakpoints, and is not typically used by application code.

The enabling of trap debug events may affect the interrupt type caused by the execution of **tw** instruction. Specifically, trap instructions may be enabled to cause Debug interrupts instead of Program interrupts. See ~~"Trap (TRAP) Debug Event" on page 15-19~~*Trap (TRAP) Debug Event* on page 446 for more details.

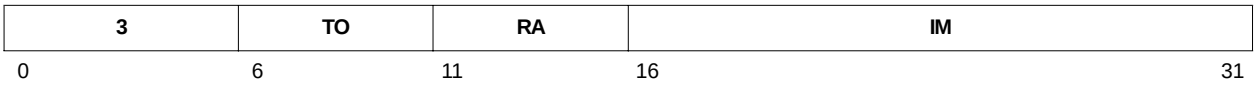
Table 28-31. Extended Mnemonics for tw

Mnemonic	Operands	Function	Other Registers Altered
trap		Trap unconditionally. *Extended mnemonic for* **tw 31,0,0**	
tweq	RA, RB	Trap if (RA) equal to (RB). *Extended mnemonic for* **tw 4,RA,RB**	
twge	RA, RB	Trap if (RA) greater than or equal to (RB). *Extended mnemonic for* **tw 12,RA,RB**	
twgt	RA, RB	Trap if (RA) greater than (RB). *Extended mnemonic for* **tw 8,RA,RB**	
twle	RA, RB	Trap if (RA) less than or equal to (RB). *Extended mnemonic for* **tw 20,RA,RB**	
twlge	RA, RB	Trap if (RA) logically greater than or equal to (RB). *Extended mnemonic for* **tw 5,RA,RB**	
twlgt	RA, RB	Trap if (RA) logically greater than (RB). *Extended mnemonic for* **tw 1,RA,RB**	
twlle	RA, RB	Trap if (RA) logically less than or equal to (RB). *Extended mnemonic for* **tw 6,RA,RB**	
twllt	RA, RB	Trap if (RA) logically less than (RB). *Extended mnemonic for* **tw 2,RA,RB**	
twlng	RA, RB	Trap if (RA) logically not greater than (RB). *Extended mnemonic for* **tw 6,RA,RB**	
twlnl	RA, RB	Trap if (RA) logically not less than (RB). *Extended mnemonic for* **tw 5,RA,RB**	
twlt	RA, RB	Trap if (RA) less than (RB). *Extended mnemonic for* **tw 16,RA,RB**	
twne	RA, RB	Trap if (RA) not equal to (RB). *Extended mnemonic for* **tw 24,RA,RB**	

Table 28-31. Extended Mnemonics for tw (continued)

Mnemonic	Operands	Function	Other Registers Altered
twng	RA, RB	Trap if (RA) not greater than (RB). *Extended mnemonic for* **tw 20,RA,RB**	
twnl	RA, RB	Trap if (RA) not less than (RB). *Extended mnemonic for* **tw 12,RA,RB**	

twi TO, RA, IM




```
if (   ((RA) <  EXTS(IM) ∧ TO0 = 1)  ∨
       ((RA) >  EXTS(IM) ∧ TO1 = 1)  ∨
       ((RA) =  EXTS(IM) ∧ TO2 = 1)  ∨
       ((RA) <u EXTS(IM) ∧ TO3 = 1)  ∨
       ((RA) >u EXTS(IM) ∧ TO4 = 1)  )
   SRR0 ← address of twi instruction
   SRR1 ← MSR
   ESR[PTR] ← 1 (other bits cleared)
   MSR[WE, EE, PR, FP, FE0, FE1, DWE, DS, IS]) ← 9 0
   PC ← IVPR0:15 || IVOR616:27 || 4 0
else no operation
```

Register RA is compared with the sign-extended IM field. If any comparison condition selected by the TO field is true, a Trap exception type Program interrupt occurs as follows (see ~~"Program Interrupt" on page 10-27~~*Program Interrupt* on page 362 for more information on Program interrupts). The contents of the MSR are copied into SRR1 and the address of the **twi** instruction) is placed into SRR0. ESR[PTR] is set to 1 and the other bits ESR bits cleared to indicate the type of exception causing the Program interrupt.

The program counter (PC) is then loaded with the interrupt vector address. The interrupt vector address is formed by concatenating the high halfword of the Interrupt Vector Prefix Register (IVPR), bits 16:27 of the Interrupt Vector Offset Register 6 (IVOR6), and 0b0000.

MSR[WE, EE, PR, FP, FE0, FE1, DWE, DS, IS] are set to 0.

Program execution continues at the new address in the PC.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- SRR0 (if trap condition is met)
- SRR1 (if trap condition is met)
- MSR[WE, EE, PR, FP, FE0, FE1, DWE, DS, IS] (if trap condition is met)
- ESR (if trap condition is met)

Invalid Instruction Forms

- Reserved fields

Programming Notes

This instruction can be inserted into the execution stream by a debugger to implement breakpoints, and is not typically used by application code.

The enabling of trap debug events may affect the interrupt type caused by the execution of **tw** instruction. Specifically, trap instructions may be enabled to cause Debug interrupts instead of Program interrupts. See ~~"Trap (TRAP) Debug Event" on page 15-19~~*Trap (TRAP) Debug Event* on page 446 for more details.

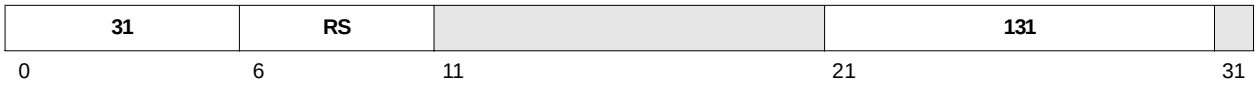
Table 28-32. Extended Mnemonics for twi

Mnemonic	Operands	Function	Other Registers Altered
tweqi	RA, IM	Trap if (RA) equal to EXTS(IM). *Extended mnemonic for* **twi 4,RA,IM**	
twgei	RA, IM	Trap if (RA) greater than or equal to EXTS(IM). *Extended mnemonic for* **twi 12,RA,IM**	
twgti	RA, IM	Trap if (RA) greater than EXTS(IM). *Extended mnemonic for* **twi 8,RA,IM**	
twlei	RA, IM	Trap if (RA) less than or equal to EXTS(IM). *Extended mnemonic for* **twi 20,RA,IM**	
twlgei	RA, IM	Trap if (RA) logically greater than or equal to EXTS(IM). *Extended mnemonic for* **twi 5,RA,IM**	
twlgti	RA, IM	Trap if (RA) logically greater than EXTS(IM). *Extended mnemonic for* **twi 1,RA,IM**	
twllei	RA, IM	Trap if (RA) logically less than or equal to EXTS(IM). *Extended mnemonic for* **twi 6,RA,IM**	
twllti	RA, IM	Trap if (RA) logically less than EXTS(IM). *Extended mnemonic for* **twi 2,RA,IM**	
twlngi	RA, IM	Trap if (RA) logically not greater than EXTS(IM). *Extended mnemonic for* **twi 6,RA,IM**	
twlnli	RA, IM	Trap if (RA) logically not less than EXTS(IM). *Extended mnemonic for* **twi 5,RA,IM**	
twlti	RA, IM	Trap if (RA) less than EXTS(IM). *Extended mnemonic for* **twi 16,RA,IM**	
twnei	RA, IM	Trap if (RA) not equal to EXTS(IM). *Extended mnemonic for* **twi 24,RA,IM**	
twngi	RA, IM	Trap if (RA) not greater than EXTS(IM). *Extended mnemonic for* **twi 20,RA,IM**	

Table 28-32. Extended Mnemonics for twi (continued)

Mnemonic	Operands	Function	Other Registers Altered
twnli	RA, IM	Trap if (RA) not less than EXTS(IM). *Extended mnemonic for* **twi 12,RA,IM**	

wrtee RS



$MSR[EE] \leftarrow (RS)_{16}$

MSR[EE] is set to the value specified by bit 16 of register RS.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- MSR[EE]

Invalid Instruction Forms:

- Reserved fields

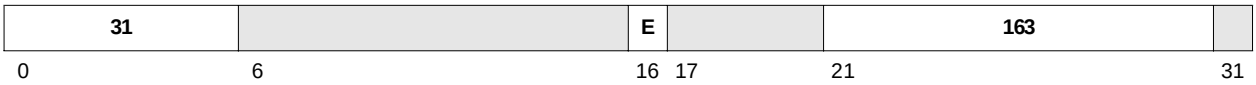
Programming Notes

Execution of this instruction is privileged.

This instruction is typically used as part of a code sequence which can provide the equivalent of an atomic read-modify-write of the MSR, as follows:

```
mfmsr Rn    #save EE in Rn[16]
wrteei 0    #Turn off EE (leaving other bits unchanged)
•           #Code with EE disabled
•
•
wrtee Rn    #restore EE without affecting any MSR changes that occurred in the disabled code
```

wrteei E



MSR[EE] ← E

MSR[EE] is set to the value specified by the E field.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- MSR[EE]

Invalid Instruction Forms:

- Reserved fields

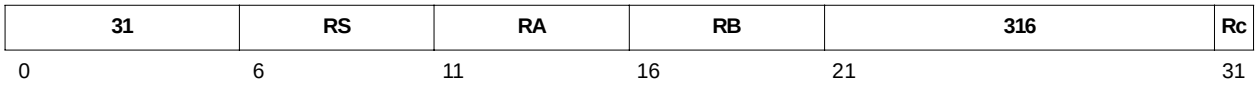
Programming Notes

Execution of this instruction is privileged.

This instruction is typically used as part of a code sequence which can provide the equivalent of an atomic read-modify-write of the MSR, as follows:

```
mfmsr Rn    #save EE in Rn[16]
wrteei 0    #Turn off EE (leaving other bits unchanged)
•           #Code with EE disabled
•
•
wrtee Rn    #restore EE without affecting any MSR changes that occurred in the disabled code
```

xor	RA, RS, RB	Rc=0
xor.	RA, RS, RB	Rc=1



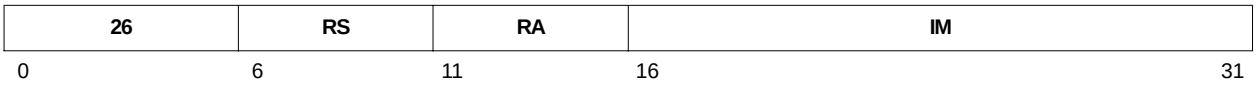
(RA) ← (RS) ⊕ (RB)

The contents of register RS are XORed with the contents of register RB; the result is placed into register RA.

Registers Altered

- RA
- CR[CR0] if Rc contains 1

xori RA, RS, IM



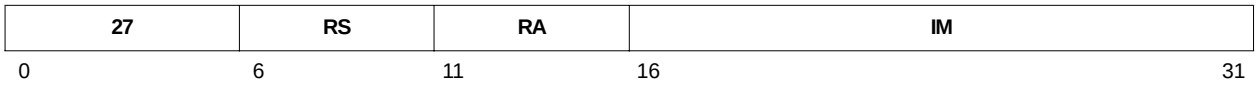
(RA) $\leftarrow$ (RS) $\oplus$ ($^{16}0$ || IM)

The IM field is extended to 32 bits by concatenating 16 0-bits on the left. The contents of register RS are XORed with the extended IM field; the result is placed into register RA.

Registers Altered

- RA

xoris RA, RS, IM



(RA) $\leftarrow$ (RS) $\oplus$ (IM || $^{16}0$)

The IM field is extended to 32 bits by concatenating 16 0-bits on the right. The contents of register RS are XORed with the extended IM field; the result is placed into register RA.

Registers Altered

- RA

29. Register Summary

This chapter provides an alphabetical listing of and bit definitions for the registers contained in the PPC440GP.

The registers, of five types, are grouped into several functional categories according to the processor functions with which they are associated. More information about the registers and register categories is provided in ~~"Registers" on page 4-10~~*Section 4.2 Registers* on page 154, and in the chapters describing the processor functions with which each register category is associated.

29.1 Register Categories

~~Table 29-1, "Register Categories," on page 29-2~~*Table 29-1*, summarizes the register categories and the registers contained in each category. Italicized register names are implementation-specific. All other registers are defined by the Book-E Enhanced PowerPC Architecture.

Table 0-1. Register Categories

Register Category	Register(s)	Model and Access	Type	Page
Branch Control	CR	User	CR	4-50
	CTR	User	SPR	4-49
	LR	User	SPR	4-49
Cache Control	*DNV0–DNV3*	Supervisor	SPR	5-2
	DTV0–DTV3	Supervisor	SPR	5-2
	DVLIM	Supervisor	SPR	5-4
	INV0–INV3	Supervisor	SPR	5-2
	ITV0–ITV3	Supervisor	SPR	5-2
	IVLIM	Supervisor	SPR	5-4
Cache Debug	*DCDBTRH, DCDBTRL*	Supervisor, read-only	SPR	5-27
	ICDBDR, ICDBTRH, ICDBTRL	Supervisor, read-only	SPR	5-15
Debug	DAC1–DAC2	Supervisor	SPR	15-30
	DBCR0–DBCR2	Supervisor	SPR	15-23
	DBDR	Supervisor	SPR	15-31
	DBSR	Supervisor	SPR	15-28
	DVC1–DVC2	Supervisor	SPR	15-31
	IAC1–IAC4	Supervisor	SPR	15-30
Device Control	Implemented outside core	Supervisor	DCR	4-16
Integer Processing	GPR0–GPR31	User	GPR	4-53
	XER	User	SPR	4-53
Interrupt Processing	CSRR0–CSRR1	Supervisor	SPR	10-10
	DEAR	Supervisor	SPR	10-11
	ESR	Supervisor	SPR	10-13
	IVOR0–IVOR15	Supervisor	SPR	10-11
	IVPR	Supervisor	SPR	10-12
	SRR0–SRR1	Supervisor	SPR	10-9

Table 0-1. Register Categories

Register Category	Register(s)	Model and Access	Type	Page
Processor Control	*CCR0*	Supervisor	SPR	5-13
	MSR	Supervisor	MSR	10-7
	PIR, PVR	Supervisor, read-only	SPR	4-58
	RSTCFG	Supervisor, read-only	SPR	4-60
	SPRG0–SPRG3	Supervisor	SPR	4-57
	SPRG4–SPRG7	User, read-only; Supervisor	SPR	4-57
	USPRG0	User, read-only; Supervisor	SPR	4-57
Storage Control	*MMUCR*	Supervisor	SPR	6-16
	PID	Supervisor	SPR	6-20
Timer	DEC	Supervisor	SPR	11-3
	DECAR	Supervisor, write-only	SPR	11-3
	TBL, TBU	User read, Supervisor write	SPR	11-2
	TCR	Supervisor	SPR	11-8
	TSR	Supervisor	SPR	11-9

Table 29-1. Register Categories

Register Category	Register(s)	Model and Access	Type	Page
Branch Control	CR	User	CR	189
	CTR	User	SPR	188
	LR	User	SPR	188
Cache Control	*DNV0–DNV3*	Supervisor	SPR	206
	DTV0–DTV3	Supervisor	SPR	206
	DVLIM	Supervisor	SPR	208
	INV0–INV3	Supervisor	SPR	206
	ITV0–ITV3	Supervisor	SPR	206
	IVLIM	Supervisor	SPR	208
Cache Debug	*DCDBTRH, DCDBTRL*	Supervisor, read-only	SPR	231
	ICDBDR, ICDBTRH, ICDBTRL	Supervisor, read-only	SPR	218
Debug	DAC1–DAC2	Supervisor	SPR	458
	DBCR0–DBCR2	Supervisor	SPR	451
	DBDR	Supervisor	SPR	459
	DBSR	Supervisor	SPR	456
	DVC1–DVC2	Supervisor	SPR	458
	IAC1–IAC4	Supervisor	SPR	457
Device Control	Implemented outside core	Supervisor	DCR	159
Integer Processing	GPR0–GPR31	User	GPR	192
	XER	User	SPR	192
Interrupt Processing	CSRR0–CSRR1	Supervisor	SPR	346
	DEAR	Supervisor	SPR	347
	ESR	Supervisor	SPR	349
	IVOR0–IVOR15	Supervisor	SPR	347
	IVPR	Supervisor	SPR	348
	SRR0–SRR1	Supervisor	SPR	345
Processor Control	*CCR0*	Supervisor	SPR	216
	MSR	Supervisor	MSR	343
	PIR, PVR	Supervisor, read-only	SPR	196
	RSTCFG	Supervisor, read-only	SPR	198
	SPRG0–SPRG3	Supervisor	SPR	195
	SPRG4–SPRG7	User, read-only; Supervisor	SPR	195
	USPRG0	User	SPR	195
Storage Control	*MMUCR*	Supervisor	SPR	247
	PID	Supervisor	SPR	251
Timer	DEC	Supervisor	SPR	385
	DECAR	Supervisor, write-only	SPR	385
	TBL, TBU	User read, Supervisor write	SPR	384
	TCR	Supervisor	SPR	389
	TSR	Supervisor	SPR	390

~~Table 29-2, "Special Purpose Registers Sorted by SPR Number," on page 29-4~~*Table 29-2 Special Purpose Registers Sorted by SPR Number* on page 1095, lists the Special Purpose Registers (SPRs) in order by SPR number (SPRN). The table provides mnemonics, names, SPRN, model (user or supervisor), and access. All SPR numbers not listed are reserved, and should be neither read nor written.

Note that two registers, DBSR and TSR, are indicated as having the access type of *read/clear*. These two registers are *status* registers, and as such behave differently than other SPRs when written. The term "read/clear" does not mean that these registers are automatically cleared upon being read. Rather, the "clear" refers to their behavior when being written. Instead of simply overwriting the SPR with the data in the source GPR, the status SPR is updated by zeroing those bit positions corresponding to 1 values in the source GPR, with those bit positions corresponding to 0 values in the source GPR being left unchanged. In this fashion, it is possible for software to read one of these status SPRs, and then write to it using the same data which was read. Any bits which were read as 1 will then be cleared, and any bits which were not yet set at the time the SPR was read will be left unchanged. If any of these previously clear bits happen to be set between the time the SPR is read and when it is written, then when the SPR is later read again, software will observe any newly set bit[s]. If it were not for this behavior, then software could erroneously clear bits which it had not yet observed as having been set, and overlook the occurrence of certain exceptions.

Table 0-2. Special Purpose Registers Sorted by SPR Number

Mnemonic	Register Name	SPRN	Model	Access
XER	Integer Exception Register	0x001	User	Read/Write
LR	Link Register	0x008	User	Read/Write
CTR	Count Register	0x009	User	Read/Write
DEC	Decrementer	0x016	Supervisor	Read/Write
SRR0	Save/Restore Register 0	0x01A	Supervisor	Read/Write
SRR1	Save/Restore Register 1	0x01B	Supervisor	Read/Write
PID	Process ID	0x030	Supervisor	Read/Write
DECAR	Decrementer Auto-Reload	0x036	Supervisor	Write-only
CSRR0	Critical Save/Restore Register 0	0x03A	Supervisor	Read/Write
CSRR1	Critical Save/Restore Register 1	0x03B	Supervisor	Read/Write
DEAR	Data Exception Address Register	0x03D	Supervisor	Read/Write
ESR	Exception Syndrome Register	0x03E	Supervisor	Read/Write
IVPR	Interrupt Vector Prefix Register	0x03F	Supervisor	Read/Write
USPRG0	User Special Purpose Register General 0	0x100	User	Read/Write
SPRG4	Special Purpose Register General 4	0x104	User	Read-only
SPRG5	Special Purpose Register General 5	0x105	User	Read-only
SPRG6	Special Purpose Register General 6	0x106	User	Read-only
SPRG7	Special Purpose Register General 7	0x107	User	Read-only
TBL	Time Base Lower	0x10C	User	Read-only
TBU	Time Base Upper	0x10D	User	Read-only
SPRG0	Special Purpose Register General 0	0x110	Supervisor	Read/Write
SPRG1	Special Purpose Register General 1	0x111	Supervisor	Read/Write
SPRG2	Special Purpose Register General 2	0x112	Supervisor	Read/Write
SPRG3	Special Purpose Register General 3	0x113	Supervisor	Read/Write
SPRG4	Special Purpose Register General 4	0x114	Supervisor	Write-only

Table 0-2. Special Purpose Registers Sorted by SPR Number (continued)

Mnemonic	Register Name	SPRN	Model	Access
SPRG5	Special Purpose Register General 5	0x115	Supervisor	Write-only
SPRG6	Special Purpose Register General 6	0x116	Supervisor	Write-only
SPRG7	Special Purpose Register General 7	0x117	Supervisor	Write-only
TBL	Time Base Lower	0x11C	Supervisor	Write-only
TBU	Time Base Upper	0x11D	Supervisor	Write-only
PIR	Processor ID Register	0x11E	Supervisor	Read-only
PVR	Processor Version Register	0x11F	Supervisor	Read-only
DBSR	Debug Status Register	0x130	Supervisor	Read/Clear
DBCR0	Debug Control Register 0	0x134	Supervisor	Read/Write
DBCR1	Debug Control Register 1	0x135	Supervisor	Read/Write
DBCR2	Debug Control Register 2	0x136	Supervisor	Read/Write
IAC1	Instruction Address Compare 1	0x138	Supervisor	Read/Write
IAC2	Instruction Address Compare 2	0x139	Supervisor	Read/Write
IAC3	Instruction Address Compare 3	0x13A	Supervisor	Read/Write
IAC4	Instruction Address Compare 4	0x13B	Supervisor	Read/Write
DAC1	Data Address Compare 1	0x13C	Supervisor	Read/Write
DAC2	Data Address Compare 2	0x13D	Supervisor	Read/Write
DVC1	Data Value Compare 1	0x13E	Supervisor	Read/Write
DVC2	Data Value Compare 2	0x13F	Supervisor	Read/Write
TSR	Timer Status Register	0x150	Supervisor	Read/Clear
TCR	Timer Control Register	0x154	Supervisor	Read/Write
IVOR0	Interrupt Vector Offset Register 0	0x190	Supervisor	Read/Write
IVOR1	Interrupt Vector Offset Register 1	0x191	Supervisor	Read/Write
IVOR2	Interrupt Vector Offset Register 2	0x192	Supervisor	Read/Write
IVOR3	Interrupt Vector Offset Register 3	0x193	Supervisor	Read/Write
IVOR4	Interrupt Vector Offset Register 4	0x194	Supervisor	Read/Write
IVOR5	Interrupt Vector Offset Register 5	0x195	Supervisor	Read/Write
IVOR6	Interrupt Vector Offset Register 6	0x196	Supervisor	Read/Write
IVOR7	Interrupt Vector Offset Register 7	0x197	Supervisor	Read/Write
IVOR8	Interrupt Vector Offset Register 8	0x198	Supervisor	Read/Write
IVOR9	Interrupt Vector Offset Register 9	0x199	Supervisor	Read/Write
IVOR10	Interrupt Vector Offset Register 10	0x19A	Supervisor	Read/Write
IVOR11	Interrupt Vector Offset Register 11	0x19B	Supervisor	Read/Write
IVOR12	Interrupt Vector Offset Register 12	0x19C	Supervisor	Read/Write
IVOR13	Interrupt Vector Offset Register 13	0x19D	Supervisor	Read/Write
IVOR14	Interrupt Vector Offset Register 14	0x19E	Supervisor	Read/Write
IVOR15	Interrupt Vector Offset Register 15	0x19F	Supervisor	Read/Write
INV0	Instruction Cache Normal Victim 0	0x370	Supervisor	Read/Write
INV1	Instruction Cache Normal Victim 1	0x371	Supervisor	Read/Write
INV2	Instruction Cache Normal Victim 2	0x372	Supervisor	Read/Write

Table 0-2. Special Purpose Registers Sorted by SPR Number (continued)

Mnemonic	Register Name	SPRN	Model	Access
INV3	Instruction Cache Normal Victim 3	0x373	Supervisor	Read/Write
ITV0	Instruction Cache Transient Victim 0	0x374	Supervisor	Read/Write
ITV1	Instruction Cache Transient Victim 1	0x375	Supervisor	Read/Write
ITV2	Instruction Cache Transient Victim 2	0x376	Supervisor	Read/Write
ITV3	Instruction Cache Transient Victim 3	0x377	Supervisor	Read/Write
DNV0	Data Cache Normal Victim 0	0x390	Supervisor	Read/Write
DNV1	Data Cache Normal Victim 1	0x391	Supervisor	Read/Write
DNV2	Data Cache Normal Victim 2	0x392	Supervisor	Read/Write
DNV3	Data Cache Normal Victim 3	0x393	Supervisor	Read/Write
DTV0	Data Cache Transient Victim 0	0x394	Supervisor	Read/Write
DTV1	Data Cache Transient Victim 1	0x395	Supervisor	Read/Write
DTV2	Data Cache Transient Victim 2	0x396	Supervisor	Read/Write
DTV3	Data Cache Transient Victim 3	0x397	Supervisor	Read/Write
DVLIM	Data Cache Victim Limit	0x398	Supervisor	Read/Write
IVLIM	Instruction Cache Victim Limit	0x399	Supervisor	Read/Write
RSTCFG	Reset Configuration	0x39B	Supervisor	Read-only
DCDBTRL	Data Cache Debug Tag Register Low	0x39C	Supervisor	Read-only
DCDBTRH	Data Cache Debug Tag Register High	0x39D	Supervisor	Read-only
ICDBTRL	Instruction Cache Debug Tag Register Low	0x39E	Supervisor	Read-only
ICDBTRH	Instruction Cache Debug Tag Register High	0x39F	Supervisor	Read-only
MMUCR	Memory Management Unit Control Register	0x3B2	Supervisor	Read/Write
CCR0	Core Configuration Register 0	0x3B3	Supervisor	Read/Write
ICDBDR	Instruction Cache Debug Data Register	0x3D3	Supervisor	Read-only
DBDR	Debug Data Register	0x3F3	Supervisor	Read/Write

Table 29-2. Special Purpose Registers Sorted by SPR Number

Mnemonic	Register Name	SPRN	Model	Access
XER	Integer Exception Register	0x001	User	Read/Write
LR	Link Register	0x008	User	Read/Write
CTR	Count Register	0x009	User	Read/Write
DEC	Decrementer	0x016	Supervisor	Read/Write
SRR0	Save/Restore Register 0	0x01A	Supervisor	Read/Write
SRR1	Save/Restore Register 1	0x01B	Supervisor	Read/Write
PID	Process ID	0x030	Supervisor	Read/Write
DECAR	Decrementer Auto-Reload	0x036	Supervisor	Write-only
CSRR0	Critical Save/Restore Register 0	0x03A	Supervisor	Read/Write
CSRR1	Critical Save/Restore Register 1	0x03B	Supervisor	Read/Write
DEAR	Data Exception Address Register	0x03D	Supervisor	Read/Write

Table 29-2. Special Purpose Registers Sorted by SPR Number

Mnemonic	Register Name	SPRN	Model	Access
ESR	Exception Syndrome Register	0x03E	Supervisor	Read/Write
IVPR	Interrupt Vector Prefix Register	0x03F	Supervisor	Read/Write
USPRG0	User Special Purpose Register General 0	0x100	User	Read/Write
SPRG4	Special Purpose Register General 4	0x104	User	Read-only
SPRG5	Special Purpose Register General 5	0x105	User	Read-only
SPRG6	Special Purpose Register General 6	0x106	User	Read-only
SPRG7	Special Purpose Register General 7	0x107	User	Read-only
TBL	Time Base Lower	0x10C	User	Read-only
TBU	Time Base Upper	0x10D	User	Read-only
SPRG0	Special Purpose Register General 0	0x110	Supervisor	Read/Write
SPRG1	Special Purpose Register General 1	0x111	Supervisor	Read/Write
SPRG2	Special Purpose Register General 2	0x112	Supervisor	Read/Write
SPRG3	Special Purpose Register General 3	0x113	Supervisor	Read/Write
SPRG4	Special Purpose Register General 4	0x114	Supervisor	Write-only
SPRG5	Special Purpose Register General 5	0x115	Supervisor	Write-only
SPRG6	Special Purpose Register General 6	0x116	Supervisor	Write-only
SPRG7	Special Purpose Register General 7	0x117	Supervisor	Write-only
TBL	Time Base Lower	0x11C	Supervisor	Write-only
TBU	Time Base Upper	0x11D	Supervisor	Write-only
PIR	Processor ID Register	0x11E	Supervisor	Read-only
PVR	Processor Version Register	0x11F	Supervisor	Read-only
DBSR	Debug Status Register	0x130	Supervisor	Read/Clear
DBCR0	Debug Control Register 0	0x134	Supervisor	Read/Write
DBCR1	Debug Control Register 1	0x135	Supervisor	Read/Write
DBCR2	Debug Control Register 2	0x136	Supervisor	Read/Write
IAC1	Instruction Address Compare 1	0x138	Supervisor	Read/Write
IAC2	Instruction Address Compare 2	0x139	Supervisor	Read/Write
IAC3	Instruction Address Compare 3	0x13A	Supervisor	Read/Write
IAC4	Instruction Address Compare 4	0x13B	Supervisor	Read/Write
DAC1	Data Address Compare 1	0x13C	Supervisor	Read/Write
DAC2	Data Address Compare 2	0x13D	Supervisor	Read/Write
DVC1	Data Value Compare 1	0x13E	Supervisor	Read/Write
DVC2	Data Value Compare 2	0x13F	Supervisor	Read/Write
TSR	Timer Status Register	0x150	Supervisor	Read/Clear
TCR	Timer Control Register	0x154	Supervisor	Read/Write
IVOR0	Interrupt Vector Offset Register 0	0x190	Supervisor	Read/Write
IVOR1	Interrupt Vector Offset Register 1	0x191	Supervisor	Read/Write

Table 29-2. Special Purpose Registers Sorted by SPR Number

Mnemonic	Register Name	SPRN	Model	Access
IVOR2	Interrupt Vector Offset Register 2	0x192	Supervisor	Read/Write
IVOR3	Interrupt Vector Offset Register 3	0x193	Supervisor	Read/Write
IVOR4	Interrupt Vector Offset Register 4	0x194	Supervisor	Read/Write
IVOR5	Interrupt Vector Offset Register 5	0x195	Supervisor	Read/Write
IVOR6	Interrupt Vector Offset Register 6	0x196	Supervisor	Read/Write
IVOR7	Interrupt Vector Offset Register 7	0x197	Supervisor	Read/Write
IVOR8	Interrupt Vector Offset Register 8	0x198	Supervisor	Read/Write
IVOR9	Interrupt Vector Offset Register 9	0x199	Supervisor	Read/Write
IVOR10	Interrupt Vector Offset Register 10	0x19A	Supervisor	Read/Write
IVOR11	Interrupt Vector Offset Register 11	0x19B	Supervisor	Read/Write
IVOR12	Interrupt Vector Offset Register 12	0x19C	Supervisor	Read/Write
IVOR13	Interrupt Vector Offset Register 13	0x19D	Supervisor	Read/Write
IVOR14	Interrupt Vector Offset Register 14	0x19E	Supervisor	Read/Write
IVOR15	Interrupt Vector Offset Register 15	0x19F	Supervisor	Read/Write
INV0	Instruction Cache Normal Victim 0	0x370	Supervisor	Read/Write
INV1	Instruction Cache Normal Victim 1	0x371	Supervisor	Read/Write
INV2	Instruction Cache Normal Victim 2	0x372	Supervisor	Read/Write
INV3	Instruction Cache Normal Victim 3	0x373	Supervisor	Read/Write
ITV0	Instruction Cache Transient Victim 0	0x374	Supervisor	Read/Write
ITV1	Instruction Cache Transient Victim 1	0x375	Supervisor	Read/Write
ITV2	Instruction Cache Transient Victim 2	0x376	Supervisor	Read/Write
ITV3	Instruction Cache Transient Victim 3	0x377	Supervisor	Read/Write
DNV0	Data Cache Normal Victim 0	0x390	Supervisor	Read/Write
DNV1	Data Cache Normal Victim 1	0x391	Supervisor	Read/Write
DNV2	Data Cache Normal Victim 2	0x392	Supervisor	Read/Write
DNV3	Data Cache Normal Victim 3	0x393	Supervisor	Read/Write
DTV0	Data Cache Transient Victim 0	0x394	Supervisor	Read/Write
DTV1	Data Cache Transient Victim 1	0x395	Supervisor	Read/Write
DTV2	Data Cache Transient Victim 2	0x396	Supervisor	Read/Write
DTV3	Data Cache Transient Victim 3	0x397	Supervisor	Read/Write
DVLIM	Data Cache Victim Limit	0x398	Supervisor	Read/Write
IVLIM	Instruction Cache Victim Limit	0x399	Supervisor	Read/Write
RSTCFG	Reset Configuration	0x39B	Supervisor	Read-only
DCDBTRL	Data Cache Debug Tag Register Low	0x39C	Supervisor	Read-only
DCDBTRH	Data Cache Debug Tag Register High	0x39D	Supervisor	Read-only
ICDBTRL	Instruction Cache Debug Tag Register Low	0x39E	Supervisor	Read-only
ICDBTRH	Instruction Cache Debug Tag Register High	0x39F	Supervisor	Read-only

Table 29-2. Special Purpose Registers Sorted by SPR Number

Mnemonic	Register Name	SPRN	Model	Access
MMUCR	Memory Management Unit Control Register	0x3B2	Supervisor	Read/Write
CCR0	Core Configuration Register 0	0x3B3	Supervisor	Read/Write
ICDBDR	Instruction Cache Debug Data Register	0x3D3	Supervisor	Read-only
DBDR	Debug Data Register	0x3F3	Supervisor	Read/Write

29.2 Reserved Fields

For all registers with fields marked as reserved, the reserved fields should be written as *zero* and read as *undefined*. That is, when writing to a reserved field, write a zero to that field. When reading from a reserved field, ignore that field.

The recommended coding practice is to perform the initial write to a register with reserved fields as described in the preceding paragraph, and to perform all subsequent writes to the register using a read-modify-write strategy: read the register, alter desired fields with logical instructions, and then write the register.

29.3 Device Control Registers

Device Control Registers (DCRs) are on-chip registers ~~that are architecturally outside of the processor core. They are~~ used to control, configure, and hold status for various functional units. DCRs are accessed using the **mfdcr** and **mtdcr** instructions.

The **mfdcr** and **mtdcr** instructions are privileged, for all DCR numbers. Therefore, all DCR accesses are privileged. All DCR numbers are reserved, and should be neither read nor written.

Some DCRs are directly accessed, that is, they are accessed using their DCR numbers. Other DCRs are indirectly accessed. Such DCRs are accessed by writing an offset to a directly accessed DCR and then reading the data at the offset in another directly accessed DCR. Table 29-3 lists the directly accessed DCRs. The indirectly accessed DCRs are described immediately following Table 29-3.

Table 0-3. Directly Accessed DCRs

Register	DCR Number	Access	Description
DCRs Used for Indirect Access			
SDRAM0_CFGADDR	0x010	R/W	DDR-SDRAM Address Register
SDRAM0_CFGDATA	0x011	R/W	DDR-SDRAM Data Register
EBC0_CFGADDR	0x012	R/W	EBCO Address Register
EBC0_CFGDATA	0x013	R/W	EBCO Data Register
EBM0_CFGADDR	0x014	R/W	EBMI Address Register
EBM0_CFGDATA	0x015	R/W	EBMI Data Register
PPM0_CFGADDR	0x016	R/W	PPM Address Register
PPM0_CFGDATA	0x017	R/W	PPM Data Register
Internal SRAM Controller			

Table 0-3. Directly Accessed DCRs

Register	DCR Number	Access	Description
SRAM0_SB0CR	0x020	R/W	SRAM Bank Configuration Register 0
SRAM0_SB1CR	0x021	R/W	SRAM Bank Configuration Register 1
SRAM0_SB2CR	0x022	R/W	SRAM Bank Configuration Register 2
SRAM0_SB3CR	0x023	R/W	SRAM Bank Configuration Register 3
SRAM0_BEAR	0x024	R/W	SRAM Bus Error Address Register
SRAM0_BESR0	0x025	R/W	SRAM Bus Error Status Register 0
SRAM0_BESR1	0x026	R/W	SRAM Bus Error Status Register 1
SRAM0_PMEG	0x027	R/W	SRAM Power Management
SRAM0_CID	0x028	R/W	SRAM Bus Core ID Register 1
SRAM0_REVID	0x029	R/W	SRAM Bus Revision ID Register
SRAM0_DPC	0x02A	R/W	SRAM Data Parity Check Register
On-Chip Buses			
PLB0_REVID	0x082	R	PLB Arbiter Revision ID
PLB0_ACR	0x083	R/W	PLB Arbiter Control Register
PLB0_BESR	0x084	R/Clear	PLB Bus Error Status Register
PLB0_BEARL	0x086	R	PLB Bus Error Address Register
PLB0_BEARH	0x087	R	PLB Bus Error Address Register
POB0_BESR0	0x090	R/Clear	PLB to OPB Bridge Error Status Register 0
POB0_BEARL	0x092	R	PLB to OPB Bridge Error Address Register
POB0_BEARH	0x093	R	PLB to OPB Bridge Error Address Register
POB0_BESR1	0x094	R/Clear	PLB to OPB Bridge Error Status Register 1
POB0_CONFG	0x096	R/Clear	PLB to OPB Bridge Error Configuration Register
POB0_LATENCY	0x098	R/Clear	PLB to OPB Bridge Burst Latency Timer
POB0_REVID	0x09A	R	PLB to OPB Bridge Revision ID Register
OPB0_BCTRL	0x0A8	R/W	OPB to PLB Bridge Control Register
OPB0_BSTAT	0x0A9	R/C	OPB to PLB Bridge Status Register
OPB0_BEARL	0x0AA	R/C	OPB to PLB Bridge Error Address Register Low
OPB0_BEARH	0x0AB	R/C	OPB to PLB Bridge Error Address Register High
OPB0_REVID	0x0AC	R/C	OPB to PLB Bridge Revision ID Register
Clocking, Power Management, and Chip Control			
CPC0_SR	0x0B0	R/W	CPM Status Register
CPC0_ER	0x0B1	R/W	CPM Enable Register
CPC0_FR	0x0B2	R/W	CPM Force Register
CPC0_SYS0	0x0E0	R/W	System Configuration Register 0
CPC0_SYS1	0x0E1	R/W	System Configuration Register 1
CPC0_CUST0	0x0E2	R/W	Customer Configuration Register 0
CPC0_CUST1	0x0E3	R/W	Customer Configuration Register 1
CPC0_STRP0	0x0E4	R	Power-on Configuration Register 0
CPC0_STRP1	0x0E5	R	Power-on Configuration Register 1
CPC0_STRP2	0x0E6	R	Power-on Configuration Register 3

Table 0-3. Directly Accessed DCRs

Register	DCR Number	Access	Description
CPC0_STRP3	0x0E7	R	Power-on Configuration Register 4
CPC0_GPIO	0x0E8	R/W	GPIO Configuration Register
CPC0_PLB	0x0E9	R/W	PLB Configuration Register
CPC0_CR1	0x0EA	R/W	CPU Master Priority Configuration Register 1
CPC0_CR0	0x0EB	R/W	UART and Timer Configuration Register 0
CPC0_MIRQ0	0x0EC	R/W	Master Interrupt Request Register 0
CPC0_MIRQ1	0x0ED	R/W	Master Interrupt Request Register 1
CPC0_JTAGID	0x0EF	R	JTAG ID Register
Universal Interrupt Controllers			
UIC0_SR	0x0C0	R/Clear	UIC 0 Status Register
UIC0_ER	0x0C2	R/W	UIC 0 Enable Register
UIC0_CR	0x0C3	R/W	UIC 0 Critical Register
UIC0_PR	0x0C4	R/W	UIC 0 Polarity Register
UIC0_TR	0x0C5	R/W	UIC 0 Triggering Register
UIC0_MSR	0x0C6	R	UIC 0 Masked Status Register
UIC0_VR	0x0C7	R	UIC 0 Vector Register
UIC0_VCR	0x0C8	W	UIC 0 Vector Configuration Register
UIC1_SR	0x0D0	R/Clear	UIC 1 Status Register
UIC1_ER	0x0D2	R/W	UIC 1 Enable Register
UIC1_CR	0x0D3	R/W	UIC 1 Critical Register
UIC1_PR	0x0D4	R/W	UIC 1 Polarity Register
UIC1_TR	0x0D5	R/W	UIC 1 Triggering Register
UIC1_MSR	0x0D6	R	UIC 1 Masked Status Register
UIC1_VR	0x0D7	R	UIC 1 Vector Register
UIC1_VCR	0x0D8	W	UIC 1 Vector Configuration Register
Direct Memory Access			
DMA0_CR0	0x100	R/W	DMA Channel Control Register 0
DMA0_CT0	0x101	R/W	DMA Count Register 0
DMA0_SAH0	0x102	R/W	DMA Source Address High Register 0
DMA0_SAL0	0x103	R/W	DMA Source Address Low Register 0
DMA0_DAH0	0x104	R/W	DMA Destination Address High Register 0
DMA0_DAL0	0x105	R/W	DMA Destination Address Low Register 0
DMA0_SGH0	0x106	R/W	DMA Scatter/Gather Descriptor Address High Register 0
DMA0_SGL0	0x107	R/W	DMA Scatter/Gather Descriptor Address Low Register 0
DMA0_CR1	0x108	R/W	DMA Channel Control Register 1
DMA0_CT1	0x109	R/W	DMA Count Register 1
DMA0_SAH1	0x10A	R/W	DMA Source Address High Register 1
DMA0_SAL1	0x10B	R/W	DMA Source Address Low Register 1

Table 0-3. Directly Accessed DCRs

Register	DCR Number	Access	Description
DMA0_DAH1	0x10C	R/W	DMA Destination Address High Register 1
DMA0_DAL1	0x10D	R/W	DMA Destination Address Low Register 1
DMA0_SGH1	0x10E	R/W	DMA Scatter/Gather Descriptor Address High Register 1
DMA0_SGL1	0x10F	R/W	DMA Scatter/Gather Descriptor Address Low Register 1
DMA0_CR2	0x110	R/W	DMA Channel Control Register 2
DMA0_CT2	0x111	R/W	DMA Count Register 2
DMA0_SAH2	0x112	R/W	DMA Source Address High Register 2
DMA0_SAL2	0x113	R/W	DMA Source Address Low Register 2
DMA0_DAH2	0x114	R/W	DMA Destination Address High Register 2
DMA0_DAL2	0x115	R/W	DMA Destination Address Low Register 2
DMA0_SGH2	0x116	R/W	DMA Scatter/Gather Descriptor Address High Register 2
DMA0_SGL2	0x117	R/W	DMA Scatter/Gather Descriptor Address Low Register 2
DMA0_CR3	0x118	R/W	DMA Channel Control Register 3
DMA0_CT3	0x119	R/W	DMA Count Register 3
DMA0_SAH3	0x11A	R/W	DMA Source Address High Register 3
DMA0_SAL3	0x11B	R/W	DMA Source Address Low Register 3
DMA0_DAH3	0x11C	R/W	DMA Destination Address High Register 3
DMA0_DAL3	0x11D	R/W	DMA Destination Address Low Register 3
DMA0_SGH3	0x11E	R/W	DMA Scatter/Gather Descriptor Address High Register 3
DMA0_SGL3	0x11F	R/W	DMA Scatter/Gather Descriptor Address Low Register 3
DMA0_SR	0x120	R/Clear	DMA Status Register
DMA0_SGC	0x123	R/W	DMA Scatter/Gather Command Register
DMA0_SLP	0x125	R/W	DMA Sleep Mode Register
DMA0_POL	0x126	R/W	DMA Polarity Configuration Register
Memory Access Layer			
MAL0_CFG	0x180	R/W	MAL Configuration Register
MAL0_ESR	0x181	R/Clear	MAL Error Status Register
MAL0_IER	0x182	R/W	MAL Interrupt Enable Register
MAL0_TXCASR	0x184	R/W	MAL Tx Channel Active Register (Set)
MAL0_TXCARR	0x185	R/W	MAL Tx Channel Active Register (Reset)
MAL0_TXEOBISR	0x186	R/Clear	MAL Tx End of Buffer Interrupt Status Register
MAL0_TXDEIR	0x187	R/Clear	MAL Tx Descriptor Error Interrupt Register
MAL0_TXTATTRR	0x188	R/W	MAL Tx PLB Attribute Register
MAL0_TXBADDR	0x189	R/W	MAL Tx Descriptor Base Address Register
MAL0_RXCASR	0x190	R/W	MAL Rx Channel Active Register (Set)

Table 0-3. Directly Accessed DCRs

Register	DCR Number	Access	Description
MAL0_RXCARR	0x191	R/W	MAL Rx Channel Active Register (Reset)
MAL0_RXEOBISR	0x192	R/Clear	MAL Rx End of Buffer Interrupt Status Register
MAL0_RXDEIR	0x193	R/Clear	MAL Rx Descriptor Error Interrupt Register
MAL0_RXTATTRR	0x194	R/W	MAL Rx PLB Attribute Register
MAL0_RXBADDR	0x195	R/W	MAL Rx Descriptor Base Address Register
MAL0_TXCTP0R	0x1A0	R/W	MAL TX Channel 0 Table Pointer Register
MAL0_TXCTP1R	0x1A1	R/W	MAL TX Channel 1 Table Pointer Register
MAL0_TXCTP2R	0x1A2	R/W	MAL TX Channel 2 Table Pointer Register
MAL0_TXCTP3R	0x1A3	R/W	MAL TX Channel 3 Table Pointer Register
MAL0_RXCTP0R	0x1C0	R/W	MAL RX Channel 0 Table Pointer Register
MAL0_RXCTP1R	0x1C1	R/W	MAL RX Channel 1 Table Pointer Register
MAL0_RCBS0	0x1E0	R/W	MAL RX Channel 0 Buffer Size Register
MAL0_RCBS1	0x1E1	R/W	MAL RX Channel 1 Buffer Size Register

Table 29-3. Directly Accessed DCRs

Register	DCR Number	Access	Description
DCRs Used for Indirect Access			
SDRAM0_CFGADDR	0x010	R/W	DDR-SDRAM Address Register
SDRAM0_CFGDATA	0x011	R/W	DDR-SDRAM Data Register
EBC0_CFGADDR	0x012	R/W	EBCO Address Register
EBC0_CFGDATA	0x013	R/W	EBCO Data Register
EBM0_CFGADDR	0x014	R/W	EBMI Address Register
EBM0_CFGDATA	0x015	R/W	EBMI Data Register
PPM0_CFGADDR	0x016	R/W	PPM Address Register
PPM0_CFGDATA	0x017	R/W	PPM Data Register
Internal SRAM Controller			
SRAM0_SB0CR	0x020	R/W	SRAM Bank Configuration Register 0
SRAM0_SB1CR	0x021	R/W	SRAM Bank Configuration Register 1
SRAM0_SB2CR	0x022	R/W	SRAM Bank Configuration Register 2
SRAM0_SB3CR	0x023	R/W	SRAM Bank Configuration Register 3
SRAM0_BEAR	0x024	R/W	SRAM Bus Error Address Register
SRAM0_BESR0	0x025	R/W	SRAM Bus Error Status Register 0
SRAM0_BESR1	0x026	R/W	SRAM Bus Error Status Register 1
SRAM0_PMEG	0x027	R/W	SRAM Power Management
SRAM0_CID	0x028	R/W	SRAM Bus Core ID Register 1
SRAM0_REVID	0x029	R/W	SRAM Bus Revision ID Register

Table 29-3. Directly Accessed DCRs

Register	DCR Number	Access	Description
SRAM0_DPC	0x02A	R/W	SRAM Data Parity Check Register
On-Chip Buses			
PLB0_REVID	0x082	R	PLB Arbiter Revision ID
PLB0_ACR	0x083	R/W	PLB Arbiter Control Register
PLB0_BESR	0x084	R/Clear	PLB Bus Error Status Register
PLB0_BEARL	0x086	R	PLB Bus Error Address Register
PLB0_BEARH	0x087	R	PLB Bus Error Address Register
POB0_BESR0	0x090	R/Clear	PLB to OPB Bridge Error Status Register 0
POB0_BEARL	0x092	R	PLB to OPB Bridge Error Address Register
POB0_BEARH	0x093	R	PLB to OPB Bridge Error Address Register
POB0_BESR1	0x094	R/Clear	PLB to OPB Bridge Error Status Register 1
POB0_CONFG	0x096	R/Clear	PLB to OPB Bridge Error Configuration Register
POB0_LATENCY	0x098	R/Clear	PLB to OPB Bridge Burst Latency Timer
POB0_REVID	0x09A	R	PLB to OPB Bridge Revision ID Register
OPB0_BCTRL	0x0A8	R/W	OPB to PLB Bridge Control Register
OPB0_BSTAT	0x0A9	R/C	OPB to PLB Bridge Status Register
OPB0_BEARL	0x0AA	R/C	OPB to PLB Bridge Error Address Register Low
OPB0_BEARH	0x0AB	R/C	OPB to PLB Bridge Error Address Register High
OPB0_REVID	0x0AC	R/C	OPB to PLB Bridge Revision ID Register
Clocking, Power Management, and Chip Control			
CPC0_SR	0x0B0	R/W	CPM Status Register
CPC0_ER	0x0B1	R/W	CPM Enable Register
CPC0_FR	0x0B2	R/W	CPM Force Register
CPC0_SYS0	0x0E0	R/W	System Configuration Register 0
CPC0_SYS1	0x0E1	R/W	System Configuration Register 1
CPC0_CUST0	0x0E2	R/W	Customer Configuration Register 0
CPC0_CUST1	0x0E3	R/W	Customer Configuration Register 1
CPC0_STRP0	0x0E4	R	Power-on Configuration Register 0
CPC0_STRP1	0x0E5	R	Power-on Configuration Register 1
CPC0_STRP2	0x0E6	R	Power-on Configuration Register 3
CPC0_STRP3	0x0E7	R	Power-on Configuration Register 4
CPC0_GPIO	0x0E8	R/W	GPIO Configuration Register
CPC0_PLB	0x0E9	R/W	PLB Configuration Register
CPC0_CR1	0x0EA	R/W	CPU Master Priority Configuration Register 1
CPC0_CR0	0x0EB	R/W	UART and Timer Configuration Register 0
CPC0_MIRQ0	0x0EC	R/W	Master Interrupt Request Register 0

Table 29-3. Directly Accessed DCRs

Register	DCR Number	Access	Description
CPC0_MIRQ1	0x0ED	R/W	Master Interrupt Request Register 1
CPC0_JTAGID	0x0EF	R	JTAG ID Register
Universal Interrupt Controllers			
UIC0_SR	0x0C0	R/Clear	UIC 0 Status Register
UIC0_ER	0x0C2	R/W	UIC 0 Enable Register
UIC0_CR	0x0C3	R/W	UIC 0 Critical Register
UIC0_PR	0x0C4	R/W	UIC 0 Polarity Register
UIC0_TR	0x0C5	R/W	UIC 0 Triggering Register
UIC0_MSR	0x0C6	R	UIC 0 Masked Status Register
UIC0_VR	0x0C7	R	UIC 0 Vector Register
UIC0_VCR	0x0C8	W	UIC 0 Vector Configuration Register
UIC1_SR	0x0D0	R/Clear	UIC 1 Status Register
UIC1_ER	0x0D2	R/W	UIC 1 Enable Register
UIC1_CR	0x0D3	R/W	UIC 1 Critical Register
UIC1_PR	0x0D4	R/W	UIC 1 Polarity Register
UIC1_TR	0x0D5	R/W	UIC 1 Triggering Register
UIC1_MSR	0x0D6	R	UIC 1 Masked Status Register
UIC1_VR	0x0D7	R	UIC 1 Vector Register
UIC1_VCR	0x0D8	W	UIC 1 Vector Configuration Register
Direct Memory Access			
DMA0_CR0	0x100	R/W	DMA Channel Control Register 0
DMA0_CT0	0x101	R/W	DMA Count Register 0
DMA0_SAH0	0x102	R/W	DMA Source Address High Register 0
DMA0_SAL0	0x103	R/W	DMA Source Address Low Register 0
DMA0_DAH0	0x104	R/W	DMA Destination Address High Register 0
DMA0_DAL0	0x105	R/W	DMA Destination Address Low Register 0
DMA0_SGH0	0x106	R/W	DMA Scatter/Gather Descriptor Address High Register 0
DMA0_SGL0	0x107	R/W	DMA Scatter/Gather Descriptor Address Low Register 0
DMA0_CR1	0x108	R/W	DMA Channel Control Register 1
DMA0_CT1	0x109	R/W	DMA Count Register 1
DMA0_SAH1	0x10A	R/W	DMA Source Address High Register 1
DMA0_SAL1	0x10B	R/W	DMA Source Address Low Register 1
DMA0_DAH1	0x10C	R/W	DMA Destination Address High Register 1
DMA0_DAL1	0x10D	R/W	DMA Destination Address Low Register 1
DMA0_SGH1	0x10E	R/W	DMA Scatter/Gather Descriptor Address High Register 1
DMA0_SGL1	0x10F	R/W	DMA Scatter/Gather Descriptor Address Low Register 1

Table 29-3. Directly Accessed DCRs

Register	DCR Number	Access	Description
DMA0_CR2	0x110	R/W	DMA Channel Control Register 2
DMA0_CT2	0x111	R/W	DMA Count Register 2
DMA0_SAH2	0x112	R/W	DMA Source Address High Register 2
DMA0_SAL2	0x113	R/W	DMA Source Address Low Register 2
DMA0_DAH2	0x114	R/W	DMA Destination Address High Register 2
DMA0_DAL2	0x115	R/W	DMA Destination Address Low Register 2
DMA0_SGH2	0x116	R/W	DMA Scatter/Gather Descriptor Address High Register 2
DMA0_SGL2	0x117	R/W	DMA Scatter/Gather Descriptor Address Low Register 2
DMA0_CR3	0x118	R/W	DMA Channel Control Register 3
DMA0_CT3	0x119	R/W	DMA Count Register 3
DMA0_SAH3	0x11A	R/W	DMA Source Address High Register 3
DMA0_SAL3	0x11B	R/W	DMA Source Address Low Register 3
DMA0_DAH3	0x11C	R/W	DMA Destination Address High Register 3
DMA0_DAL3	0x11D	R/W	DMA Destination Address Low Register 3
DMA0_SGH3	0x11E	R/W	DMA Scatter/Gather Descriptor Address High Register 3
DMA0_SGL3	0x11F	R/W	DMA Scatter/Gather Descriptor Address Low Register 3
DMA0_SR	0x120	R/Clear	DMA Status Register
DMA0_SGC	0x123	R/W	DMA Scatter/Gather Command Register
DMA0_SLP	0x125	R/W	DMA Sleep Mode Register
DMA0_POL	0x126	R/W	DMA Polarity Configuration Register
Memory Access Layer			
MAL0_CFG	0x180	R/W	MAL Configuration Register
MAL0_ESR	0x181	R/Clear	MAL Error Status Register
MAL0_IER	0x182	R/W	MAL Interrupt Enable Register
MAL0_TXCASR	0x184	R/W	MAL Tx Channel Active Register (Set)
MAL0_TXCARR	0x185	R/W	MAL Tx Channel Active Register (Reset)
MAL0_TXEOBISR	0x186	R/Clear	MAL Tx End of Buffer Interrupt Status Register
MAL0_TXDEIR	0x187	R/Clear	MAL Tx Descriptor Error Interrupt Register
MAL0_TXTATTRR	0x188	R/W	MAL Tx PLB Attribute Register
MAL0_TXBADDR	0x189	R/W	MAL Tx Descriptor Base Address Register
MAL0_RXCASR	0x190	R/W	MAL Rx Channel Active Register (Set)
MAL0_RXCARR	0x191	R/W	MAL Rx Channel Active Register (Reset)
MAL0_RXEOBISR	0x192	R/Clear	MAL Rx End of Buffer Interrupt Status Register
MAL0_RXDEIR	0x193	R/Clear	MAL Rx Descriptor Error Interrupt Register
MAL0_RXTATTRR	0x194	R/W	MAL Rx PLB Attribute Register
MAL0_RXBADDR	0x195	R/W	MAL Rx Descriptor Base Address Register

Table 29-3. Directly Accessed DCRs

Register	DCR Number	Access	Description
MAL0_TXCTP0R	0x1A0	R/W	MAL TX Channel 0 Table Pointer Register
MAL0_TXCTP1R	0x1A1	R/W	MAL TX Channel 1 Table Pointer Register
MAL0_TXCTP2R	0x1A2	R/W	MAL TX Channel 2 Table Pointer Register
MAL0_TXCTP3R	0x1A3	R/W	MAL TX Channel 3 Table Pointer Register
MAL0_RXCTP0R	0x1C0	R/W	MAL RX Channel 0 Table Pointer Register
MAL0_RXCTP1R	0x1C1	R/W	MAL RX Channel 1 Table Pointer Register
MAL0_RCBS0	0x1E0	R/W	MAL RX Channel 0 Buffer Size Register
MAL0_RCBS1	0x1E1	R/W	MAL RX Channel 1 Buffer Size Register

~~Indirectly Accessed DCRs~~

~~The DCRs for the DDR-SDRAM controller, external bus controller (EBCO), and external bus master interface (EBMI) are indirectly accessed.~~

~~Indirect Access of DDR-SDRAM Controller DCRs~~

~~The following procedure accesses the DDR-SDRAM controller DCRs listed in Table 29-4.~~

~~1. Write the offset from Table 29-5 to the DDR-SDRAM Address Register (SDRAM0_CFGADDR).~~

~~2. Read data from or write data to the DDR-SDRAM Data Register (SDRAM0_CFGDATA).~~

Table 0-4. SDRAM Controller DCR Usage

Register	DCR Number	Access	Description
SDRAM0_CFGADDR	0x010	R/W	DDR-SDRAM Address Register
SDRAM0_CFGDATA	0x011	R/W	DDR-SDRAM Data Register

Table 0-5. Offsets for SDRAM Controller Registers

Register	Offset	R/W	Description
SDRAM0_BESR0	0x00	R/W	DDR-SDRAM Bus Error Syndrome Register 0
SDRAM0_BESR1	0x08	R/W	DDR-SDRAM Bus Error Syndrome Register 1
SDRAM0_BEAR	0x10	R	DDR-SDRAM Bus Error Address Register
SDRAM0_MIRQ	0x11	R/W	DDR-SDRAM Master Write Interrupt
SDRAM0_SLIO	0x18	R/W	DDR-SDRAM Slave Interface Options
SDRAM0_CFG0	0x20	R/W	DDR-SDRAM Options 0
SDRAM0_CFG1	0x21	R/W	DDR-SDRAM Options 1
SDRAM0_DEVOPT	0x22	R/W	DDR-SDRAM Device Options
SDRAM0_MCSTS	0x24	R	DDR-SDRAM Memory Controller Status
SDRAM0_RTR	0x30	R/W	DDR-SDRAM Refresh Timer Register
SDRAM0_PMIT	0x34	R/W	DDR-SDRAM Power Management Idle Timer

Table 0-5. Offsets for SDRAM Controller Registers (continued)

Register	Offset	R/W	Description
SDRAM0_UABBA	0x38	R/W	DDR-SDRAM PLB UA Bus Base Address
SDRAM0_B0CR	0x40	R/W	DDR-SDRAM Bank 0 Configuration Register
SDRAM0_B1CR	0x44	R/W	DDR-SDRAM Bank 1 Configuration Register
SDRAM0_B2CR	0x48	R/W	DDR-SDRAM Bank 2 Configuration Register
SDRAM0_B3CR	0x4C	R/W	DDR-SDRAM Bank 3 Configuration Register
SDRAM0_TR0	0x80	R/W	DDR-SDRAM Timing Register 0
SDRAM0_TR1	0x81	R/W	DDR-SDRAM Timing Register 1
SDRAM0_CLKTR	0x82	R/W	DDR-SDRAM Clock Timing Register
SDRAM0_WDDCTR	0x83	R/W	DDR-SDRAM Write Data, DQS, DM Clock Timing Register
SDRAM0_DLYCAL	0x84	R/W	DDR-SDRAM Delay Line Calibration Register
SDRAM0_ECCESR	0x98	R/W	DDR-SDRAM ECC Error Status Register
SDRAM0_CID	0xA4	R	DDR-SDRAM Core ID Register
SDRAM0_RID	0xA8	R	DDR-SDRAM Revision ID Register

~~Indirect Access of External Bus Controller DCRs~~

~~The following procedure accesses the EBCO DCRs listed in Table 29-6.~~

~~1. Write the offset from Table 29-7 to the EBCO Address Register (EBC0_CFGADDR).~~

~~2. Read data from or write data to the EBCO Data Register (EBC0_CFGDATA).~~

Table 0-6. External Bus Controller DCR Usage

Register	DCR Number	Access	Description
EBC0_CFGADDR	0x012	R/W	EBCO Controller Address Register
EBC0_CFGDATA	0x013	R/W	EBCO Controller Data Register

Indirect Access of External Bus Controller DCRs

Table 0-7. Offsets for External Bus Controller Registers

Register	Offset	Access	Description
EBC0_B0CR	0x00	R/W	EBCO Bank 0 Configuration Register
EBC0_B1CR	0x01	R/W	EBCO Bank 1 Configuration Register
EBC0_B2CR	0x02	R/W	EBCO Bank 2 Configuration Register
EBC0_B3CR	0x03	R/W	EBCO Bank 3 Configuration Register
EBC0_B4CR	0x04	R/W	EBCO Bank 4 Configuration Register
EBC0_B5CR	0x05	R/W	EBCO Bank 5 Configuration Register
EBC0_B6CR	0x06	R/W	EBCO Bank 6 Configuration Register
EBC0_B7CR	0x07	R/W	EBCO Bank 7 Configuration Register
EBC0_B0AP	0x10	R/W	EBCO Bank 0 Access Parameters
EBC0_B1AP	0x11	R/W	EBCO Bank 1 Access Parameters
EBC0_B2AP	0x12	R/W	EBCO Bank 2 Access Parameters
EBC0_B3AP	0x13	R/W	EBCO Bank 3 Access Parameters
EBC0_B4AP	0x14	R/W	EBCO Bank 4 Access Parameters
EBC0_B5AP	0x15	R/W	EBCO Bank 5 Access Parameters
EBC0_B6AP	0x16	R/W	EBCO Bank 6 Access Parameters
EBC0_B7AP	0x17	R/W	EBCO Bank 7 Access Parameters
EBC0_BEAR	0x20	R/W	EBCO Bus Error Address Register
EBC0_BESR	0x21	R/W	EBCO Bus Error Status Register 0
EBC0_CFG	0x23	R/W	EBCO Peripheral Control Register
EBC0_CID	0x24	R/W	EBCO Peripheral Core ID Register

Indirect Access of External Bus Master DCRs

The following procedure accesses the EBMI DCRs listed in Table 29-8.

1. Write the offset from Table 29-9 to the EBMI Address Register (EBM0_CFGADDR).
2. Read data from or write data to the EBMI Data Register (EBM0_CFGDATA).

Table 0-8. External Bus Master DCR Usage

Register	DCR Number	Access	Description
EBM0_CFGADDR	0x014	R/W	EBMI Controller Address Register
EBM0_CFGDATA	0x015	R/W	EBMI Controller Data Register

Table 0-9. Offsets for External Bus Master Registers

Register	Offset	Access	Description
EBM0_CTL	0x00	R/W	EBMI Control Register
EBM0_LCNT	0x01	R/W	EBMI OPB Latency Count Register
EBM0_BEAR	0x02	R/W	EBMI Bus Error Address Register

Table 0-9. Offsets for External Bus Master Registers (continued)

Register	Offset	Access	Description
EBM0_BESR	0x03	R/W	EBMI Bus Error Status Register
EBM0_BEMR	0x04	R/W	EBMI Bus Error Mask Register
EBM0_UAR	0x05	R/W	EBMI OPB Upper Address Register
EBM0_UAM	0x06	R/W	EBMI OPB Upper Address Mask
EBM0_SLPMD	0x07	R/W	EBMI Sleep Mode Register
EBM0_FAIR	0x08	R/W	EBMI Fairness Control Register
EBM0_CID	0x11	R	EBMI Core ID Register

~~Indirect Access of PLB Performance Monitor DCRs~~

~~The following procedure accesses the PPM DCRs listed in Table 29-10.~~

~~1. Write the offset from Table 29-11 to the PPM Address Register (PPM0_CFGADDR).~~

~~2. Read data from or write data to the PPM Data Register (PPM0_CFGDATA).~~

Table 0-10. PLB Performance Monitor DCR Usage

Register	DCR Number	Access	Description
PPM0_CFGADDR	0x016	R/W	PPM Address Register
PPM0_CFGDATA	0x017	R/W	PPM Data Register

Table 0-11. Offsets for PLB Performance Monitor Registers

Register	Offset	Access	Description
PPM0_ISR	0x0	Read	Interrupt Status Register
PPM0_CR	0x2	R/W	Control Register
PPM0_CCR	0x3	R/W	Cycle Control Register
PPM0_UAR	0x4	R/W	Upper Address Register
PPM0_LAR	0x5	R/W	Lower Address Register
PPM0_UAMR	0x6	R/W	Upper Address Mask Register
PPM0_LAMR	0x7	R/W	Lower Address Mask Register
PPM0_RIDR	0x8	R	Revision ID Register
PPM0_MCSR0	0x9	R/W	Master Event Counter Selection Register 0
PPM0_MCSR1	0xA	R/W	Master Event Counter Selection Register 1
PPM0_MCSR2	0xB	R/W	Master Event Counter Selection Register 2
PPM0_MCSR3	0xC	R/W	Master Event Counter Selection Register 3
PPM0_SCSR0	0x11	R/W	Slave Event Counter Selection Register 0
PPM0_SCSR1	0x12	R/W	Slave Event Counter Selection Register 1
PPM0_SCSR2	0x13	R/W	Slave Event Counter Selection Register 2
PPM0_SCSR3	0x14	R/W	Slave Event Counter Selection Register 3
PPM0_GCSR0	0x19	R/W	Generic Event Counter Selection Register 0
PPM0_GCSR1	0x1A	R/W	Generic Event Counter Selection Register 1

Table 0-11. Offsets for PLB Performance Monitor Registers (continued)

Register	Offset	Access	Description
PPM0_GCSR2	0x1B	R/W	Generic Event Counter Selection Register 2
PPM0_GCSR3	0x1C	R/W	Generic Event Counter Selection Register 3
PPM0_MCR0	0x1D	R/W	Master Event Counter Register 0
PPM0_MCR1	0x1E	R/W	Master Event Counter Register 1
PPM0_MCR2	0x1F	R/W	Master Event Counter Register 2
PPM0_MCR3	0x20	R/W	Master Event Counter Register 3
PPM0_SCR0	0x25	R/W	Slave Event Counter Register 0
PPM0_SCR1	0x26	R/W	Slave Event Counter Register 1
PPM0_SCR2	0x27	R/W	Slave Event Counter Register 2
PPM0_SCR3	0x28	R/W	Slave Event Counter Register 3
PPM0_GCR0	0x2D	R/W	Generic Pipeline Event Counter Register 0
PPM0_GCR1	0x2E	R/W	Generic Pipeline Event Counter Register 1
PPM0_GCR2	0x2F	R/W	Generic Pipeline Event Counter Register 2
PPM0_GCR3	0x30	R/W	Generic Pipeline Event Counter Register 3
PPM0_DCSR0	0x31	R/W	Duration Counter Selection Register 0
PPM0_DCSR1	0x32	R/W	Duration Counter Selection Register 1
PPM0_DCMXR0	0x33	R/W	Duration Counter Max Register 0
PPM0_DCMXR1	0x34	R/W	Duration Counter Max Register 1
PPM0_DCMNR0	0x35	R/W	Duration Counter Min Register 0
PPM0_DCMNR1	0x36	R/W	Duration Counter Minimum Register 1
PPM0_DCTVR0	0x37	R/W	Duration Counter Total Value Register 0
PPM0_DCTVR1	0x38	R/W	Duration Counter Total Value Register 1
PPM0_DCOTR0	0x39	R/W	Duration Counter Occurrence Total Register 0
PPM0_DCOTR1	0x3A	R/W	Duration Counter Occurrence Total Register 1

Indirectly Accessed DCRs

The DCRs for the DDR-SDRAM controller, external bus controller (EBCO), and external bus master interface (EBMI) are indirectly accessed.

Indirect Access of DDR-SDRAM Controller DCRs

The following procedure accesses the DDR-SDRAM controller DCRs listed in Table 29-4.

1. Write the offset from Table 29-5 to the DDR-SDRAM Address Register (SDRAM0_CFGADDR).
2. Read data from or write data to the DDR-SDRAM Data Register (SDRAM0_CFGDATA).

Table 29-4. SDRAM Controller DCR Usage

Register	DCR Number	Access	Description
SDRAM0_CFGADDR	0x010	R/W	DDR-SDRAM Address Register
SDRAM0_CFGDATA	0x011	R/W	DDR-SDRAM Data Register

Table 29-5. Offsets for SDRAM Controller Registers

Register	Offset	R/W	Description
SDRAM0_BESR0	0x00	R/W	DDR-SDRAM Bus Error Syndrome Register 0
SDRAM0_BESR1	0x08	R/W	DDR-SDRAM Bus Error Syndrome Register 1
SDRAM0_BEAR	0x10	R	DDR-SDRAM Bus Error Address Register
SDRAM0_MIRQ	0x11	R/W	DDR-SDRAM Master Write Interrupt
SDRAM0_SLIO	0x18	R/W	DDR-SDRAM Slave Interface Options
SDRAM0_CFG0	0x20	R/W	DDR-SDRAM Options 0
SDRAM0_CFG1	0x21	R/W	DDR-SDRAM Options 1
SDRAM0_DEVOPT	0x22	R/W	DDR-SDRAM Device Options
SDRAM0_MCSTS	0x24	R	DDR-SDRAM Memory Controller Status
SDRAM0_RTR	0x30	R/W	DDR-SDRAM Refresh Timer Register
SDRAM0_PMIT	0x34	R/W	DDR-SDRAM Power Management Idle Timer
SDRAM0_UABBA	0x38	R/W	DDR-SDRAM PLB UA Bus Base Address
SDRAM0_B0CR	0x40	R/W	DDR-SDRAM Bank 0 Configuration Register
SDRAM0_B1CR	0x44	R/W	DDR-SDRAM Bank 1 Configuration Register
SDRAM0_B2CR	0x48	R/W	DDR-SDRAM Bank 2 Configuration Register
SDRAM0_B3CR	0x4C	R/W	DDR-SDRAM Bank 3 Configuration Register
SDRAM0_TR0	0x80	R/W	DDR-SDRAM Timing Register 0
SDRAM0_TR1	0x81	R/W	DDR-SDRAM Timing Register 1
SDRAM0_CLKTR	0x82	R/W	DDR-SDRAM Clock Timing Register
SDRAM0_WDDCTR	0x83	R/W	DDR-SDRAM Write Data, DQS, DM Clock Timing Register
SDRAM0_DLYCAL	0x84	R/W	DDR-SDRAM Delay Line Calibration Register
SDRAM0_ECCESR	0x98	R/W	DDR-SDRAM ECC Error Status Register
SDRAM0_CID	0xA4	R	DDR-SDRAM Core ID Register
SDRAM0_RID	0xA8	R	DDR-SDRAM Revision ID Register

Indirect Access of External Bus Controller DCRs

The following procedure accesses the EBCO DCRs listed in Table 29-6.

1. Write the offset from Table 29-7 to the EBCO Address Register (EBC0_CFGADDR).
2. Read data from or write data to the EBCO Data Register (EBC0_CFGDATA).

Table 29-6. External Bus Controller DCR Usage

Register	DCR Number	Access	Description
EBC0_CFGADDR	0x012	R/W	EBCO Controller Address Register
EBC0_CFGDATA	0x013	R/W	EBCO Controller Data Register

Indirect Access of External Bus Controller DCRs

Table 29-7. Offsets for External Bus Controller Registers

Register	Offset	Access	Description
EBC0_B0CR	0x00	R/W	EBCO Bank 0 Configuration Register
EBC0_B1CR	0x01	R/W	EBCO Bank 1 Configuration Register
EBC0_B2CR	0x02	R/W	EBCO Bank 2 Configuration Register
EBC0_B3CR	0x03	R/W	EBCO Bank 3 Configuration Register
EBC0_B4CR	0x04	R/W	EBCO Bank 4 Configuration Register
EBC0_B5CR	0x05	R/W	EBCO Bank 5 Configuration Register
EBC0_B6CR	0x06	R/W	EBCO Bank 6 Configuration Register
EBC0_B7CR	0x07	R/W	EBCO Bank 7 Configuration Register
EBC0_B0AP	0x10	R/W	EBCO Bank 0 Access Parameters
EBC0_B1AP	0x11	R/W	EBCO Bank 1 Access Parameters
EBC0_B2AP	0x12	R/W	EBCO Bank 2 Access Parameters
EBC0_B3AP	0x13	R/W	EBCO Bank 3 Access Parameters
EBC0_B4AP	0x14	R/W	EBCO Bank 4 Access Parameters
EBC0_B5AP	0x15	R/W	EBCO Bank 5 Access Parameters
EBC0_B6AP	0x16	R/W	EBCO Bank 6 Access Parameters
EBC0_B7AP	0x17	R/W	EBCO Bank 7 Access Parameters
EBC0_BEAR	0x20	R/W	EBCO Bus Error Address Register
EBC0_BESR	0x21	R/W	EBCO Bus Error Status Register 0
EBC0_CFG	0x23	R/W	EBCO Peripheral Control Register
EBC0_CID	0x24	R/W	EBCO Peripheral Core ID Register

Indirect Access of External Bus Master DCRs

The following procedure accesses the EBMI DCRs listed in Table 29-8.

1. Write the offset from Table 29-9 to the EBMI Address Register (EBM0_CFGADDR).
2. Read data from or write data to the EBMI Data Register (EBM0_CFGDATA).

Table 29-8. External Bus Master DCR Usage

Register	DCR Number	Access	Description
EBM0_CFGADDR	0x014	R/W	EBMI Controller Address Register
EBM0_CFGDATA	0x015	R/W	EBMI Controller Data Register

Table 29-9. Offsets for External Bus Master Registers

Register	Offset	Access	Description
EBM0_CTL	0x00	R/W	EBMI Control Register
EBM0_LCNT	0x01	R/W	EBMI OPB Latency Count Register
EBM0_BEAR	0x02	R/W	EBMI Bus Error Address Register
EBM0_BESR	0x03	R/W	EBMI Bus Error Status Register
EBM0_BEMR	0x04	R/W	EBMI Bus Error Mask Register
EBM0_UAR	0x05	R/W	EBMI OPB Upper Address Register
EBM0_UAM	0x06	R/W	EBMI OPB Upper Address Mask
EBM0_SLPMD	0x07	R/W	EBMI Sleep Mode Register
EBM0_FAIR	0x08	R/W	EBMI Fairness Control Register
EBM0_CID	0x11	R	EBMI Core ID Register

Indirect Access of PLB Performance Monitor DCRs

The following procedure accesses the PPM DCRs listed in Table 29-10.

1. Write the offset from Table 29-11 to the PPM Address Register (PPM0_CFGADDR).
2. Read data from or write data to the PPM Data Register (PPM0_CFGDATA).

Table 29-10. PLB Performance Monitor DCR Usage

Register	DCR Number	Access	Description
PPM0_CFGADDR	0x016	R/W	PPM Address Register
PPM0_CFGDATA	0x017	R/W	PPM Data Register

Table 29-11. Offsets for PLB Performance Monitor Registers

Register	Offset	Access	Description
PPM0_ISR	0x0	Read	Interrupt Status Register
PPM0_CR	0x2	R/W	Control Register
PPM0_CCR	0x3	R/W	Cycle Control Register
PPM0_UAR	0x4	R/W	Upper Address Register
PPM0_LAR	0x5	R/W	Lower Address Register
PPM0_UAMR	0x6	R/W	Upper Address Mask Register
PPM0_LAMR	0x7	R/W	Lower Address Mask Register
PPM0_RIDR	0x8	R	Revision ID Register
PPM0_MCSR0	0x9	R/W	Master Event Counter Selection Register 0
PPM0_MCSR1	0xA	R/W	Master Event Counter Selection Register 1
PPM0_MCSR2	0xB	R/W	Master Event Counter Selection Register 2
PPM0_MCSR3	0xC	R/W	Master Event Counter Selection Register 3
PPM0_SCSR0	0x11	R/W	Slave Event Counter Selection Register 0

Table 29-11. Offsets for PLB Performance Monitor Registers

Register	Offset	Access	Description
PPM0_SCSR1	0x12	R/W	Slave Event Counter Selection Register 1
PPM0_SCSR2	0x13	R/W	Slave Event Counter Selection Register 2
PPM0_SCSR3	0x14	R/W	Slave Event Counter Selection Register 3
PPM0_GCSR0	0x19	R/W	Generic Event Counter Selection Register 0
PPM0_GCSR1	0x1A	R/W	Generic Event Counter Selection Register 1
PPM0_GCSR2	0x1B	R/W	Generic Event Counter Selection Register 2
PPM0_GCSR3	0x1C	R/W	Generic Event Counter Selection Register 3
PPM0_MCR0	0x1D	R/W	Master Event Counter Register 0
PPM0_MCR1	0x1E	R/W	Master Event Counter Register 1
PPM0_MCR2	0x1F	R/W	Master Event Counter Register 2
PPM0_MCR3	0x20	R/W	Master Event Counter Register 3
PPM0_SCR0	0x25	R/W	Slave Event Counter Register 0
PPM0_SCR1	0x26	R/W	Slave Event Counter Register 1
PPM0_SCR2	0x27	R/W	Slave Event Counter Register 2
PPM0_SCR3	0x28	R/W	Slave Event Counter Register 3
PPM0_GCR0	0x2D	R/W	Generic Pipeline Event Counter Register 0
PPM0_GCR1	0x2E	R/W	Generic Pipeline Event Counter Register 1
PPM0_GCR2	0x2F	R/W	Generic Pipeline Event Counter Register 2
PPM0_GCR3	0x30	R/W	Generic Pipeline Event Counter Register 3
PPM0_DCSR0	0x31	R/W	Duration Counter Selection Register 0
PPM0_DCSR1	0x32	R/W	Duration Counter Selection Register 1
PPM0_DCMXR0	0x33	R/W	Duration Counter Max Register 0
PPM0_DCMXR1	0x34	R/W	Duration Counter Max Register 1
PPM0_DCMNR0	0x35	R/W	Duration Counter Min Register 0
PPM0_DCMNR1	0x36	R/W	Duration Counter Minimum Register 1
PPM0_DCTVR0	0x37	R/W	Duration Counter Total Value Register 0
PPM0_DCTVR1	0x38	R/W	Duration Counter Total Value Register 1
PPM0_DCOTR0	0x39	R/W	Duration Counter Occurrence Total Register 0
PPM0_DCOTR1	0x3A	R/W	Duration Counter Occurrence Total Register 1

29.4 Memory-Mapped Input/Output Registers

Some registers associated with on-chip peripherals are memory-mapped input/output (MMIO) registers. Such registers are mapped into the system memory space and are accessed using load/store instructions that contain the register addresses. The tables that follow list all MMIO registers.

Table 0-12. MMIO Registers

Register	Address	Access	Description
Serial Port 0			
UART0_RBR	0x1 40000200	R	UART 0 Receiver Buffer Register **Note:** Set UART0_LCR[DLAB] = 0 to access.
UART0_THR		W	UART 0 Transmitter Holding Register **Note:** Set UART0_LCR[DLAB] = 0 to access.
UART0_DLL		R/W	UART 0 Baud-rate Divisor Latch LSB **Note:** Set UART0_LCR[DLAB] = 1 to access.
UART0_IER	0x1 40000201	R/W	UART 0 Interrupt Enable Register **Note:** Set UART0_LCR[DLAB] = 0 to access.
UART0_DLM		R/W	UART 0 Baud-rate Divisor Latch MSB **Note:** Set UART0_LCR[DLAB] = 1 to access.
UART0_IIR	0x1 40000202	R	UART 0 Interrupt Identification Register
UART0_FCR	0x1 40000202	W	UART 0 FIFO Control Register
UART0_LCR	0x1 40000203	R/W	UART 0 Line Control Register
UART0_MCR	0x1 40000204	R/W	UART 0 Modem Control Register
UART0_LSR	0x1 40000205	R/W	UART 0 Line Status Register
UART0_MSR	0x1 40000206	R/W	UART 0 Modem Status Register
UART0_SCR	0x1 40000207	R/W	UART 0 Scratch Register
Serial Port 1			
UART1_RBR	0x1 40000300	R	UART 1 Receiver Buffer Register **Note:** Set UART1_LCR[DLAB] = 0 to access.
UART1_THR		W	UART 1 Transmitter Holding Register **Note:** Set UART1_LCR[DLAB] = 0 to access.
UART1_DLL		R/W	UART 1 Baud-rate Divisor Latch LSB **Note:** Set UART1_LCR[DLAB] = 1 to access.
UART1_IER	0x1 40000301	R/W	UART 1 Interrupt Enable Register **Note:** Set UART1_LCR[DLAB] = 0 to access.
UART1_DLM		R/W	UART 1 Baud-rate Divisor Latch MSB **Note:** Set UART1_LCR[DLAB] = 1 to access.
UART1_IIR	0x1 40000302	R	UART 1 Interrupt Identification Register
UART1_FCR	0x1 40000302	W	UART 1 FIFO Control Register
UART1_LCR	0x1 40000303	R/W	UART 1 Line Control Register
UART1_MCR	0x1 40000304	R/W	UART 1 Modem Control Register
UART1_LSR	0x1 40000305	R/W	UART 1 Line Status Register

Table 0-12. MMIO Registers

Register	Address	Access	Description
UART1_MSR	0x1 40000306	R/W	UART 1 Modem Status Register
UART1_SCR	0x1 40000307	R/W	UART 1 Scratch Register
Inter-Integrated Circuit 0			
IIC0_MDBUF	0x1 40000400	R/W	IIC 0 Master Data Buffer
IIC0_SDBUF	0x1 40000402	R/W	IIC 0 Slave Data Buffer
IIC0_LMADR	0x1 40000404	R/W	IIC 0 Low Master Address
IIC0_HMADR	0x1 40000405	R/W	IIC 0 High Master Address
IIC0_CNTL	0x1 40000406	R/W	IIC 0 Control
IIC0_MDCNTL	0x1 40000407	R/W	IIC 0 Mode Control
IIC0_STS	0x1 40000408	R/W	IIC 0 Status
IIC0_EXTSTS	0x1 40000409	R/W	IIC 0 Extended Status
IIC0_LSADR	0x1 4000040A	R/W	IIC 0 Low Slave Address
IIC0_HSADR	0x1 4000040B	R/W	IIC 0 High Slave Address
IIC0_CLKDIV	0x1 4000040C	R/W	IIC 0 Clock Divide
IIC0_INTRMSK	0x1 4000040D	R/W	IIC 0 Interrupt Mask
IIC0_XFRCNT	0x1 4000040E	R/W	IIC 0 Transfer Count
IIC0_XTCNTLSS	0x1 4000040F	R/W	IIC 0 Extended Control and Slave Status
IIC0_DIRECTCNTL	0x1 40000410	R/W	IIC 0 Direct Control
Inter-Integrated Circuit 1			
IIC1_MDBUF	0x1 40000500	R/W	IIC 1 Master Data Buffer
IIC1_SDBUF	0x1 40000502	R/W	IIC 1 Slave Data Buffer
IIC1_LMADR	0x1 40000504	R/W	IIC 1 Low Master Address
IIC1_HMADR	0x1 40000505	R/W	IIC 1 High Master Address
IIC1_CNTL	0x1 40000506	R/W	IIC 1 Control
IIC1_MDCNTL	0x1 40000507	R/W	IIC 1 Mode Control
IIC1_STS	0x1 40000508	R/W	IIC 1 Status
IIC1_EXTSTS	0x1 40000509	R/W	IIC 1 Extended Status
IIC1_LSADR	0x1 4000050A	R/W	IIC 1 Low Slave Address
IIC1_HSADR	0x1 4000050B	R/W	IIC 1 High Slave Address
IIC1_CLKDIV	0x1 4000050C	R/W	IIC 1 Clock Divide
IIC1_INTRMSK	0x1 4000050D	R/W	IIC 1 Interrupt Mask
IIC1_XFRCNT	0x1 4000050E	R/W	IIC 1 Transfer Count
IIC1_XTCNTLSS	0x1 4000050F	R/W	IIC 1 Extended Control and Slave Status
IIC1_DIRECTCNTL	0x1 40000510	R/W	IIC 1 Direct Control
OPB Arbiter			
OPBA0_PR	0x1 40000600	R/W	OPB Arbiter Priority Register
OPBA0_CR	0x1 40000601	R/W	OPB Arbiter Control Register

Table 0-12. MMIO Registers

Register	Address	Access	Description
General-Purpose I/O			
GPIO0_OR	0x1 40000700	R/W	GPIO Output Register
GPIO0_TCR	0x1 40000704	R/W	GPIO Three-State Control Register
GPIO0_ODR	0x1 40000718	R/W	GPIO Open Drain Register
GPIO0_IR	0x1 4000071C	R	GPIO Input Register
ZMII			
ZMII0_FER	0x1 40000780	R/W	ZMII Function Enable Register
ZMII0_SSR	0x1 40000784	R/W	ZMII Speed Select Register
ZMII0_SMIISR	0x1 40000788	R/W	ZMII SMII Status Register
Ethernet MAC 0			
EMAC0_MR0	0x1 40000800	R/W	EMAC 0 Mode Register 0
EMAC0_MR1	0x1 40000804	R/W	EMAC 0 Mode Register 1
EMAC0_TMR0	0x1 40000808	R/W	EMAC 0 Transmit Mode Register 0
EMAC0_TMR1	0x1 4000080C	R/W	EMAC 0 Transmit Mode Register 1
EMAC0_RMR	0x1 40000810	R/W	EMAC 0 Receive Mode Register
EMAC0_ISR	0x1 40000814	R/W	EMAC 0 Interrupt Status Register
EMAC0_ISER	0x1 40000818	R/W	EMAC 0 Interrupt Status Enable Register
EMAC0_IAHR	0x1 4000081C	R/W	EMAC 0 Individual Address High
EMAC0_IALR	0x1 40000820	R/W	EMAC 0 Individual Address Low
EMAC0_VTPID	0x1 40000824	R/W	EMAC 0 VLAN TPID Register
EMAC0_VTCI	0x1 40000828	R/W	EMAC 0 VLAN TCI Register
EMAC0_PTR	0x1 4000082C	R/W	EMAC 0 Pause Timer Register
EMAC0_IAHT1	0x1 40000830	R/W	EMAC 0 Individual Address Hash Table 1
EMAC0_IAHT2	0x1 40000834	R/W	EMAC 0 Individual Address Hash Table 2
EMAC0_IAHT3	0x1 40000838	R/W	EMAC 0 Individual Address Hash Table 3
EMAC0_IAHT4	0x1 4000083C	R/W	EMAC 0 Individual Address Hash Table 4
EMAC0_GAHT1	0x1 40000840	R/W	EMAC 0 Group Address Hash Table 1
EMAC0_GAHT2	0x1 40000844	R/W	EMAC 0 Group Address Hash Table 2
EMAC0_GAHT3	0x1 40000848	R/W	EMAC 0 Group Address Hash Table 3
EMAC0_GAHT4	0x1 4000084C	R/W	EMAC 0 Group Address Hash Table 4
EMAC0_LSAH	0x1 40000850	R	EMAC 0 Last Source Address Low
EMAC0_LSAL	0x1 40000854	R	EMAC 0 Last Source Address High
EMAC0_IPGVR	0x1 40000858	R/W	EMAC 0 Inter-Packet Gap Value Register
EMAC0_STACR	0x1 4000085C	R/W	EMAC 0 STA Control Register
EMAC0_TRTR	0x1 40000860	R/W	EMAC 0 Transmit Request Threshold Register
EMAC0_RWMR	0x1 40000864	R/W	EMAC 0 Receive Low/High Water Mark Register
EMAC0_OCTX	0x1 40000868	R	EMAC 0 Number of Octets Transimitted
EMAC0_OCRX	0x1 4000086C	R	EMAC 0 Number of Octets Received
Ethernet MAC 1			
EMAC1_MR0	0x1 40000900	R/W	EMAC 1 Mode Register 0

Table 0-12. MMIO Registers

Register	Address	Access	Description
EMAC1_MR1	0x1 40000904	R/W	EMAC 1 Mode Register 1
EMAC1_TMR0	0x1 40000908	R/W	EMAC 1 Transmit Mode Register 0
EMAC1_TMR1	0x1 4000090C	R/W	EMAC 1 Transmit Mode Register 1
EMAC1_RMR	0x1 40000910	R/W	EMAC 1 Receive Mode Register
EMAC1_ISR	0x1 40000914	R/W	EMAC 1 Interrupt Status Register
EMAC1_ISER	0x1 40000918	R/W	EMAC 1 Interrupt Status Enable Register
EMAC1_IAHR	0x1 4000091C	R/W	EMAC 1 Individual Address High
EMAC1_IALR	0x1 40000920	R/W	EMAC 1 Individual Address Low
EMAC1_VTPID	0x1 40000924	R/W	EMAC 1 VLAN TPID Register
EMAC1_VTCI	0x1 40000928	R/W	EMAC 1 VLAN TCI Register
EMAC1_PTR	0x1 4000092C	R/W	EMAC 1 Pause Timer Register
EMAC1_IAHT1	0x1 40000930	R/W	EMAC 1 Individual Address Hash Table 1
EMAC1_IAHT2	0x1 40000934	R/W	EMAC 1 Individual Address Hash Table 2
EMAC1_IAHT3	0x1 40000938	R/W	EMAC 1 Individual Address Hash Table 3
EMAC1_IAHT4	0x1 4000093C	R/W	EMAC 1 Individual Address Hash Table 4
EMAC1_GAHT1	0x1 40000940	R/W	EMAC 1 Group Address Hash Table 1
EMAC1_GAHT2	0x1 40000944	R/W	EMAC 1 Group Address Hash Table 2
EMAC1_GAHT3	0x1 40000948	R/W	EMAC 1 Group Address Hash Table 3
EMAC1_GAHT4	0x1 4000094C	R/W	EMAC 1 Group Address Hash Table 4
EMAC1_LSAH	0x1 40000950	R	EMAC 1 Last Source Address Low
EMAC1_LSAL	0x1 40000954	R	EMAC 1 Last Source Address High
EMAC1_IPGVR	0x1 40000958	R/W	EMAC 1 Inter-Packet Gap Value Register
EMAC1_STACR	0x1 4000095C	R/W	EMAC 1 STA Control Register
EMAC1_TRTR	0x1 40000960	R/W	EMAC 1 Transmit Request Threshold Register
EMAC1_RWMR	0x1 40000964	R/W	EMAC 1 Receive Low/High Water Mark Register
EMAC1_OCTX	0x1 40000968	R	EMAC 1 Number of Octets Transimitted
EMAC1_OCRX	0x1 4000096C	R	EMAC 1 Number of Octets Received
General Purpose Timer			
GPT0_TBC	0x1 40000A00	R/W	GPT Time Base Counter
GPT0_OE	0x1 40000A10	R/W	GPT Output Enable
GPT0_OL	0x1 40000A14	R/W	GPT Output Level
GPT0_IM	0x1 40000A18	R/W	GPT Interrupt Mask
GPT0_ISS	0x1 40000A1C	R/W	GPT Interrupt Status (Set bits if write 1)
GPT0_ISC	0x1 40000A20	R/W	GPT Interrupt Status (Clear bits if write 1)
GPT0_IE	0x1 40000A24	R/W	GPT Interrupt Enable
GPT0_COMP0	0x1 40000A80	R/W	GPT Compare Timer 0
GPT0_COMP1	0x1 40000A84	R/W	GPT Compare Timer 1
GPT0_COMP2	0x1 40000A88	R/W	GPT Compare Timer 2
GPT0_COMP3	0x1 40000A8C	R/W	GPT Compare Timer 3
GPT0_COMP4	0x1 40000A90	R/W	GPT Compare Timer 4

Table 0-12. MMIO Registers

Register	Address	Access	Description
GPT0_MASK0	0x1 40000AC0	R/W	GPT Compare Mask 0
GPT0_MASK1	0x1 40000AC4	R/W	GPT Compare Mask 1
GPT0_MASK2	0x1 40000AC8	R/W	GPT Compare Mask 2
GPT0_MASK3	0x1 40000ACC	R/W	GPT Compare Mask 3
GPT0_MASK4	0x1 40000AD0	R/W	GPT Compare Mask 4

~~The two registers listed in Table 29-13 are used to access the configuration registers of external PCI-X devices only.~~

Table 0-13. External PCI-X Device Configuration Register Access

Register	Address	Access	Description
PCIX0_CFGADDR	0x2 0EC00000	R/W	PCI-X Configuration Address Register
PCIX0_CFGDATA	0x2 0EC00004	R/W	PCI-X Configuration Data Register

~~Registers listed in Table 29-14 can be accessed in two ways:~~

~~1. By the processor using the PLB address~~

~~2. By an external PCI-X device using configuration cycles to the proper offset.~~

Table 0-14. Internal PCI-X Configuration Registers

Register	PLB		PCI-X Configuration		Description
	Address	Access	Offset	Access	
PCIX0_VENDID	0x2 0EC80000	R/W	0x01–0x00	R	PCI-X Vendor ID
PCIX0_DEVID	0x2 0EC80002	R/W	0x03–0x02	R	PCI-X Device ID
PCIX0_CMD	0x2 0EC80004	R/W	0x05–0x04	R/W	PCI-X Command Register
PCIX0_STATUS	0x2 0EC80006	R/W	0x07–0x06	R/W	PCI-X Status Register
PCIX0_REVID	0x2 0EC80008	R/W	0x08	R	PCI-X Revision ID
PCIX0_CLS	0x2 0EC80009	R/W	0x0B–0x09	R	PCI-X Class Register
PCIX0_CACHELS	0x2 0EC8000C	R/W	0x0C	R/W	PCI-X Cache Line Size
PCIX0_LATTIM	0x2 0EC8000D	R/W	0x0D	R/W	PCI-X Latency Timer
PCIX0_HDTYPE	0x2 0EC8000E	R	0x0E	R	PCI-X Header Type
PCIX0_BIST	0x2 0EC8000F	R	0x0F	R	PCI-X Built In Self Test Control
PCIX0_ BAR0L	0x2 0EC80010	R/W	0x13–0x10	R/W	PCI-X BAR 0 Low
PCIX0_ BAR0H	0x2 0EC80014	R/W	0x17–0x14	R/W	PCI-X BAR 0 High
PCIX0_BAR1	0x2 0EC80018	R/W	0x1B–0x18	R/W	PCI-X BAR 1
PCIX0_ BAR2L	0x2 0EC8001C	R/W	0x1F–0x1C	R/W	PCI-X BAR 2 Low
PCIX0_ BAR2H	0x2 0EC80020	R/W	0x23–0x20	R/W	PCI-X BAR 2 High
PCIX0_ BAR3	0x2 0EC80024	R	0x27–0x24	R	Unused BAR 3
PCIX0_CISPTR	0x2 0EC80028	R	0x2B–0x28	R	Unused Cardbus CIS Pointer
PCIX0_SBSYSVID	0x2 0EC8002C	R/W	0x2D–0x2C	R	PCI-X Subsystem Vendor ID
PCIX0_SBSYSID	0x2 0EC8002E	R/W	0x2F–0x2E	R	PCI-X Subsystem ID

Table 0-14. Internal PCI-X Configuration Registers

Register	PLB		PCI-X Configuration		Description
	Address	Access	Offset	Access	
PCIX0_EROMBA	0x2 0EC80030	R/W	0x33–0x30	R/W	PCI-X Expansion ROM Base Address
PCIX0_CAP	0x2 0EC80034	R	0x34	R	PCI-X Capabilities Pointer
PCIX0_RES0	0x2 0EC80035	R	0x35	R	Reserved
PCIX0_RES1	0x2 0EC80036	R	0x37–0x6	R	Reserved
PCIX0_RES2	0x2 0EC80038	R	0x3B–0x8	R	Reserved
PCIX0_INTLN	0x2 0EC8003C	R/W	0x3C	R/W	PCI-X Interrupt Line
PCIX0_INTPN	0x2 0EC8003D	R	0x3D	R	PCI-X Interrupt Pin
PCIX0_MINGNT	0x2 0EC8003E	R	0x3E	R	PCI-X Minimum Grant
PCIX0_MAXLTNCY	0x2 0EC8003F	R	0x3F	R	PCI-X Maximum Latency
PCIX0_BRDGOPT1	0x2 0EC80040	R/W	0x43–0x40	R/W	PCI-X Bridge Options 1
PCIX0_BRDGOPT2	0x2 0EC80044	R/W	0x47–0x44	R/W	PCI-X Bridge Options 2
PCIX0_ERREN	0x2 0EC80050	R/W	0x53–0x50	R/W	PCI-X Error Enable
PCIX0_ERRSTS	0x2 0EC80054	R/W	0x57–0x54	R/W	PCI-X Error Status
PCIX0_PLBBESR	0x2 0EC80058	R	0x5B–0x58	R	PCI-X PLB Slave Error Syndrome Register
PCIX0_PLBBEARL	0x2 0EC8005C	R	0x5F–0x5C	R	PCI-X PLB Slave Error Address Register Low
PCIX0_PLBBEARH	0x2 0EC80060	R	0x63–0x60	R	PCI-X PLB Slave Error Address Register High
PCIX0_POM0LAL	0x2 0EC80068	R/W	0x6B–0x68	R/W	PCI-X POM0 Local Address Low
PCIX0_POM0LAH	0x2 0EC8006C	R/W	0x6F–0x6C	R/W	PCI-X POM0 Local Address High
PCIX0_POM0SA	0x2 0EC80070	R/W	0x73–0x70	R/W	PCI-X POM0 Size Attribute
PCIX0_POM0PCIAL	0x2 0EC80074	R/W	0x77–0x74	R/W	PCI-X POM0 PCI Address Low
PCIX0_POM0PCIAH	0x2 0EC80078	R/W	0x7B–0x78	R/W	PCI-X POM0 PCI Address High
PCIX0_POM1LAL	0x2 0EC8007C	R/W	0x7F–0x7C	R/W	PCI-X POM1 Local Address Low
PCIX0_POM1LAH	0x2 0EC80080	R/W	0x83–0x80	R/W	PCI-X POM1 Local Address High
PCIX0_POM1SA	0x2 0EC80084	R/W	0x87–0x84	R/W	PCI-X POM1 Size Attribute
PCIX0_POM1PCIAL	0x2 0EC80088	R/W	0x8B–0x88	R/W	PCI-X POM1 PCI Address Low
PCIX0_POM1PCIAH	0x2 0EC8008C	R/W	0x8F–0x8C	R/W	PCI-X POM1 PCI Address High
PCIX0_POM2SA	0x2 0EC80090	R/W	0x93–0x90	R/W	PCI-X POM2 Size Attribute
PCIX0_PIM0SA	0x2 0EC80098	R/W	0x9B–0x98	R/W	PCI-X PIM0 Size/Attribute
PCIX0_PIM0LAL	0x2 0EC8009C	R/W	0x9F–0x9C	R/W	PCI-X PIM0 Local Address Low
PCIX0_PIM0LAH	0x2 0EC800A0	R/W	0xA3–0xA0	R/W	PCI-X PIM0 Local Address High
PCIX0_PIM1SA	0x2 0EC800A4	R/W	0xA7–0xA4	R/W	PCI-X PIM1 Size/Attribute
PCIX0_PIM1LAL	0x2 0EC800A8	R/W	0xAB–0xA8	R/W	PCI-X PIM1 Local Address Low
PCIX0_PIM1LAH	0x2 0EC800AC	R/W	0xAF–0xAC	R/W	PCI-X PIM1 Local Address High

Table 0-14. Internal PCI-X Configuration Registers

Register	PLB		PCI-X Configuration		Description
	Address	Access	Offset	Access	
PCIX0_PIM2SA	0x2 0EC800B0	R/W	0xB3–0xB0	R/W	PCI-X PIM2 Size/Attribute
PCIX0_PIM2LAL	0x2 0EC800B4	R/W	0xB7–0xB4	R/W	PCI-X PIM2 Local Address Low
PCIX0_PIM2LAH	0x2 0EC800B8	R/W	0xBB–0xB8	R/W	PCI-X PIM2 Local Address High
PCIX0_OMCAPID	0x2 0EC800C0	R	0xC0	R	PCI-X Outbound MSI Capability Identifier
PCIX0_OMNIPTR	0x2 0EC800C1	R/W	0xC1	R	PCI-X Outbound MSI Next Item Pointer
PCIX0_OMMC	0x2 0EC800C2	R/W	0xC3–0xC2	R/W	PCI-X Outbound MSI Message Control
PCIX0_OMMA	0x2 0EC800C4	R/W	0xC7–0xC4	R/W	PCI-X Outbound MSI Message Address
PCIX0_OMMUA	0x2 0EC800C8	R/W	0xCB–0xC8	R/W	PCI-X Outbound MSI Message Upper Address
PCIX0_OMMDATA	0x2 0EC800CC	R/W	0xCD–0xCC	R/W	PCI-X Outbound MSI Message Data
PCIX0_OMMEOI	0x2 0EC800CE	R/W	0xCE	R/W	PCI-X Outbound MSI Message End Of Interrupt
PCIX0_PMCAPID	0x2 0EC800D0	R/W	0xD0	R	PCI-X PMC Capability Identifier
PCIX0_PMNIPTR	0x2 0EC800D1	R/W	0xD1	R	PCI-X PMC Next Item Pointer
PCIX0_PMC	0x2 0EC800D2	R	0xD3–0xD2	R	PCI-X Power Management Capabilities
PCIX0_PMCSR	0x2 0EC800D4	R/W	0xD5–0xD4	R/W	PCI-X Power Management Control Status
PCIX0_PMCSRBSE	0x2 0EC800D6	R	0xD6	R	PCI-X PMCSR PCI to PCI Bridge Support Extensions
PCIX0_PMDATA	0x2 0EC800D7	R	0xD7	R	PCI-X PMC Unused Data Register
PCIX0_PMSCRR	0x2 0EC800D8	R/W	0xD8	R/W	PCI-X Power Management State Change Request Register
PCIX0_PCIXCAPID	0x2 0EC800DC	R	0xDC	R	PCI-X Capability Identifier
PCIX0_PCIXNIPTR	0x2 0EC800DD	R	0xDD	R	PCI-X Next Item Pointer
PCIX0_PCIXCMD	0x2 0EC800DE	R/W	0xDF–0xDE	R/W	PCI-X Command
PCIX0_PCIXSTS	0x2 0EC800E0	R/W	0xE3–0xE0	R/W	PCI-X Status
PCIX0_PCIXIDR	0x2 0EC800E4	R/W	0xE7–0xE4	R/W	PCI-X Internal Debug Register
PCIX0_PCIXCID	0x2 0EC800E8	R	0xEB–0xE8	R	PCI-X Internal Core Device ID
PCIX0_PCIXRID	0x2 0EC800EC	R	0xEF–0xEC	R	PCI-X Internal Core Revision ID

~~Registers listed in Table 29-15 can be accessed in two ways:~~

~~1. By the processor using the PLB address~~

~~2. By an external PCI-X device using memory cycles to the proper address.~~

Table 0-15. PCI-X SImple Message Passing and Inbound MSI Registers

Register	PLB		PCI-X Memory		Description
	Address	Access	Address	Access	
PCIX0_MSGIL	0x2 0EC80100	R/W	(PCIX0_BAR0H || PCIX0_BAR0L) + (0x03–0x00)	R/W	PCI-X Message In Low
PCIX0_MSGIH	0x2 0EC80104	R/W	(PCIX0_BAR0H || PCIX0_BAR0L) + (0x07–0x04)	R/W	PCI-X Message In High
PCIX0_MSGOL	0x2 0EC80108	R/W	(PCIX0_BAR0H || PCIX0_BAR0L) + (0x0B–0x08)	R/W	PCI-X Message Out Low
PCIX0_MSGOH	0x2 0EC8010C	R/W	(PCIX0_BAR0H || PCIX0_BAR0L) + (0x0F–0x0C)	R/W	PCI-X Message Out High
PCIX0_IM	0x2 0EC801F8	W	(PCIX0_BAR0H || PCIX0_BAR0L) + (0xFB–0xF8)	W	PCI-X Inbound MSI

Some registers associated with on-chip peripherals are memory-mapped input/output (MMIO) registers. Such registers are mapped into the system memory space and are accessed using load/store instructions that contain the register addresses. The tables that follow list all MMIO registers.

Table 29-12. MMIO Registers

Register	Address	Access	Description
Serial Port 0			
UART0_RBR	0x1 4000 0200	R	UART 0 Receiver Buffer Register **Note:** Set UART0_LCR[DLAB] = 0 to access.
UART0_THR		W	UART 0 Transmitter Holding Register **Note:** Set UART0_LCR[DLAB] = 0 to access.
UART0_DLL		R/W	UART 0 Baud-rate Divisor Latch LSB **Note:** Set UART0_LCR[DLAB] = 1 to access.
UART0_IER	0x1 4000 0201	R/W	UART 0 Interrupt Enable Register **Note:** Set UART0_LCR[DLAB] = 0 to access.
UART0_DLM		R/W	UART 0 Baud-rate Divisor Latch MSB **Note:** Set UART0_LCR[DLAB] = 1 to access.
UART0_IIR	0x1 4000 0202	R	UART 0 Interrupt Identification Register
UART0_FCR	0x1 4000 0202	W	UART 0 FIFO Control Register
UART0_LCR	0x1 4000 0203	R/W	UART 0 Line Control Register
UART0_MCR	0x1 4000 0204	R/W	UART 0 Modem Control Register
UART0_LSR	0x1 4000 0205	R/W	UART 0 Line Status Register
UART0_MSR	0x1 4000 0206	R/W	UART 0 Modem Status Register
UART0_SCR	0x1 4000 0207	R/W	UART 0 Scratch Register

Table 29-12. MMIO Registers

Register	Address	Access	Description
Serial Port 1			
UART1_RBR	0x1 4000 0300	R	UART 1 Receiver Buffer Register **Note:** Set UART1_LCR[DLAB] = 0 to access.
UART1_THR		W	UART 1 Transmitter Holding Register **Note:** Set UART1_LCR[DLAB] = 0 to access.
UART1_DLL		R/W	UART 1 Baud-rate Divisor Latch LSB **Note:** Set UART1_LCR[DLAB] = 1 to access.
UART1_IER	0x1 4000 0301	R/W	UART 1 Interrupt Enable Register **Note:** Set UART1_LCR[DLAB] = 0 to access.
UART1_DLM		R/W	UART 1 Baud-rate Divisor Latch MSB **Note:** Set UART1_LCR[DLAB] = 1 to access.
UART1_IIR	0x1 4000 0302	R	UART 1 Interrupt Identification Register
UART1_FCR	0x1 4000 0302	W	UART 1 FIFO Control Register
UART1_LCR	0x1 4000 0303	R/W	UART 1 Line Control Register
UART1_MCR	0x1 4000 0304	R/W	UART 1 Modem Control Register
UART1_LSR	0x1 4000 0305	R/W	UART 1 Line Status Register
UART1_MSR	0x1 4000 0306	R/W	UART 1 Modem Status Register
UART1_SCR	0x1 4000 0307	R/W	UART 1 Scratch Register
Inter-Integrated Circuit 0			
IIC0_MDBUF	0x1 40000400	R/W	IIC 0 Master Data Buffer
IIC0_SDBUF	0x1 40000402	R/W	IIC 0 Slave Data Buffer
IIC0_LMADR	0x1 40000404	R/W	IIC 0 Low Master Address
IIC0_HMADR	0x1 40000405	R/W	IIC 0 High Master Address
IIC0_CNTL	0x1 40000406	R/W	IIC 0 Control
IIC0_MDCNTL	0x1 40000407	R/W	IIC 0 Mode Control
IIC0_STS	0x1 40000408	R/W	IIC 0 Status
IIC0_EXTSTS	0x1 40000409	R/W	IIC 0 Extended Status
IIC0_LSADR	0x1 4000040A	R/W	IIC 0 Low Slave Address
IIC0_HSADR	0x1 4000040B	R/W	IIC 0 High Slave Address
IIC0_CLKDIV	0x1 4000040C	R/W	IIC 0 Clock Divide
IIC0_INTRMSK	0x1 4000040D	R/W	IIC 0 Interrupt Mask
IIC0_XFRCNT	0x1 4000040E	R/W	IIC 0 Transfer Count
IIC0_XTCNTLSS	0x1 4000040F	R/W	IIC 0 Extended Control and Slave Status
IIC0_DIRECTCNTL	0x1 40000410	R/W	IIC 0 Direct Control
Inter-Integrated Circuit 1			
IIC1_MDBUF	0x1 40000500	R/W	IIC 1 Master Data Buffer
IIC1_SDBUF	0x1 40000502	R/W	IIC 1 Slave Data Buffer
IIC1_LMADR	0x1 40000504	R/W	IIC 1 Low Master Address
IIC1_HMADR	0x1 40000505	R/W	IIC 1 High Master Address

Table 29-12. MMIO Registers

Register	Address	Access	Description
IIC1_CNTL	0x1 40000506	R/W	IIC 1 Control
IIC1_MDCNTL	0x1 40000507	R/W	IIC 1 Mode Control
IIC1_STS	0x1 40000508	R/W	IIC 1 Status
IIC1_EXTSTS	0x1 40000509	R/W	IIC 1 Extended Status
IIC1_LSADR	0x1 4000050A	R/W	IIC 1 Low Slave Address
IIC1_HSADR	0x1 4000050B	R/W	IIC 1 High Slave Address
IIC1_CLKDIV	0x1 4000050C	R/W	IIC 1 Clock Divide
IIC1_INTRMSK	0x1 4000050D	R/W	IIC 1 Interrupt Mask
IIC1_XFRCNT	0x1 4000050E	R/W	IIC 1 Transfer Count
IIC1_XTCNTLSS	0x1 4000050F	R/W	IIC 1 Extended Control and Slave Status
IIC1_DIRECTCNTL	0x1 40000510	R/W	IIC 1 Direct Control
OPB Arbiter			
OPBA0_PR	0x1 40000600	R/W	OPB Arbiter Priority Register
OPBA0_CR	0x1 40000601	R/W	OPB Arbiter Control Register
General-Purpose I/O			
GPIO0_OR	0x1 40000700	R/W	GPIO Output Register
GPIO0_TCR	0x1 40000704	R/W	GPIO Three-State Control Register
GPIO0_ODR	0x1 40000718	R/W	GPIO Open Drain Register
GPIO0_IR	0x1 4000071C	R	GPIO Input Register
Ethernet to PHY Bridge (ZMII)			
ZMII0_FER	0x1 40000780	R/W	ZMII Function Enable Register
ZMII0_SSR	0x1 40000784	R/W	ZMII Speed Select Register
ZMII0_SMIISR	0x1 40000788	R/W	ZMII SMII Status Register
Ethernet MAC 0			
EMAC0_MR0	0x1 40000800	R/W	EMAC 0 Mode Register 0
EMAC0_MR1	0x1 40000804	R/W	EMAC 0 Mode Register 1
EMAC0_TMR0	0x1 40000808	R/W	EMAC 0 Transmit Mode Register 0
EMAC0_TMR1	0x1 4000080C	R/W	EMAC 0 Transmit Mode Register 1
EMAC0_RMR	0x1 40000810	R/W	EMAC 0 Receive Mode Register
EMAC0_ISR	0x1 40000814	R/W	EMAC 0 Interrupt Status Register
EMAC0_ISER	0x1 40000818	R/W	EMAC 0 Interrupt Status Enable Register
EMAC0_IAHR	0x1 4000081C	R/W	EMAC 0 Individual Address High
EMAC0_IALR	0x1 40000820	R/W	EMAC 0 Individual Address Low
EMAC0_VTPID	0x1 40000824	R/W	EMAC 0 VLAN TPID Register
EMAC0_VTCI	0x1 40000828	R/W	EMAC 0 VLAN TCI Register
EMAC0_PTR	0x1 4000082C	R/W	EMAC 0 Pause Timer Register
EMAC0_IAHT1	0x1 40000830	R/W	EMAC 0 Individual Address Hash Table 1

Table 29-12. MMIO Registers

Register	Address	Access	Description
EMAC0_IAHT2	0x1 40000834	R/W	EMAC 0 Individual Address Hash Table 2
EMAC0_IAHT3	0x1 40000838	R/W	EMAC 0 Individual Address Hash Table 3
EMAC0_IAHT4	0x1 4000083C	R/W	EMAC 0 Individual Address Hash Table 4
EMAC0_GAHT1	0x1 40000840	R/W	EMAC 0 Group Address Hash Table 1
EMAC0_GAHT2	0x1 40000844	R/W	EMAC 0 Group Address Hash Table 2
EMAC0_GAHT3	0x1 40000848	R/W	EMAC 0 Group Address Hash Table 3
EMAC0_GAHT4	0x1 4000084C	R/W	EMAC 0 Group Address Hash Table 4
EMAC0_LSAH	0x1 40000850	R	EMAC 0 Last Source Address Low
EMAC0_LSAL	0x1 40000854	R	EMAC 0 Last Source Address High
EMAC0_IPGVR	0x1 40000858	R/W	EMAC 0 Inter-Packet Gap Value Register
EMAC0_STACR	0x1 4000085C	R/W	EMAC 0 STA Control Register
EMAC0_TRTR	0x1 40000860	R/W	EMAC 0 Transmit Request Threshold Register
EMAC0_RWMR	0x1 40000864	R/W	EMAC 0 Receive Low/High Water Mark Register
EMAC0_OCTX	0x1 40000868	R	EMAC 0 Number of Octets Transimitted
EMAC0_OCRX	0x1 4000086C	R	EMAC 0 Number of Octets Received
Ethernet MAC 1			
EMAC1_MR0	0x1 40000900	R/W	EMAC 1 Mode Register 0
EMAC1_MR1	0x1 40000904	R/W	EMAC 1 Mode Register 1
EMAC1_TMR0	0x1 40000908	R/W	EMAC 1 Transmit Mode Register 0
EMAC1_TMR1	0x1 4000090C	R/W	EMAC 1 Transmit Mode Register 1
EMAC1_RMR	0x1 40000910	R/W	EMAC 1 Receive Mode Register
EMAC1_ISR	0x1 40000914	R/W	EMAC 1 Interrupt Status Register
EMAC1_ISER	0x1 40000918	R/W	EMAC 1 Interrupt Status Enable Register
EMAC1_IAHR	0x1 4000091C	R/W	EMAC 1 Individual Address High
EMAC1_IALR	0x1 40000920	R/W	EMAC 1 Individual Address Low
EMAC1_VTPID	0x1 40000924	R/W	EMAC 1 VLAN TPID Register
EMAC1_VTCI	0x1 40000928	R/W	EMAC 1 VLAN TCI Register
EMAC1_PTR	0x1 4000092C	R/W	EMAC 1 Pause Timer Register
EMAC1_IAHT1	0x1 40000930	R/W	EMAC 1 Individual Address Hash Table 1
EMAC1_IAHT2	0x1 40000934	R/W	EMAC 1 Individual Address Hash Table 2
EMAC1_IAHT3	0x1 40000938	R/W	EMAC 1 Individual Address Hash Table 3
EMAC1_IAHT4	0x1 4000093C	R/W	EMAC 1 Individual Address Hash Table 4
EMAC1_GAHT1	0x1 40000940	R/W	EMAC 1 Group Address Hash Table 1
EMAC1_GAHT2	0x1 40000944	R/W	EMAC 1 Group Address Hash Table 2
EMAC1_GAHT3	0x1 40000948	R/W	EMAC 1 Group Address Hash Table 3
EMAC1_GAHT4	0x1 4000094C	R/W	EMAC 1 Group Address Hash Table 4
EMAC1_LSAH	0x1 40000950	R	EMAC 1 Last Source Address Low

Table 29-12. MMIO Registers

Register	Address	Access	Description
EMAC1_LSAL	0x1 40000954	R	EMAC 1 Last Source Address High
EMAC1_IPGVR	0x1 40000958	R/W	EMAC 1 Inter-Packet Gap Value Register
EMAC1_STACR	0x1 4000095C	R/W	EMAC 1 STA Control Register
EMAC1_TRTR	0x1 40000960	R/W	EMAC 1 Transmit Request Threshold Register
EMAC1_RWMR	0x1 40000964	R/W	EMAC 1 Receive Low/High Water Mark Register
EMAC1_OCTX	0x1 40000968	R	EMAC 1 Number of Octets Transimitted
EMAC1_OCRX	0x1 4000096C	R	EMAC 1 Number of Octets Received
General Purpose Timer			
GPT0_TBC	0x1 40000A00	R/W	GPT Time Base Counter
GPT0_OE	0x1 40000A10	R/W	GPT Output Enable
GPT0_OL	0x1 40000A14	R/W	GPT Output Level
GPT0_IM	0x1 40000A18	R/W	GPT Interrupt Mask
GPT0_ISS	0x1 40000A1C	R/W	GPT Interrupt Status (Set bits if write 1)
GPT0_ISC	0x1 40000A20	R/W	GPT Interrupt Status (Clear bits if write 1)
GPT0_IE	0x1 40000A24	R/W	GPT Interrupt Enable
GPT0_COMP0	0x1 40000A80	R/W	GPT Compare Timer 0
GPT0_COMP1	0x1 40000A84	R/W	GPT Compare Timer 1
GPT0_COMP2	0x1 40000A88	R/W	GPT Compare Timer 2
GPT0_COMP3	0x1 40000A8C	R/W	GPT Compare Timer 3
GPT0_COMP4	0x1 40000A90	R/W	GPT Compare Timer 4
GPT0_MASK0	0x1 40000AC0	R/W	GPT Compare Mask 0
GPT0_MASK1	0x1 40000AC4	R/W	GPT Compare Mask 1
GPT0_MASK2	0x1 40000AC8	R/W	GPT Compare Mask 2
GPT0_MASK3	0x1 40000ACC	R/W	GPT Compare Mask 3
GPT0_MASK4	0x1 40000AD0	R/W	GPT Compare Mask 4

The two registers listed in Table 29-13 are used to access the configuration registers of external PCI-X devices only.

Table 29-13. External PCI-X Device Configuration Register Access

Register	Address	Access	Description
PCIX0_CFGADDR	0x2 0EC00000	R/W	PCI-X Configuration Address Register
PCIX0_CFGDATA	0x2 0EC00004	R/W	PCI-X Configuration Data Register

Registers listed in Table 29-14 can be accessed in two ways:

1. By the processor using the PLB address

2. By an external PCI-X device using configuration cycles to the proper offset.

Table 29-14. Internal PCI-X Configuration Registers (continued)

Register	PLB		PCI-X Configuration		Description
	Address	Access	Offset	Access	
PCIX0_VENDID	0x2 0EC80000	R/W	0x01–0x00	R	PCI-X Vendor ID
PCIX0_DEVID	0x2 0EC80002	R/W	0x03–0x02	R	PCI-X Device ID
PCIX0_CMD	0x2 0EC80004	R/W	0x05–0x04	R/W	PCI-X Command Register
PCIX0_STATUS	0x2 0EC80006	R/W	0x07–0x06	R/W	PCI-X Status Register
PCIX0_REVID	0x2 0EC80008	R/W	0x08	R	PCI-X Revision ID
PCIX0_CLS	0x2 0EC80009	R/W	0x0B–0x09	R	PCI-X Class Register
PCIX0_CACHELS	0x2 0EC8000C	R/W	0x0C	R/W	PCI-X Cache Line Size
PCIX0_LATTIM	0x2 0EC8000D	R/W	0x0D	R/W	PCI-X Latency Timer
PCIX0_HDTYPE	0x2 0EC8000E	R	0x0E	R	PCI-X Header Type
PCIX0_BIST	0x2 0EC8000F	R	0x0F	R	PCI-X Built In Self Test Control
PCIX0_ BAR0L	0x2 0EC80010	R/W	0x13–0x10	R/W	PCI-X BAR 0 Low
PCIX0_ BAR0H	0x2 0EC80014	R/W	0x17–0x14	R/W	PCI-X BAR 0 High
PCIX0_BAR1	0x2 0EC80018	R/W	0x1B–0x18	R/W	PCI-X BAR 1
PCIX0_ BAR2L	0x2 0EC8001C	R/W	0x1F–0x1C	R/W	PCI-X BAR 2 Low
PCIX0_ BAR2H	0x2 0EC80020	R/W	0x23–0x20	R/W	PCI-X BAR 2 High
PCIX0_ BAR3	0x2 0EC80024	R	0x27–0x24	R	Unused BAR 3
PCIX0_CISPTR	0x2 0EC80028	R	0x2B–0x28	R	Unused Cardbus CIS Pointer
PCIX0_SBSYSVID	0x2 0EC8002C	R/W	0x2D–0x2C	R	PCI-X Subsystem Vendor ID
PCIX0_SBSYSID	0x2 0EC8002E	R/W	0x2F–0x2E	R	PCI-X Subsystem ID
PCIX0_EROMBA	0x2 0EC80030	R/W	0x33–0x30	R/W	PCI-X Expansion ROM Base Address
PCIX0_CAP	0x2 0EC80034	R	0x34	R	PCI-X Capabilities Pointer
PCIX0_RES0	0x2 0EC80035	R	0x35	R	Reserved
PCIX0_RES1	0x2 0EC80036	R	0x37–0x6	R	Reserved
PCIX0_RES2	0x2 0EC80038	R	0x3B–0x8	R	Reserved
PCIX0_INTLN	0x2 0EC8003C	R/W	0x3C	R/W	PCI-X Interrupt Line
PCIX0_INTPN	0x2 0EC8003D	R	0x3D	R	PCI-X Interrupt Pin
PCIX0_MINGNT	0x2 0EC8003E	R	0x3E	R	PCI-X Minimum Grant
PCIX0_MAXLTNCY	0x2 0EC8003F	R	0x3F	R	PCI-X Maximum Latency
PCIX0_BRDGOPT1	0x2 0EC80040	R/W	0x43–0x40	R/W	PCI-X Bridge Options 1
PCIX0_BRDGOPT2	0x2 0EC80044	R/W	0x47–0x44	R/W	PCI-X Bridge Options 2
PCIX0_ERREN	0x2 0EC80050	R/W	0x53–0x50	R/W	PCI-X Error Enable
PCIX0_ERRSTS	0x2 0EC80054	R/W	0x57–0x54	R/W	PCI-X Error Status
PCIX0_PLBBESR	0x2 0EC80058	R	0x5B–0x58	R	PCI-X PLB Slave Error Syndrome Register

Table 29-14. Internal PCI-X Configuration Registers (continued)

Register	PLB		PCI-X Configuration		Description
	Address	Access	Offset	Access	
PCIX0_PLBBEARL	0x2 0EC8005C	R	0x5F–0x5C	R	PCI-X PLB Slave Error Address Register Low
PCIX0_PLBBEARH	0x2 0EC80060	R	0x63–0x60	R	PCI-X PLB Slave Error Address Register High
PCIX0_POM0LAL	0x2 0EC80068	R/W	0x6B–0x68	R/W	PCI-X POM0 Local Address Low
PCIX0_POM0LAH	0x2 0EC8006C	R/W	0x6F–0x6C	R/W	PCI-X POM0 Local Address High
PCIX0_POM0SA	0x2 0EC80070	R/W	0x73–0x70	R/W	PCI-X POM0 Size Attribute
PCIX0_POM0PCIAL	0x2 0EC80074	R/W	0x77–0x74	R/W	PCI-X POM0 PCI Address Low
PCIX0_POM0PCIAH	0x2 0EC80078	R/W	0x7B–0x78	R/W	PCI-X POM0 PCI Address High
PCIX0_POM1LAL	0x2 0EC8007C	R/W	0x7F–0x7C	R/W	PCI-X POM1 Local Address Low
PCIX0_POM1LAH	0x2 0EC80080	R/W	0x83–0x80	R/W	PCI-X POM1 Local Address High
PCIX0_POM1SA	0x2 0EC80084	R/W	0x87–0x84	R/W	PCI-X POM1 Size Attribute
PCIX0_POM1PCIAL	0x2 0EC80088	R/W	0x8B–0x88	R/W	PCI-X POM1 PCI Address Low
PCIX0_POM1PCIAH	0x2 0EC8008C	R/W	0x8F–0x8C	R/W	PCI-X POM1 PCI Address High
PCIX0_POM2SA	0x2 0EC80090	R/W	0x93–0x90	R/W	PCI-X POM2 Size Attribute
PCIX0_PIM0SA	0x2 0EC80098	R/W	0x9B–0x98	R/W	PCI-X PIM0 Size/Attribute
PCIX0_PIM0LAL	0x2 0EC8009C	R/W	0x9F–0x9C	R/W	PCI-X PIM0 Local Address Low
PCIX0_PIM0LAH	0x2 0EC800A0	R/W	0xA3–0xA0	R/W	PCI-X PIM0 Local Address High
PCIX0_PIM1SA	0x2 0EC800A4	R/W	0xA7–0xA4	R/W	PCI-X PIM1 Size/Attribute
PCIX0_PIM1LAL	0x2 0EC800A8	R/W	0xAB–0xA8	R/W	PCI-X PIM1 Local Address Low
PCIX0_PIM1LAH	0x2 0EC800AC	R/W	0xAF–0xAC	R/W	PCI-X PIM1 Local Address High
PCIX0_PIM2SA	0x2 0EC800B0	R/W	0xB3–0xB0	R/W	PCI-X PIM2 Size/Attribute
PCIX0_PIM2LAL	0x2 0EC800B4	R/W	0xB7–0xB4	R/W	PCI-X PIM2 Local Address Low
PCIX0_PIM2LAH	0x2 0EC800B8	R/W	0xBB–0xB8	R/W	PCI-X PIM2 Local Address High
PCIX0_OMCAPID	0x2 0EC800C0	R	0xC0	R	PCI-X Outbound MSI Capability Identifier
PCIX0_OMNIPTR	0x2 0EC800C1	R/W	0xC1	R	PCI-X Outbound MSI Next Item Pointer
PCIX0_OMMC	0x2 0EC800C2	R/W	0xC3–0xC2	R/W	PCI-X Outbound MSI Message Control
PCIX0_OMMA	0x2 0EC800C4	R/W	0xC7–0xC4	R/W	PCI-X Outbound MSI Message Address
PCIX0_OMMUA	0x2 0EC800C8	R/W	0xCB–0xC8	R/W	PCI-X Outbound MSI Message Upper Address
PCIX0_OMMDATA	0x2 0EC800CC	R/W	0xCD–0xCC	R/W	PCI-X Outbound MSI Message Data
PCIX0_OMMEOI	0x2 0EC800CE	R/W	0xCE	R/W	PCI-X Outbound MSI Message End Of Interrupt
PCIX0_PMCAPID	0x2 0EC800D0	R/W	0xD0	R	PCI-X PMC Capability Identifier
PCIX0_PMNIPTR	0x2 0EC800D1	R/W	0xD1	R	PCI-X PMC Next Item Pointer
PCIX0_PMC	0x2 0EC800D2	R	0xD3–0xD2	R	PCI-X Power Management Capabilities

Table 29-14. Internal PCI-X Configuration Registers (continued)

Register	PLB		PCI-X Configuration		Description
	Address	Access	Offset	Access	
PCIX0_PMCSR	0x2 0EC800D4	R/W	0xD5–0xD4	R/W	PCI-X Power Management Control Status
PCIX0_PMCSRBSE	0x2 0EC800D6	R	0xD6	R	PCI-X PMCSR PCI to PCI Bridge Support Extensions
PCIX0_PMDATA	0x2 0EC800D7	R	0xD7	R	PCI-X PMC Unused Data Register
PCIX0_PMSCRR	0x2 0EC800D8	R/W	0xD8	R/W	PCI-X Power Management State Change Request Register
PCIX0_PCIXCAPID	0x2 0EC800DC	R	0xDC	R	PCI-X Capability Identifier
PCIX0_PCIXNIPTR	0x2 0EC800DD	R	0xDD	R	PCI-X Next Item Pointer
PCIX0_PCIXCMD	0x2 0EC800DE	R/W	0xDF–0xDE	R/W	PCI-X Command
PCIX0_PCIXSTS	0x2 0EC800E0	R/W	0xE3–0xE0	R/W	PCI-X Status
PCIX0_PCIXIDR	0x2 0EC800E4	R/W	0xE7–0xE4	R/W	PCI-X Internal Debug Register
PCIX0_PCIXCID	0x2 0EC800E8	R	0xEB–0xE8	R	PCI-X Internal Core Device ID
PCIX0_PCIXRID	0x2 0EC800EC	R	0xEF–0xEC	R	PCI-X Internal Core Revision ID

Registers listed in Table 29-15 can be accessed in two ways:

1. By the processor using the PLB address
2. By an external PCI-X device using memory cycles to the proper address.

Table 29-15. PCI-X SImple Message Passing and Inbound MSI Registers

Register	PLB		PCI-X Memory		Description
	Address	Access	Address	Access	
PCIX0_MSGIL	0x2 0EC80100	R/W	(PCIX0_BAR0H || PCIX0_BAR0L) + (0x03–0x00)	R/W	PCI-X Message In Low
PCIX0_MSGIH	0x2 0EC80104	R/W	(PCIX0_BAR0H || PCIX0_BAR0L) + (0x07–0x04)	R/W	PCI-X Message In High
PCIX0_MSGOL	0x2 0EC80108	R/W	(PCIX0_BAR0H || PCIX0_BAR0L) + (0x0B–0x08)	R/W	PCI-X Message Out Low
PCIX0_MSGOH	0x2 0EC8010C	R/W	(PCIX0_BAR0H || PCIX0_BAR0L) + (0x0F–0x0C)	R/W	PCI-X Message Out High
PCIX0_IM	0x2 0EC801F8	W	(PCIX0_BAR0H || PCIX0_BAR0L) + (0xFB–0xF8)	W	PCI-X Inbound MSI

29.5 Alphabetical Register Listing of Processor Core Registers

The following pages list the registers available in the PPC440GP. For each register, the following information is supplied:

- Register mnemonic and name
- Cross reference to detailed register information
- Register type (if SPR); the types of the other registers are the same as the register names (CR, GPR, MSR)
- Register number (address)
- Register programming model (user or supervisor) and access (read-clear, read-only, read/write (R/W), write-only)
- A diagram illustrating the register fields (all register fields have mnemonics, unless there is only one field)
- A table describing the register fields, giving field mnemonics, field bit locations, field names, and the functions associated with the various field values

SPR 0x3B3 Supervisor R/W

See *Core Configuration Register 0 (CCR0)* on page 216.

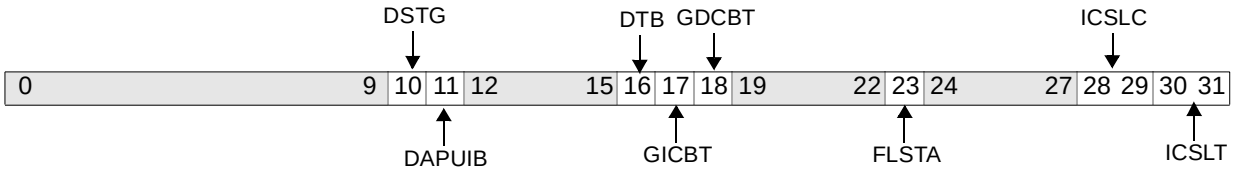


Figure 0-1. Core Configuration Register 0 (CCR0)

0:9		Reserved	
10	DSTG	Disable Store Gathering 0 Enabled; stores to contiguous addresses may be gathered into a single transfer 1 Disabled; all stores to memory will be performed independently	See "Store Gathering" on page 5-21.
11	DAPUIB	Disable APU Instruction Broadcast 0 Enabled. 1 Disabled; instructions not broadcast to APU for decoding	This mechanism is provided as a means of reducing power consumption when an auxilliary processor is not attached and/or is not being used. See Chapter 7, "Reset and Initialization".
12:15		Reserved	
16	DTB	Disable Trace Broadcast 0 Enabled. 1 Disabled; no trace information is broadcast.	This mechanism is provided as a means of reducing power consumption when instruction tracing is not needed. See Chapter 7, "Reset and Initialization".
17	GICBT	Guaranteed Instruction Cache Block Touch 0 **icbt** may be abandoned without having filled cache line if instruction pipeline stalls. 1 **icbt** is guaranteed to fill cache line even if instruction pipeline stalls.	See "icbt Operation" on page 5-15.
18	GDCBT	Guaranteed Data Cache Block Touch 0 **dcbt/dcbtst** may be abandoned without having filled cache line if instruction pipeline stalls. 1 **dcbt/dcbtst** are guaranteed to fill cache line even if instruction pipeline stalls.	See "Data Cache Control and Debug" on page 5-25.
19:22		Reserved	
23	FLSTA	Force Load/Store Alignment 0 No Alignment exception on integer storage access instructions, regardless of alignment 1 An alignment exception occurs on integer storage access instructions if data address is not on an operand boundary.	See "Load and Store Alignment" on page 5-19.

24:27		Reserved	
28:29	ICSLC	Instruction Cache Speculative Line Count	Number of additional lines (0–3) to fill on instruction fetch miss. See "Speculative Prefetch Mechanism" on page 5-10.
30:31	ICSLT	Instruction Cache Speculative Line Threshold	Number of doublewords that must have already been filled in order that the current speculative line fill is *not* abandoned on a redirection of the instruction stream. See "Speculative Prefetch Mechanism" on page 5-10.

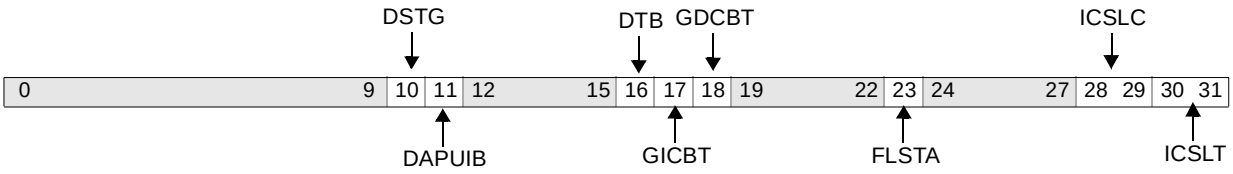


Figure 29-1. Core Configuration Register 0 (CCR0)

0:9		Reserved	
10	DSTG	Disable Store Gathering 0 Enabled; stores to contiguous addresses may be gathered into a single transfer 1 Disabled; all stores to memory will be performed independently	See "Store Gathering" on page -223.
11	DAPUIB	Disable APU Instruction Broadcast 0 Enabled. 1 Disabled; instructions not broadcast to APU for decoding	This mechanism is provided as a means of reducing power consumption when an auxilliary processor is not attached and/or is not being used. See *Reset and Initialization* on page 259.
12:15		Reserved	
16	DTB	Disable Trace Broadcast 0 Enabled. 1 Disabled; no trace information is broadcast.	This mechanism is provided as a means of reducing power consumption when instruction tracing is not needed. See *Reset and Initialization* on page 259.
17	GICBT	Guaranteed Instruction Cache Block Touch 0 **icbt** may be abandoned without having filled cache line if instruction pipeline stalls. 1 **icbt** is guaranteed to fill cache line even if instruction pipeline stalls.	See *icbt Operation* on page 218.
18	GDCBT	Guaranteed Data Cache Block Touch 0 **dcbt/dcbtst** may be abandoned without having filled cache line if load/store pipeline stalls. 1 **dcbt/dcbtst** are guaranteed to fill cache line even if load/store pipeline stalls.	See *Data Cache Control and Debug* on page 228.
19:22		Reserved	

23	FLSTA	Force Load/Store Alignment 0 No Alignment exception on integer storage access instructions, regardless of alignment 1 An alignment exception occurs on integer storage access instructions if data address is not on an operand boundary.	See *Load and Store Alignment* on page 222.
24:27		Reserved	
28:29	ICSLC	Instruction Cache Speculative Line Count	Number of additional lines (0–3) to fill on instruction fetch miss. See *Speculative Prefetch Mechanism* on page 213.
30:31	ICSLT	Instruction Cache Speculative Line Threshold	Number of doublewords that must have already been filled in order that the current speculative line fill is *not* abandoned on a redirection of the instruction stream. See *Speculative Prefetch Mechanism* on page 213.

User Read/Write

See *Condition Register (CR)* on page 189.

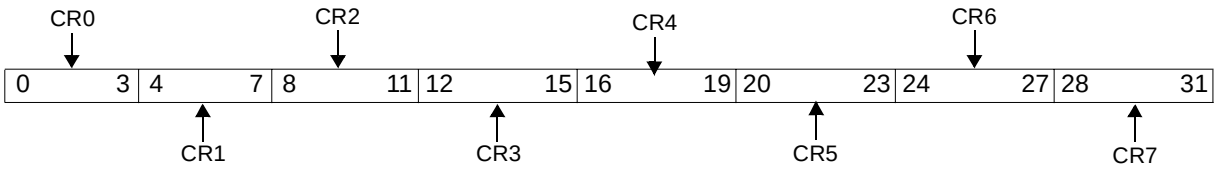


Figure 0-2. Condition Register (CR)

0:3	CR0	Condition Register Field 0
4:7	CR1	Condition Register Field 1
8:11	CR2	Condition Register Field 2
12:15	CR3	Condition Register Field 3
16:19	CR4	Condition Register Field 4
20:23	CR5	Condition Register Field 5
24:27	CR6	Condition Register Field 6
28:31	CR7	Condition Register Field 7

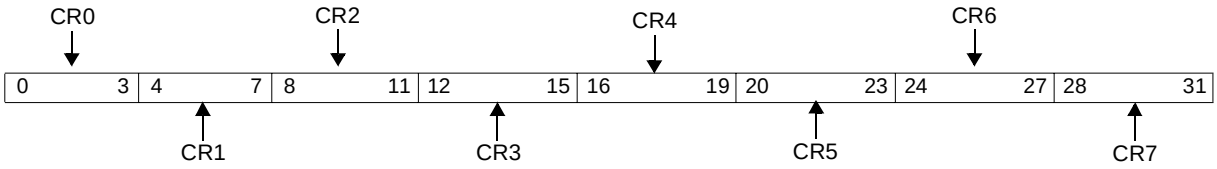


Figure 29-2. Condition Register (CR)

0:3	CR0	Condition Register Field 0
4:7	CR1	Condition Register Field 1
8:11	CR2	Condition Register Field 2
12:15	CR3	Condition Register Field 3
16:19	CR4	Condition Register Field 4
20:23	CR5	Condition Register Field 5
24:27	CR6	Condition Register Field 6
28:31	CR7	Condition Register Field 7

SPR 0x03A Supervisor R/W

See *Critical Save/Restore Register 0 (CSRR0)* on page 346.

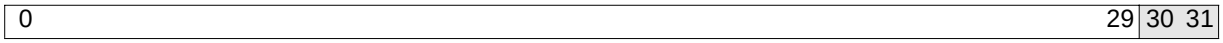


Figure 0-3. Critical Save/Restore Register 0 (CSRR0)

0:29		Return address for critical interrupts
30:31		Reserved

Figure 29-3. Critical Save/Restore Register 0 (CSRR0)

0:29		Return address for critical interrupts
30:31		Reserved

SPR 0x03B Supervisor R/W

See *Critical Save/Restore Register 1 (CSRR1)* on page 346.

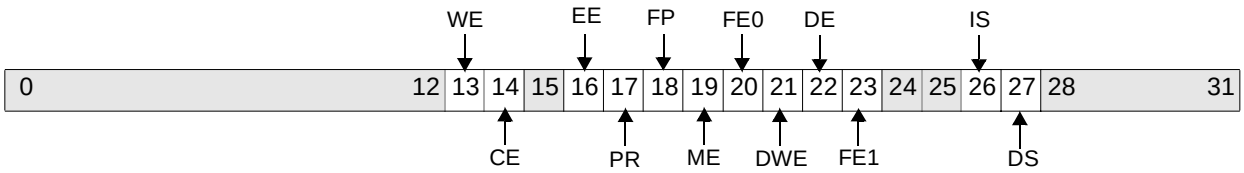


Figure 0-4. Critical Save/Restore Register 1 (CSRR1)

0:31	Copy of the MSR when a critical interrupt is taken

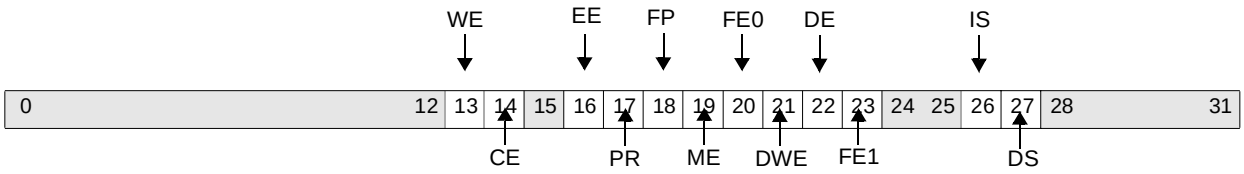


Figure 29-4. Critical Save/Restore Register 1 (CSRR1)

0:31	Copy of the MSR when a critical interrupt is taken

SPR 0x009 User R/W

See *Count Register (CTR)* on page 188.

Figure 0-5. Count Register (CTR)

0:31		Count	Used as count for branch conditional with decrement instructions, or as target address for **bcctr** instructions

Figure 29-5. Count Register (CTR)

0:31		Count	Used as count for branch conditional with decrement instructions, or as target address for **bcctr** instructions

SPR 0x13C–0x13D Supervisor R/W

See *Data Address Compare Registers (DAC1–DAC2)* on page 458.

Figure 0-6. Data Address Compare Registers (DAC1–DAC2)

0:31		Data Address Compare (DAC) byte address

Figure 29-6. Data Address Compare Registers (DAC1–DAC2)

0:31		Data Address Compare (DAC) byte address

SPR 0x134 Supervisor R/W

See *Debug Control Register 0 (DBCR0)* on page 451.

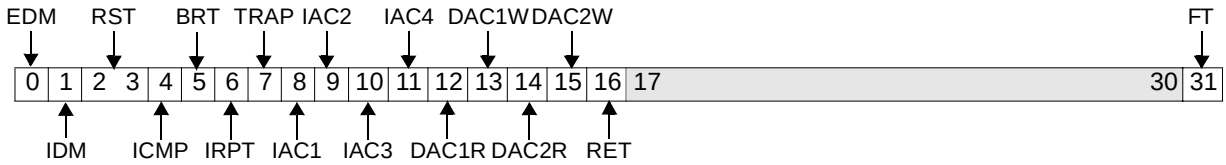


Figure 0-7. Debug Control Register 0 (DBCR0)

0	EDM	External Debug Mode 0 Disable external debug mode. 1 Enable external debug mode.	
1	IDM	Internal Debug Mode 0 Disable internal debug mode. 1 Enable internal debug mode.	
2:3	RST	Reset 00 No action 01 Core reset 10 Chip reset 11 System reset	
		Attention: Writing 01, 10, or 11 to this field causes a processor reset to occur.	
4	ICMP	Instruction Completion Debug Event 0 Disable instruction completion debug event. 1 Enable instruction completion debug event.	Instruction completions do not cause instruction completion debug events if MSR[DE] = 0 in internal debug mode, unless also in external debug mode or debug wait mode.
5	BRT	Branch Taken Debug Event 0 Disable branch taken debug event. 1 Enable branch taken debug event.	Taken branches do not cause branch taken debug events if MSR[DE] = 0 in internal debug mode, unless also in external debug mode or debug wait mode.
6	IRPT	Interrupt Debug Event 0 Disable interrupt debug event. 1 Enable interrupt debug event.	Critical interrupts do not cause interrupt debug events in internal debug mode, unless also in external debug mode or debug wait mode.
7	TRAP	Trap Debug Event 0 Disable trap debug event. 1 Enable trap debug event.	
8	IAC1	Instruction Address Compare (IAC) 1 Debug Event 0 Disable IAC 1 debug event. 1 Enable IAC 1 debug event.	

9	IAC2	IAC 2 Debug Event 0 Disable IAC 2 debug event. 1 Enable IAC 2 debug event.	
10	IAC3	IAC 3 Debug Event 0 Disable IAC 3 debug event. 1 Enable IAC 3 debug event.	
11	IAC4	IAC 4 Debug Event 0 Disable IAC 4 debug event. 1 Enable IAC 4 debug event.	
12	DAC1R	Data Address Compare (DAC) 1 Read Debug Event 0 Disable DAC 1 read debug event. 1 Enable DAC 1 read debug event.	
13	DAC1W	DAC 1 Write Debug Event 0 Disable DAC 1 write debug event. 1 Enable DAC 1 write debug event.	
14	DAC2R	DAC 2 Read Debug Event 0 Disable DAC 2 read debug event. 1 Enable DAC 2 read debug event.	
15	DAC2W	DAC 2 Write Debug Event 0 Disable DAC 2 write debug event. 1 Enable DAC 2 write debug event.	
16	RET	Return Debug Event 0 Disable return (**rfi/rfci**) debug event. 1 Enable return (**rfi/rfci**) debug event.	**rfci** does not cause a return debug event if MSR[DE] = 0 in internal debug mode, unless also in external debug mode or debug wait mode.
17:30		Reserved	
31	FT	Freeze timers on debug event 0 Timers are not frozen. 1 Freeze timers if a DBSR field associated with a debug event is set.	

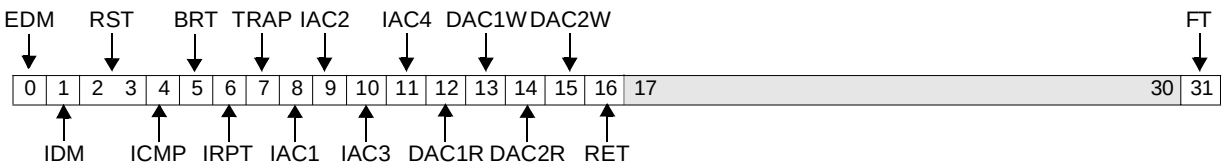


Figure 29-7. Debug Control Register 0 (DBCR0)

0	EDM	External Debug Mode 0 Disable external debug mode. 1 Enable external debug mode.
1	IDM	Internal Debug Mode 0 Disable internal debug mode. 1 Enable internal debug mode.

2:3	RST	Reset 00 No action 01 Core reset 10 Chip reset 11 System reset	
		Attention: Writing 01, 10, or 11 to this field causes a processor reset to occur.	
4	ICMP	Instruction Completion Debug Event 0 Disable instruction completion debug event. 1 Enable instruction completion debug event.	Instruction completions do not cause instruction completion debug events if MSR[DE] = 0 in internal debug mode, unless also in external debug mode or debug wait mode.
5	BRT	Branch Taken Debug Event 0 Disable branch taken debug event. 1 Enable branch taken debug event.	Taken branches do not cause branch taken debug events if MSR[DE] = 0 in internal debug mode, unless also in external debug mode or debug wait mode.
6	IRPT	Interrupt Debug Event 0 Disable interrupt debug event. 1 Enable interrupt debug event.	Critical interrupts do not cause interrupt debug events in internal debug mode, unless also in external debug mode or debug wait mode.
7	TRAP	Trap Debug Event 0 Disable trap debug event. 1 Enable trap debug event.	
8	IAC1	Instruction Address Compare (IAC) 1 Debug Event 0 Disable IAC 1 debug event. 1 Enable IAC 1 debug event.	
9	IAC2	IAC 2 Debug Event 0 Disable IAC 2 debug event. 1 Enable IAC 2 debug event.	
10	IAC3	IAC 3 Debug Event 0 Disable IAC 3 debug event. 1 Enable IAC 3 debug event.	
11	IAC4	IAC 4 Debug Event 0 Disable IAC 4 debug event. 1 Enable IAC 4 debug event.	
12	DAC1R	Data Address Compare (DAC) 1 Read Debug Event 0 Disable DAC 1 read debug event. 1 Enable DAC 1 read debug event.	
13	DAC1W	DAC 1 Write Debug Event 0 Disable DAC 1 write debug event. 1 Enable DAC 1 write debug event.	
14	DAC2R	DAC 2 Read Debug Event 0 Disable DAC 2 read debug event. 1 Enable DAC 2 read debug event.	
15	DAC2W	DAC 2 Write Debug Event 0 Disable DAC 2 write debug event. 1 Enable DAC 2 write debug event.	
16	RET	Return Debug Event 0 Disable return (**rfi/rfci**) debug event. 1 Enable return (**rfi/rfci**) debug event.	**rfci** does not cause a return debug event if MSR[DE] = 0 in internal debug mode, unless also in external debug mode or debug wait mode.
17:30		Reserved	

31	FT	Freeze timers on debug event 0 Timers are not frozen. 1 Freeze timers if a DBSR field associated with a debug event is set.

SPR 0x135 Supervisor R/W

See *Debug Control Register 1 (DBCR1)* on page 452.

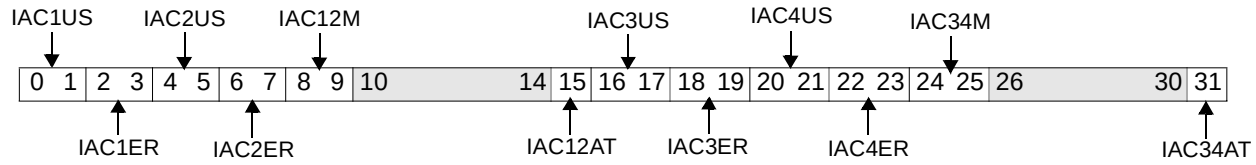


Figure 0-8. Debug Control Register 1 (DBCR1)

0:1	IAC1US	Instruction Address Compare (IAC) 1 User/Supervisor 00 Both 01 Reserved 10 Supervisor only (MSR[PR] = 0) 11 User only (MSR[PR] = 1)	
2:3	IAC1ER	IAC 1 Effective/Real 00 Effective (MSR[IS] = don't care) 01 Reserved 10 Virtual (MSR[IS] = 0 11 Virtual (MSR[IS] = 1)	
4:5	IAC2US	IAC 2 User/Supervisor 00 Both 01 Reserved 10 Supervisor only (MSR[PR] = 0) 11 User only (MSR[PR] = 1)	
6:7	IAC2ER	IAC 2 Effective/Real 00 Effective (MSR[IS] = don't care) 01 Reserved 10 Virtual (MSR[IS] = 0 11 Virtual (MSR[IS] = 1)	
8:9	IAC12M	IAC 1/2 Mode 00 Exact match 01 Reserved 10 Range inclusive 11 Range exclusive	Match if address[0:29] = IAC 1/2[0:29]; two independent compares Match if IAC1 ≤ address < IAC2 Match if address < IAC1 OR address ≥ IAC2
10:14		Reserved	
15	IAC12AT	IAC 1/2 Auto-Toggle Enable 0 Disable IAC 1/2 auto-toggle 1 Enable IAC 1/2 auto-toggle	
16:17	IAC3US	IAC 3 User/Supervisor 00 Both 01 Reserved 10 Supervisor only (MSR[PR] = 0) 11 User only (MSR[PR] = 1)	

18:19	IAC3ER	IAC 3 Effective/Real 00 Effective (MSR[IS] = don't care) 01 Reserved 10 Virtual (MSR[IS] = 0 11 Virtual (MSR[IS] = 1)
20:21	IAC4US	IAC 4 User/Supervisor 00 Both 01 Reserved 10 Supervisor only (MSR[PR] = 0) 11 User only (MSR[PR] = 1)
22:23	IAC4ER	IAC 4 Effective/Real 00 Effective (MSR[IS] = don't care) 01 Reserved 10 Virtual (MSR[IS] = 0 11 Virtual (MSR[IS] = 1)
24:25	IAC34M	IAC 3/4 Mode 00 Exact match — Match if address[0:29] = IAC 3/4[0:29]; two independent compares 01 Reserved 10 Range inclusive — Match if IAC3 ≤ address < IAC4 11 Range exclusive — Match if address < IAC3 OR address ≥ IAC4
26:30		Reserved
31	IAC34AT	IAC3/4 Auto-Toggle Enable 0 Disable IAC 3/4 auto-toggle 1 Enable IAC 3/4 auto-toggle

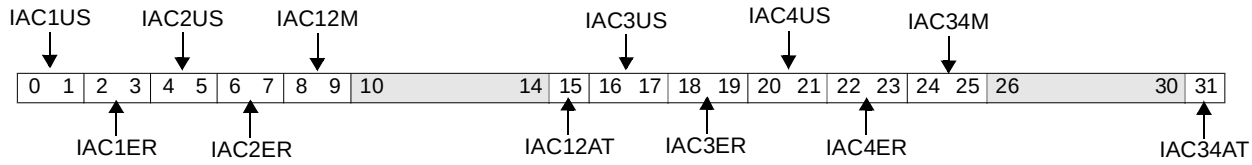


Figure 29-8. Debug Control Register 1 (DBCR1)

0:1	IAC1US	Instruction Address Compare (IAC) 1 User/Supervisor 00 Both 01 Reserved 10 Supervisor only (MSR[PR] = 0) 11 User only (MSR[PR] = 1)
2:3	IAC1ER	IAC 1 Effective/Real 00 Effective (MSR[IS] = don't care) 01 Reserved 10 Virtual (MSR[IS] = 0) 11 Virtual (MSR[IS] = 1)

4:5	IAC2US	IAC 2 User/Supervisor 00 Both 01 Reserved 10 Supervisor only (MSR[PR] = 0) 11 User only (MSR[PR] = 1)
6:7	IAC2ER	IAC 2 Effective/Real 00 Effective (MSR[IS] = don't care) 01 Reserved 10 Virtual (MSR[IS] = 0) 11 Virtual (MSR[IS] = 1)
8:9	IAC12M	IAC 1/2 Mode 00 Exact match — Match if address[0:29] = IAC 1/2[0:29]; two independent compares 01 Reserved 10 Range inclusive — Match if IAC1 ≤ address < IAC2 11 Range exclusive — Match if address < IAC1 OR address ≥ IAC2
10:14		Reserved
15	IAC12AT	IAC 1/2 Auto-Toggle Enable 0 Disable IAC 1/2 auto-toggle 1 Enable IAC 1/2 auto-toggle
16:17	IAC3US	IAC 3 User/Supervisor 00 Both 01 Reserved 10 Supervisor only (MSR[PR] = 0) 11 User only (MSR[PR] = 1)
18:19	IAC3ER	IAC 3 Effective/Real 00 Effective (MSR[IS] = don't care) 01 Reserved 10 Virtual (MSR[IS] = 0) 11 Virtual (MSR[IS] = 1)
20:21	IAC4US	IAC 4 User/Supervisor 00 Both 01 Reserved 10 Supervisor only (MSR[PR] = 0) 11 User only (MSR[PR] = 1)
22:23	IAC4ER	IAC 4 Effective/Real 00 Effective (MSR[IS] = don't care) 01 Reserved 10 Virtual (MSR[IS] = 0) 11 Virtual (MSR[IS] = 1)
24:25	IAC34M	IAC 3/4 Mode 00 Exact match — Match if address[0:29] = IAC 3/4[0:29]; two independent compares 01 Reserved 10 Range inclusive — Match if IAC3 ≤ address < IAC4 11 Range exclusive — Match if address < IAC3 OR address ≥ IAC4
26:30		Reserved
31	IAC34AT	IAC3/4 Auto-Toggle Enable 0 Disable IAC 3/4 auto-toggle 1 Enable IAC 3/4 auto-toggle

SPR 0x136 Supervisor R/W

See *Debug Control Register 2 (DBCR2)* on page 455.

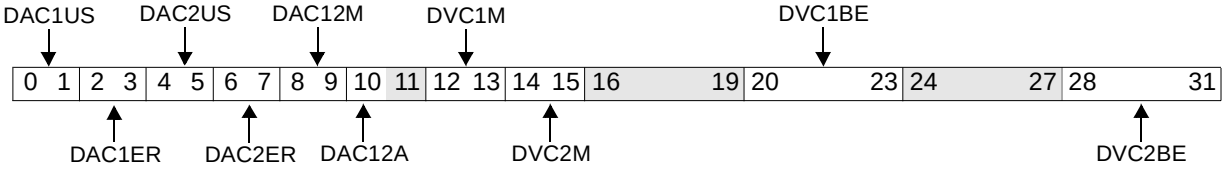


Figure 0-9. Debug Control Register 2 (DBCR2)

0:1	DAC1US	Data Address Compare (DAC) 1 User/Supervisor 00 Both 01 Reserved 10 Supervisor only (MSR[PR] = 0) 11 User only (MSR[PR] = 1)
2:3	DAC1ER	DAC 1 Effective/Real 00 Effective (MSR[DS] = don't care) 01 Reserved 10 Virtual (MSR[DS] = 0) 11 Virtual (MSR[DS] = 1)
4:5	DAC2US	DAC 2 User/Supervisor 00 Both 01 Reserved 10 Supervisor only (MSR[PR] = 0) 11 User only (MSR[PR] = 1)
6:7	DAC2ER	DAC 2 Effective/Real 00 Effective (MSR[DS] = don't care) 01 Reserved 10 Virtual (MSR[DS] = 0) 11 Virtual (MSR[DS] = 1)
8:9	DAC12M	DAC 1/2 Mode 00 Exact match — Match if address[0:31] = DAC 1/2[0:31]; two independent compares 01 Address bit mask — Match if address = DAC1; only compare bits corresponding to 1 bits in DAC2 10 Range inclusive — Match if DAC1 ≤ address < DAC2 11 Range exclusive — Match if address < DAC1 OR address ≥ DAC2
10	DAC12A	DAC 1/2 Asynchronous 0 Debug interrupt caused by DAC1/2 exception will be synchronous 1 Debug interrupt caused by DAC1/2 exception will be asynchronous
11		Reserved

12:13	DVC1M	Data Value Compare (DVC) 1 Mode 00 Reserved 01 AND all bytes enabled by DVC1BE 10 OR all bytes enabled by DVC1BE 11 AND-OR pairs of bytes enabled by DVC1BE (0 AND 1) OR (2 AND 3)
14:15	DVC2M	DVC 2 Mode 00 Reserved 01 AND all bytes enabled by DVC2BE 10 OR all bytes enabled by DVC2BE 11 AND-OR pairs of bytes enabled by DVC2BE (0 AND 1) OR (2 AND 3)
16:19		Reserved
20:23	DVC1BE	DVC 1 Byte Enables 0:3
24:27		Reserved
28:31	DVC2BE	DVC 2 Byte Enables 0:3

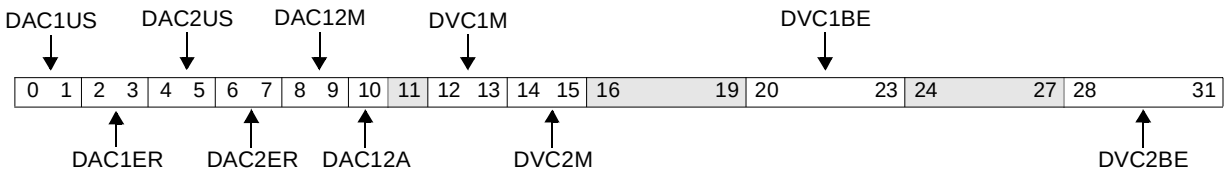


Figure 29-9. Debug Control Register 2 (DBCR2)

0:1	DAC1US	Data Address Compare (DAC) 1 User/Supervisor 00 Both 01 Reserved 10 Supervisor only (MSR[PR] = 0) 11 User only (MSR[PR] = 1)
2:3	DAC1ER	DAC 1 Effective/Real 00 Effective (MSR[DS] = don't care) 01 Reserved 10 Virtual (MSR[DS] = 0) 11 Virtual (MSR[DS] = 1)
4:5	DAC2US	DAC 2 User/Supervisor 00 Both 01 Reserved 10 Supervisor only (MSR[PR] = 0) 11 User only (MSR[PR] = 1)
6:7	DAC2ER	DAC 2 Effective/Real 00 Effective (MSR[DS] = don't care) 01 Reserved 10 Virtual (MSR[DS] = 0) 11 Virtual (MSR[DS] = 1)

8:9	DAC12M	DAC 1/2 Mode 00 Exact match 01 Address bit mask 10 Range inclusive 11 Range exclusive	 Match if address[0:31] = DAC 1/2[0:31]; two independent compares Match if address = DAC1; only compare bits corresponding to 1 bits in DAC2 Match if DAC1 ≤ address < DAC2 Match if address < DAC1 OR address ≥ DAC2
10	DAC12A	DAC 1/2 Asynchronous 0 Debug interrupt caused by DAC1/2 exception will be synchronous 1 Debug interrupt caused by DAC1/2 exception will be asynchronous	
11		Reserved	
12:13	DVC1M	Data Value Compare (DVC) 1 Mode 00 Reserved 01 AND all bytes enabled by DVC1BE 10 OR all bytes enabled by DVC1BE 11 AND-OR pairs of bytes enabled by DVC1BE	 (0 AND 1) OR (2 AND 3)
14:15	DVC2M	DVC 2 Mode 00 Reserved 01 AND all bytes enabled by DVC2BE 10 OR all bytes enabled by DVC2BE 11 AND-OR pairs of bytes enabled by DVC2BE	 (0 AND 1) OR (2 AND 3)
16:19		Reserved	
20:23	DVC1BE	DVC 1 Byte Enables 0:3	
24:27		Reserved	
28:31	DVC2BE	DVC 2 Byte Enables 0:3	

SPR 0x3F3 Supervisor R/W

See *Debug Data Register (DBDR)* on page 459.

0	31

Figure 0-10. Debug Data Register (DBDR)

0:31		Debug Data

0	31

Figure 29-10. Debug Data Register (DBDR)

0:31		Debug Data

SPR 0x130 Supervisor Read/Clear

See *Debug Status Register (DBSR)* on page 456.

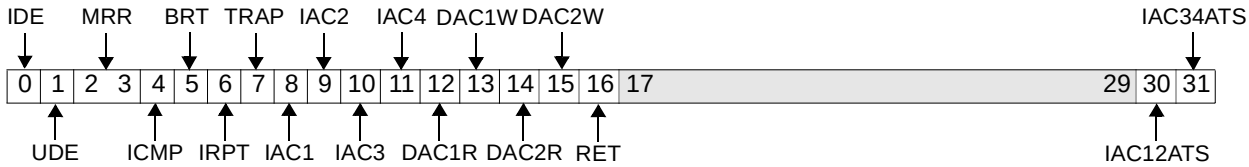


Figure 0-11. Debug Status Register (DBSR)

0	IDE	Imprecise Debug Event 0 Debug event ocurred while MSR[DE] = 1 1 Debug event occurred while MSR[DE] = 0	For synchronous debug events in internal debug mode, this field indicates whether the corresponding Debug interrupt occurs precisely or imprecisely
1	UDE	Unconditional Debug Event 0 Event didn't occur 1 Event occurred	
2:3	MRR	Most Recent Reset 00 No reset has occurred since this field was last cleared by software. 01 Core reset 10 Chip reset 11 System reset	This field is set upon any processor reset to a value indicating the type of reset.
4	ICMP	Instruction Completion Debug Event 0 Event didn't occur 1 Event occurred	
5	BRT	Branch Taken Debug Event 0 Event didn't occur 1 Event occurred	
6	IRPT	Interrupt Debug Event 0 Event didn't occur 1 Event occurred	
7	TRAP	Trap Debug Event 0 Event didn't occur 1 Event occurred	
8	IAC1	IAC 1 Debug Event 0 Event didn't occur 1 Event occurred	
9	IAC2	IAC 2 Debug Event 0 Event didn't occur 1 Event occurred	
10	IAC3	IAC 3 Debug Event 0 Event didn't occur 1 Event occurred	

11	IAC4	IAC 4 Debug Event 0 Event didn't occur 1 Event occurred
12	DAC1R	DAC 1 Read Debug Event 0 Event didn't occur 1 Event occurred
13	DAC1W	DAC 1 Write Debug Event 0 Event didn't occur 1 Event occurred
14	DAC2R	DAC 2 Read Debug Event 0 Event didn't occur 1 Event occurred
15	DAC2W	DAC 2 Write Debug Event 0 Event didn't occur 1 Event occurred
16	RET	Return Debug Event 0 Event didn't occur 1 Event occurred
17:29		Reserved
30	IAC12ATS	IAC 1/2 Auto-Toggle Status 0 Range is not reversed from value specified in DBCR1[IAC12M] 1 Range is reversed from value specified in DBCR1[IAC12M]
31	IAC34ATS	IAC 3/4 Auto-Toggle Status 0 Range is not reversed from value specified in DBCR1[IAC34M] 1 Range is reversed from value specified in DBCR1[IAC34M]

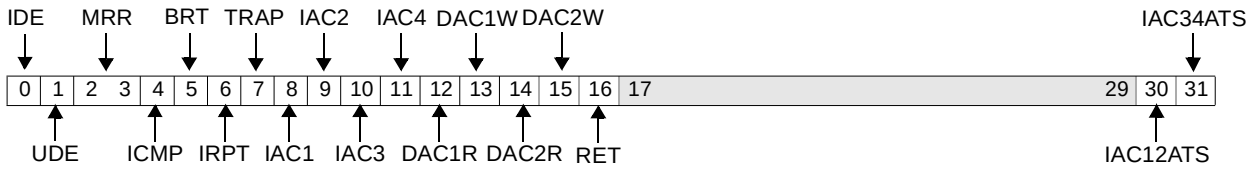


Figure 29-11. Debug Status Register (DBSR)

0	IDE	Imprecise Debug Event 0 Debug event ocurred while MSR[DE] = 1 1 Debug event occurred while MSR[DE] = 0	For synchronous debug events in internal debug mode, this field indicates whether the corresponding Debug interrupt occurs precisely or imprecisely
1	UDE	Unconditional Debug Event 0 Event didn't occur 1 Event occurred	

2:3	MRR	Most Recent Reset 00 No reset has occurred since this field was last cleared by software. 01 Core reset 10 Chip reset 11 System reset	This field is set upon any processor reset to a value indicating the type of reset.
4	ICMP	Instruction Completion Debug Event 0 Event didn't occur 1 Event occurred	
5	BRT	Branch Taken Debug Event 0 Event didn't occur 1 Event occurred	
6	IRPT	Interrupt Debug Event 0 Event didn't occur 1 Event occurred	
7	TRAP	Trap Debug Event 0 Event didn't occur 1 Event occurred	
8	IAC1	IAC 1 Debug Event 0 Event didn't occur 1 Event occurred	
9	IAC2	IAC 2 Debug Event 0 Event didn't occur 1 Event occurred	
10	IAC3	IAC 3 Debug Event 0 Event didn't occur 1 Event occurred	
11	IAC4	IAC 4 Debug Event 0 Event didn't occur 1 Event occurred	
12	DAC1R	DAC 1 Read Debug Event 0 Event didn't occur 1 Event occurred	
13	DAC1W	DAC 1 Write Debug Event 0 Event didn't occur 1 Event occurred	
14	DAC2R	DAC 2 Read Debug Event 0 Event didn't occur 1 Event occurred	
15	DAC2W	DAC 2 Write Debug Event 0 Event didn't occur 1 Event occurred	
16	RET	Return Debug Event 0 Event didn't occur 1 Event occurred	
17:29		Reserved	
30	IAC12ATS	IAC 1/2 Auto-Toggle Status 0 Range is not reversed from value specified in DBCR1[IAC12M] 1 Range is reversed from value specified in DBCR1[IAC12M]	

31	IAC34ATS	IAC 3/4 Auto-Toggle Status 0 Range is not reversed from value specified in DBCR1[IAC34M] 1 Range is reversed from value specified in DBCR1[IAC34M]

SPR 0x39D Supervisor Read-Only

See *dcread Operation* on page 231.

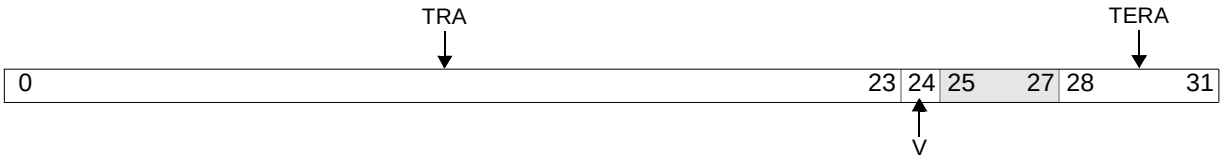


Figure 0-12. Data Cache Debug Tag Register High (DCDBTRH)

0:23	TRA	Tag Real Address	Bits 0:23 of the lower 32 bits of the 36-bit real address associated with the cache line read by **dcread**.
24	V	Cache Line Valid 0 Cache line is not valid. 1 Cache line is valid.	The valid indicator for the cache line read by **dcread**.
25:27		Reserved	
28:31	TERA	Tag Extended Real Address	Upper 4 bits of the 36-bit real address associated with the cache line read by **dcread**.

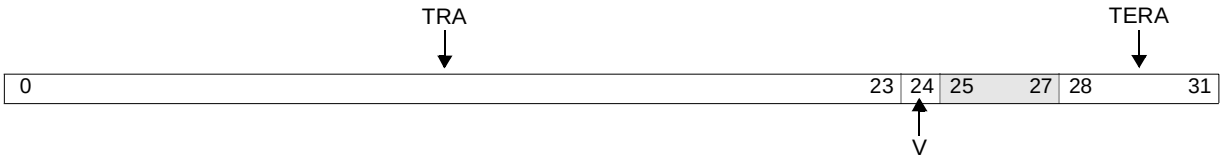


Figure 29-12. Data Cache Debug Tag Register High (DCDBTRH)

0:23	TRA	Tag Real Address	Bits 0:23 of the lower 32 bits of the 36-bit real address associated with the cache line read by **dcread**.
24	V	Cache Line Valid 0 Cache line is not valid. 1 Cache line is valid.	The valid indicator for the cache line read by **dcread**.
25:27		Reserved	
28:31	TERA	Tag Extended Real Address	Upper 4 bits of the 36-bit real address associated with the cache line read by **dcread**.

SPR 0x39C Supervisor Read-Only

See *dcread Operation* on page 231.

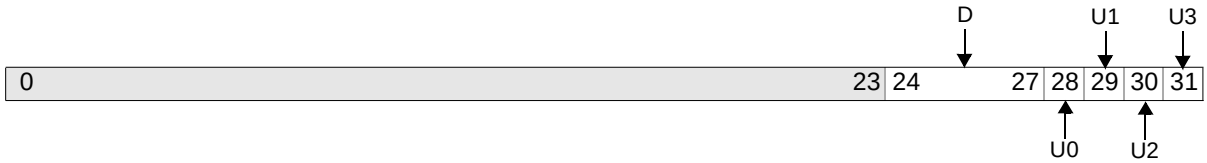


Figure 0-13. Data Cache Debug Tag Register Low (DCDBTRL)

0:23		Reserved	
24:27	D	Dirty Indicators	The "dirty" (modified) indicators for each of the four doublewords in the cache line read by **dcread**.
28	U0	U0 Storage Attribute	The U0 storage attribute for the memory page associated with this cache line read by **dcread**.
29	U1	U1 Storage Attribute	The U0 storage attribute for the memory page associated with this cache line read by **dcread**.
30	U2	U2 Storage Attribute	The U0 storage attribute for the memory page associated with this cache line read by **dcread**.
31	U3	U3 Storage Attribute	The U0 storage attribute for the memory page associated with this cache line read by **dcread**.

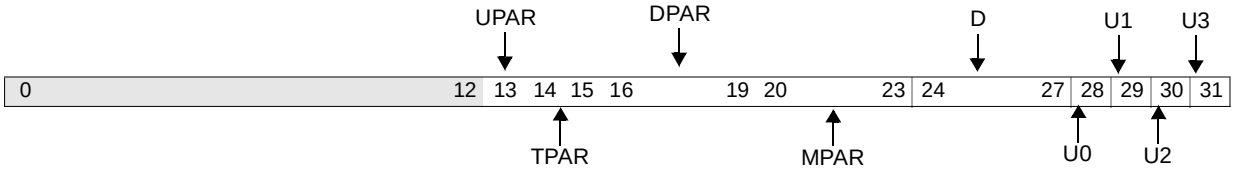


Figure 29-13. Data Cache Debug Tag Register Low (DCDBTRL)

0:12		Reserved	
13	UPAR	U bit parity	The parity for the U0-U3 bits in the cache line read by **dcread**
14:15	TPAR	Tag parity	The parity for the tag bits in the cache line read by **dcread**
16:19	DPAR	Data parity	The parity *check values* for the data bytes in the word read by **dcread**
20:23	MPAR	Modified (dirty) parity	The parity for the modified (dirty) indicators for each of the four doublewords in the cache line read by **dcread**.
24:27	D	Dirty Indicators	The "dirty" (modified) indicators for each of the four doublewords in the cache line read by **dcread**.

28	U0	U0 Storage Attribute	The U0 storage attribute for the memory page associated with this cache line read by **dcread**.
29	U1	U1 Storage Attribute	The U1 storage attribute for the memory page associated with this cache line read by **dcread**.
30	U2	U2 Storage Attribute	The U2 storage attribute for the memory page associated with this cache line read by **dcread**.
31	U3	U3 Storage Attribute	The U3 storage attribute for the memory page associated with this cache line read by **dcread**.

SPR 0x03D Supervisor R/W

See *Data Exception Address Register (DEAR)* on page 347.

Figure 0-14. Data Exception Address Register (DEAR)

0:31		Address of data exception for Data Storage, Alignment, and Data TLB Error interrupts

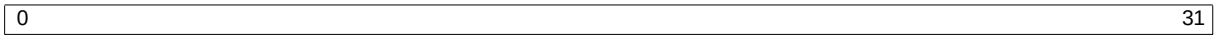


Figure 29-14. Data Exception Address Register (DEAR)

0:31		Address of data exception for Data Storage, Alignment, and Data TLB Error interrupts

SPR 0x016 Supervisor R/W

See *Decrementer (DEC)* on page 385.

Figure 0-15. Decrementer (DEC)

0:31		Decrement value

Figure 29-15. Decrementer (DEC)

0:31		Decrement value

SPR 0x036 Supervisor Write-Only

See *Decrementer (DEC)* on page 385.

Figure 0-16. Decrementer Auto-Reload (DECAR)

0:31		Decrementer auto-reload value	Copied to DEC at next time base clock when DEC = 1 and auto-reload is enabled (TCR[ARE] = 1).

Figure 29-16. Decrementer Auto-Reload (DECAR)

0:31		Decrementer auto-reload value	Copied to DEC at next time base clock when DEC = 1 and auto-reload is enabled (TCR[ARE] = 1).

SPR 0x390–0x393 Supervisor R/W

See *Cache Line Replacement Policy* on page 206.

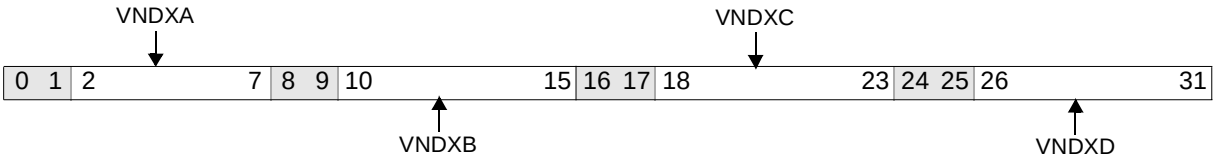


Figure 0-17. Data Cache Normal Victim Registers (DNV0–DNV3)

0:1		Reserved	
2:7	VNDXA	Victim Index A (for cache lines with EA[25:26] = 0b00)	
8:9		Reserved	
10:15	VNDXB	Victim Index A (for cache lines with EA[25:26] = 0b01)	
16:17		Reserved	
18:23	VNDXC	Victim Index A (for cache lines with EA[25:26] = 0b10)	
24:25		Reserved	
26:31	VNDXD	Victim Index A (for cache lines with EA[25:26] = 0b11)	

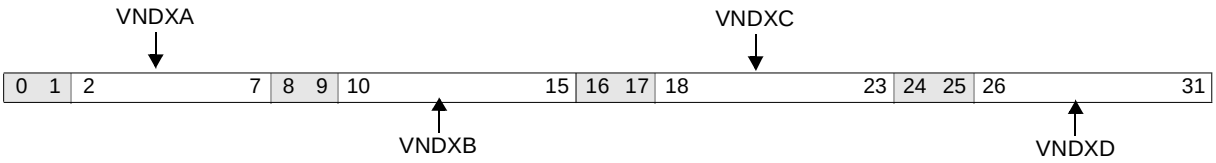


Figure 29-17. Data Cache Normal Victim Registers (DNV0–DNV3)

0:1		Reserved	
2:7	VNDXA	Victim Index A (for cache lines with EA[25:26] = 0b00)	
8:9		Reserved	
10:15	VNDXB	Victim Index A (for cache lines with EA[25:26] = 0b01)	
16:17		Reserved	
18:23	VNDXC	Victim Index A (for cache lines with EA[25:26] = 0b10)	
24:25		Reserved	
26:31	VNDXD	Victim Index A (for cache lines with EA[25:26] = 0b11)	

SPR 0x394–0x397 Supervisor R/W

See *Cache Line Replacement Policy* on page 206.

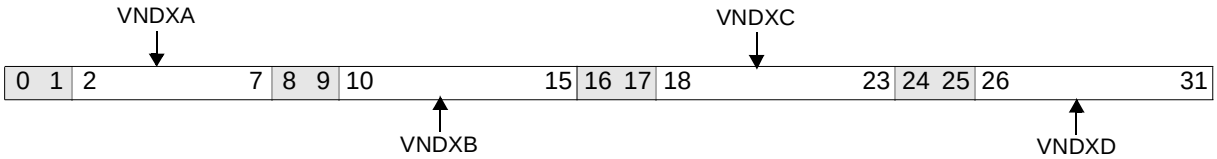


Figure 0-18. Data Cache Transient Victim Registers (DTV0–DTV3)

0:1		Reserved	
2:7	VNDXA	Victim Index A (for cache lines with EA[25:26] = 0b00)	
8:9		Reserved	
10:15	VNDXB	Victim Index A (for cache lines with EA[25:26] = 0b01)	
16:17		Reserved	
18:23	VNDXC	Victim Index A (for cache lines with EA[25:26] = 0b10)	
24:25		Reserved	
26:31	VNDXD	Victim Index A (for cache lines with EA[25:26] = 0b11)	

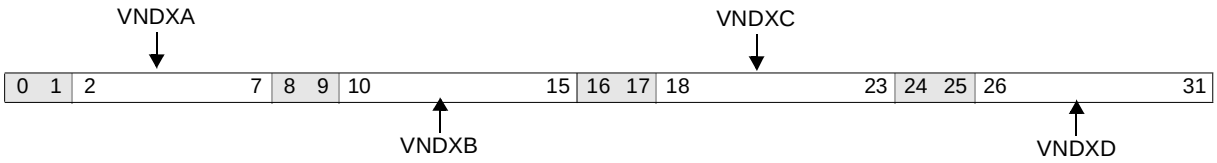


Figure 29-18. Data Cache Transient Victim Registers (DTV0–DTV3)

0:1		Reserved	
2:7	VNDXA	Victim Index A (for cache lines with EA[25:26] = 0b00)	
8:9		Reserved	
10:15	VNDXB	Victim Index A (for cache lines with EA[25:26] = 0b01)	
16:17		Reserved	
18:23	VNDXC	Victim Index A (for cache lines with EA[25:26] = 0b10)	
24:25		Reserved	
26:31	VNDXD	Victim Index A (for cache lines with EA[25:26] = 0b11)	

SPR 0x13E–0x13F Supervisor R/W

See *Data Value Compare Registers (DVC1–DVC2)* on page 458.

Figure 0-19. Data Value Compare Registers (DVC1–DVC2)

0:31		Data value to compare

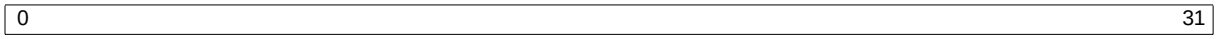


Figure 29-19. Data Value Compare Registers (DVC1–DVC2)

0:31		Data value to compare

SPR 0x398 Supervisor R/W

See *Cache Locking and Transient Mechanism* on page 207.

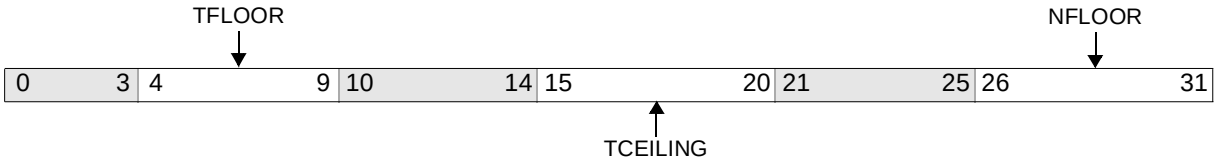


Figure 0-20. Data Cache Victim Limit (DVLIM)

0:3		Reserved	
4:9	TFLOOR	Transient Floor	
10:14		Reserved	
15:20	TCEILING	Transient Ceiling	
21:25		Reserved	
26:31	NFLOOR	Normal Floor	

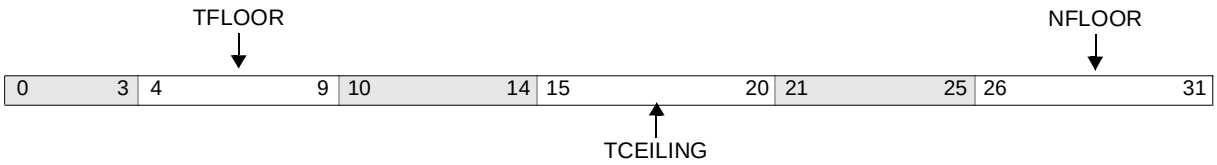


Figure 29-20. Data Cache Victim Limit (DVLIM)

0:3		Reserved	
4:9	TFLOOR	Transient Floor	
10:14		Reserved	
15:20	TCEILING	Transient Ceiling	
21:25		Reserved	
26:31	NFLOOR	Normal Floor	

SPR 0x03E Supervisor R/W

See *Exception Syndrome Register (ESR)* on page 349.

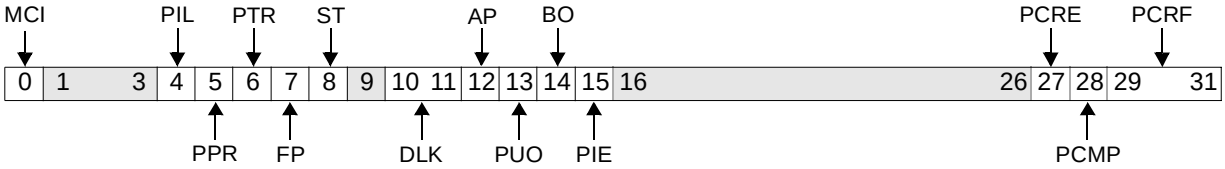


Figure 0-21. Exception Syndrome Register (ESR)

0	MCI	Machine Check—Instruction Fetch Exception 0 Instruction Machine Check exception did not occur. 1 Instruction Machine Check exception occurred.	This is an implementation-dependent field of the ESR and is not part of the PowerPC Book-E Architecture.
1:3		Reserved	
4	PIL	Program Interrupt—Illegal Instruction Exception 0 Illegal Instruction exception did not occur. 1 Illegal Instruction exception occurred.	
5	PPR	Program Interrupt—Privileged Instruction Exception 0 Privileged Instruction exception did not occur. 1 Privileged Instruction exception occurred.	
6	PTR	Program Interrupt—Trap Exception 0 Trap exception did not occur. 1 Trap exception occurred.	
7	FP	Floating Point Operation 0 Exception was not caused by a floating point instruction. 1 Exception was caused by a floating point instruction.	
8	ST	Store Operation 0 Exception was not caused by a store-type storage access or cache management instruction. 1 Exception was caused by a store-type storage access or cache management instruction.	
9		Reserved	

10:11	DLK	Data Storage Interrupt—Locking Exception 00 Locking exception did not occur. 01 Locking exception was caused by **dcbf**. 10 Locking exception was caused by **icbi**. 11 Reserved	
12	AP	AP Operation 0 Exception was not caused by an auxiliary processor instruction. 1 Exception was caused by an auxiliary processor instruction.	
13	PUO	Program Interrupt—Unimplemented Operation Exception 0 Unimplemented Operation exception did not occur. 1 Unimplemented Operation exception occurred.	
14	BO	Byte Ordering Exception 0 Byte Ordering exception did not occur. 1 Byte Ordering exception occurred.	
15	PIE	Program Interrupt—Imprecise Exception 0 Exception occurred precisely; SRR0 contains the address of the instruction that caused the exception. 1 Exception occurred imprecisely; SRR0 contains the address of an instruction after the one which caused the exception.	This field is only set for a Floating-Point Enabled exception type Program interrupt, and then only when the interrupt occurs imprecisely due to MSR[FE0,FE1] being set to a non-zero value when an attached floating-point unit is already signaling the Floating-Point Enabled exception (that is, FPSCR[FEX] is already 1).
16:26		Reserved	
27	PCRE	Program Interrupt—Condition Register Enable 0 Instruction which caused the exception is not a floating-point CR-updating instruction. 1 Instruction which caused the exception is a floating-point CR-updating instruction.	This is an implementation-dependent field of the ESR and is not part of the PowerPC Book-E Architecture. This field is only defined for a Floating-Point Enabled exception type Program interrupt, and then only when ESR[PIE] is 0.
28	PCMP	Program Interrupt—Compare 0 Instruction which caused the exception is not a floating-point compare type instruction 1 Instruction which caused the exception is a floating-point compare type instruction.	This is an implementation-dependent field of the ESR and is not part of the PowerPC Book-E Architecture. This field is only defined for a Floating-Point Enabled exception type Program interrupt, and then only when ESR[PIE] is 0.

29:31	PCRF	Program Interrupt—Condition Register Field If ESR[PCRE]=1, this field indicates which CR field was to be updated by the floating-point instruction which caused the exception.	This is an implementation-dependent field of the ESR and is not part of the PowerPC Book-E Architecture. This field is only defined for a Floating-Point Enabled exception type Program interrupt, and then only when ESR[PIE] is 0.

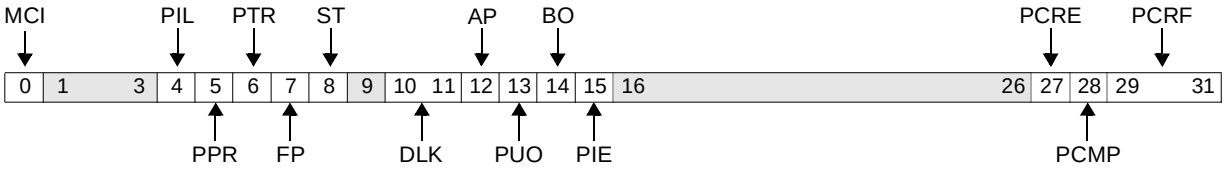


Figure 29-21. Exception Syndrome Register (ESR)

0	MCI	Machine Check—Instruction Fetch Exception 0 Instruction Machine Check exception did not occur. 1 Instruction Machine Check exception occurred.	This is an implementation-dependent field of the ESR and is not part of the PowerPC Book-E Architecture.
1:3		Reserved	
4	PIL	Program Interrupt—Illegal Instruction Exception 0 Illegal Instruction exception did not occur. 1 Illegal Instruction exception occurred.	
5	PPR	Program Interrupt—Privileged Instruction Exception 0 Privileged Instruction exception did not occur. 1 Privileged Instruction exception occurred.	
6	PTR	Program Interrupt—Trap Exception 0 Trap exception did not occur. 1 Trap exception occurred.	
7	FP	Floating Point Operation 0 Exception was not caused by a floating point instruction. 1 Exception was caused by a floating point instruction.	
8	ST	Store Operation 0 Exception was not caused by a store-type storage access or cache management instruction. 1 Exception was caused by a store-type storage access or cache management instruction.	
9		Reserved	
10:11	DLK	Data Storage Interrupt—Locking Exception 00 Locking exception did not occur. 01 Locking exception was caused by **dcbf**. 10 Locking exception was caused by **icbi**. 11 Reserved	

12	AP	AP Operation 0 Exception was not caused by an auxiliary processor instruction. 1 Exception was caused by an auxiliary processor instruction.	
13	PUO	Program Interrupt—Unimplemented Operation Exception 0 Unimplemented Operation exception did not occur. 1 Unimplemented Operation exception occurred.	
14	BO	Byte Ordering Exception 0 Byte Ordering exception did not occur. 1 Byte Ordering exception occurred.	
15	PIE	Program Interrupt—Imprecise Exception 0 Exception occurred precisely; SRR0 contains the address of the instruction that caused the exception. 1 Exception occurred imprecisely; SRR0 contains the address of an instruction after the one which caused the exception.	This field is only set for a Floating-Point Enabled exception type Program interrupt, and then only when the interrupt occurs imprecisely due to MSR[FE0,FE1] being set to a non-zero value when an attached floating-point unit is already signaling the Floating-Point Enabled exception (that is, FPSCR[FEX] is already 1).
16:26		Reserved	
27	PCRE	Program Interrupt—Condition Register Enable 0 Instruction which caused the exception is not a floating-point CR-updating instruction. 1 Instruction which caused the exception is a floating-point CR-updating instruction.	This is an implementation-dependent field of the ESR and is not part of the PowerPC Book-E Architecture. This field is only defined for a Floating-Point Enabled exception type Program interrupt, and then only when ESR[PIE] is 0.
28	PCMP	Program Interrupt—Compare 0 Instruction which caused the exception is not a floating-point compare type instruction 1 Instruction which caused the exception is a floating-point compare type instruction.	This is an implementation-dependent field of the ESR and is not part of the PowerPC Book-E Architecture. This field is only defined for a Floating-Point Enabled exception type Program interrupt, and then only when ESR[PIE] is 0.
29:31	PCRF	Program Interrupt—Condition Register Field If ESR[PCRE]=1, this field indicates which CR field was to be updated by the floating-point instruction which caused the exception.	This is an implementation-dependent field of the ESR and is not part of the PowerPC Book-E Architecture. This field is only defined for a Floating-Point Enabled exception type Program interrupt, and then only when ESR[PIE] is 0.

User R/W

See *General Purpose Registers (GPRs)* on page 192.

Figure 0-22. General Purpose Registers (R0-R31)

0:31		General Purpose Register data

Figure 29-22. General Purpose Registers (R0-R31)

0:31		General Purpose Register data

SPR 0x138–0x13B Supervisor R/W

See *Instruction Address Compare Registers (IAC1–IAC4)* on page 457.

Figure 0-23. Instruction Address Compare Registers (IAC1–IAC4)

0:29		Instruction Address Compare (IAC) word address
30:31		Reserved

Figure 29-23. Instruction Address Compare Registers (IAC1–IAC4)

0:29		Instruction Address Compare (IAC) word address
30:31		Reserved

SPR 0x3D3 Supervisor Read-Only

See *icread Operation* on page 218.

Figure 0-24. Instruction Cache Debug Data Register (ICDBDR)

0:31		Instruction machine code from instruction cache

Figure 29-24. Instruction Cache Debug Data Register (ICDBDR)

0:31		Instruction machine code from instruction cache

SPR 0x39F Supervisor Read-Only

See *icread Operation* on page 218.

Figure 0-25. Instruction Cache Debug Tag Register High (ICDBTRH)

0:23		Tag Effective Address	Bits 0:23 of the 32-bit effective address associated with the cache line read by **icread**.
24	V	Cache Line Valid 0 Cache line is not valid. 1 Cache line is valid.	The valid indicator for the cache line read by **icread**.
25:31		Reserved	

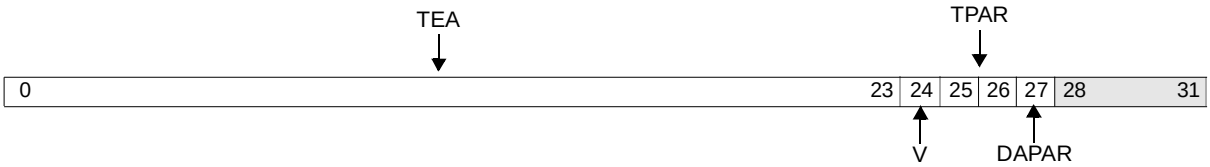


Figure 29-25. Instruction Cache Debug Tag Register High (ICDBTRH)

0:23		Tag Effective Address	Bits 0:23 of the 32-bit effective address associated with the cache line read by **icread**.
24	V	Cache Line Valid 0 Cache line is not valid. 1 Cache line is valid.	The valid indicator for the cache line read by **icread**.
25:26	TPAR	Tag Parity	The parity bits for the address tag for the cache line read by **icread**, if CCR0[CRPE] is set.
27	DAPAR	Instruction Data parity	The parity bit for the instruction word at the 32-bit effective address specified in the **icread** instruction, if CCR0[CRPE] is set.
28:31		Reserved	

SPR 0x39E Supervisor Read-Only

See *icread Operation* on page 218.

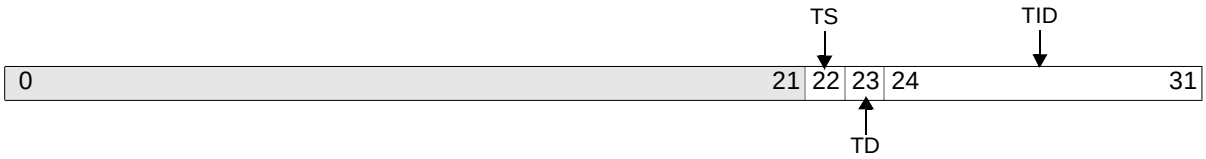


Figure 0-26. Instruction Cache Debug Tag Register Low (ICDBTRL)

0:21		Reserved	
22	TS	Translation Space	The address space portion of the virtual address associated with the cache line read by **icread**.
23	TD	Translation ID (TID) Disable 0 TID enable 1 TID disable	TID Disable field for the memory page associated with the cache line read by **icread**.
24:31	TID	Translation ID	TID field portion of the virtual address associated with the cache line read by **icread**.

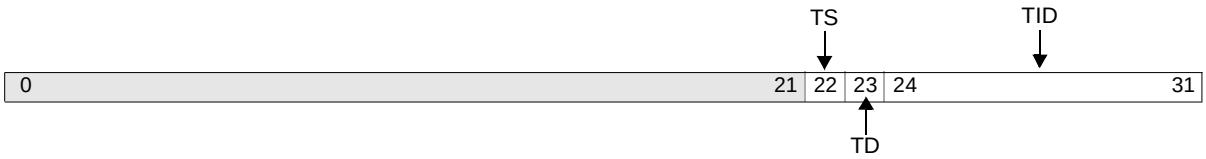


Figure 29-26. Instruction Cache Debug Tag Register Low (ICDBTRL)

0:21		Reserved	
22	TS	Translation Space	The address space portion of the virtual address associated with the cache line read by **icread**.
23	TD	Translation ID (TID) Disable 0 TID enable 1 TID disable	TID Disable field for the memory page associated with the cache line read by **icread**.
24:31	TID	Translation ID	TID field portion of the virtual address associated with the cache line read by **icread**.

SPR 0x370–0x373 Supervisor R/W

See *Cache Line Replacement Policy* on page 206.

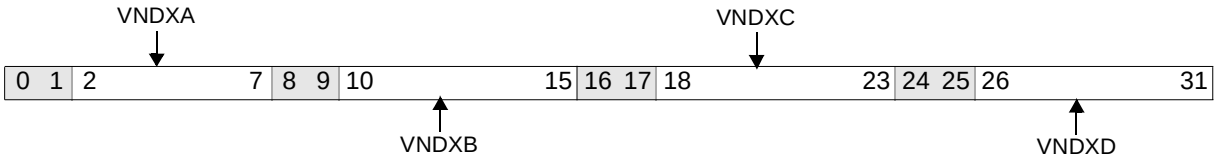


Figure 0-27. Instruction Cache Normal Victim Registers (INV0–INV3)

0:1		Reserved	
2:7	VNDXA	Victim Index A (for cache lines with EA[25:26] = 0b00)	
8:9		Reserved	
10:15	VNDXB	Victim Index A (for cache lines with EA[25:26] = 0b01)	
16:17		Reserved	
18:23	VNDXC	Victim Index A (for cache lines with EA[25:26] = 0b10)	
24:25		Reserved	
26:31	VNDXD	Victim Index A (for cache lines with EA[25:26] = 0b11)	

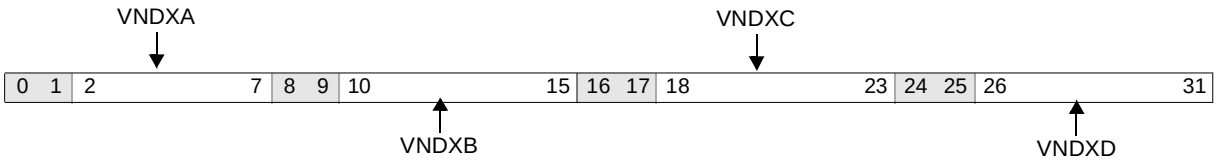


Figure 29-27. Instruction Cache Normal Victim Registers (INV0–INV3)

0:1		Reserved	
2:7	VNDXA	Victim Index A (for cache lines with EA[25:26] = 0b00)	
8:9		Reserved	
10:15	VNDXB	Victim Index A (for cache lines with EA[25:26] = 0b01)	
16:17		Reserved	
18:23	VNDXC	Victim Index A (for cache lines with EA[25:26] = 0b10)	
24:25		Reserved	
26:31	VNDXD	Victim Index A (for cache lines with EA[25:26] = 0b11)	

SPR 0x374–0x377 Supervisor R/W

See *Cache Line Replacement Policy* on page 206.

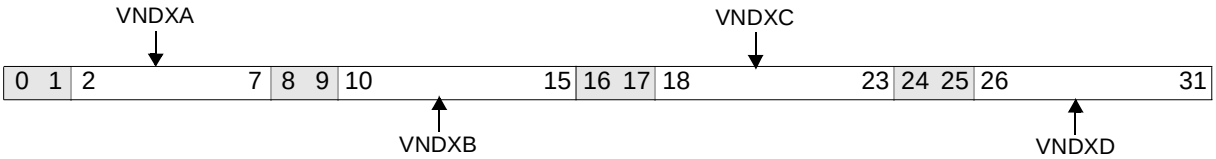


Figure 0-28. Instruction Cache Transient Victim Registers (ITV0–ITV3)

0:1		Reserved	
2:7	VNDXA	Victim Index A (for cache lines with EA[25:26] = 0b00)	
8:9		Reserved	
10:15	VNDXB	Victim Index A (for cache lines with EA[25:26] = 0b01)	
16:17		Reserved	
18:23	VNDXC	Victim Index A (for cache lines with EA[25:26] = 0b10)	
24:25		Reserved	
26:31	VNDXD	Victim Index A (for cache lines with EA[25:26] = 0b11)	

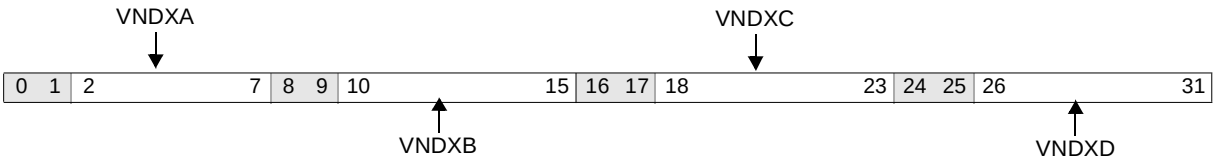


Figure 29-28. Instruction Cache Transient Victim Registers (ITV0–ITV3)

0:1		Reserved	
2:7	VNDXA	Victim Index A (for cache lines with EA[25:26] = 0b00)	
8:9		Reserved	
10:15	VNDXB	Victim Index A (for cache lines with EA[25:26] = 0b01)	
16:17		Reserved	
18:23	VNDXC	Victim Index A (for cache lines with EA[25:26] = 0b10)	
24:25		Reserved	
26:31	VNDXD	Victim Index A (for cache lines with EA[25:26] = 0b11)	

SPR 0x399 Supervisor R/W

See *Cache Locking and Transient Mechanism* on page 207.

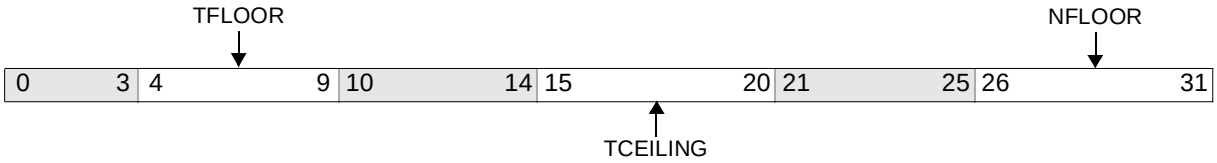


Figure 0-29. Instruction Cache Victim Limit (IVLIM)

0:3		Reserved	
4:9	TFLOOR	Transient Floor	
10:14		Reserved	
15:20	TCEILING	Transient Ceiling	
21:25		Reserved	
26:31	NFLOOR	Normal Floor	

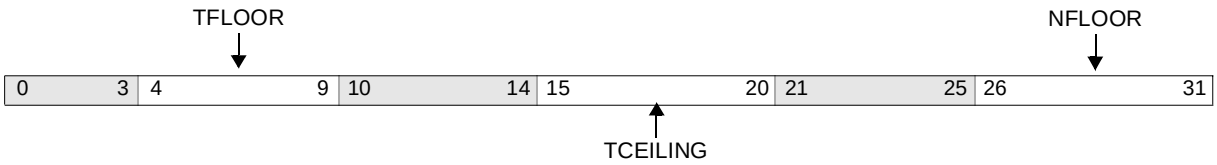


Figure 29-29. Instruction Cache Victim Limit (IVLIM)

0:3		Reserved	
4:9	TFLOOR	Transient Floor	
10:14		Reserved	
15:20	TCEILING	Transient Ceiling	
21:25		Reserved	
26:31	NFLOOR	Normal Floor	

SPR 0x190–0x19F Supervisor R/W

See *Interrupt Vector Offset Registers (IVOR0–IVOR15)* on page 347.

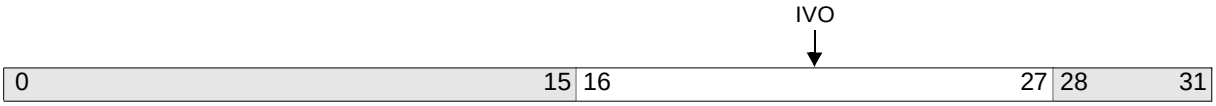


Figure 0-30. Interrupt Vector Offset Registers (IVOR0–IVOR15)

0:15		Reserved
16:27	IVO	Interrupt Vector Offset
28:31		Reserved

Table 0-1. Interrupt Types Associated with each IVOR

IVOR	Interrupt Type
IVOR0	Critical Input
IVOR1	Machine Check
IVOR2	Data Storage
IVOR3	Instruction Storage
IVOR4	External Input
IVOR5	Alignment
IVOR6	Program
IVOR7	Floating Point Unavailable
IVOR8	System Call
IVOR9	Auxiliary Processor Unavailable
IVOR10	Decrementer
IVOR11	Fixed Interval Timer
IVOR12	Watchdog Timer
IVOR13	Data TLB Error
IVOR14	Instruction TLB Error
IVOR15	Debug

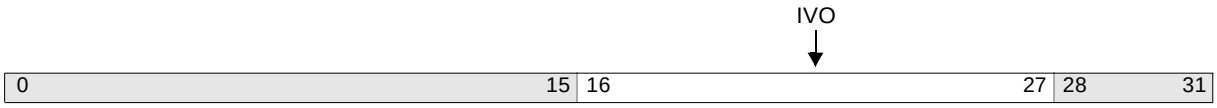


Figure 29-30. Interrupt Vector Offset Registers (IVOR0–IVOR15)

0:15		Reserved
16:27	IVO	Interrupt Vector Offset
28:31		Reserved

Table 29-16. Interrupt Types Associated with each IVOR

IVOR	Interrupt Type
IVOR0	Critical Input
IVOR1	Machine Check
IVOR2	Data Storage
IVOR3	Instruction Storage
IVOR4	External Input
IVOR5	Alignment
IVOR6	Program
IVOR7	Floating Point Unavailable
IVOR8	System Call
IVOR9	Auxiliary Processor Unavailable
IVOR10	Decrementer
IVOR11	Fixed Interval Timer
IVOR12	Watchdog Timer
IVOR13	Data TLB Error
IVOR14	Instruction TLB Error
IVOR15	Debug

SPR 0x03F Supervisor R/W

See *Interrupt Vector Prefix Register (IVPR)* on page 348.

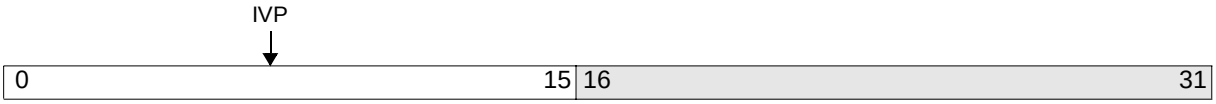


Figure 0-31. Interrupt Vector Prefix Register (IVPR)

0:15	IVP	Interrupt Vector Prefix
16:31		Reserved

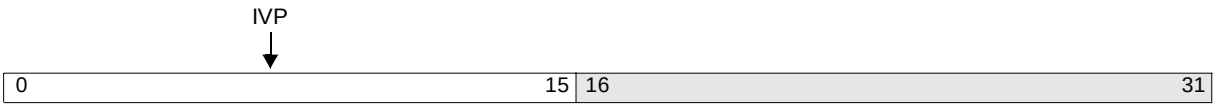


Figure 29-31. Interrupt Vector Prefix Register (IVPR)

0:15	IVP	Interrupt Vector Prefix
16:31		Reserved

SPR 0x008 User R/W

See *Link Register (LR)* on page 188.

Figure 0-32. Link Register (LR)

0:31		Link Register contents	Target address of **bclr** instruction

Figure 29-32. Link Register (LR)

0:31		Link Register contents	Target address of **bclr** instruction

SPR 0x3B2 Supervisor R/W

See *Memory Management Unit Control Register (MMUCR)* on page 247.

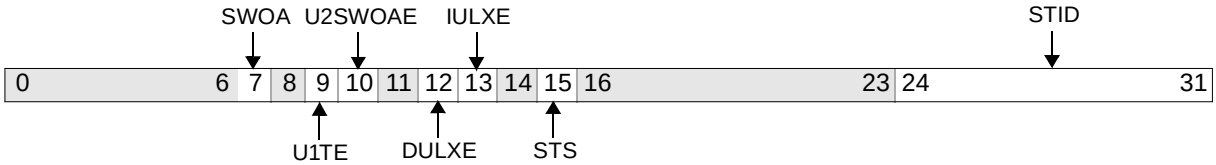


Figure 0-33. Memory Management Unit Control Register (MMUCR)

0:6		Reserved	
7	SWOA	Store Without Allocate 0 Cacheable store misses allocate a line in the data cache. 1 Cacheable store misses do not allocate a line in the data cache.	If MMUCR[U2SWOAE] = 1, this field is ignored.
8		Reserved	
9	U1TE	U1 Transient Enable 0 Disable U1 storage attribute as transient storage attribute. 1 Enable U1 storage attribute as transient storage attribute.	
10	U2SWOAE	U2 Store without Allocate Enable 0 Disable U2 storage attribute control of store without allocate. 1 Enable U2 storage attribute control of store without allocate.	If MMUCR[U2SWOAE] = 1, the U2 storage attribute overrides MMUCR[SWOA].
11		Reserved	
12	DULXE	Data Cache Unlock Exception Enable 0 Data cache unlock exception is disabled. 1 Data cache unlock exception is enabled.	**dcbf** in user mode causes a Cache Locking exception type Data Storage interrupt when MMUCR[DULXE] is 1.
13	IULXE	Instruction Cache Unlock Exception Enable 0 Instruction cache unlock exception is disabled. 1 Instruction cache unlock exception is enabled.	**icbi** in user mode causes a Cache Locking exception type Data Storage interrupt when MMUCR[IULXE] is 1.
14		Reserved	
15	STS	Search Translation Space	Specifies the value of the translation space (TS) field for **tlbsx[.]**
16:23		Reserved	
24:31	STID	Search Translation ID	Specifies the value of the TID field for the **tlbsx[.]**; also used to transfer a TLB entry's TID value for **tlbre** and **tlbwe** .

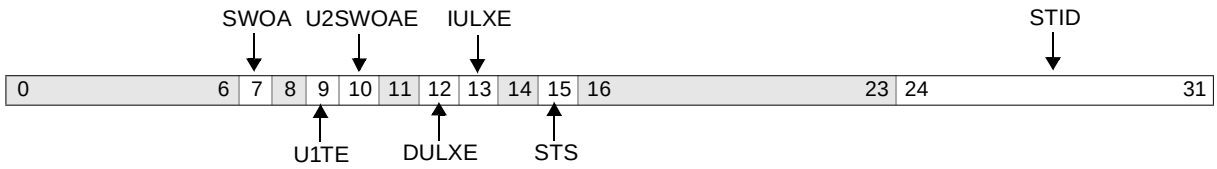


Figure 29-33. Memory Management Unit Control Register (MMUCR)

0:6		Reserved	
7	SWOA	Store Without Allocate 0 Cacheable store misses allocate a line in the data cache. 1 Cacheable store misses do not allocate a line in the data cache.	If MMUCR[U2SWOAE] = 1, this field is ignored.
8		Reserved	
9	U1TE	U1 Transient Enable 0 Disable U1 storage attribute as transient storage attribute. 1 Enable U1 storage attribute as transient storage attribute.	
10	U2SWOAE	U2 Store without Allocate Enable 0 Disable U2 storage attribute control of store without allocate. 1 Enable U2 storage attribute control of store without allocate.	If MMUCR[U2SWOAE] = 1, the U2 storage attribute overrides MMUCR[SWOA].
11		Reserved	
12	DULXE	Data Cache Unlock Exception Enable 0 Data cache unlock exception is disabled. 1 Data cache unlock exception is enabled.	**dcbf** in user mode causes a Cache Locking exception type Data Storage interrupt when MMUCR[DULXE] is 1.
13	IULXE	Instruction Cache Unlock Exception Enable 0 Instruction cache unlock exception is disabled. 1 Instruction cache unlock exception is enabled.	**icbi** in user mode causes a Cache Locking exception type Data Storage interrupt when MMUCR[IULXE] is 1.
14		Reserved	
15	STS	Search Translation Space	Specifies the value of the translation space (TS) field for **tlbsx[.]**
16:23		Reserved	
24:31	STID	Search Translation ID	Specifies the value of the TID field for the **tlbsx[.]**; also used to transfer a TLB entry's TID value for **tlbre** and **tlbwe** .

Supervisor R/W

See *Machine State Register (MSR)* on page 343.

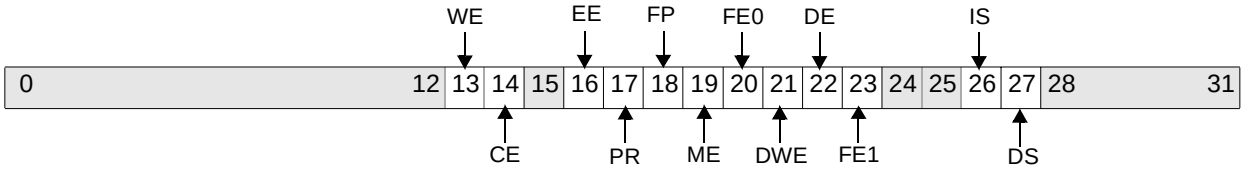


Figure 0-34. Machine State Register (MSR)

0:12		Reserved	
13	WE	Wait State Enable 0 The processor is not in the wait state. 1 The processor is in the wait state.	If MSR[WE] = 1, the processor remains in the wait state until an interrupt is taken, a reset occurs, or an external debug tool clears WE.
14	CE	Critical Interrupt Enable 0 Critical Input and Watchdog Timer interrupts are disabled. 1 Critical Input and Watchdog Timer interrupts are enabled.	
15		Reserved	
16	EE	External Interrupt Enable 0 External Input, Decrementer, and Fixed Interval Timer interrupts are disabled. 1 External Input, Decrementer, and Fixed Interval Timer interrupts are enabled.	
17	PR	Problem State 0 Supervisor state (privileged instructions can be executed) 1 Problem state (privileged instructions can not be executed)	
18	FP	Floating Point Available 0 The processor cannot execute floating-point instructions 1 The processor can execute floating-point instructions	
19	ME	Machine Check Enable 0 Machine Check interrupts are disabled 1 Machine Check interrupts are enabled.	
20	FE0	Floating-point exception mode 0 0 If MSR[FE1] = 0, ignore exceptions mode; if MSR[FE1] = 1, imprecise nonrecoverable mode 1 If MSR[FE1] = 0, imprecise recoverable mode; if MSR[FE1] = 1, precise mode	

21	DWE	Debug Wait Enable 0 Disable debug wait mode. 1 Enable debug wait mode.
22	DE	Debug interrupt Enable 0 Debug interrupts are disabled. 1 Debug interrupts are enabled.
23	FE1	Floating-point exception mode 1 0 If MSR[FE0] = 0, ignore exceptions mode; if MSR[FE0] = 1, imprecise recoverable mode 1 If MSR[FE0] = 0, imprecise non-recoverable mode; if MSR[FE0] = 1, precise mode
24:25		Reserved
26	IS	Instruction Address Space 0 All instruction storage accesses are directed to address space 0 (TS = 0 in the relevant TLB entry). 1 All instruction storage accesses are directed to address space 1 (TS = 1 in the relevant TLB entry).
27	DS	Data Address Space 0 All data storage accesses are directed to address space 0 (TS = 0 in the relevant TLB entry). 1 All data storage accesses are directed to address space 1 (TS = 1 in the relevant TLB entry).
28:31		Reserved

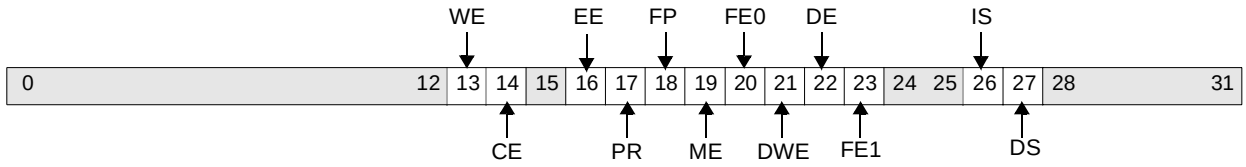


Figure 29-34. Machine State Register (MSR)

0:12		Reserved	
13	WE	Wait State Enable 0 The processor is not in the wait state. 1 The processor is in the wait state.	If MSR[WE] = 1, the processor remains in the wait state until an interrupt is taken, a reset occurs, or an external debug tool clears WE.
14	CE	Critical Interrupt Enable 0 Critical Input and Watchdog Timer interrupts are disabled. 1 Critical Input and Watchdog Timer interrupts are enabled.	
15		Reserved	

16	EE	External Interrupt Enable 0 External Input, Decrementer, and Fixed Interval Timer interrupts are disabled. 1 External Input, Decrementer, and Fixed Interval Timer interrupts are enabled.
17	PR	Problem State 0 Supervisor state (privileged instructions can be executed) 1 Problem state (privileged instructions can not be executed)
18	FP	Floating Point Available 0 The processor cannot execute floating-point instructions 1 The processor can execute floating-point instructions
19	ME	Machine Check Enable 0 Machine Check interrupts are disabled 1 Machine Check interrupts are enabled.
20	FE0	Floating-point exception mode 0 0 If MSR[FE1] = 0, ignore exceptions mode; if MSR[FE1] = 1, imprecise nonrecoverable mode 1 If MSR[FE1] = 0, imprecise recoverable mode; if MSR[FE1] = 1, precise mode
21	DWE	Debug Wait Enable 0 Disable debug wait mode. 1 Enable debug wait mode.
22	DE	Debug interrupt Enable 0 Debug interrupts are disabled. 1 Debug interrupts are enabled.
23	FE1	Floating-point exception mode 1 0 If MSR[FE0] = 0, ignore exceptions mode; if MSR[FE0] = 1, imprecise recoverable mode 1 If MSR[FE0] = 0, imprecise non-recoverable mode; if MSR[FE0] = 1, precise mode
24:25		Reserved
26	IS	Instruction Address Space 0 All instruction storage accesses are directed to address space 0 (TS = 0 in the relevant TLB entry). 1 All instruction storage accesses are directed to address space 1 (TS = 1 in the relevant TLB entry).
27	DS	Data Address Space 0 All data storage accesses are directed to address space 0 (TS = 0 in the relevant TLB entry). 1 All data storage accesses are directed to address space 1 (TS = 1 in the relevant TLB entry).
28:31		Reserved

SPR 0x030 Supervisor R/W

See *Process ID (PID)* on page 251.

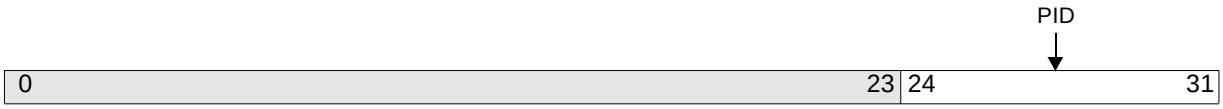


Figure 0-35. Process ID (PID)

0:23		Reserved
24:31	PID	Process ID

Figure 29-35. Process ID (PID)

0:23		Reserved
24:31	PID	Process ID

SPR 0x11E Supervisor Read-Only

See *Processor Identification Register (PIR)* on page 196.

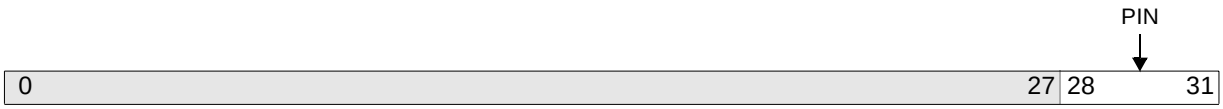


Figure 0-36. Processor Identification Register (PIR)

0:27		Reserved
28:31	PIN	Processor Identification Number (PIN)

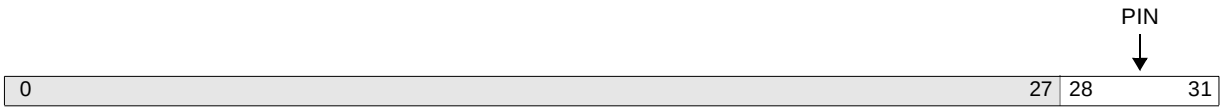


Figure 29-36. Processor Identification Register (PIR)

0:27		Reserved
28:31	PIN	Processor Identification Number (PIN)

SPR 0x11F Supervisor Read-Only

See *Processor Version Register (PVR)* on page 196.

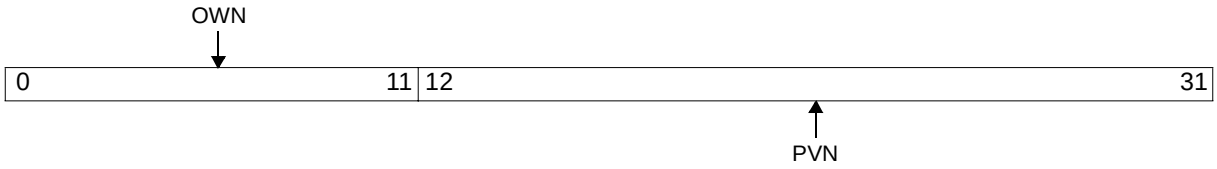


Figure 0-37. Processor Version Register (PVR)

0:31		Processor Version	Refer to *PowerPC 440GP Embedded Processor Data Sheet* for the PVR value.

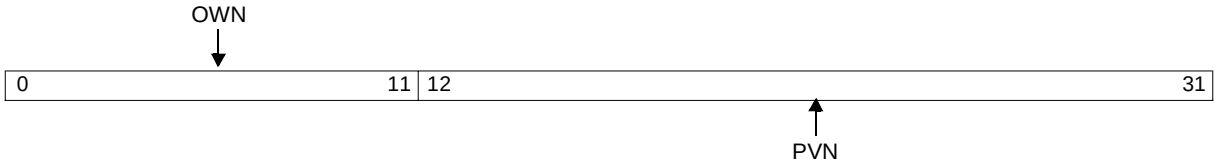


Figure 29-37. Processor Version Register (PVR)

0:31		Processor Version	Refer to *PowerPC 440GP Embedded Processor Data Sheet* for the PVR value.

SPR 39B Supervisor Read-Only

See *Reset Configuration (RSTCFG)* on page 198.

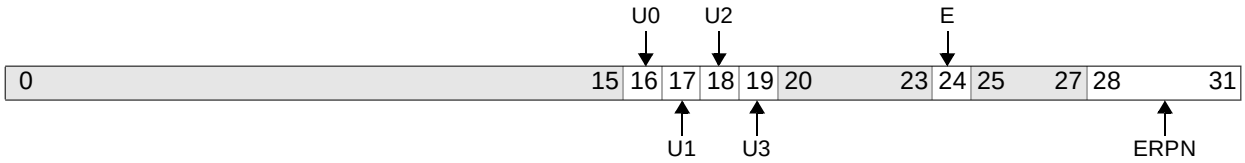


Figure 0-38. Reset Configuration

0:15		Reserved	
16	U0	U0 Storage Attribute 0 U0 storage attribute is disabled 1 U0 storage attribute is enabled	U0 has no effect in the PPC440GP.
17	U1	U1 Storage Attribute 0 Memory page contains normal instructions and data 1 Memory page contains transient instructions or data	
18	U2	U2 Storage Attribute 0 A storage miss does not cause a line to be allocated in the data cache 1 A storage miss causes a line to be allocated in the data cache	
19	U3	U3 Storage Attribute 0 U3 storage attribute is disabled 1 U3 storage attribute is enabled	U3 has no effect in the PPC440GP.
20:23		Reserved	
24	E	E Storage Attribute 0 Accesses to the page are big endian. 1 Accesses to the page are little endian.	
25:27		Reserved	
28:31	ERPN	Extended Real Page Number	f ROM is connected to EBCO, ERPN is 0b0001. If ROM is connected to PCI, ERPN is 0b0010.

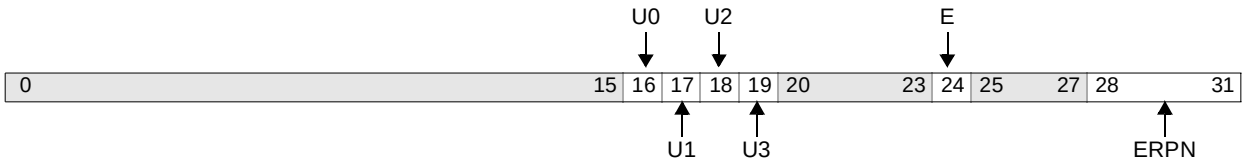


Figure 29-38. Reset Configuration

0:15		Reserved	
16	U0	U0 Storage Attribute 0 U0 storage attribute is disabled 1 U0 storage attribute is enabled	U0 has no effect in the PPC440GP.
17	U1	U1 Storage Attribute 0 Memory page contains normal instructions and data 1 Memory page contains transient instructions or data	
18	U2	U2 Storage Attribute 0 A storage miss does not cause a line to be allocated in the data cache 1 A storage miss causes a line to be allocated in the data cache	
19	U3	U3 Storage Attribute 0 U3 storage attribute is disabled 1 U3 storage attribute is enabled	U3 has no effect in the PPC440GP.
20:23		Reserved	
24	E	E Storage Attribute 0 Accesses to the page are big endian. 1 Accesses to the page are little endian.	
25:27		Reserved	
28:31	ERPN	Extended Real Page Number	f ROM is connected to EBCO, ERPN is 0b0001. If ROM is connected to PCI, ERPN is 0b0010.

SPR 0x104–0x107 (User/Supervisor Read-Only); SPR 0x110–0x113 (Supervisor R/W); SPR 0x114–0x117 (Supervisor Write-Only)

See *Special Purpose Registers General (USPRG0, SPRG0–SPRG7)* on page 195.

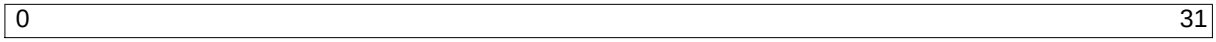


Figure 0-39. Special Purpose Registers General (SPRG0–SPRG7)

0:31		General data	Software value; no hardware usage.

Figure 29-39. Special Purpose Registers General (SPRG0–SPRG7)

0:31		General data	Software value; no hardware usage.

SPR 0x01A Supervisor R/W

See *Save/Restore Register 0 (SRR0)* on page 345.

Figure 0-40. Save/Restore Register 0 (SRR0)

0:29		Return address for non-critical interrupts
30:31		Reserved

Figure 29-40. Save/Restore Register 0 (SRR0)

0:29		Return address for non-critical interrupts
30:31		Reserved

SPR 0x01B Supervisor R/W

See *Save/Restore Register 1 (SRR1)* on page 345.

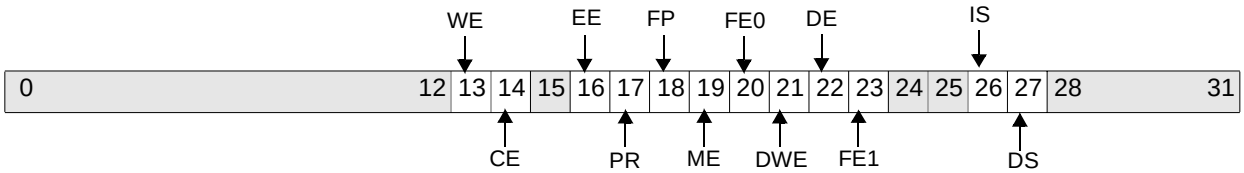


Figure 0-41. Save/Restore Register 1 (SRR1)

0:31	Copy of the MSR at the time of a non-critical interrupt.

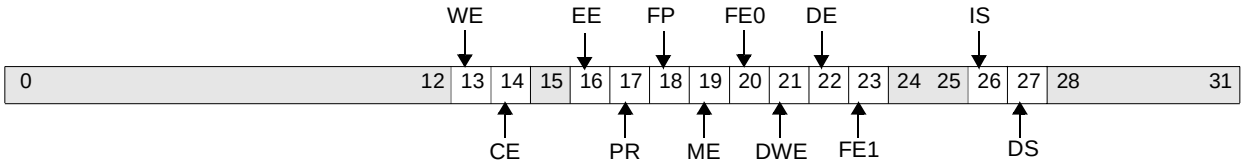


Figure 29-41. Save/Restore Register 1 (SRR1)

0:31	Copy of the MSR at the time of a non-critical interrupt.

SPR 0x10C (User/Supervisor Read-Only); SPR 0x11C (Supervisor Write-Only)

See *Time Base* on page 384.

Figure 0-42. Time Base Lower (TBL)

0:31		Time Base Lower	Low-order 32 bits of time base.

Figure 29-42. Time Base Lower (TBL)

0:31		Time Base Lower	Low-order 32 bits of time base.

SPR 0x10D (User/Supervisor Read-Only); SPR 0x11D (Supervisor Write-Only)

See *Time Base* on page 384.

Figure 0-43. Time Base Upper (TBU)

0:31		Time Base Upper	High-order 32 bits of time base.

Figure 29-43. Time Base Upper (TBU)

0:31		Time Base Upper	High-order 32 bits of time base.

SPR 0x154 Supervisor R/W

See *Timer Control Register (TCR)* on page 389.

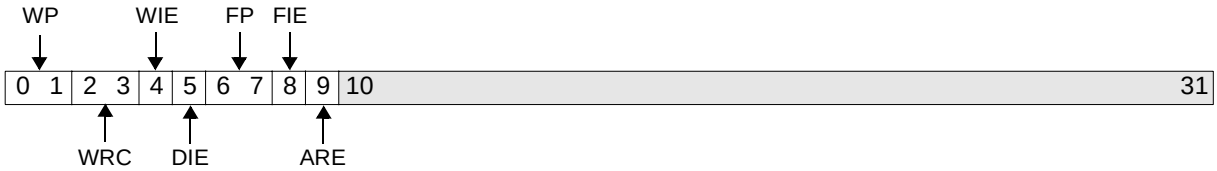


Figure 0-44. Timer Control Register (TCR)

0:1	WP	Watchdog Timer Period 00 2^{21} time base clocks 01 2^{25} time base clocks 10 2^{29} time base clocks 11 2^{33} time base clocks	
2:3	WRC	Watchdog Timer Reset Control 00 No Watchdog Timer reset will occur. 01 Core reset 10 Chip reset 11 System reset	TCR[WRC] resets to 0b00. Type of reset to cause upon Watchdog Timer exception with TSR[ENW,WIS]=0b11. This field can be set by software, but cannot be cleared by software, except by a software-induced reset.
4	WIE	Watchdog Timer Interrupt Enable 0 Disable Watchdog Timer interrupt. 1 Enable Watchdog Timer interrupt.	
5	DIE	Decrementer Interrupt Enable 0 Disable Decrementer interrupt. 1 Enable Decrementer interrupt.	
6:7	FP	Fixed Interval Timer (FIT) Period 00 2^{13} time base clocks 01 2^{17} time base clocks 10 2^{21} time base clocks 11 2^{25} time base clocks	
8	FIE	FIT Interrupt Enable 0 Disable Fixed Interval Timer interrupt. 1 Enable Fixed Interval Timer interrupt.	
9	ARE	Auto-Reload Enable 0 Disable auto reload. 1 Enable auto reload.	TCR[ARE] resets to 0b0.
10:31		Reserved	

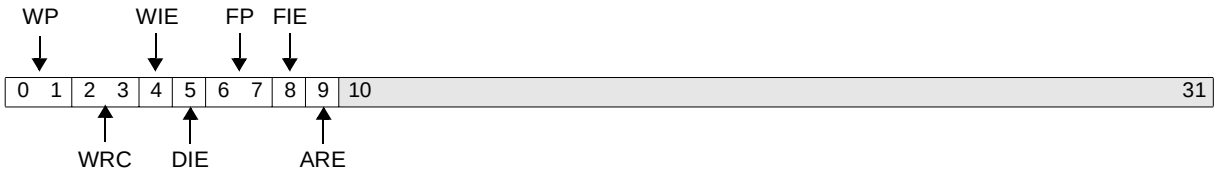


Figure 29-44. Timer Control Register (TCR)

0:1	WP	Watchdog Timer Period 00 2^{21} time base clocks 01 2^{25} time base clocks 10 2^{29} time base clocks 11 2^{33} time base clocks	
2:3	WRC	Watchdog Timer Reset Control 00 No Watchdog Timer reset will occur. 01 Core reset 10 Chip reset 11 System reset	TCR[WRC] resets to 0b00. Type of reset to cause upon Watchdog Timer exception with TSR[ENW,WIS]=0b11. This field can be set by software, but cannot be cleared by software, except by a software-induced reset.
4	WIE	Watchdog Timer Interrupt Enable 0 Disable Watchdog Timer interrupt. 1 Enable Watchdog Timer interrupt.	
5	DIE	Decrementer Interrupt Enable 0 Disable Decrementer interrupt. 1 Enable Decrementer interrupt.	
6:7	FP	Fixed Interval Timer (FIT) Period 00 2^{13} time base clocks 01 2^{17} time base clocks 10 2^{21} time base clocks 11 2^{25} time base clocks	
8	FIE	FIT Interrupt Enable 0 Disable Fixed Interval Timer interrupt. 1 Enable Fixed Interval Timer interrupt.	
9	ARE	Auto-Reload Enable 0 Disable auto reload. 1 Enable auto reload.	TCR[ARE] resets to 0b0.
10:31		Reserved	

SPR 0x150 Supervisor Read/Clear

See *Timer Status Register (TSR)* on page 390.

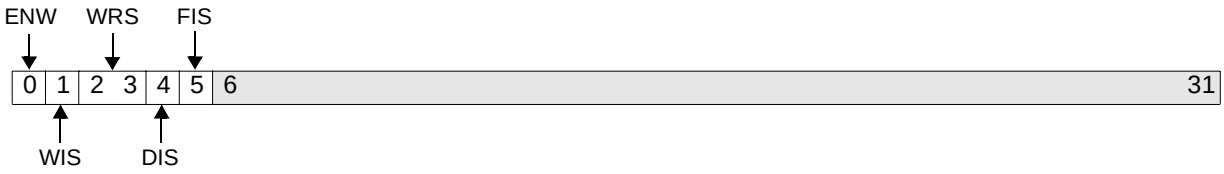


Figure 0-45. Timer Status Register (TSR)

0	ENW	Enable Next Watchdog Timer Exception 0 Action on next Watchdog Timer exception is to set TSR[ENW] = 1. 1 Action on next Watchdog Timer exception is governed by TSR[WIS].
1	WIS	Watchdog Timer Interrupt Status 0 Watchdog Timer exception has not occurred. 1 Watchdog Timer exception has occurred.
2:3	WRS	Watchdog Timer Reset Status 00 No Watchdog Timer reset has occurred. 01 Core reset was forced by Watchdog Timer. 10 Chip reset was forced by Watchdog Timer. 11 System reset was forced by Watchdog Timer.
4	DIS	Decrementer Interrupt Status 0 Decrementer exception has not occurred. 1 Decrementer exception has occurred.
5	FIS	Fixed Interval Timer (FIT) Interrupt Status 0 Fixed Interval Timer exception has not occurred. 1 Fixed Interval Timer exception has occurred.
6:31		Reserved

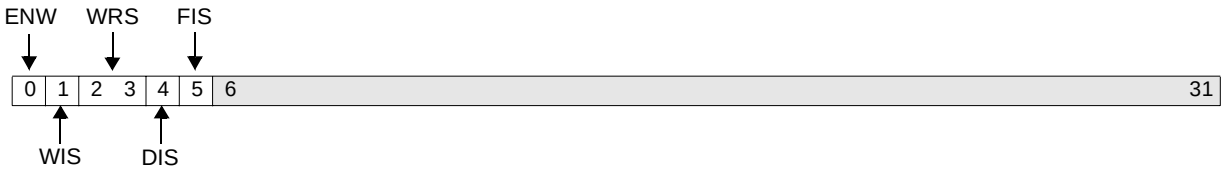


Figure 29-45. Timer Status Register (TSR)

0	ENW	Enable Next Watchdog Timer Exception 0 Action on next Watchdog Timer exception is to set TSR[ENW] = 1. 1 Action on next Watchdog Timer exception is governed by TSR[WIS].
1	WIS	Watchdog Timer Interrupt Status 0 Watchdog Timer exception has not occurred. 1 Watchdog Timer exception has occurred.

2:3	WRS	Watchdog Timer Reset Status 00 No Watchdog Timer reset has occurred. 01 Core reset was forced by Watchdog Timer. 10 Chip reset was forced by Watchdog Timer. 11 System reset was forced by Watchdog Timer.
4	DIS	Decrementer Interrupt Status 0 Decrementer exception has not occurred. 1 Decrementer exception has occurred.
5	FIS	Fixed Interval Timer (FIT) Interrupt Status 0 Fixed Interval Timer exception has not occurred. 1 Fixed Interval Timer exception has occurred.
6:31		Reserved

SPR 0x100 (User R/W)

See *Special Purpose Registers General (USPRG0, SPRG0–SPRG7)* on page 195.

Figure 0-46. User Special Purpose Register General (USPRG0)

0:31		General data	Software value; no hardware usage.

Figure 29-46. User Special Purpose Register General (USPRG0)

0:31		General data	Software value; no hardware usage.

SPR 0x001 User R/W

See *Integer Exception Register (XER)* on page 192.

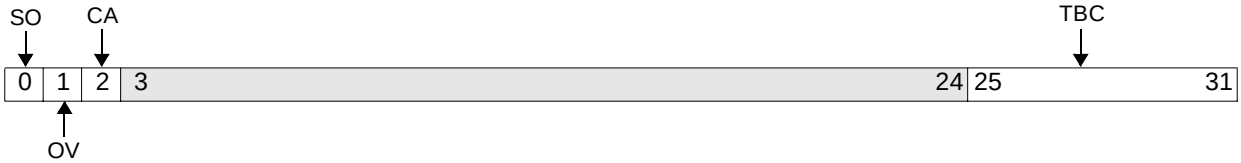


Figure 0-47. Integer Exception Register (XER)

0	SO	Summary Overflow 0 No overflow has occurred. 1 Overflow has occurred.	Can be *set* by **mtspr** or by integer or auxiliary processor instructions with the [**o**] option; can be *reset* by **mtspr** or by **mcrxr**.
1	OV	Overflow 0 No overflow has occurred. 0 Overflow has occurred.	Can be *set* by **mtspr** or by integer or allocated structions with the [**o**] option; can be *reset* by **mtspr**, by **mcrxr**, or by integer or allocated instructions with the [**o**] option.
2	CA	Carry 0 Carry has not occurred. 1 Carry has occurred.	Can be *set* by **mtspr** or by certain integer arithmetic and shift instructions; can be *reset* by **mtspr**, by **mcrxr**, or by certain integer arithmetic and shift instructions.
3:24		Reserved	
25:31	TBC	Transfer Byte Count	Used as a byte count by **lswx** and **stswx**; written by **dlmzb**[**.**] and by **mtspr.**

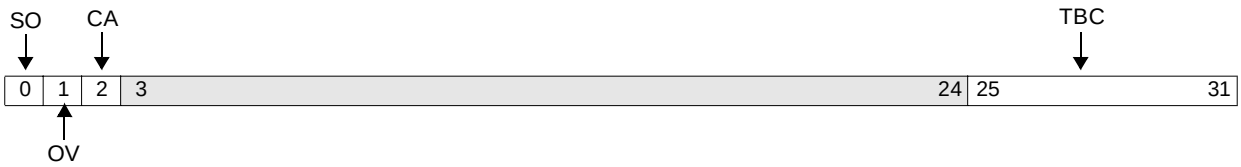


Figure 29-47. Integer Exception Register (XER)

0	SO	Summary Overflow 0 No overflow has occurred. 1 Overflow has occurred.	Can be *set* by **mtspr** or by integer or auxiliary processor instructions with the [**o**] option; can be *reset* by **mtspr** or by **mcrxr**.
1	OV	Overflow 0 No overflow has occurred. 1 Overflow has occurred.	Can be *set* by **mtspr** or by integer or allocated instructions with the [**o**] option; can be *reset* by **mtspr**, by **mcrxr**, or by integer or allocated instructions with the [**o**] option.
2	CA	Carry 0 Carry has not occurred. 1 Carry has occurred.	Can be *set* by **mtspr** or by certain integer arithmetic and shift instructions; can be *reset* by **mtspr**, by **mcrxr**, or by certain integer arithmetic and shift instructions.
3:24		Reserved	
25:31	TBC	Transfer Byte Count	Used as a byte count by **lswx** and **stswx**; written by **dlmzb**[**.**] and by **mtspr.**

29.6 Alphabetical Chip Control and Peripheral Core Register Listing

The following pages list all the device control registers (DCRs) and memory mapped registers (MMIOs) implemented in the PPC440GP For each register, the following information is supplied:

- Register mnemonic and name
- Cross reference to detailed register information
- Register type (DCR, MMIO)
- Register number (address)
- Register programming model (user or supervisor) and access (read-clear, read-only, read/write (R/W), write-only)
- A diagram illustrating the register fields (all register fields have mnemonics, unless there is only one field)
- A table describing the register fields, giving field mnemonics, field bit locations, field names, and the functions associated with the various field values

DCR 0x0EB Read/Write

See *Control Register 0 (CPC0_CR0)* on page 795.

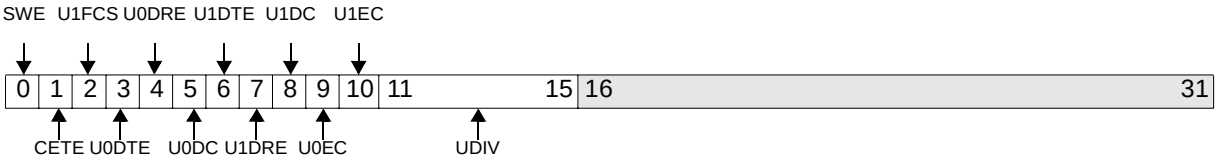


Figure 0-1. Control Register 0 (CPC0_CR0)

0	SWE	System Write Enable 0 DCR updates to the SYS0 and SYS1 registers are disabled 1 DCR updates to the SYS0 and SYS1 registers are enabled
1	CETE	CPU EXT Timer Enable 0 CPU timers increment at CPU clock frequency 1 CPU timers increment at TMR_CLK frequency
2	U1FCS	UART 1 Flow Control Select 0 Configure UART1 as DSR and DTR 1 Configure UART1 as CTS and RTS
3	U0DTE	UART 0 DMA Transmit Enable 0 UART0 DMA transmit is disabled 1 UART0 DMA transmit is enabled
4	U0DRE	UART 0 DMA Receive Enable 0 UART0 DMA receive is disabled 1 UART0 DMA receive is enabled
5	U0DC	UART 0 DMA Clear 0 U0DTE and U0DRE are not cleared when UART receives a corresponding terminal count. 1 U0DTE and U0DRE are cleared when UART receives a corresponding terminal count.
6	U1DTE	UART 1 DMA Transmit Enable 0 UART1 DMA transmit is disabled 1 UART1 DMA transmit is enabled

7	U1DRE	UART 1 DMA Receive Enable 0 UART1 DMA receive is disabled 1 UART1 DMA receive is enabled	
8	U1DC	UART 1 DMA Clear 0 U1DTE and U1DRE are not cleared when UART receives a corresponding terminal count. 1 U1DTE and U1DRE are cleared when UART receives a corresponding terminal count.	
9	U0EC	UART 0 EXT Clock Enable 0 UART0 uses the internal serial clock 1 UART0 uses the external serial clock	
10	U1EC	UART 1 EXT Clock Enable 0 UART1 uses the internal serial clock 1 UART1 uses the external serial clock	
11:15	UDIV	UART Divider 00000 - divider = 1 00001 - divider = 2 00010 - divider = 3 ..... 11110 - divider = 31 11111 - divider = 32	These bits control the bus frequency divider ratio between the 2xPLB and UART serial clock frequencies. Each UART can be individually configured to use the serial clock that is divided down according to these bits, or to use an externally provided serial clock. For example, if the 2xPLB clock is running at 266MHz, a divider value of 20 will set the serial clock frequency at 13.3MHz. **Note:** Maximum serial clock frequency allowed is slightly less than 1/2 OPB frequency.
16:31		Reserved	

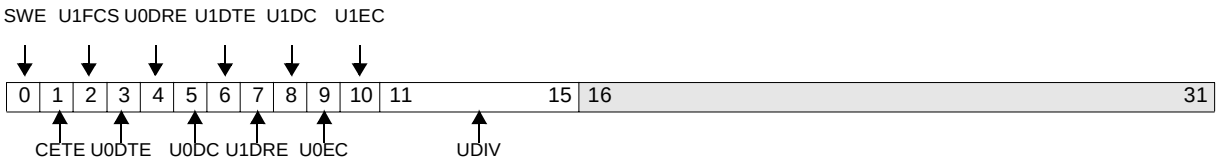


Figure 29-48. Control Register 0 (CPC0_CR0)

0	SWE	System Write Enable 0 DCR updates to the SYS0 and SYS1 registers are disabled 1 DCR updates to the SYS0 and SYS1 registers are enabled
1	CETE	CPU EXT Timer Enable 0 CPU timers increment at CPU clock frequency 1 CPU timers increment at TMR_CLK frequency

2	U1FCS	UART 1 Flow Control Select 0 Configure UART1 as DSR and DTR 1 Configure UART1 as CTS and RTS	
3	U0DTE	UART 0 DMA Transmit Enable 0 UART0 DMA transmit is disabled 1 UART0 DMA transmit is enabled	
4	U0DRE	UART 0 DMA Receive Enable 0 UART0 DMA receive is disabled 1 UART0 DMA receive is enabled	
5	U0DC	UART 0 DMA Clear 0 U0DTE and U0DRE are not cleared when UART receives a corresponding terminal count. 1 U0DTE and U0DRE are cleared when UART receives a corresponding terminal count.	
6	U1DTE	UART 1 DMA Transmit Enable 0 UART1 DMA transmit is disabled 1 UART1 DMA transmit is enabled	
7	U1DRE	UART 1 DMA Receive Enable 0 UART1 DMA receive is disabled 1 UART1 DMA receive is enabled	
8	U1DC	UART 1 DMA Clear 0 U1DTE and U1DRE are not cleared when UART receives a corresponding terminal count. 1 U1DTE and U1DRE are cleared when UART receives a corresponding terminal count.	
9	U0EC	UART 0 EXT Clock Enable 0 UART0 uses the internal serial clock 1 UART0 uses the external serial clock	
10	U1EC	UART 1 EXT Clock Enable 0 UART1 uses the internal serial clock 1 UART1 uses the external serial clock	
11:15	UDIV	UART Divider 00000 - divider = 1 00001 - divider = 2 00010 - divider = 3 ..... 11110 - divider = 31 11111 - divider = 32	These bits control the bus frequency divider ratio between the 2xPLB and UART serial clock frequencies. Each UART can be individually configured to use the serial clock that is divided down according to these bits, or to use an externally provided serial clock. For example, if the 2xPLB clock is running at 266MHz, a divider value of 20 will set the serial clock frequency at 13.3MHz. **Note:** Maximum serial clock frequency allowed is slightly less than 1/2 OPB frequency.
16:31		Reserved	

DCR 0x0EA Read/Write

See *Control Register 1 (CPC0_CR1)* on page 94.

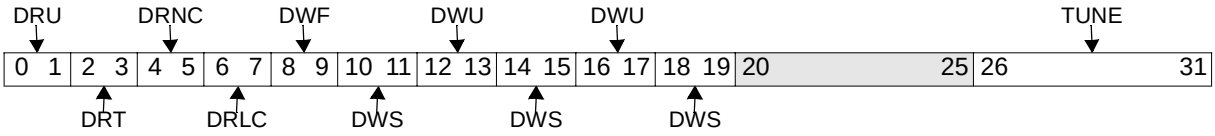


Figure 0-2. Control Register 1 (CPC0_CR1)

0:1	DRU	DcuRdUrgent priority setting used by CPU in some scenarios
2:3	DRT	DcuRdTouch priority setting used by CPU in some scenarios
4:5	DRNC	DcuRdNonCache priority setting used by CPU in some scenarios
6:7	DRLC	DcuRdLdCache priority setting used by CPU in some scenarios
8:9	DWF	DcuWrFlush priority setting used by CPU in some scenarios
10:11	DWS	DcuWrStore priority setting used by CPU in some scenarios
12:13	DWU	DcuWrUrgent priority setting used by CPU in some scenarios
14:15	IRF	IcuRdFetch priority setting used by CPU in some scenarios
16:17	IRT	IcuRdTouch priority setting used by CPU in some scenarios
18:19	IRS	IcuRdSpec priority setting used by CPU in some scenarios
20:25		Reserved
26:31	TUNE	TUNE (5:0) bits for DDR delay line element

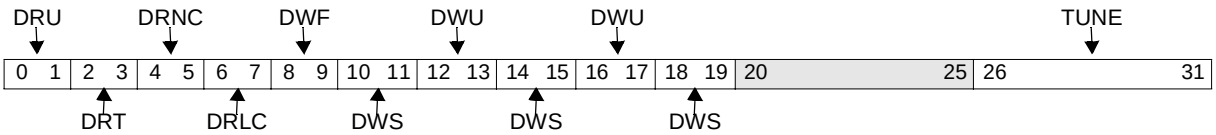


Figure 29-49. Control Register 1 (CPC0_CR1)

0:1	DRU	DcuRdUrgent priority setting used by CPU in some scenarios

2:3	DRT	DcuRdTouch priority setting used by CPU in some scenarios
4:5	DRNC	DcuRdNonCache priority setting used by CPU in some scenarios
6:7	DRLC	DcuRdLdCache priority setting used by CPU in some scenarios
8:9	DWF	DcuWrFlush priority setting used by CPU in some scenarios
10:11	DWS	DcuWrStore priority setting used by CPU in some scenarios
12:13	DWU	DcuWrUrgent priority setting used by CPU in some scenarios
14:15	IRF	IcuRdFetch priority setting used by CPU in some scenarios
16:17	IRT	IcuRdTouch priority setting used by CPU in some scenarios
18:19	IRS	IcuRdSpec priority setting used by CPU in some scenarios
20:25		Reserved
26:31	TUNE	TUNE (5:0) bits for DDR delay line element

DCR 0x0E2 Read/Write

See *Customer Configuration Register 0 (CPC0_CUST0)* on page 297.

Figure 0-3. Customer Configuration Register 0 (CPC0_CUST0)

0:31		Reserved	Reserved for customer use

Figure 29-50. Customer Configuration Register 0 (CPC0_CUST0)

0:31		Reserved	Reserved for customer use

DCR 0x0E3 Read/Write

See *Customer Configuration Register 0 (CPC0_CUST1)* on page 298.

Figure 0-4. Customer Configuration Register 1 (CPC0_CUST1)

0:31		Reserved	Reserved for customer use

Figure 29-51. Customer Configuration Register 1 (CPC0_CUST1)

0:31		Reserved	Reserved for customer use

DCR 0x0B1 Read/Write

See *CPM Enable Register (CPC0_ER)* on page 423.

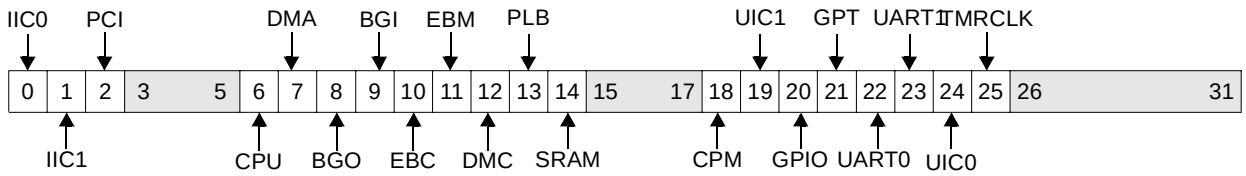


Figure 0-5. CPM Enable Register (CPC0_ER)

0	IIC0	Inter-Integrated Circuit 0	Class 3
1	IIC1	Inter-Integrated Circuit 1	Class 3
2	PCI	Peripheral Component Interconnect	Class 3
3:5		Reserved	
6	CPU	Processor Core	Class 2
7	DMA	Direct Memory Access Controller	Class 2
8	BGO	PLB to OPB Bridge	Class 2
9	BGI	OPB to PLB Bridge	Class 2
10	EBC	External Bus Controller	Class 2
11	EBM	External Bus Master Interface	Class 2
12	DMC	DDR SDRAM Controller	Class 2
13	PLB	PLB Arbiter	Class 2
14	SRAM	Internal SRAM Controller	Class 2
15:17		Reserved	
18	PPM	PLB Performance Monitor	Class 1
19	UIC1	Universal Interrupt Controller 1	Class 1
20	GPIO	General Purpose IO	Class 1
21	GPT	General Purpose Timer	Class 1
22	UART0	Universal Asynchronous Receiver/Transmitter 0	Class 1
23	UART1	Universal Asynchronous Receiver/Transmitter 1	Class 1
24	UIC0	Universal Interrupt Controller 0	Class 1
25	TMRCLK	CPU Timer	Class 1
26:31		Reserved	

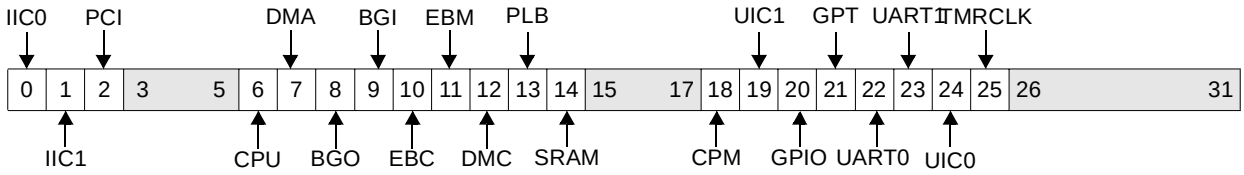


Figure 29-52. CPM Enable Register (CPC0_ER)

0	IIC0	Inter-Integrated Circuit 0	Class 3
1	IIC1	Inter-Integrated Circuit 1	Class 3
2	PCI	Peripheral Component Interconnect	Class 3
3:5		Reserved	
6	CPU	PPC440GP Processor Core	Class 2
7	DMA	Direct Memory Access Controller	Class 2
8	BGO	PLB to OPB Bridge	Class 2
9	BGI	OPB to PLB Bridge	Class 2
10	EBC	External Bus Controller	Class 2
11	EBM	External Bus Master Interface	Class 2
12	DMC	DDR SDRAM Controller	Class 2
13	PLB	PLB Arbiter	Class 2
14	SRAM	Internal SRAM Controller	Class 2
15:17		Reserved	
18	PPM	PLB Performance Monitor	Class 1
19	UIC1	Universal Interrupt Controller 1	Class 1
20	GPIO	General Purpose IO	Class 1
21	GPT	General Purpose Timer	Class 1
22	UART0	Universal Asynchronous Receiver/Transmitter 0	Class 1
23	UART1	Universal Asynchronous Receiver/Transmitter 1	Class 1
24	UIC0	Universal Interrupt Controller 0	Class 1
25	TMRCLK	CPU Timer	Class 1
26:31		Reserved	

DCR 0x0B2 Read/Write

See *CPM Force Register (CPC0_FR)* on page 425.

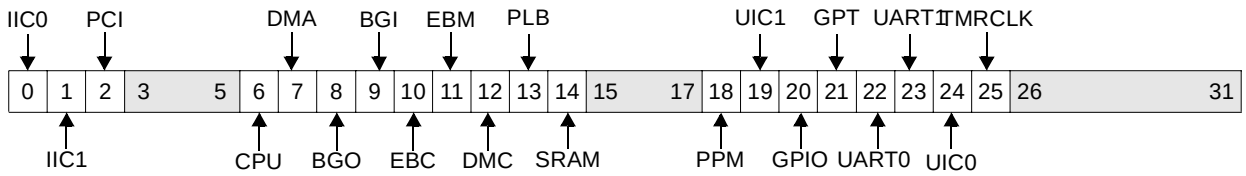


Figure 0-6. CPM Force Register (CPC0_FR)

0	IIC0	Inter-Integrated Circuit 0	Class 3
1	IIC1	Inter-Integrated Circuit 1	Class 3
2	PCI	Peripheral Component Interconnect	Class 3
3:5		Reserved	
6	CPU	Processor Core	Class 2
7	DMA	Direct Memory Access Controller	Class 2
8	BGO	PLB to OPB Bridge	Class 2
9	BGI	OPB to PLB Bridge	Class 2
10	EBC	External Bus Controller	Class 2
11	EBM	External Bus Master Interface	Class 2
12	DMC	DDR SDRAM Controller	Class 2
13	PLB	PLB Arbiter	Class 2
14	SRAM	Internal SRAM Controller	Class 2
15:17		Reserved	
18	PPM	PLB Performance Monitor	Class 1
19	UIC1	Universal Interrupt Controller 1	Class 1
20	GPIO	General Purpose IO	Class 1
21	GPT	General Purpose Timer	Class 1
22	UART0	Universal Asynchronous Receiver/Transmitter 0	Class 1
23	UART1	Universal Asynchronous Receiver/Transmitter 1	Class 1
24	UIC0	Universal Interrupt Controller 0	Class 1
25	TMRCLK	CPU Timer	Class 1
26:31		Reserved	

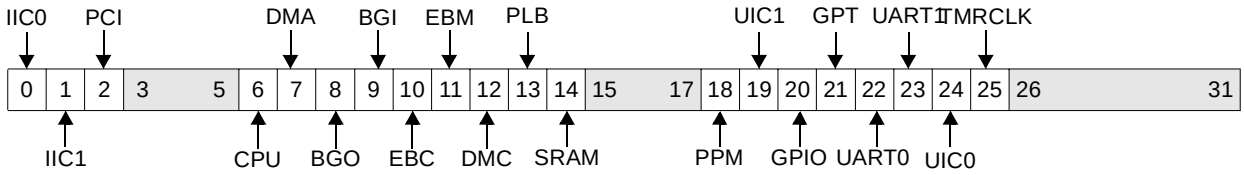


Figure 29-53. CPM Force Register (CPC0_FR)

0	IIC0	Inter-Integrated Circuit 0	Class 3
1	IIC1	Inter-Integrated Circuit 1	Class 3
2	PCI	Peripheral Component Interconnect	Class 3
3:5		Reserved	
6	CPU	PPC440GP Processor Core	Class 2
7	DMA	Direct Memory Access Controller	Class 2
8	BGO	PLB to OPB Bridge	Class 2
9	BGI	OPB to PLB Bridge	Class 2
10	EBC	External Bus Controller	Class 2
11	EBM	External Bus Master Interface	Class 2
12	DMC	DDR SDRAM Controller	Class 2
13	PLB	PLB Arbiter	Class 2
14	SRAM	Internal SRAM Controller	Class 2
15:17		Reserved	
18	PPM	PLB Performance Monitor	Class 1
19	UIC1	Universal Interrupt Controller 1	Class 1
20	GPIO	General Purpose IO	Class 1
21	GPT	General Purpose Timer	Class 1
22	UART0	Universal Asynchronous Receiver/Transmitter 0	Class 1
23	UART1	Universal Asynchronous Receiver/Transmitter 1	Class 1
24	UIC0	Universal Interrupt Controller 0	Class 1
25	TMRCLK	CPU Timer	Class 1
26:31		Reserved	

DCR 0x0E8 Read/Write

See *GPIO Configuration Register (CPC0_GPIO)* on page 803.

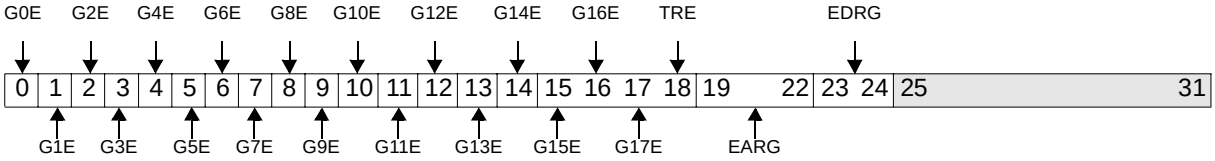


Figure 0-7. GPIO Configuration Register (CPC0_GPIO)

0	G0E	GPIO 0 Enable 0 Enable interrupt IRQ00 as an interrupt 1 Enable interrupt IRQ00 as GPIO0
1	G1E	GPIO 1 Enable 0 Enable interrupt IRQ01 as an interrupt 1 Enable interrupt IRQ01 as GPIO1
2	G2E	GPIO 2 Enable 0 Enable interrupt IRQ02 as an interrupt 1 Enable interrupt IRQ02 as GPIO2
3	G3E	GPIO 3 Enable 0 Enable interrupt IRQ03 as an interrupt 1 Enable interrupt IRQ03 as GPIO3
4	G4E	GPIO 4 Enable 0 Enable interrupt IRQ04 as an interrupt 1 Enable interrupt IRQ04 as GPIO4
5	G5E	GPIO 5 Enable 0 Enable interrupt IRQ05 as an interrupt 1 Enable interrupt IRQ05 as GPIO5
6	G6E	GPIO 6 Enable 0 Enable interrupt IRQ06 as an interrupt 1 Enable interrupt IRQ06 as GPIO6
7	G7E	GPIO 7 Enable 0 Enable interrupt IRQ07 as an interrupt 1 Enable interrupt IRQ07 as GPIO7
8	G8E	GPIO 8 Enable 0 Enable interrupt IRQ08 as an interrupt 1 Enable interrupt IRQ08 as GPIO8
9	G9E	GPIO 9 Enable 0 Enable interrupt IRQ09 as an interrupt 1 Enable interrupt IRQ09 as GPIO9
10	G10E	GPIO 10 Enable 0 Enable interrupt IRQ010 as an interrupt 1 Enable interrupt IRQ010 as GPIO10

11	G11E	GPIO 11 Enable 0 Enable interrupt IRQ011 as an interrupt 1 Enable interrupt IRQ011 as GPIO11	
12	G12E	GPIO 12 Enable 0 Enable UART1RX as UART1RX 1 Enable UART1RX as GPIO 12	
13	G13E	GPIO 13 Enable 0 Enable UART1TX as UART1TX 1 Enable UART1TX as GPIO 13	
14	G14E	GPIO 14 Enable 0 Enable UART1DSR_CTS as UART1DSR_CTS 1 Enable UART1DSR_CTS as GPIO 14	
15	G15E	GPIO 15 Enable 0 Enable UART1RTS_DTR as UART1RTS_DTR 1 Enable UART1RTS_DTR as GPIO 15	
16	G16E	GPIO 16 Enable 0 Enable IIC1SCL as IIC1SCL 1 Enable IIC1SCL as GPIO 16	
17	G17E	GPIO 17 Enable 0 Enable IIC1SDA as IIC1SDA 1 Enable IIC1SDA as GPIO 17	
18	TRE	GPIO Trace Enable 0 GPIO18-31 are enabled 1 GPIO18-31 are disabled	Trace interface cannot be used when GPIO is enabled. Trace interface can be used when GPIO is disabled.
19:22	EARG	EBC Address Receiver Gating 0000 Receiver gating disabled 0001 Disable address bus inputs 0:1 0010 Disable address bus inputs 0:2 0011 Disable address bus inputs 0:3 .. .. .. 1111 Disable address bus inputs 0:15	Encoded value for gating external bus controller address bus
23:24	EDRG	EBC Data Receiver Gating 00 Receiver gating disabled 01 Disable data bus input bytes 1, 2, 3 10 Disable data bus input bytes 2, 3 11 Disable data bus input byte 3	Encoded value for gating external bus controller data bus
25:31		Reserved	

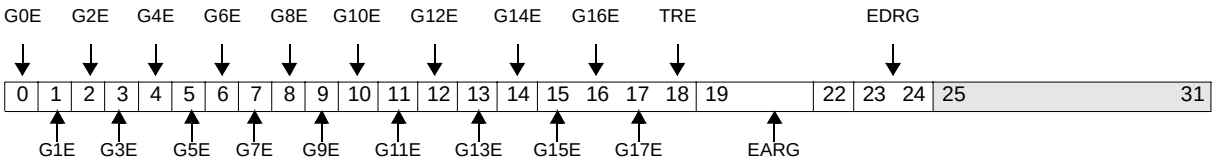


Figure 29-54. GPIO Configuration Register (CPC0_GPIO)

0	G0E	GPIO 0 Enable 0 Enable interrupt IRQ00 as an interrupt 1 Enable interrupt IRQ00 as GPIO0
1	G1E	GPIO 1 Enable 0 Enable interrupt IRQ01 as an interrupt 1 Enable interrupt IRQ01 as GPIO1
2	G2E	GPIO 2 Enable 0 Enable interrupt IRQ02 as an interrupt 1 Enable interrupt IRQ02 as GPIO2
3	G3E	GPIO 3 Enable 0 Enable interrupt IRQ03 as an interrupt 1 Enable interrupt IRQ03 as GPIO3
4	G4E	GPIO 4 Enable 0 Enable interrupt IRQ04 as an interrupt 1 Enable interrupt IRQ04 as GPIO4
5	G5E	GPIO 5 Enable 0 Enable interrupt IRQ05 as an interrupt 1 Enable interrupt IRQ05 as GPIO5
6	G6E	GPIO 6 Enable 0 Enable interrupt IRQ06 as an interrupt 1 Enable interrupt IRQ06 as GPIO6
7	G7E	GPIO 7 Enable 0 Enable interrupt IRQ07 as an interrupt 1 Enable interrupt IRQ07 as GPIO7
8	G8E	GPIO 8 Enable 0 Enable interrupt IRQ08 as an interrupt 1 Enable interrupt IRQ08 as GPIO8
9	G9E	GPIO 9 Enable 0 Enable interrupt IRQ09 as an interrupt 1 Enable interrupt IRQ09 as GPIO9
10	G10E	GPIO 10 Enable 0 Enable interrupt IRQ010 as an interrupt 1 Enable interrupt IRQ010 as GPIO10
11	G11E	GPIO 11 Enable 0 Disable GPIO11 1 Enable GPIO11
12	G12E	GPIO 12 Enable 0 Enable UART1RX as UART1RX 1 Enable UART1RX as GPIO 12

13	G13E	GPIO 13 Enable 0 Enable UART1TX as UART1TX 1 Enable UART1TX as GPIO 13	
14	G14E	GPIO 14 Enable 0 Enable UART1DSR_CTS as UART1DSR_CTS 1 Enable UART1DSR_CTS as GPIO 14	
15	G15E	GPIO 15 Enable 0 Enable UART1RTS_DTR as UART1RTS_DTR 1 Enable UART1RTS_DTR as GPIO 15	
16	G16E	GPIO 16 Enable 0 Enable IIC1SCL as IIC1SCL 1 Enable IIC1SCL as GPIO 16	
17	G17E	GPIO 17 Enable 0 Enable IIC1SDA as IIC1SDA 1 Enable IIC1SDA as GPIO 17	
18	TRE	GPIO Trace Enable 0 GPIO18-31 are enabled 1 GPIO18-31 are disabled	Trace interface cannot be used when GPIO is enabled. Trace interface can be used when GPIO is disabled.
19:22	EARG	EBC Address Receiver Gating 0000 Receiver gating disabled 0001 Disable address bus inputs 0:1 0010 Disable address bus inputs 0:2 0011 Disable address bus inputs 0:3 .. .. .. 1111 Disable address bus inputs 0:15	Encoded value for gating external bus controller address bus
23:24	EDRG	EBC Data Receiver Gating 00 Receiver gating disabled 01 Disable data bus input bytes 1, 2, 3 10 Disable data bus input bytes 2, 3 11 Disable data bus input byte 3	Encoded value for gating external bus controller data bus
25:31		Reserved	

DCR 0X0EF Read-Only

See *JTAG ID Register (CPC0_JTAGID)* on page 431.

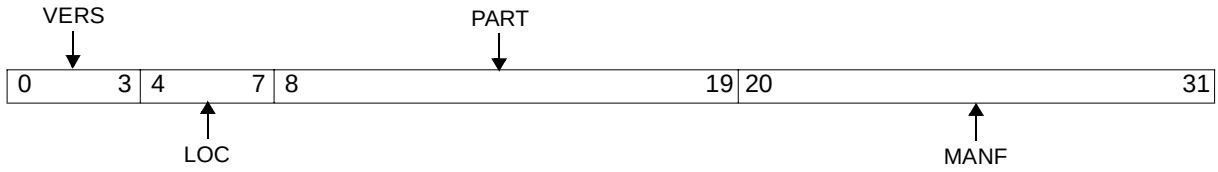


Figure 0-8. JTAG ID Register (CPC0_JTAGID)

0:3	VERS	Version
4:7	LOC	Developer Location
8:19	PART	Part Number
20:31	MANF	Manufacturer Identifier

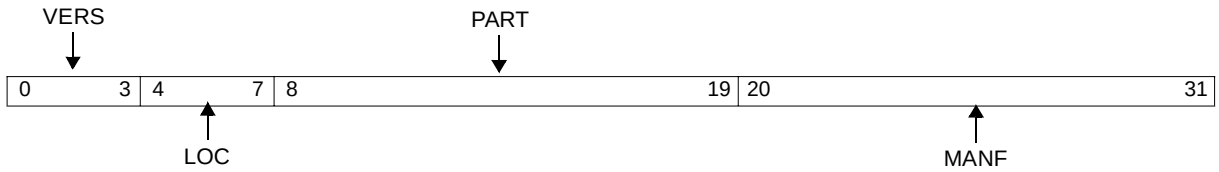


Figure 29-55. JTAG ID Register (CPC0_JTAGID)

0:3	VERS	Version
4:7	LOC	Developer Location
8:19	PART	Part Number
20:31	MANF	Manufacturer Identifier

DCR 0x0EC Read/Write

See *Master Interrupt Request Register 0 (CPC0_MIRQ0)* on page 95.

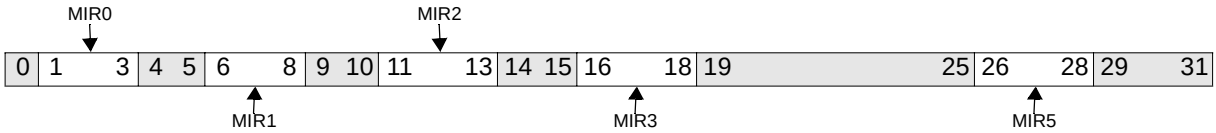


Figure 0-9. Master Interrupt Request Register 0 (CPC0_MIRQ0)

0		Reserved	
1:3	MIR0	Master interrupt request 0	ICU read interrupt request from DDR, PCI, and PLB to OPB bridge controller.
4:5		Reserved	
6:8	MIR1	Master interrupt request 1	DCU read interrupt request from DDR, PCI, and PLB to OPB bridge controller.
9:10		Reserved	
11:13	MIR2	Master interrupt request 2	DCU write interrupt request from DDR, PCI, and PLB to OPB bridge controller.
14:15		Reserved	
16:18	MIR3	Master interrupt request 3	PCIX controller interrupt request from DDR, PCI, and PLB to OPB bridge controller
19:25		Reserved	
26:28	MIR5	Master interrupt request 5	MAL controller interrupt request from DDR, PCI, and PLB to OPB bridge controller
29:31		Reserved	

Figure 29-56. Master Interrupt Request Register 0 (CPC0_MIRQ0)

0		Reserved	
1:3	MIR0	Master interrupt request 0	ICU read interrupt request from DDR, PCI, and PLB to OPB bridge controller.
4:5		Reserved	
6:8	MIR1	Master interrupt request 1	DCU read interrupt request from DDR, PCI, and PLB to OPB bridge controller.
9:10		Reserved	

11:13	MIR2	Master interrupt request 2	DCU write interrupt request from DDR, PCI, and PLB to OPB bridge controller.
14:15		Reserved	
16:18	MIR3	Master interrupt request 3	PCIX controller interrupt request from DDR, PCI, and PLB to OPB bridge controller
19:25		Reserved	
26:28	MIR5	Master interrupt request 5	MAL controller interrupt request from DDR, PCI, and PLB to OPB bridge controller
29:31		Reserved	

DCR 0x0ED Read/Write

See *Master Interrupt Request Register 1 (CPC0_MIRQ1)* on page 96.

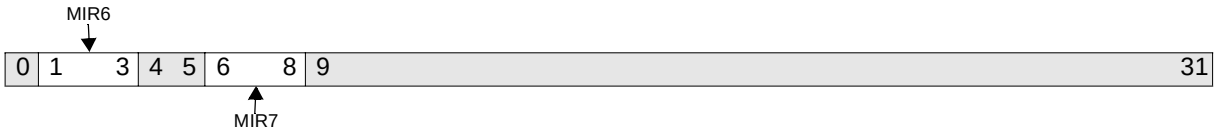


Figure 0-10. Master Interrupt Request Register 1 (CPC0_MIRQ1)

0		Reserved	
1:3	MIR6	Master interrupt request 6	DMA controller interrupt request from DDR, PCI, and PLB to OPB bridge controller.
4:5		Reserved	
6:8	MIR7	Master interrupt request 7	OPB to PLB bridge controller interrupt request from DDR, PCI, and PLB to OPB bridge controller.
9:31		Reserved	

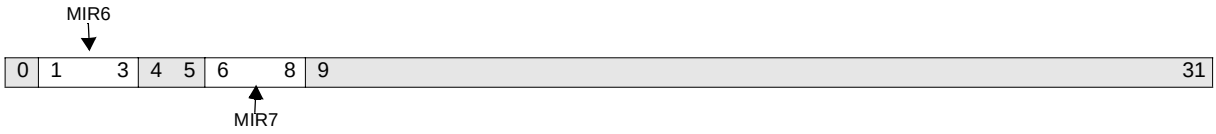


Figure 29-57. Master Interrupt Request Register 1 (CPC0_MIRQ1)

0		Reserved	
1:3	MIR6	Master interrupt request 6	DMA controller interrupt request from DDR, PCI, and PLB to OPB bridge controller.
4:5		Reserved	
6:8	MIR7	Master interrupt request 7	OPB to PLB bridge controller interrupt request from DDR, PCI, and PLB to OPB bridge controller.
9:31		Reserved	

DCR 0x0E9 Read/Write

See *PLB Master Priority Register (CPC0_PLB)* on page 93.

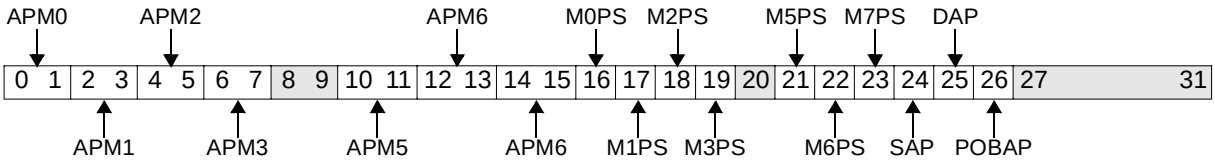


Figure 0-11. PLB Master Priority Register (CPC0_PLB)

0:1	APM0	Alternate PLB master priority setting 0	Alternate ICU read priority setting
2:3	APM1	Alternate PLB master priority setting 1	Alternate DCU read priority setting
4:5	APM2	Alternate PLB master priority setting 2	Alternate DCU write priority setting
6:7	APM3	Alternate PLB master priority setting 3	ALternate PCIX priority setting
8:9		Reserved	
10:11	APM5	Alternate PLB master priority setting 5	Alternate MAL priority setting
12:13	APM6	Alternate PLB master priority setting 6	Alternate DMA priority setting
14:15	APM7	Alternate PLB master priority setting 7	Alternate OPB to PLB bridge priority setting
16	M0PS	Master 0 Priority Setting 0 Master 0 controls own priority setting 1 Alternate PLB master priority setting	ICU read priority setting
17	M1PS	Master 1 Priority Setting 0 Master 1 controls own priority setting 1 Alternate PLB master priority setting	DCU read priority setting
18	M2PS	Master 2 Priority Setting 0 Master 2 controls own priority setting 1 Alternate PLB master priority setting	DCU write priority setting
19	M3PS	Master 3 Priority Setting 0 Master 3 controls own priority setting 1 Alternate PLB master priority setting	PCIX priority setting
20		Reserved	
21	M5PS	Master 5 Priority Setting 0 Master 5 controls own priority setting 1 Alternate PLB master priority setting	MAL priority setting
22	M6PS	Master 6 Priority Setting 0 Master 6 controls own priority setting 1 Alternate PLB master priority setting	DMA priority setting

23	M7PS	Master 7 Priority Setting 0 Master 7 controls own priority setting 1 Alternate PLB master priority setting	OPB to PLB bridge priority setting
24	SAP	SRAM Address Pipelining 0 Address pipelining disabled to SRAM slave 1 Address pipelining enabled to SRAM slave	
25	DAP	DDRSDRAM Address Pipelining 0 Address pipelining disabled to DDR SDRAM slave 1 Address pipelining enabled to DDR SDRAM slave	
26	POBAP	PLB to OPB Bridge Address Pipelining 0 Address pipelining disabled to PLB to OPB bridge slave 1 Address pipelining enabled to PLB to OPB bridge slave	
27:31		Reserved	

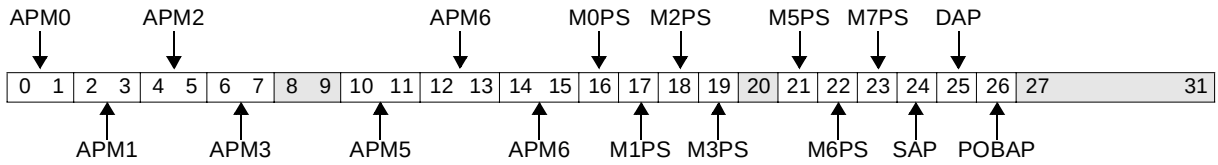


Figure 29-58. PLB Master Priority Register (CPC0_PLB)

0:1	APM0	Alternate PLB master priority setting 0	Alternate ICU read priority setting
2:3	APM1	Alternate PLB master priority setting 1	Alternate DCU read priority setting
4:5	APM2	Alternate PLB master priority setting 2	Alternate DCU write priority setting
6:7	APM3	Alternate PLB master priority setting 3	ALternate PCIX priority setting
8:9		Reserved	
10:11	APM5	Alternate PLB master priority setting 5	Alternate MAL priority setting
12:13	APM6	Alternate PLB master priority setting 6	Alternate DMA priority setting
14:15	APM7	Alternate PLB master priority setting 7	Alternate OPB to PLB bridge priority setting
16	M0PS	Master 0 Priority Setting 0 Master 0 controls own priority setting 1 Alternate PLB master priority setting	ICU read priority setting
17	M1PS	Master 1 Priority Setting 0 Master 1 controls own priority setting 1 Alternate PLB master priority setting	DCU read priority setting

18	M2PS	Master 2 Priority Setting 0 Master 2 controls own priority setting 1 Alternate PLB master priority setting	DCU write priority setting
19	M3PS	Master 3 Priority Setting 0 Master 3 controls own priority setting 1 Alternate PLB master priority setting	PCIX priority setting
20		Reserved	
21	M5PS	Master 5 Priority Setting 0 Master 5 controls own priority setting 1 Alternate PLB master priority setting	MAL priority setting
22	M6PS	Master 6 Priority Setting 0 Master 6 controls own priority setting 1 Alternate PLB master priority setting	DMA priority setting
23	M7PS	Master 7 Priority Setting 0 Master 7 controls own priority setting 1 Alternate PLB master priority setting	OPB to PLB bridge priority setting
24	SAP	SRAM Address Pipelining 0 Address pipelining disabled to SRAM slave 1 Address pipelining enabled to SRAM slave	
25	DAP	DDRSDRAM Address Pipelining 0 Address pipelining disabled to DDR SDRAM slave 1 Address pipelining enabled to DDR SDRAM slave	
26	POBAP	PLB to OPB Bridge Address Pipelining 0 Address pipelining disabled to PLB to OPB bridge slave 1 Address pipelining enabled to PLB to OPB bridge slave	
27:31		Reserved	

DCR 0x0B0 Read Only

See *CPM Status Register (CPC0_SR)* on page 426.

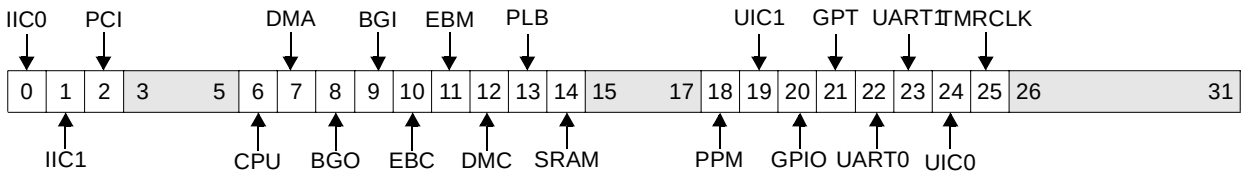


Figure 0-12. CPM Status Register (CPC0_SR)

0	IIC0	Inter-Integrated Circuit 0	Class 3
1	IIC1	Inter-Integrated Circuit 1	Class 3
2	PCI	Peripheral Component Interconnect	Class 3
3:5		Reserved	
6	CPU	Processor Core	Class 2
7	DMA	Direct Memory Access Controller	Class 2
8	BGO	PLB to OPB Bridge	Class 2
9	BGI	OPB to PLB Bridge	Class 2
10	EBC	External Bus Controller	Class 2
11	EBM	External Bus Master Interface	Class 2
12	DMC	DDR SDRAM Controller	Class 2
13	PLB	PLB Arbiter	Class 2
14	SRAM	Internal SRAM Controller	Class 2
15:17		Reserved	
18	PPM	PLB Performance Monitor	Class 1
19	UIC1	Universal Interrupt Controller 1	Class 1
20	GPIO	General Purpose IO	Class 1
21	GPT	General Purpose Timer	Class 1
22	UART0	Universal Asynchronous Receiver/Transmitter 0	Class 1
23	UART1	Universal Asynchronous Receiver/Transmitter 1	Class 1
24	UIC0	Universal Interrupt Controller 0	Class 1
25	TMRCLK	CPU Timer	Class 1
26:31		Reserved	

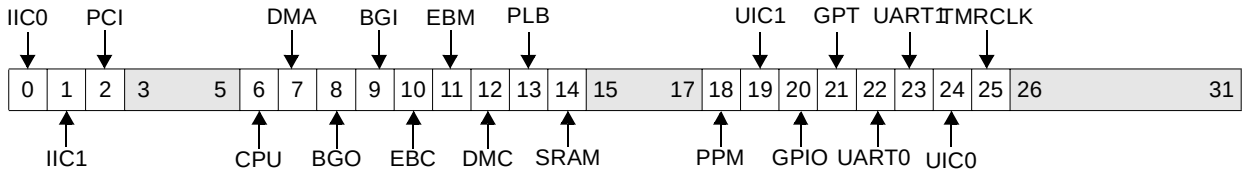


Figure 29-59. CPM Status Register (CPC0_SR)

0	IIC0	Inter-Integrated Circuit 0	Class 3
1	IIC1	Inter-Integrated Circuit 1	Class 3
2	PCI	Peripheral Component Interconnect	Class 3
3:5		Reserved	
6	CPU	PPC440GP Processor Core	Class 2
7	DMA	Direct Memory Access Controller	Class 2
8	BGO	PLB to OPB Bridge	Class 2
9	BGI	OPB to PLB Bridge	Class 2
10	EBC	External Bus Controller	Class 2
11	EBM	External Bus Master Interface	Class 2
12	DMC	DDR SDRAM Controller	Class 2
13	PLB	PLB Arbiter	Class 2
14	SRAM	Internal SRAM Controller	Class 2
15:17		Reserved	
18	PPM	PLB Performance Monitor	Class 1
19	UIC1	Universal Interrupt Controller 1	Class 1
20	GPIO	General Purpose IO	Class 1
21	GPT	General Purpose Timer	Class 1
22	UART0	Universal Asynchronous Receiver/Transmitter 0	Class 1
23	UART1	Universal Asynchronous Receiver/Transmitter 1	Class 1
24	UIC0	Universal Interrupt Controller 0	Class 1
25	TMRCLK	CPU Timer	Class 1
26:31		Reserved	

DCR 0x0E4 Read-Only

See *Power-On Configuration Register 0 (CPC0_STRP0)* on page 289.

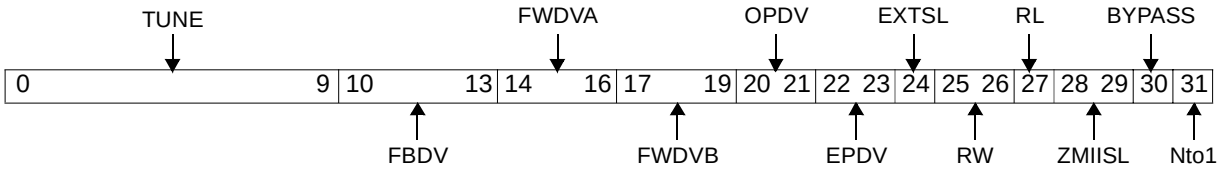


Figure 0-13. Power-On Configuration Register 0 (CPC0_STRP0)

0:9	TUNE	System PLL TUNE bits	See Table 13-5, on page 13-5 for tune bit setting information.
10:13	FBDV	PLL Feedback Divisor 0000 PLL Feedback Divisor = 16 0001 PLL Feedback Divisor = 1 0010 PLL Feedback Divisor = 2 0011 PLL Feedback Divisor = 3 0100 PLL Feedback Divisor = 4 0101 PLL Feedback Divisor = 5 0110 PLL Feedback Divisor = 6 0111 PLL Feedback Divisor = 7 1000 PLL Feedback Divisor = 8 1001 PLL Feedback Divisor = 9 1010 PLL Feedback Divisor = 10 1011 PLL Feedback Divisor = 11 1100 PLL Feedback Divisor = 12 1101 PLL Feedback Divisor = 13 1110 PLL Feedback Divisor = 14 1111 PLL Feedback Divisor = 15	
14:16	FWDVA	PLL Forward Divisor A 000 PLL Forward Divisor A = 8 001 PLL Forward Divisor A = 7 010 PLL Forward Divisor A = 6 011 PLL Forward Divisor A = 5 100 PLL Forward Divisor A = 4 101 PLL Forward Divisor A = 3 110 PLL Forward Divisor A = 2 111 PLL Forward Divisor A = 1	
17:19	FWDVB	PLL Forward Divisor B 000 PLL Forward Divisor B = 8 001 PLL Forward Divisor B = 7 010 PLL Forward Divisor B = 6 011 PLL Forward Divisor B = 5 100 PLL Forward Divisor B = 4 101 PLL Forward Divisor B = 3 110 PLL Forward Divisor B = 2	

20:21	OPDV	PLB to OPB Clock Divisor Ratio 00 PLB to OPB Clock Divisor = 1 01 PLB to OPB Clock Divisor = 2 10 PLB to OPB Clock Divisor = 3 11 PLB to OPB Clock Divisor = 4	The OPDV is the divisor ratio between the PLB and OPB clock. For example, if the PLB clock frequency is 133MHz, a divisor ratio of two sets the OPB clock frequency to 66MHz.
22:23	EPDV	EBCO to OPB Clock Divisor Ratio 00 EBCO to OPB Clock Divisor = 1 01 EBCO to OPB Clock Divisor = 2 10 EBCO to OPB Clock Divisor = 3 11 EBCO to OPB Clock Divisor = 4	The EPDV is the divisor ratio between the EBC0 (PerClk) and OPB clock. For example, if the OPB clock frequency is 66MHz, a divisor ratio of two sets the EBC0 clock (PerClk) frequency to 33MHz.
24	EXTSL	EBC0 clock (PerClk) Feedback Path Selection 0 CPU clock selected as PLL feedback path 1 EBC0 clock (PerClk) selected as PLL feedback path	Selecting the CPU clock as the PLL feedback path phase aligns PerClk with the CPU clock. Selecting the PerClk as the PLL feedback path phase aligns PerClk with SysClk.
25:26	RW	ROM Width 00 8-bit ROM 01 16-bit ROM 10 reserved 11 32-bit ROM	
27	RL	ROM Location 0 ROM is connected to EBCO 1 ROM is connected to PCI	If ROM is connected to EBCO, the real address of the reset vector is 0x1FFFFFFC. If ROM is connected to PCI, the real address of the reset vector is 0x2FFFFFFC.
28:29	ZMIISL	ZMII Selection 00 MII mode selected 01 SMII mode selected 10 RMII 10Mb mode selected 11 RMII 100Mb mode selected	See "EMAC-ZMII Bridge Interfaces" on page 21-3.
30	BYPASS	Bypass System PLL 0 System PLL enabled 1 Bypass system PLL	
31	Nto1	CPU:PLB N to 1 clock ratio 0 CPU:PLB clock ratio is N:P where P is greater than 1 1 CPU:PLB clock ratio is N:1	

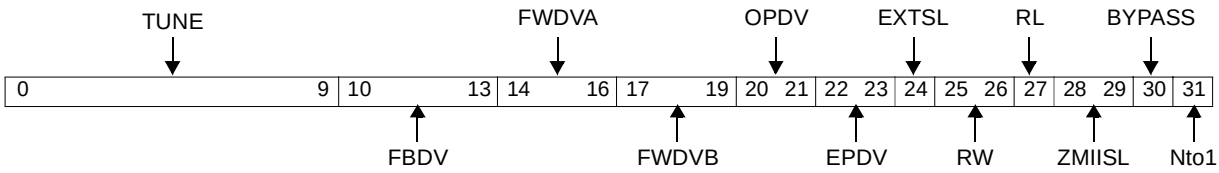


Figure 29-60. Power-On Configuration Register 0 (CPC0_STRP0)

0:9	TUNE	System PLL TUNE bits	See Table 13-5, on page -411 for tune bit setting information.
10:13	FBDV	PLL Feedback Divisor 0000 PLL Feedback Divisor = 16 0001 PLL Feedback Divisor = 1 0010 PLL Feedback Divisor = 2 0011 PLL Feedback Divisor = 3 0100 PLL Feedback Divisor = 4 0101 PLL Feedback Divisor = 5 0110 PLL Feedback Divisor = 6 0111 PLL Feedback Divisor = 7 1000 PLL Feedback Divisor = 8 1001 PLL Feedback Divisor = 9 1010 PLL Feedback Divisor = 10 1011 PLL Feedback Divisor = 11 1100 PLL Feedback Divisor = 12 1101 PLL Feedback Divisor = 13 1110 PLL Feedback Divisor = 14 1111 PLL Feedback Divisor = 15	
14:16	FWDVA	PLL Forward Divisor A 000 PLL Forward Divisor A = 8 001 PLL Forward Divisor A = 7 010 PLL Forward Divisor A = 6 011 PLL Forward Divisor A = 5 100 PLL Forward Divisor A = 4 101 PLL Forward Divisor A = 3 110 PLL Forward Divisor A = 2 111 PLL Forward Divisor A = 1	
17:19	FWDVB	PLL Forward Divisor B 000 PLL Forward Divisor B = 8 001 PLL Forward Divisor B = 7 010 PLL Forward Divisor B = 6 011 PLL Forward Divisor B = 5 100 PLL Forward Divisor B = 4 101 PLL Forward Divisor B = 3 110 PLL Forward Divisor B = 2	
20:21	OPDV	PLB to OPB Clock Divisor Ratio 00 PLB to OPB Clock Divisor = 1 01 PLB to OPB Clock Divisor = 2 10 PLB to OPB Clock Divisor = 3 11 PLB to OPB Clock Divisor = 4	The OPDV is the divisor ratio between the PLB and OPB clock. For example, if the PLB clock frequency is 133MHz, a divisor ratio of two sets the OPB clock frequency to 66MHz.
22:23	EPDV	EBCO to OPB Clock Divisor Ratio 00 EBCO to OPB Clock Divisor = 1 01 EBCO to OPB Clock Divisor = 2 10 EBCO to OPB Clock Divisor = 3 11 EBCO to OPB Clock Divisor = 4	The EPDV is the divisor ratio between the EBC0 (PerClk) and OPB clock. For example, if the OPB clock frequency is 66MHz, a divisor ratio of two sets the EBC0 clock (PerClk) frequency to 33MHz.

24	EXTSL	EBC0 clock (PerClk) Feedback Path Selection 0 CPU clock selected as PLL feedback path 1 EBC0 clock (PerClk) selected as PLL feedback path	Selecting the CPU clock as the PLL feedback path phase aligns PerClk with the CPU clock. Selecting the PerClk as the PLL feedback path phase aligns PerClk with SysClk.
25:26	RW	ROM Width 00 8-bit ROM 01 16-bit ROM 10 reserved 11 32-bit ROM	
27	RL	ROM Location 0 ROM is connected to EBCO 1 ROM is connected to PCI	If ROM is connected to EBCO, the real address of the reset vector is 0x1FFFFFFFC. If ROM is connected to PCI, the real address of the reset vector is 0x2FFFFFFFC.
28:29	ZMIISL	ZMII Selection 00 MII mode selected 01 SMII mode selected 10 RMII 10Mb mode selected 11 RMII 100Mb mode selected	See *EMAC-ZMII Bridge Interfaces* on page 721.
30	BYPASS	Bypass System PLL 0 System PLL enabled 1 Bypass system PLL	
31	Nto1	CPU:PLB N to 1 clock ratio 0 CPU:PLB clock ratio is N:P where P is greater than 1 1 CPU:PLB clock ratio is N:1	

DCR 0x0E5 Read-Only

See *Power-On Configuration Register 1 (CPC0_STRP1)* on page 293.

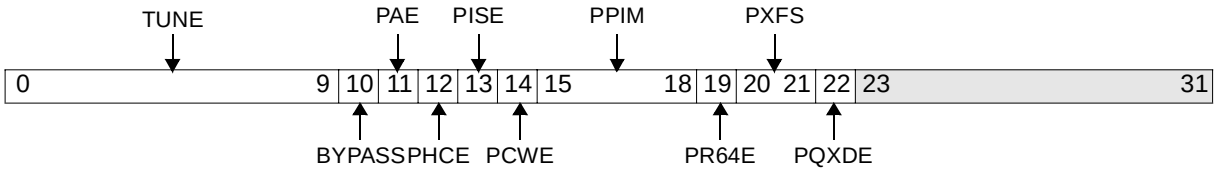


Figure 0-14. Power-On Configuration Register 1 (CPC0_STRP1)

0:9	TUNE	PCI PLL TUNE Bits	See Table 13-5, on page 13-5 for tune bit setting information.
10	BYPASS	Bypass PCI PLL 0 PCI PLL enabled 1 Bypass PCI PLL	
11	PAE	PCI arbiter enable	
12	PHCE	PCI host configuration enable	
13	PISE	PCI initial sequence enable	
14	PCWE	PCI local CPU wait enable	
15:18	PPIM	PCI Inbound Map (PIM) Settings 0000 PIM0 off, PIM1 off, PIM2 off 0001 PIM0 4k, PIM1 off, PIM2 off 0010 PIM0 1M, PIM1 off, PIM2 off 0011 PIM0 64M, PIM1 off, PIM2 off 0100 PIM0 4kp, PIM1 off, PIM2 off 0101 PIM0 1Mp, PIM1 off, PIM2 off 0110 PIM0 64Mp, PIM1 off, PIM2 off 0111 PIM0 64k, PIM1 off, PIM2 16k 1000 PIM0 1M, PIM1 off, PIM2 64k 1001 PIM0 64kp, PIM1 off, PIM2 16k 1010 PIM0 1Mp, PIM1 off, PIM2 64k 1011 PIM0 64k, PIM1 off, PIM2 64Kp 1100 PIM0 1M, PIM1 off, PIM2 1Mp 1101 PIM0 1Mp, PIM1 off, PIM2 1Mp 1110 PIM0 1M, PIM1 on, PIM2 off 1111 PIM0 1M, PIM1 on, PIM2 16k	These bits control the default settings for various PIM fields. See “PCI Inbound Map Setting” on page 8-3.
19	PR64E	PCI initialize Req64 Enable	
20:21	PXFS	PCIX Frequency Selection 00 100 - 133MHz 01 66 - 100MHz 10 50 - 66MHz 11 reserved	
22	PQXDE	Quick PCIX capable detect enable	
23:31		Reserved	

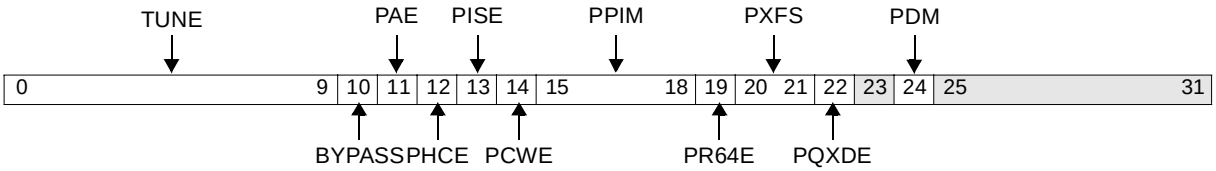


Figure 29-61. Power-On Configuration Register 1 (CPC0_STRP1)

0:9	TUNE	PCI PLL TUNE Bits	See Table 13-5, on page -411 for tune bit setting information.
10	BYPASS	Bypass PCI PLL 0 PCI PLL enabled 1 Bypass PCI PLL	
11	PAE	PCI Arbiter Enable 0 PCI arbiter disabled 1 PCI arbiter enabled	
12	PHCE	PCI Host Configuration Enable 0 PCI host configuration disabled 1 PCI host configuration enabled	
13	PISE	PCI initial sequence enable 0 PCI initial sequence disabled 1 PCI initial sequence enabled	
14	PCWE	PCI local CPU wait enable 0 PCI local CPU wait disabled 1 PCI local CPU wait enabled	
15:18	PPIM	PCI Inbound Map (PIM) Settings 0000 PIM0 off, PIM1 off, PIM2 off 0001 PIM0 4k, PIM1 off, PIM2 off 0010 PIM0 1M, PIM1 off, PIM2 off 0011 PIM0 64M, PIM1 off, PIM2 off 0100 PIM0 4kp, PIM1 off, PIM2 off 0101 PIM0 1Mp, PIM1 off, PIM2 off 0110 PIM0 64Mp, PIM1 off, PIM2 off 0111 PIM0 64k, PIM1 off, PIM2 16k 1000 PIM0 1M, PIM1 off, PIM2 64k 1001 PIM0 64kp, PIM1 off, PIM2 16k 1010 PIM0 1Mp, PIM1 off, PIM2 64k 1011 PIM0 64k, PIM1 off, PIM2 64Kp 1100 PIM0 1M, PIM1 off, PIM2 1Mp 1101 PIM0 1Mp, PIM1 off, PIM2 1Mp 1110 PIM0 1M, PIM1 on, PIM2 off 1111 PIM0 1M, PIM1 on, PIM2 16k	These bits control the default settings for various PIM fields. See *PCI Inbound Map Setting* on page 285.
19	PR64E	PCI initialize Req64 Enable 0 PCI initialize Req64 disabled 1 PCI initialize Req64 enabled	
20:21	PXFS	PCIX Frequency Selection 00 100 - 133MHz 01 66 - 100MHz 10 50 - 66MHz 11 reserved	
22	PQXDE	Quick PCIX capable detect enable 0 Quick PCIX capable detect disabled 1 Quick PCIX capable detect enabled	

23		Reserved
24	PDM	PCIX Driver Mode 0 PCIX driver impedence is 20 ohms for multi point mode 1 PCIX driver impedence is 40 ohms for point to point mode
25:31		Reserved

DCR 0x0E6 Read-Only

See *Power-On Configuration Register 2 (CPC0_STRP2)* on page 296.

Figure 0-15. Power-On Configuration Register 2 (CPC0_STRP2)

0:31		Reserved	Reserved for customer use

Figure 29-62. Power-On Configuration Register 2 (CPC0_STRP2)

0:31		Reserved	Reserved for customer use

DCR 0x0E7 Read-Only

See *Power-On Configuration Register 3 (CPC0_STRP3)* on page 297.

Figure 0-16. Power-On Configuration Register 3 (CPC0_STRP3)

0:31		Reserved	Reserved for customer use

Figure 29-63. Power-On Configuration Register 3 (CPC0_STRP3)

0:31		Reserved	Reserved for customer use

DCR 0x0E0 Read/Write

See *System Configuration Register 0 (CPC0_SYS0)* on page 418.

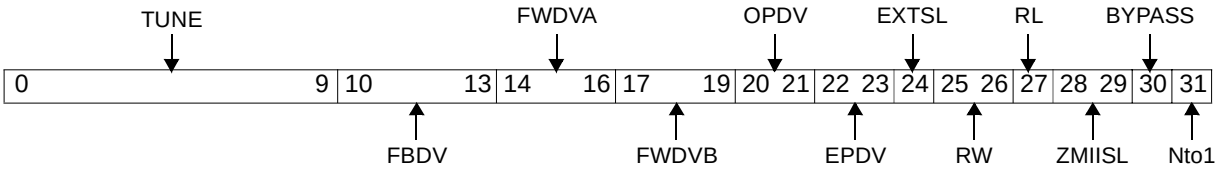


Figure 0-17. System Configuration Register 0 (CPC0_SYS0)

0:9	TUNE	System PLL TUNE bits	See Table 13-5, on page 13-5 for tune bit setting information.
10:13	FBDV	PLL Feedback Divisor 0000 PLL Feedback Divisor = 16 0001 PLL Feedback Divisor = 1 0010 PLL Feedback Divisor = 2 0011 PLL Feedback Divisor = 3 0100 PLL Feedback Divisor = 4 0101 PLL Feedback Divisor = 5 0110 PLL Feedback Divisor = 6 0111 PLL Feedback Divisor = 7 1000 PLL Feedback Divisor = 8 1001 PLL Feedback Divisor = 9 1010 PLL Feedback Divisor = 10 1011 PLL Feedback Divisor = 11 1100 PLL Feedback Divisor = 12 1101 PLL Feedback Divisor = 13 1110 PLL Feedback Divisor = 14 1111 PLL Feedback Divisor = 15	
14:16	FWDVA	PLL Forward Divisor A 000 PLL Forward Divisor A = 8 001 PLL Forward Divisor A = 7 010 PLL Forward Divisor A = 6 011 PLL Forward Divisor A = 5 100 PLL Forward Divisor A = 4 101 PLL Forward Divisor A = 3 110 PLL Forward Divisor A = 2 111 PLL Forward Divisor A = 1	
17:19	FWDVB	PLL Forward Divisor B 000 PLL Forward Divisor B = 8 001 PLL Forward Divisor B = 7 010 PLL Forward Divisor B = 6 011 PLL Forward Divisor B = 5 100 PLL Forward Divisor B = 4 101 PLL Forward Divisor B = 3 110 PLL Forward Divisor B = 2 111 PLL Forward Divisor B = 1	

20:21	OPDV	PLB to OPB Clock Divisor Ratio 00 PLB to OPB Clock Divisor = 1 01 PLB to OPB Clock Divisor = 2 10 PLB to OPB Clock Divisor = 3 11 PLB to OPB Clock Divisor = 4	The OPDV is the divisor ratio between the PLB and OPB clock. For example, if the PLB clock frequency is 133MHz, a divisor ratio of two sets the OPB clock frequency to 66MHz.
22:23	EPDV	EBCO to OPB Clock Divisor Ratio 00 EBCO to OPB Clock Divisor = 1 01 EBCO to OPB Clock Divisor = 2 10 EBCO to OPB Clock Divisor = 3 11 EBCO to OPB Clock Divisor = 4	The EPDV is the divisor ratio between the EBC0 (PerClk) and OPB clock. For example, if the OPB clock frequency is 66MHz, a divisor ratio of two sets the EBC0 clock (PerClk) frequency to 33MHz.
24	EXTSL	EBC0 clock (PerClk) Feedback Path Selection 0 CPU clock selected as PLL feedback path 1 EBC0 clock (PerClk) selected as PLL feedback path	Selecting the CPU clock as the PLL feedback path phase aligns PerClk with the CPU clock. Selecting the PerClk as the PLL feedback path phase aligns PerClk with SysClk.
25:26	RW	ROM Width 00 8-bit ROM 01 16-bit ROM 10 reserved 11 32-bit ROM	
27	RL	ROM Location 0 ROM connected to EBCO 1 ROM connected to PCI	If ROM is connected to EBCO, the real address of the reset vector is 0x1FFFFFFFC. If ROM is connected to PCI, the real address of the reset vector is 0x2FFFFFFFC.
28:29	ZMIISL	ZMII Selection 00 MII mode selected 01 SMII mode selected 10 RMII 10Mb mode selected 11 RMII 100Mb mode selected	See "EMAC-ZMII Bridge Interfaces" on page 21-3.
30	BYPASS	Bypass System PLL 0 System PLL enabled 1 Bypass system PLL	
31	Nto1	CPU:PLB N to 1 clock ratio 0 CPU:PLB clock ratio is N:P where P is greater than 1 1 CPU:PLB clock ratio is N:1	

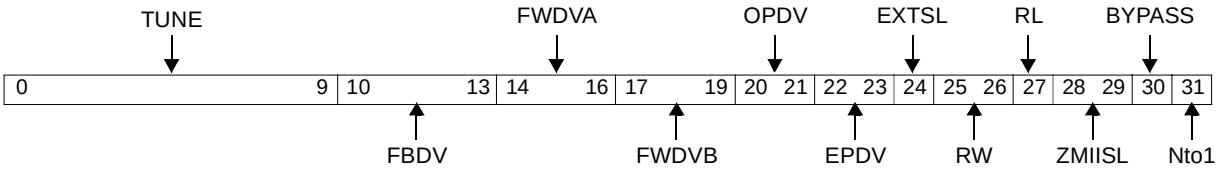


Figure 29-64. System Configuration Register 0 (CPC0_SYS0)

0:9	TUNE	System PLL TUNE bits	See Table 13-5, on page -411 for tune bit setting information.
10:13	FBDV	PLL Feedback Divisor 0000 PLL Feedback Divisor = 16 0001 PLL Feedback Divisor = 1 0010 PLL Feedback Divisor = 2 0011 PLL Feedback Divisor = 3 0100 PLL Feedback Divisor = 4 0101 PLL Feedback Divisor = 5 0110 PLL Feedback Divisor = 6 0111 PLL Feedback Divisor = 7 1000 PLL Feedback Divisor = 8 1001 PLL Feedback Divisor = 9 1010 PLL Feedback Divisor = 10 1011 PLL Feedback Divisor = 11 1100 PLL Feedback Divisor = 12 1101 PLL Feedback Divisor = 13 1110 PLL Feedback Divisor = 14 1111 PLL Feedback Divisor = 15	
14:16	FWDVA	PLL Forward Divisor A 000 PLL Forward Divisor A = 8 001 PLL Forward Divisor A = 7 010 PLL Forward Divisor A = 6 011 PLL Forward Divisor A = 5 100 PLL Forward Divisor A = 4 101 PLL Forward Divisor A = 3 110 PLL Forward Divisor A = 2 111 PLL Forward Divisor A = 1	
17:19	FWDVB	PLL Forward Divisor B 000 PLL Forward Divisor B = 8 001 PLL Forward Divisor B = 7 010 PLL Forward Divisor B = 6 011 PLL Forward Divisor B = 5 100 PLL Forward Divisor B = 4 101 PLL Forward Divisor B = 3 110 PLL Forward Divisor B = 2 111 PLL Forward Divisor B = 1	
20:21	OPDV	PLB to OPB Clock Divisor Ratio 00 PLB to OPB Clock Divisor = 1 01 PLB to OPB Clock Divisor = 2 10 PLB to OPB Clock Divisor = 3 11 PLB to OPB Clock Divisor = 4	The OPDV is the divisor ratio between the PLB and OPB clock. For example, if the PLB clock frequency is 133MHz, a divisor ratio of two sets the OPB clock frequency to 66MHz.
22:23	EPDV	EBCO to OPB Clock Divisor Ratio 00 EBCO to OPB Clock Divisor = 1 01 EBCO to OPB Clock Divisor = 2 10 EBCO to OPB Clock Divisor = 3 11 EBCO to OPB Clock Divisor = 4	The EPDV is the divisor ratio between the EBC0 (PerClk) and OPB clock. For example, if the OPB clock frequency is 66MHz, a divisor ratio of two sets the EBC0 clock (PerClk) frequency to 33MHz.

24	EXTSL	EBC0 clock (PerClk) Feedback Path Selection 0 CPU clock selected as PLL feedback path 1 EBC0 clock (PerClk) selected as PLL feedback path	Selecting the CPU clock as the PLL feedback path phase aligns PerClk with the CPU clock. Selecting the PerClk as the PLL feedback path phase aligns PerClk with SysClk.
25:26	RW	ROM Width 00 8-bit ROM 01 16-bit ROM 10 reserved 11 32-bit ROM	
27	RL	ROM Location 0 ROM connected to EBCO 1 ROM connected to PCI	If ROM is connected to EBCO, the real address of the reset vector is 0x1FFFFFFFC. If ROM is connected to PCI, the real address of the reset vector is 0x2FFFFFFFC.
28:29	ZMIISL	ZMII Selection 00 MII mode selected 01 SMII mode selected 10 RMII 10Mb mode selected 11 RMII 100Mb mode selected	See *EMAC-ZMII Bridge Interfaces* on page 721.
30	BYPASS	Bypass System PLL 0 System PLL enabled 1 Bypass system PLL	
31	Nto1	CPU:PLB N to 1 clock ratio 0 CPU:PLB clock ratio is N:P where P is greater than 1 1 CPU:PLB clock ratio is N:1	

DCR 0x0E1 Read/Write

See *System Configuration Register 1 (CPC0_SYS1)* on page 420.

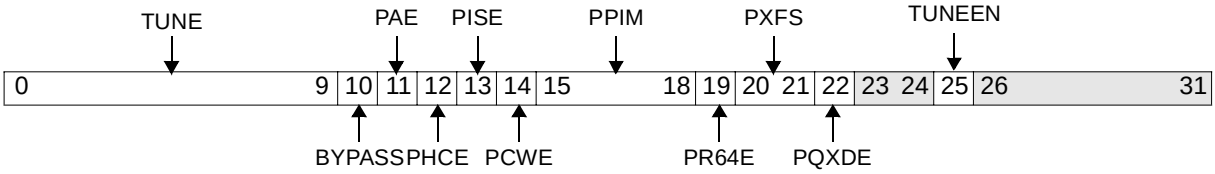


Figure 0-18. System Configuration Register 1 (CPC0_SYS1)

0:9	TUNE	CPC0_SYS1[TUNEEN] 0 CPC0_SYS1[TUNEEN] reads will return the tune bit value which was dynamically determined by the PCI interface at power-on (default). 1 CPC0_SYS1[TUNEEN] reads will return the current value programmed into CPC0_SYS1[TUNE]	CPC0_SYS1[TUNEEN] controls a mux which affects the value read from this field. Also see Table 13-13, on page 13-11 for PCI PLL tune bit setting information.
10	BYPASS	Bypass PCI PLL 0 PCI PLL enabled 1 Bypass PCI PLL	
11	PAE	PCI arbiter enable 0 PCI arbiter disabled 1 PCI arbiter enabled	
12	PHCE	PCI host configuration enable 0 PCI host configuration disabled 1 PCI host configuration enabled	
13	PISE	PCI initial sequence enable 0 PCI initial sequence disabled 1 PCI initial sequence enabled	
14	PCWE	PCI local CPU wait enable 0 PCI local CPU wait disabled 1 PCI local CPU wait enabled	

15:18	PPIM	PCI Inbound Map (PIM) Settings 0000 PIM0 off, PIM1 off, PIM2 off 0001 PIM0 4k, PIM1 off, PIM2 off 0010 PIM0 1M, PIM1 off, PIM2 off 0011 PIM0 64M, PIM1 off, PIM2 off 0100 PIM0 4kp, PIM1 off, PIM2 off 0101 PIM0 1Mp, PIM1 off, PIM2 off 0110 PIM0 64Mp, PIM1 off, PIM2 off 0111 PIM0 64k, PIM1 off, PIM2 16k 1000 PIM0 1M, PIM1 off, PIM2 64k 1001 PIM0 64kp, PIM1 off, PIM2 16k 1010 PIM0 1Mp, PIM1 off, PIM2 64k 1011 PIM0 64k, PIM1 off, PIM2 64Kp 1100 PIM0 1M, PIM1 off, PIM2 1Mp 1101 PIM0 1Mp, PIM1 off, PIM2 1Mp 1110 PIM0 1M, PIM1 on, PIM2 off 1111 PIM0 1M, PIM1 on, PIM2 16k	These bits control the default settings for various PIM fields. See "PCI Inbound Map Setting" on page 8-3.
19	PR64E	PCI initialize Req64 Enable	
20:21	PXFS	PCIX Frequency Selection 00 100 - 133MHz frequency selection 01 66 - 100MHz frequency selection 10 50 - 66MHz frequency selection 11 reserved	
22	PQXDE	Quick PCIX capable detect enable 0 Quick PCIX capable detect disabled 1 Quick PCIX capable detect enabled	
23:24		Reserved	
25	TUNEEN	Custom PCI Tune Bit Enable 0 Use dynamically determined tune bits (default) 1 Use tune bits from CPC0_SYS1[TUNE]	Override the PCI PLL tune bit settings which are dynamically determined at power-on with the value programmed in CPC0_SYS1[TUNE]
26:31		Reserved	

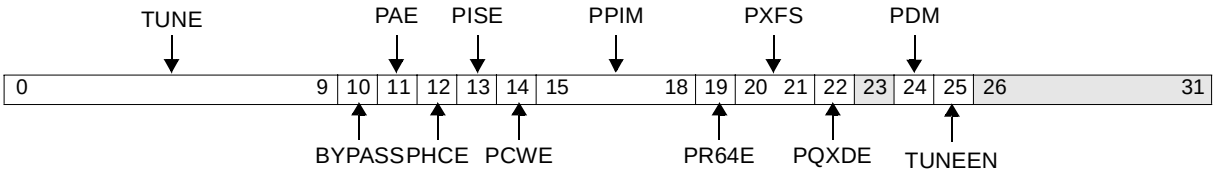


Figure 29-65. System Configuration Register 1 (CPC0_SYS1)

0:9	TUNE	CPC0_SYS1[TUNEEN] 0 CPC0_SYS1[TUNEEN] reads will return the tune bit value which was dynamically determined by the PCI interface at power-on (default). 1 CPC0_SYS1[TUNEEN] reads will return the current value programmed into CPC0_SYS1[TUNE]	CPC0_SYS1[TUNEEN] controls a mux which affects the value read from this field. Also see Table 13-13, on page -417 for PCI PLL tune bit setting information.
10	BYPASS	Bypass PCI PLL 0 PCI PLL enabled 1 Bypass PCI PLL	
11	PAE	PCI arbiter enable 0 PCI arbiter disabled 1 PCI arbiter enabled	
12	PHCE	PCI host configuration enable 0 PCI host configuration disabled 1 PCI host configuration enabled	
13	PISE	PCI initial sequence enable 0 PCI initial sequence disabled 1 PCI initial sequence enabled	
14	PCWE	PCI local CPU wait enable 0 PCI local CPU wait disabled 1 PCI local CPU wait enabled	
15:18	PPIM	PCI Inbound Map (PIM) Settings 0000 PIM0 off, PIM1 off, PIM2 off 0001 PIM0 4k, PIM1 off, PIM2 off 0010 PIM0 1M, PIM1 off, PIM2 off 0011 PIM0 64M, PIM1 off, PIM2 off 0100 PIM0 4kp, PIM1 off, PIM2 off 0101 PIM0 1Mp, PIM1 off, PIM2 off 0110 PIM0 64Mp, PIM1 off, PIM2 off 0111 PIM0 64k, PIM1 off, PIM2 16k 1000 PIM0 1M, PIM1 off, PIM2 64k 1001 PIM0 64kp, PIM1 off, PIM2 16k 1010 PIM0 1Mp, PIM1 off, PIM2 64k 1011 PIM0 64k, PIM1 off, PIM2 64Kp 1100 PIM0 1M, PIM1 off, PIM2 1Mp 1101 PIM0 1Mp, PIM1 off, PIM2 1Mp 1110 PIM0 1M, PIM1 on, PIM2 off 1111 PIM0 1M, PIM1 on, PIM2 16k	These bits control the default settings for various PIM fields. See *PCI Inbound Map Setting* on page 285.
19	PR64E	PCI initialize Req64 Enable 0 PCI initialize Req64 disabled 1 PCI initialize Req64 enabled	

20:21	PXFS	PCIX Frequency Selection 00 100 - 133MHz frequency selection 01 66 - 100MHz frequency selection 10 50 - 66MHz frequency selection 11 reserved	
22	PQXDE	Quick PCIX capable detect enable 0 Quick PCIX capable detect disabled 1 Quick PCIX capable detect enabled	
23		Reserved	
24	PDM	PCIX driver mode enable 0 PCIX driver impedence is 20 ohms for multi point mode 1 PCIX driver impedence is 40 ohms for point to point mode	
25	TUNEEN	Custom PCI Tune Bit Enable 0 Use dynamically determined tune bits (default) 1 Use tune bits from CPC0_SYS1[TUNE]	Override the PCI PLL tune bit settings which are dynamically determined at power-on with the value programmed in CPC0_SYS1[TUNE]
26:31		Reserved	

DCR 0x100, 0x108, 0x110, 0x118 R/W

See *DMA Channel Control Registers (DMA0_CRn)* on page 661.

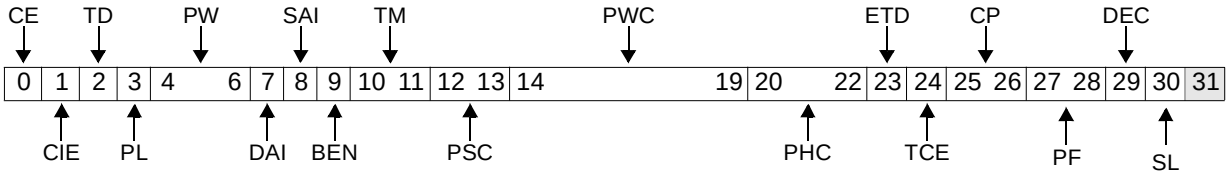


Figure 0-1. DMA Channel Control Registers (DMA0_CR0-DMA0_CR3)

0	CE	Channel Enable 0 Channel is disabled 1 Channel is enabled	This field is automatically cleared when the transfer completes or an error occurs.
1	CIE	Channel Interrupt Enable 0 Disable interrupts from this channel 1 Enable interrupts from this channel	When enabled, interrupts can be generated for terminal count, end of transfer, and errors conditions. Each of these interrupt types must be individually enabled in DMA0_CTn. See "Interrupts."
2	TD	In peripheral mode: 0 Transfers are from memory-to-peripheral 1 Transfers are from peripheral-to-memory In device-paced memory-to-memory mode: 0 Peripheral is at the destination address 1 Peripheral is at the source address	TD is not used (don't care) for software-initiated memory-to-memory transfers.
3	PL	Peripheral Location/Destination Memory Location For memory-to-memory transfers: 0 Destination address located in PLB memory space 1 Destination address located in OPB memory space For peripheral/DPM-to-memory or memory-to-DPM/peripheral transfers: 0 Device located on external bus 1 Device located on the OPB.	
4:6	PW	Peripheral Width/Transfer Width 000 Byte (8 bits) 001 Halfword (16 bits) 010 Word (32 bits) 011 Doubleword (64 bits) 100 Quadword (128 bits)	Transfers involving peripheral, device-paced-memory, and OPB FIFO devices can only be byte, halfword, or word. This limitation also applies to transfers involving EBC memory when SAI=0 or DAI=0. PLB FIFO devices can only be quadword size.

7	DAI	Destination Address Increment 0 Do not increment destination address 1 After each data transfer increment the destination address by: 1, for byte (8-bit) transfers 2, for halfword (16-bit) transfers 4, for word (32-bit) transfers 8, for doubleword (64-bit) transfers 16, for quadword (128-bit) transfers	Valid only when DEC=0. Is required that DAI=1 for peripheral-to-memory and device-paced memory-to-memory transfers since peripheral-to-FIFO type devices or DPM-to-FIFO type devices are not supported.
8	SAI	Source Address Increment 0 Do not increment source address 1 After each data transfer increment the source address by: 1, for byte (8-bit) transfers 2, for halfword (16-bit) transfers 4, for word (32-bit) transfers 8, for doubleword (64-bit) transfers 16, for quadword (128-bit) transfers	Valid only when DEC=0. Is required that SAI=1 for memory-to-peripheral and memory-to-device-paced memory transfers since FIFO-to-peripheral/DPM transfers are not supported.
9	BEN	Buffer Enable 0 Disable DMA 128-byte buffer 1 Enable DMA 128-byte buffer	If BEN=0 discrete read and write operations occur for each data transfer.
10:11	TM	Transfer mode 00 Peripheral 01 Reserved 10 Software-initiated memory-to-memory 11 Device-paced memory-to-memory	
12:13	PSC	Peripheral Setup Cycles 0-3	Number of PerClk cycles that the peripheral bus is idle from the last peripheral bus transaction to DMAAckn becoming active. Used only for the peripheral side of peripheral mode transfers.
14:19	PWC	Peripheral Wait Cycles 0-63	DMAAckn remains active for PWC+1 PerClk cycles. Used only for the peripheral side of peripheral mode transfers.
20:22	PHC	Peripheral Hold Cycles 0-7	The number of PerClk cycles between the time that DMAAckn becomes inactive until the peripheral bus is available for the next bus access. Used only during the peripheral side of peripheral mode transfers.
23	ETD	End-of-Transfer/Terminal Count (EOTn[TCn]) Pin Direction 0 EOTn[TCn] is an EOT input 1 EOTn[TCn] is a TC output	ETD must be set to 1 if the channel is configured for software-initiated memory-to-memory transfers.
24	TCE	Terminal Count (TC) Enable 0 Channel does not stop when TC is reached 1 Channel stops when TC is reached	If TCE=1, it is required that ETD=1.

25:26	CP	Channel Priority 00 Low priority 01 Medium low priority 10 Medium high priority 11 High priority	Actively requesting channels of the same priority are ranked in order by channel number, channel 0 having the highest priority. For more information, see "DMA Arbitration Transfer Priorities."
27:28	PF	Memory Read Prefetch Transfer 00 Prefetch 1 quadword (128-bits) 01 Prefetch 2 quadwords 10 Prefetch 4 quadwords 11 Prefetch 8 quadwords	Used only during memory-to-peripheral and memory to deviced-paced memory transfers. To enable prefetching it is required that BEN=1.
29	DEC	Address Decrement 0 SAI and DAI fields control memory address incrementing. 1 After each data transfer the memory address is decremented by the transfer width.	If DEC=1, it is required that BEN=0. This field is valid only for peripheral mode transfers (TM=00).
30	SL	Source Address Location/Memory Location 0 PLB 1 OPB	For memory-to-memory transfers (TM=1x) SL indicates whether the source memory controller is on the PLB or OPB. For peripheral mode transfers, this field denotes the location of the memory controller.
31		Reserved	

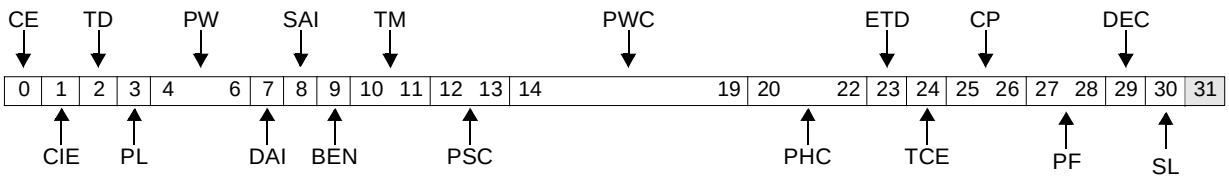


Figure 29-66. DMA Channel Control Registers (DMA0_CR0-DMA0_CR3)

0	CE	Channel Enable 0 Channel is disabled 1 Channel is enabled	This field is automatically cleared when the transfer completes or an error occurs.
1	CIE	Channel Interrupt Enable 0 Disable interrupts from this channel 1 Enable interrupts from this channel	When enabled, interrupts can be generated for terminal count, end of transfer, and errors conditions. Each of these interrupt types must be individually enabled in DMA0_CTn. See "Interrupts."
2	TD	In peripheral mode: 0 Transfers are from memory-to-peripheral 1 Transfers are from peripheral-to-memory In device-paced memory-to-memory mode: 0 Peripheral is at the destination address 1 Peripheral is at the source address	TD is not used (don't care) for software-initiated memory-to-memory transfers.

3	PL	Peripheral Location/Destination Memory Location For memory-to-memory transfers: 0 Destination address located in PLB memory space 1 Destination address located in OPB memory space For peripheral/DPM-to-memory or memory-to-DPM/peripheral transfers: 0 Device located on external bus 1 Device located on the OPB.	
4:6	PW	Peripheral Width/Transfer Width 000 Byte (8 bits) 001 Halfword (16 bits) 010 Word (32 bits) 011 Doubleword (64 bits) 100 Quadword (128 bits)	Transfers involving peripheral, device-paced-memory, and OPB FIFO devices can only be byte, halfword, or word. This limitation also applies to transfers involving EBC memory when SAI=0 or DAI=0. PLB FIFO devices can only be quadword size.
7	DAI	Destination Address Increment 0 Do not increment destination address 1 After each data transfer increment the destination address by: 1, for byte (8-bit) transfers 2, for halfword (16-bit) transfers 4, for word (32-bit) transfers 8, for doubleword (64-bit) transfers 16, for quadword (128-bit) transfers	Valid only when DEC=0. Is required that DAI=1 for peripheral-to-memory and device-paced memory-to-memory transfers since peripheral-to-FIFO type devices or DPM-to-FIFO type devices are not supported.
8	SAI	Source Address Increment 0 Do not increment source address 1 After each data transfer increment the source address by: 1, for byte (8-bit) transfers 2, for halfword (16-bit) transfers 4, for word (32-bit) transfers 8, for doubleword (64-bit) transfers 16, for quadword (128-bit) transfers	Valid only when DEC=0. Is required that SAI=1 for memory-to-peripheral and memory-to-device-paced memory transfers since FIFO-to-peripheral/DPM transfers are not supported.
9	BEN	Buffer Enable 0 Disable DMA 128-byte buffer 1 Enable DMA 128-byte buffer	If BEN=0 discrete read and write operations occur for each data transfer.
10:11	TM	Transfer mode 00 Peripheral 01 Reserved 10 Software-initiated memory-to-memory 11 Device-paced memory-to-memory	
12:13	PSC	Peripheral Setup Cycles 0-3	Number of PerClk cycles that the peripheral bus is idle from the last peripheral bus transaction to DMAAckn becoming active. Used only for the peripheral side of peripheral mode transfers.
14:19	PWC	Peripheral Wait Cycles 0-63	DMAAckn remains active for PWC+1 PerClk cycles. Used only for the peripheral side of peripheral mode transfers.
20:22	PHC	Peripheral Hold Cycles 0-7	The number of PerClk cycles between the time that DMAAckn becomes inactive until the peripheral bus is available for the next bus access. Used only during the peripheral side of peripheral mode transfers.
23	ETD	End-of-Transfer/Terminal Count (EOTn[TCn]) Pin Direction 0 EOTn[TCn] is an EOT input 1 EOTn[TCn] is a TC output	ETD must be set to 1 if the channel is configured for software-initiated memory-to-memory transfers.

24	TCE	Terminal Count (TC) Enable 0 Channel does not stop when TC is reached 1 Channel stops when TC is reached	If TCE=1, it is required that ETD=1.
25:26	CP	Channel Priority 00 Low priority 01 Medium low priority 10 Medium high priority 11 High priority	Actively requesting channels of the same priority are ranked in order by channel number, channel 0 having the highest priority. For more information, see "DMA Arbitration Transfer Priorities."
27:28	PF	Memory Read Prefetch Transfer 00 Prefetch 1 quadword (128-bits) 01 Prefetch 2 quadwords 10 Prefetch 4 quadwords 11 Prefetch 8 quadwords	Used only during memory-to-peripheral and memory to deviced-paced memory transfers. To enable prefetching it is required that BEN=1.
29	DEC	Address Decrement 0 SAI and DAI fields control memory address incrementing. 1 After each data transfer the memory address is decremented by the transfer width.	If DEC=1, it is required that BEN=0. This field is valid only for peripheral mode transfers (TM=00).
30	SL	Source Address Location/Memory Location 0 PLB 1 OPB	For memory-to-memory transfers (TM=1x) SL indicates whether the source memory controller is on the PLB or OPB. For peripheral mode transfers, this field denotes the location of the memory controller.
31		Reserved	

DCR 0x101, 0x109, 0x111, 0x119 R/W

See *DMA Count and Control Registers (DMA0_CTCn)* on page 663.

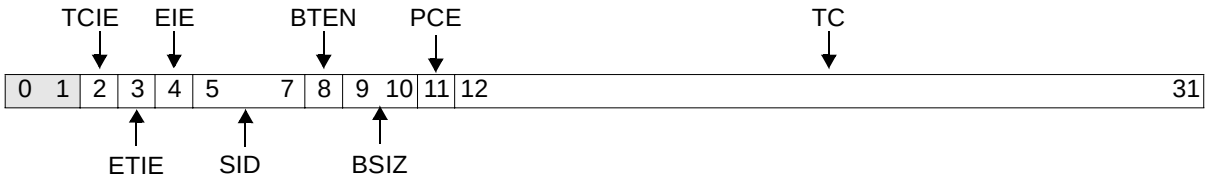


Figure 0-2. DMA Count and Control Registers (DMA0_CT0-DMA0_CT3)

0:1		Reserved	
2	TCIE	Terminal Count Interrupt Enable 0 Disable terminal count interrupts 1 Enable terminal count interrupts	Terminal Count interrupts are further qualified by DMA0_CR0-DMA0_CR3[CIE] and UIC0_ER[DMAn].
3	ETIE	EOT Interrupt Enable 0 Enable end of transfer interrupts 1 Disable end of transfer interrupts	EOT interrupts are further qualified by DMA0_CR0-DMA0_CR3[CIE] and UIC0_ER[DMAn].
4	EIE	Error Interrupt Enable 0 Enable error interrupts 1 Disable error interrupts	Error interrupts are further qualified by DMA0_CR0-DMA0_CR3[CIE] and UIC0_ER[DMAn].
5:7	SID	Subchannel ID	
8	BTEN	Burst Enable 0 Disable bursting 1 Enable bursting	Controls bursting to and from peripheral devices and EBC-attached device-paced memory. See "Peripheral and Deviced-Paced Memory Bursts."
9:10	BSIZ	Burst Size 00 Burst of 2 01 Burst of 4 10 Burst of 8 11 Burst of 16	Determines the burst length used when BEN=1. When BEN=1, TC must be programmed to a multiple of BRSTSIZ. For OPB memory burst transfers, these bits determine the maximum burst size.
11	PCE	Parity Check Enable 0 Disable parity checking 1 Enable parity checking	Enables parity for peripheral mode transfers. See "Data Parity During DMA Peripheral Transfers."
12:31	TC	Transfer Count 0 - 1024K-1	Programming TC=0 results in the maximum count of 1024K transfers.

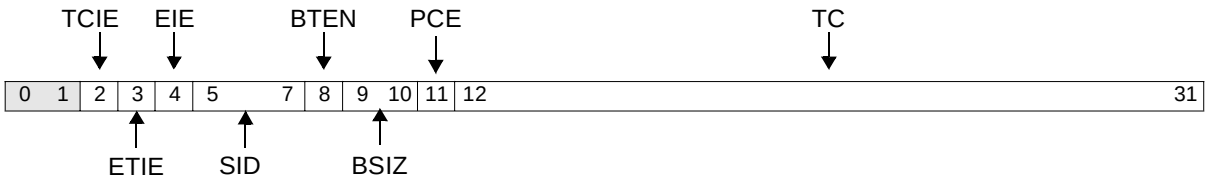


Figure 29-67. DMA Count and Control Registers (DMA0_CT0-DMA0_CT3)

0:1		Reserved	
2	TCIE	Terminal Count Interrupt Enable 0 Disable terminal count interrupts 1 Enable terminal count interrupts	Terminal Count interrupts are further qualified by DMA0_CR0-DMA0_CR3[CIE] and UIC0_ER[DMAn].
3	ETIE	EOT Interrupt Enable 0 Enable end of transfer interrupts 1 Disable end of transfer interrupts	EOT interrupts are further qualified by DMA0_CR0-DMA0_CR3[CIE] and UIC0_ER[DMAn].
4	EIE	Error Interrupt Enable 0 Enable error interrupts 1 Disable error interrupts	Error interrupts are further qualified by DMA0_CR0-DMA0_CR3[CIE] and UIC0_ER[DMAn].
5:7	SID	Subchannel ID	
8	BTEN	Burst Enable 0 Disable bursting 1 Enable bursting	Controls bursting to and from peripheral devices and EBC-attached device-paced memory. See "Peripheral and Deviced-Paced Memory Bursts."
9:10	BSIZ	Burst Size 00 Burst of 2 01 Burst of 4 10 Burst of 8 11 Burst of 16	Determines the burst length used when BEN=1. When BEN=1, TC must be programmed to a multiple of BRSTSIZ. For OPB memory burst transfers, these bits determine the maximum burst size.
11	PCE	Parity Check Enable 0 Disable parity checking 1 Enable parity checking	Enables parity for peripheral mode transfers. See "Data Parity During DMA Peripheral Transfers."
12:31	TC	Transfer Count 0 - 1024K-1	Programming TC=0 results in the maximum count of 1024K transfers.

DCR 0x104, 0x10C, 0x114, 0x11C R/W

See *DMA Destination Address Registers* on page 665.

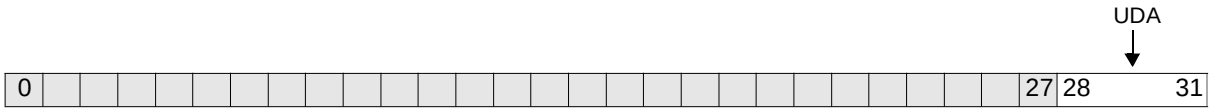


Figure 0-3. DMA Destination Address High Registers (DMA0_DAH0-DMA0_DAH3)

0:27		Reserved
28:31	UDA	Upper 4-bits of destination address for memory-to-memory and peripheral-to-memory transfers.

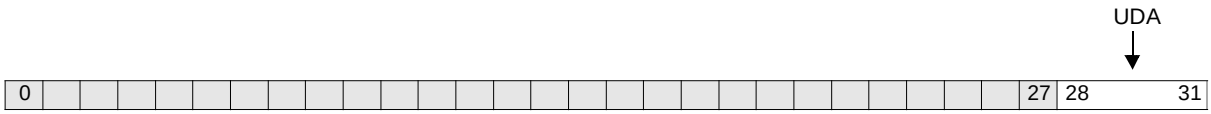


Figure 29-68. DMA Destination Address High Registers (DMA0_DAH0-DMA0_DAH3)

0:27		Reserved
28:31	UDA	Upper 4-bits of destination address for memory-to-memory and peripheral-to-memory transfers.

DCR 0x105, 0x10D, 0x115, 0x11D R/W

See *DMA Destination Address Registers* on page 665.

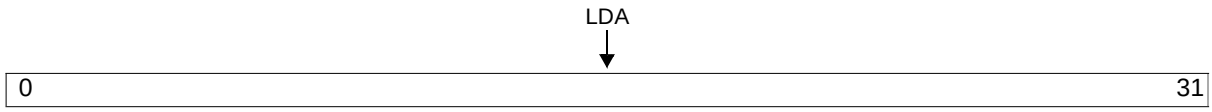


Figure 0-4. DMA Destination Address Low Registers ((DMA0_DAL0-DMA0_DAL3)

0:31	LDA	Lower 32-bits of destination address for memory-to-memory and peripheral-to-memory transfers.

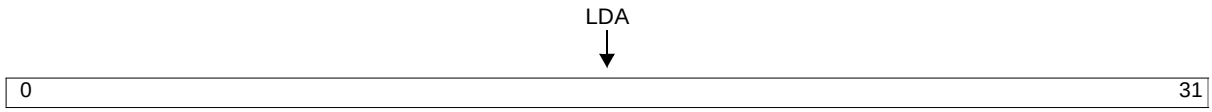


Figure 29-69. DMA Destination Address Low Registers ((DMA0_DAL0-DMA0_DAL3)

0:31	LDA	Lower 32-bits of destination address for memory-to-memory and peripheral-to-memory transfers.

DCR 0x126 R/W

See *DMA Polarity Configuration Register (DMA0_POL)* on page 670.

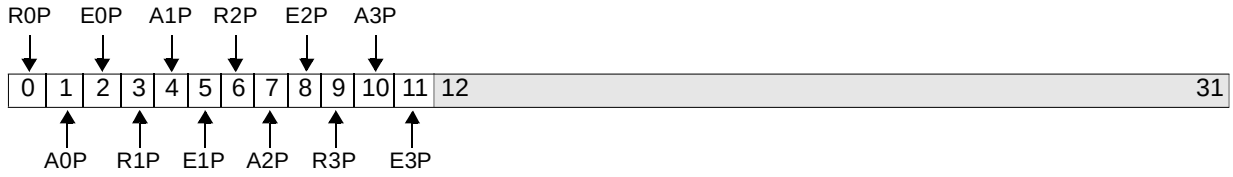


Figure 0-5. DMA Polarity Configuration Register (DMA0_POL)

0	R0P	DMAReq0 Polarity 0 Active high 1 Active low
1	A0P	DMAAck0 Polarity 0 Active high 1 Active low
2	E0P	EOT0[TC0] Polarity 0 Active high 1 Active low
3	R1P	DMAReq1 Polarity 0 Active high 1 Active low
4	A1P	DMAAck1 Polarity 0 Active high 1 Active low
5	E1P	EOT1[TC1] Polarity 0 Active high 1 Active low
6	R2P	DMAReq2 Polarity 0 Active high 1 Active low
7	A2P	DMAAck2 Polarity 0 Active high 1 Active low
8	E2P	EOT2[TC2] Polarity 0 Active high 1 Active low
9	R3P	DMAReq3 Polarity 0 Active high 1 Active low
10	A3P	DMAAck3 Polarity 0 Active high 1 Active low

11	E3P	EOT3[TC3] Polarity 0 Active high 1 Active low
12:31		Reserved

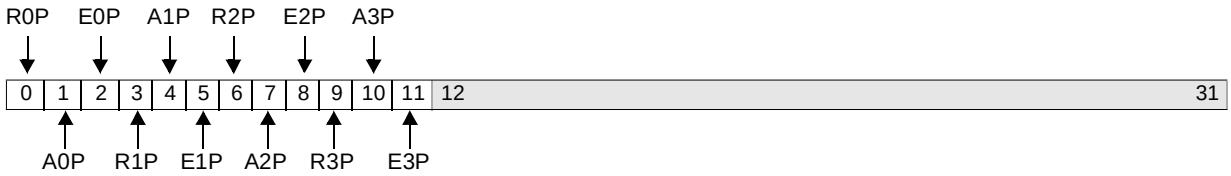


Figure 29-70. DMA Polarity Configuration Register (DMA0_POL)

0	R0P	DMAReq0 Polarity 0 Active high 1 Active low
1	A0P	DMAAck0 Polarity 0 Active high 1 Active low
2	E0P	EOT0[TC0] Polarity 0 Active high 1 Active low
3	R1P	DMAReq1 Polarity 0 Active high 1 Active low
4	A1P	DMAAck1 Polarity 0 Active high 1 Active low
5	E1P	EOT1[TC1] Polarity 0 Active high 1 Active low
6	R2P	DMAReq2 Polarity 0 Active high 1 Active low
7	A2P	DMAAck2 Polarity 0 Active high 1 Active low
8	E2P	EOT2[TC2] Polarity 0 Active high 1 Active low
9	R3P	DMAReq3 Polarity 0 Active high 1 Active low
10	A3P	DMAAck3 Polarity 0 Active high 1 Active low
11	E3P	EOT3[TC3] Polarity 0 Active high 1 Active low
12:31		Reserved

DCR 0x102, 0x10A, 0x112, 0x11A R/W

See *DMA Source Address Registers)* on page 664.

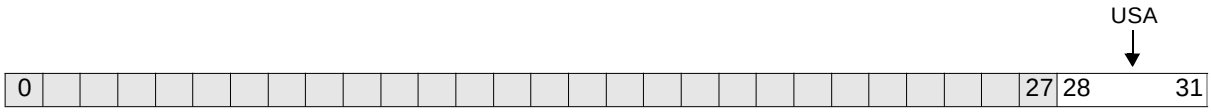


Figure 0-6. DMA Source Address High Registers (DMA0-SAH0-DMA0-SAH3)

0:27		Reserved
28:31	USA	Upper 4-bits of source address for memory-to-memory and memory-to-peripheral transfers.

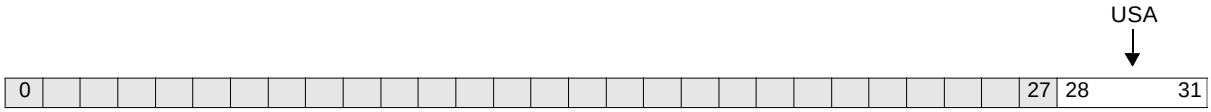


Figure 29-71. DMA Source Address High Registers (DMA0-SAH0-DMA0-SAH3)

0:27		Reserved
28:31	USA	Upper 4-bits of source address for memory-to-memory and memory-to-peripheral transfers.

DCR 0x103, 0x10B, 0x113, 0x11B R/W

See *DMA Source Address Registers)* on page 664.

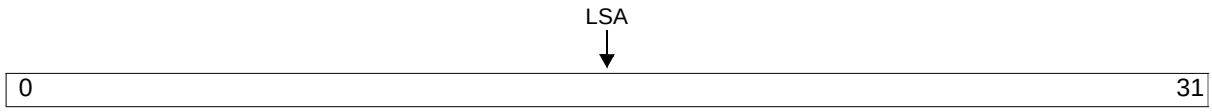


Figure 0-7. DMA Source Address Low Registers (DMA0-SAL0-DMA0-SAL3)

0:31	LSA	Lower 32-bits of source address for memory-to-memory and memory-to-peripheral transfers.

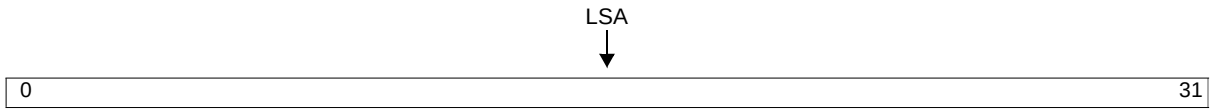


Figure 29-72. DMA Source Address Low Registers (DMA0-SAL0-DMA0-SAL3)

0:31	LSA	Lower 32-bits of source address for memory-to-memory and memory-to-peripheral transfers.

DCR 0x123 R/W

See *DMA Scatter/Gather Command Register (DMA0_SGC)* on page 668.

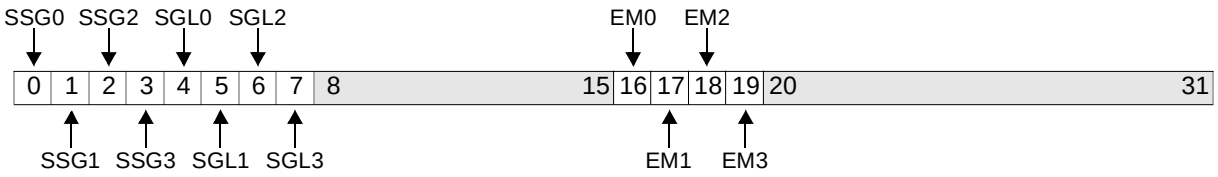


Figure 0-8. DMA Scatter/Gather Command Register (DMA0_SGC)

0:3	SSG[0:3]	Start Scatter/Gather for channels 0-3 0 Scatter/gather support is disabled 1 Scatter/gather support is enabled	To start a scatter/gather operation for channel n, EM[n] must also be set.
4:7	SGL[0:3]	Scatter/Gather Descriptor Table Location for channels 0-3 0 PLB memory space 1 OPB memory space	A channel's descriptor table may not cross between PLB and OPB address space. See "Scatter/Gather Transfers" on page 19-21.
8:15		Reserved	
16:19	EM[0:3]	Enable Mask for channels 0-3 0 Writes to SSG[n] and SGL[n] are ignored 1 Allow writing to SSG[n] and SGL[n]	To write SSG[n] or SGL[n], EM[n] must be set. Otherwise, writing SSG[n] or SGL[n] has no effect.
20:31		Reserved	

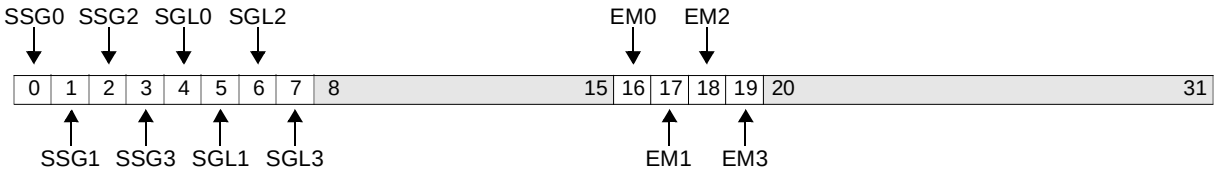


Figure 29-73. DMA Scatter/Gather Command Register (DMA0_SGC)

0:3	SSG[0:3]	Start Scatter/Gather for channels 0-3 0 Scatter/gather support is disabled 1 Scatter/gather support is enabled	To start a scatter/gather operation for channel n, EM[n] must also be set.
4:7	SGL[0:3]	Scatter/Gather Descriptor Table Location for channels 0-3 0 PLB memory space 1 OPB memory space	A channel's descriptor table may not cross between PLB and OPB address space. See *Scatter/Gather Transfers* on page 674.
8:15		Reserved	
16:19	EM[0:3]	Enable Mask for channels 0-3 0 Writes to SSG[n] and SGL[n] are ignored 1 Allow writing to SSG[n] and SGL[n]	To write SSG[n] or SGL[n], EM[n] must be set. Otherwise, writing SSG[n] or SGL[n] has no effect.
20:31		Reserved	

DCR 0x106, 0x10E, 0x116, 0x11E R/W

See *DMA Scatter/Gather Descriptor Address Registers* on page 666.

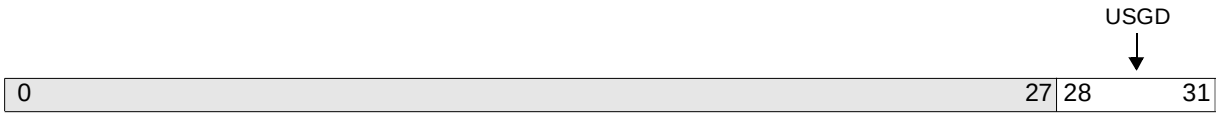


Figure 0-9. DMA Scatter/Gather Descriptor Address High Registers (DMA0_SGH0-DMA0_SGH3)

0:27		Reserved
28:31	USGD	Upper 4-bits of address of next scatter/gather descriptor table.

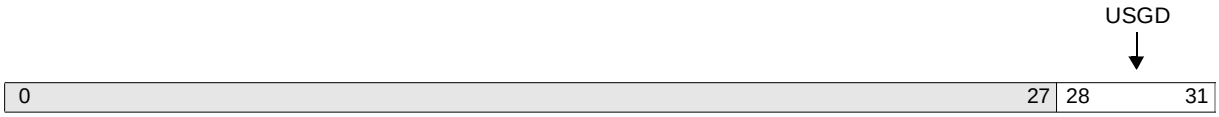


Figure 29-74. DMA Scatter/Gather Descriptor Address High Registers (DMA0_SGH0-DMA0_SGH3)

0:27		Reserved
28:31	USGD	Upper 4-bits of address of next scatter/gather descriptor table.

DCR 0x107, 0x10F, 0x117, 0x11F R/W

See *DMA Scatter/Gather Descriptor Address Registers* on page 666.

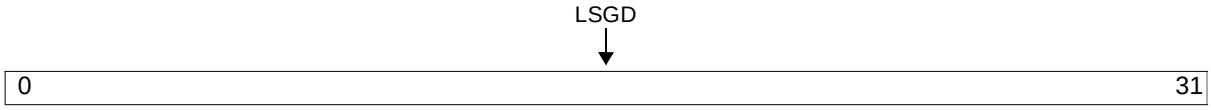


Figure 0-10. DMA Scatter/Gather Descriptor Address Low Registers (DMA0_SGL0-DMA0_SGL3)

0:31	LSGD	Lower 32-bits of address of next scatter/gather descriptor table.

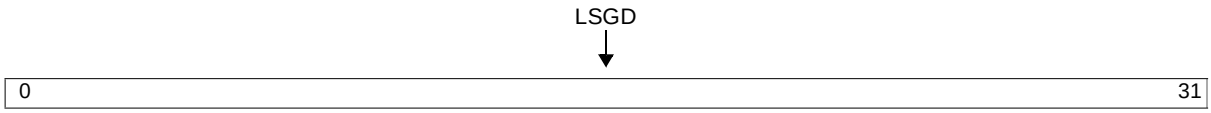


Figure 29-75. DMA Scatter/Gather Descriptor Address Low Registers (DMA0_SGL0-DMA0_SGL3)

0:31	LSGD	Lower 32-bits of address of next scatter/gather descriptor table.

DCR 0x125 R/W

See *DMA Sleep Mode Register (DMA0_SLP)* on page 669.

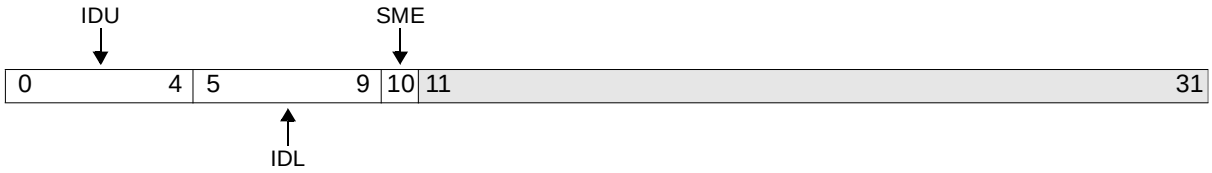


Figure 0-11. DMA Sleep Mode Register (DMA0_SLP)

0:4	IDU	Idle Timer Upper 0-31	Upper 5 bits of the idle timer.
5:9	IDL	Idle Timer Lower Hardcoded to 0b11111	Lower 5-bit portion of the idle timer. Writing this field has no effect.
10	SME	Sleep Mode Enable 0 Sleep disabled 1 Sleep enabled	If SME=1, also set CPM0_ER[DMA] to enable the Clock and Power Management macro to put the DMA controller to sleep.
11:31		Reserved	

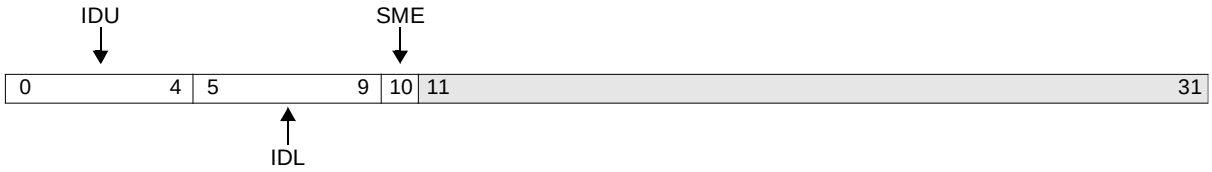


Figure 29-76. DMA Sleep Mode Register (DMA0_SLP)

0:4	IDU	Idle Timer Upper 0-31	Upper 5 bits of the idle timer.
5:9	IDL	Idle Timer Lower Hardcoded to 0b11111	Lower 5-bit portion of the idle timer. Writing this field has no effect.
10	SME	Sleep Mode Enable 0 Sleep disabled 1 Sleep enabled	If SME=1, also set CPM0_ER[DMA] to enable the Clock and Power Management macro to put the DMA controller to sleep.
11:31		Reserved	

DCR 0x120 Read/Clear

See *DMA Status Register (DMA0_SR)* on page 667.

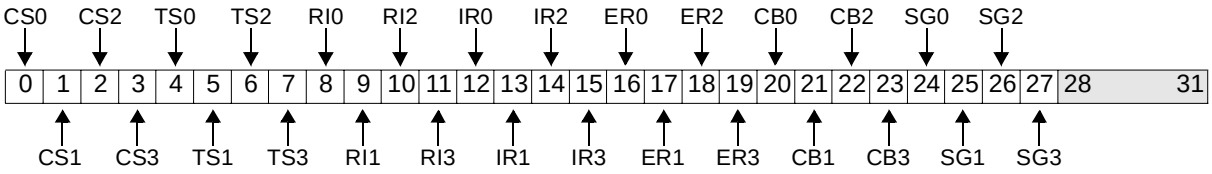


Figure 0-12. DMA Status Register (DMA0_SR)

0:3	CS[0:3]	Channel 0-3 Terminal Count Status 0 Terminal count has not occurred 1 Terminal count has been reached	Set when the transfer count reaches 0 and the DMA0_CRn(TCE) bit is set.
4:7	TS[0:3]	Channel 0-3 End of Transfer Status 0 End of transfer has not been requested 1 End of transfer has been requested	Only valid for channels with DMA0_CRn[ETD]=0 (no memory-to-memory transfers).
8:11	RI[0:3]	Channel 0-3 Error Status 0 No error 1 Error occurred	See Errors section for more information.
12:15	IR[0:3]	Internal DMA Request 0 No internal DMA request pending 1 Internal DMA request is pending	
16:19	ER[0:3]	External DMA Request 0 No external DMA request pending 1 External DMA request is pending	
20:23	CB[0:3]	Channel Busy 0 Channel is idle 1 Channel currently active	During catter/gather fetches, these bits are active.
24:27	SG[0:3]	Scatter/Gather Status 0 No scatter/gather operation in progress 1 Scatter/gather operation in progress	For multiple scatter/gather links, the scatter/gather status bit is set when the first link is loaded and is kept adctive until the last link is completed.
28:31		Reserved	

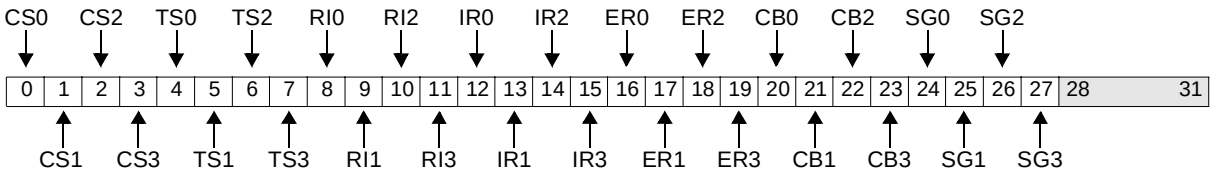


Figure 29-77. DMA Status Register (DMA0_SR)

0:3	CS[0:3]	Channel 0-3 Terminal Count Status 0 Terminal count has not occurred 1 Terminal count has been reached	Set when the transfer count reaches 0 and the DMA0_CRn(TCE) bit is set.
4:7	TS[0:3]	Channel 0-3 End of Transfer Status 0 End of transfer has not been requested 1 End of transfer has been requested	Only valid for channels with DMA0_CRn[ETD]=0 (no memory-to-memory transfers).
8:11	RI[0:3]	Channel 0-3 Error Status 0 No error 1 Error occurred	See Errors section for more information.
12:15	IR[0:3]	Internal DMA Request 0 No internal DMA request pending 1 Internal DMA request is pending	
16:19	ER[0:3]	External DMA Request 0 No external DMA request pending 1 External DMA request is pending	
20:23	CB[0:3]	Channel Busy 0 Channel is idle 1 Channel currently active	During catter/gather fetches, these bits are active.
24:27	SG[0:3]	Scatter/Gather Status 0 No scatter/gather operation in progress 1 Scatter/gather operation in progress	For multiple scatter/gather links, the scatter/gather status bit is set when the first link is loaded and is kept adctive until the last link is completed.
28:31		Reserved	

DCR Accessed using EBC0_CFGADDR; EBC0_CFGDATA; Offset 0x10–0x17 R/W

See *Peripheral Bank 0–7 Access Parameters (EBC0_B0AP-EBC0_B7AP)* on page 882.

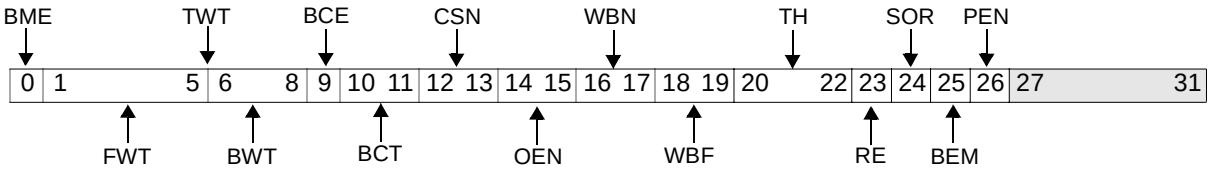


Figure 0-1. Peripheral Bank Access Parameters (EBC0_B0AP-EBC0_B7AP)

0	BME	Burst Mode Enable 0 Bursting is disabled 1 Bursting is enabled	
1:8	TWT	Transfer Wait 0 to 255 PerClk cycles	If bursting is disabled, BME=0, wait states on all non-burst transfers.
1:5	FWT	First Wait 0 to 31 PerClk cycles	If bursting is enabled, BME=1, number of wait states on the first transfer of a burst.
6:8	BWT	Burst Wait 0 to 7 PerClk cycles	If bursting is enabled, BME=1, number of wait states on non-first transfers of a burst.
9	BCE	Fixed Length Burst Reads 0 Disabled 1 Enabled	If BCE=1, all burst read operations consist of exactly BCT transfers. When BCE=0, burst read operations may read more than the requested amount of data.
10:11	BCT	Fixed Length Burst Count 00 2 transfers 01 4 transfers 10 8 transfers 11 16 transfers	The amount of data transferred depends upon the programmed bank width, EBC0_BnCR[BW].
12:13	CSN	Chip Select On Timing 0 to 3 PerClk cycles.	Number of cycles from peripheral address driven to $\overline{\text{PerCSn}}$ low.
14:15	OEN	Output Enable On TIming 0 to 3 PerClk cycles	Number of cycles from $\overline{\text{PerCSn}}$ low to $\overline{\text{PerOE}}$ low.
16:17	WBN	Write Byte Enable On Timing 0 to 3 PerClk cycles	If BEM=0, number of cycles from $\overline{\text{PerCSn}}$ low to $\overline{\text{PerWBE0:3}}$ active.
18:19	WBF	Write Byte Enable Off Timing (If bursting and ready pin input are disabled) 0 to 3 PerClk cycles	If BEM=0 and RE=0, number of cycles $\overline{\text{PerWBE0:3}}$ becomes inactive prior to $\overline{\text{PerCSn}}$ inactive.
20:22	TH	Transfer Hold 0 to 7 PerClk cycles	Contains the number of hold cycles inserted at the end of a transfer. Hold cycles insert idle bus cycles between transfers to enable slow peripherals to remove data from the data bus before the next transfer begins.

23	RE	Ready Enable 0 PerReady input is disabled 1 PerReady input is enabled	Set for Device Paced Transfers
24	SOR	Sampling of Ready for Device Paced 0 Ready is asserted by the device one clock before data is ready on the external bus 1 Ready is asserted by the device on the same clock that data is ready on the external bus	
25	BEM	Byte Enable Mode 0 PerWBE0:3 pins are only active on write cycles 1 PerWBE0:3 pins are active for both read and write cycles	
26	PEN	Parity Enable 0 Disable Parity checking 1 Enable Parity checking	The EBCO implements odd parity.
27:31		Reserved	

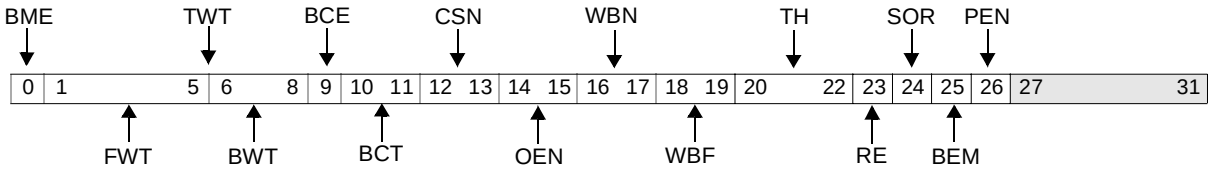


Figure 29-78. Peripheral Bank Access Parameters (EBC0_B0AP-EBC0_B7AP)

0	BME	Burst Mode Enable Bursting is disabled Bursting is enabled	
1:8	TWT	Transfer Wait 0 to 255 PerClk cycles	If bursting is disabled, BME=0, wait states on all non-burst transfers.
1:5	FWT	First Wait 0 to 31 PerClk cycles	If bursting is enabled, BME=1, number of wait states on the first transfer of a burst.
6:8	BWT	Burst Wait 0 to 7 PerClk cycles	If bursting is enabled, BME=1, number of wait states on non-first transfers of a burst.
9	BCE	Fixed Length Burst Reads Disabled Enabled	If BCE=1, all burst read operations consist of exactly BCT transfers. When BCE=0, burst read operations may read more than the requested amount of data.
10:11	BCT	Fixed Length Burst Count 2 transfers 4 transfers 8 transfers 16 transfers	The amount of data transferred depends upon the programmed bank width, EBC0_BnCR[BW].
12:13	CSN	Chip Select On Timing 0 to 3 PerClk cycles.	Number of cycles from peripheral address driven to PerCSn low.

14:15	OEN	Output Enable On TIming 0 to 3 PerClk cycles	Number of cycles from $\overline{\text{PerCSn}}$ low to $\overline{\text{PerOE}}$ low.
16:17	WBN	Write Byte Enable On Timing 0 to 3 PerClk cycles	If BEM=0, number of cycles from $\overline{\text{PerCSn}}$ low to $\overline{\text{PerWBE0:3}}$ active.
18:19	WBF	Write Byte Enable Off Timing (If bursting and ready pin input are disabled) 0 to 3 PerClk cycles	If BEM=0 and RE=0, number of cycles $\overline{\text{PerWBE0:3}}$ becomes inactive prior to $\overline{\text{PerCSn}}$ inactive.
20:22	TH	Transfer Hold 0 to 7 PerClk cycles	Contains the number of hold cycles inserted at the end of a transfer. Hold cycles insert idle bus cycles between transfers to enable slow peripherals to remove data from the data bus before the next transfer begins.
23	RE	Ready Enable PerReady input is disabled PerReady input is enabled	Set for Device Paced Transfers
24	SOR	Sampling of Ready for Device Paced Ready is asserted by the device one clock before data is ready on the external bus Ready is asserted by the device on the same clock that data is ready on the external bus	
25	BEM	Byte Enable Mode $\overline{\text{PerWBE0:3}}$ pins are only active on write cycles $\overline{\text{PerWBE0:3}}$ pins are active for both read and write cycles	
26	PEN	Parity Enable Disable Parity checking Enable Parity checking	The EBCO implements odd parity.
27:31		Reserved	

DCR Accessed using EBC0_CFGADDR; EBC0_CFGDATA; Offset 0x00–0x07 R/W

See *Peripheral Bank Configuration Registers (EBC0_B0CR-EBC0_B7CR)* on page 880.

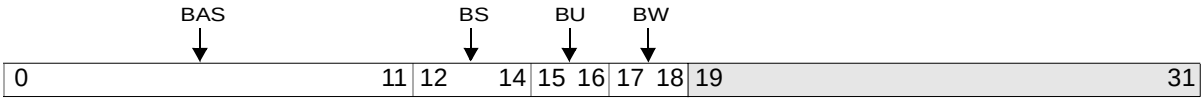


Figure 0-2. Peripheral Bank Configuration Registers (EBC0_B0CR-EBC0_B7CR)

0:11	BAS	Base Address Select	Specifies the bank starting address, which must be a multiple of the bank size.
12:14	BS	Bank Size 000 1 MB bank 001 2 MB bank 010 4 MB bank 011 8 MB bank 100 16 MB bank 101 32 MB bank 110 64 MB bank 111 128 MB bank	
15:16	BU	Bank Usage 00 Disabled 01 Read-only 10 Write-only 11 Read/Write	Specifies the type of accesses allowed for the bank. A protect error occurs if a write is attempted to a read-only bank or a read is attempted from a write-only bank.
17:18	BW	Bus Width 00 8-bit bus 01 16-bit bus 10 Reserved 11 32-bit bus	
19:31		Reserved	

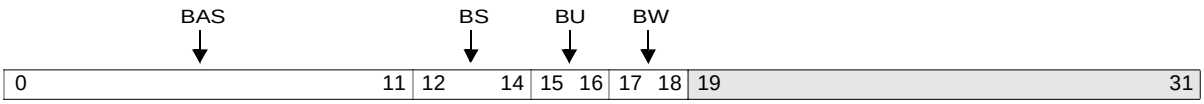


Figure 29-79. Peripheral Bank Configuration Registers (EBC0_B0CR-EBC0_B7CR)

0:11	BAS	Base Address Select	Specifies the bank starting address, which must be a multiple of the bank size.

12:14	BS	Bank Size 000 1 MB bank 001 2 MB bank 010 4 MB bank 011 8 MB bank 100 16 MB bank 101 32 MB bank 110 64 MB bank 111 128 MB bank	
15:16	BU	Bank Usage 00 Disabled 01 Read-only 10 Write-only 11 Read/Write	Specifies the type of accesses allowed for the bank. A protect error occurs if a write is attempted to a read-only bank or a read is attempted from a write-only bank.
17:18	BW	Bus Width 00 8-bit bus 01 16-bit bus 10 Reserved 11 32-bit bus	
19:31		Reserved	

DCR Accessed using EBC0_CFGADDR; EBC0_CFGDATA; Offset 0x20 R

See *Peripheral Bus Error Address Register (EBC0_BEAR)* on page 886.

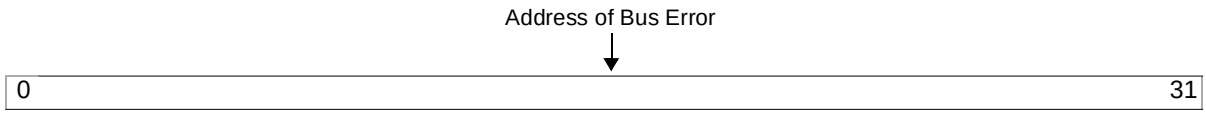


Figure 0-3. Peripheral Bus Error Address Register (EBC0_BEAR)

0:31		Address of Bus Error

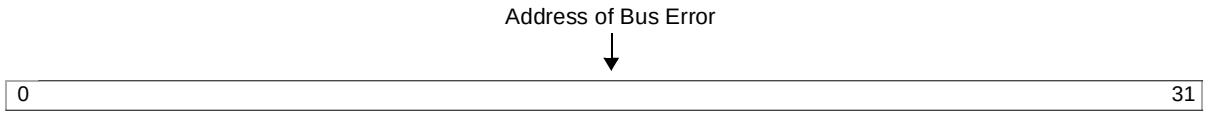


Figure 29-80. Peripheral Bus Error Address Register (EBC0_BEAR)

0:31		Address of Bus Error

DCR Accessed using EBC0_CFGADDR; EBC0_CFGDATA; Offset 0x21 R/W

See *Peripheral Bus Error Status Register (EBC0_BESR)* on page 886.

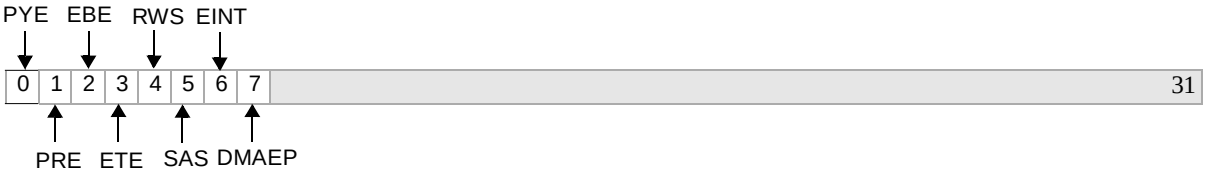


Figure 0-4. Bus Error Status Register (EBC0_BESR)

0	PYE	Parity Error Detected on Read 0 No Parity Error Detected 1 Parity Error Detected	O_ebco_opb_errAck is asserted.
1	PRE	Bank Protection Error 0 No Protection Error 1 Protection Error	O_ebco_opb_errAck is asserted for read or write.
2	EBE	PerErr 0 No PerErr Detected during EBCO transaction. 1 PerErr Detected during EBCO transaction.	If PerErr is detected at any other time during a read, errAck is not asserted. ErrAck is not asserted for write operations.
3	ETE	External Bus Timeout Error 0 No External Bus Timeout Error Detected 1 External Bus Timeout Error Detected	See PDT bit in Figure 27.5.6 on page 27-36 for timeout enable information.
4	RWS	Read/write Error Status for 0 Errant operation was a write operation 1 Errant operation was a read operation	
5	SAS	Sequential Address status 0 Sequential Address Not Active during error detection 1 Sequential Address Active during error detection	
6	EINT	Error Interrupt This bit is a logical “OR” of the four different errors that may be reported and indicates an interrupt has been asserted to the Interrupt Controller	
7	DMAEP	DMA External Peripheral Error 0 No error 1 Error	DMA external peripheral error detected during EBCO transaction.
8:31		Reserved	

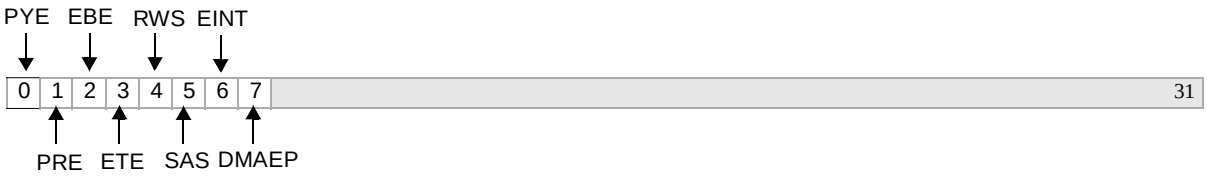


Figure 29-81. Bus Error Status Register (EBC0_BESR)

0	PYE	Parity Error Detected on Read 0 No Parity Error Detected 1 Parity Error Detected	O_ebco_opb_errAck is asserted.
1	PRE	Bank Protection Error 0 No Protection Error 1 Protection Error	O_ebco_opb_errAck is asserted for read or write.
2	EBE	PerErr 0 No PerErr Detected during EBCO transaction. 1 PerErr Detected during EBCO transaction.	If PerErr is detected at any other time during a read, errAck is not asserted. ErrAck is not asserted for write operations.
3	ETE	External Bus Timeout Error 0 No External Bus Timeout Error Detected 1 External Bus Timeout Error Detected	See PDT bit in *Figure 27.5.6* on page 887 for time-out enable information.
4	RWS	Read/write Error Status for 0 Errant operation was a write operation 1 Errant operation was a read operation	
5	SAS	Sequential Address status 0 Sequential Address Not Active during error detection 1 Sequential Address Active during error detection	
6	EINT	Error Interrupt This bit is a logical "OR" of the four different errors that may be reported and indicates an interrupt has been asserted to the Interrupt Controller	
7	DMAEP	DMA External Peripheral Error 0 No error 1 Error	DMA external peripheral error detected during EBCO transaction.
8:31		Reserved	

DCR Accessed using EBC0_CFGADDR; EBC0_CFGDATA; Offset 0x23 R/W

See *EBC Configuration Register (EBC0_CFG)* on page 887.

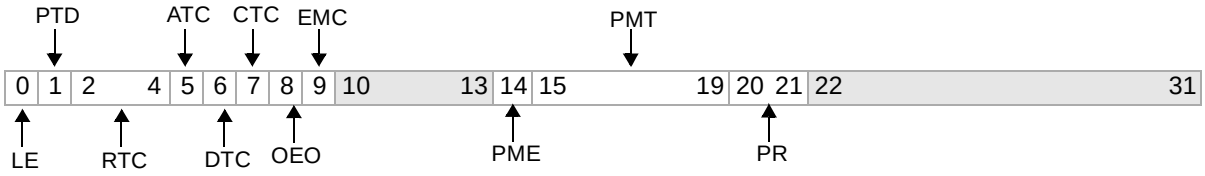


Figure 0-5. EBC Configuration Register (EBC0_CFG)

0	LE	Lock Error 0 Do not lock error 1 Lock error	This bit is used for error locking in the EBC. If this bit is set, then the EBC will latch the first error detected and save it with associated parameters and address. If this bit is not set, then the EBC will continue to latch new errors and associated parameters and addresses that will overwrite previous errors.
1	PTD	Device-Paced Time-out Disable 0 Enabled time-outs 1 Disable time-outs	This bit is valid only when EBC0_BnAP[RE]=1. If PTD=1, the EBC waits indefinitely for assertion of PerReady during device-paced accesses.
2:4	RTC	Ready Timeout Count 000 16 PerClk cycles 001 32 PerClk cycles 010 64 PerClk cycles 011 128 PerClk cycles 100 256 PerClk cycles 101 512 PerClk cycles 110 1024 PerClk cycles 111 2048 PerClk cycles	When PTD=0, the number of clock cycles from the assertion of PerAddr0:31 until a time-out error occurs.
5	ATC	Address Bus High Impedance Control 0 External address bus is high impedance when EBC is idle 1 External address bus drive previous value when EBC is idle and has ownership of the peripheral interface	Default after reset is ATC=1. For details on how the ATC affects driver enables, See “Driver Enables” on page 27-4.
6	DTC	Data Bus High Impedance Control 0 External data bus is high impedance when EBC is idle 1 External data bus drive previous write data value when EBC is idle and has ownership of the peripheral interface	Default after reset is DTC=1. For details on how the DTC affects driver enables, See “Driver Enables” on page 27-4.

7	CTC	Control Signal High Impedance Control 0 External bus Control Signals are high impedance when EBC is idle 1 External bus Control Signals are driven inactive and held when EBC is idle and has ownership of the peripheral interface	Default after reset is CTC=1. For details on how the CTC affects driver enables, See "Driver Enables" on page 27-4.
8	OEO	External Bus Override High Impedance Control 0 External Bus Output Enable Override Disabled 1 External Bus Output Enable Override Enabled	Default after reset is OEO=1. For details on how the OEO affects driver enables, See "Driver Enables" on page 27-4.
9	EMC	External Master High Impedance Control 0 High impedance all EBC outputs when HOLD_ACK = 1. 1 High impedance all EBC outputs when HOLD_ACK = 1 except PerCS0:7 are always driven.	Default after reset is EMC=1. For details, See "Driver Enables" on page 27-4.
10:13		Reserved	
14	PME	Power Management Enable 0 Disabled 1 Enabled	
15:19	PMT	Power Management Timer 0-31	The EBC makes a sleep request to the Clock and Power Management unit when PME=1 and the EBC has been idle for 32*PMT PerClk cycles.
20:21	PR	Pending Request Timer 00 -16 01 - 32 10 - 64 11 - 128	Number of PerClk cycles to wait for a retried OPB operation[1] to return before relinquishing external bus ownership back to the external master.
22:31		Reserved	

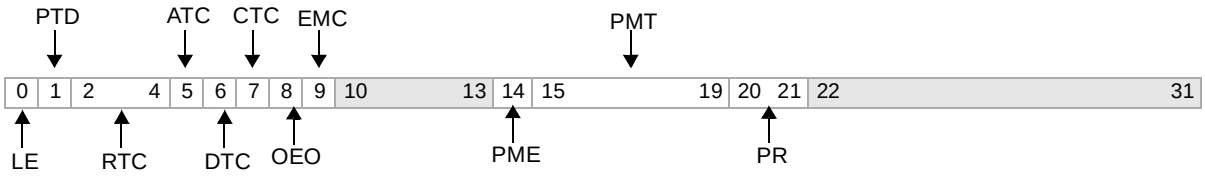


Figure 29-82. EBC Configuration Register (EBC0_CFG)

0	LE	Lock Error 0 Do not lock error 1 Lock error	This bit is used for error locking in the EBC. If this bit is set, then the EBC will latch the first error detected and save it with associated parameters and address. If this bit is not set, then the EBC will continue to latch new errors and associated parameters and addresses that will overwrite previous errors.
1	PTD	Device-Paced Time-out Disable 0 Enabled time-outs 1 Disable time-outs	This bit is valid only when EBC0_BnAP[RE]=1. If PTD=1, the EBC waits indefinitely for assertion of PerReady during device-paced accesses.
2:4	RTC	Ready Timeout Count 000 16 PerClk cycles 001 32 PerClk cycles 010 64 PerClk cycles 011 128 PerClk cycles 100 256 PerClk cycles 101 512 PerClk cycles 110 1024 PerClk cycles 111 2048 PerClk cycles	When PTD=0, the number of clock cycles from the assertion of PerAddr0:31 until a time-out error occurs.
5	ATC	Address Bus High Impedance Control 0 External address bus is high impedance when EBC is idle 1 External address bus drive previous value when EBC is idle and has ownership of the peripheral interface	Default after reset is ATC=1. For details on how the ATC affects driver enables, See *Driver Enables* on page 857.
6	DTC	Data Bus High Impedance Control 0 External data bus is high impedance when EBC is idle 1 External data bus drive previous write data value when EBC is idle and has ownership of the peripheral interface	Default after reset is DTC=1. For details on how the DTC affects driver enables, See *Driver Enables* on page 857.
7	CTC	Control Signal High Impedance Control 0 External bus Control Signals are high impedance when EBC is idle 1 External bus Control Signals are driven inactive and held when EBC is idle and has ownership of the peripheral interface	Default after reset is CTC=1. For details on how the CTC affects driver enables, See *Driver Enables* on page 857.
8	OEO	External Bus Override High Impedance Control 0 External Bus Output Enable Override Disabled 1 External Bus Output Enable Override Enabled	Default after reset is OEO=1. For details on how the OEO affects driver enables, See *Driver Enables* on page 857.
9	EMC	External Master High Impedance Control 0 High impedance all EBC outputs when HOLD_ACK = 1. 1 High impedance all EBC outputs when HOLD_ACK = 1 except PerCS0:7 are always driven.	Default after reset is EMC=1. For details, See *Driver Enables* on page 857.

10:13		Reserved	
14	PME	Power Management Enable 0 Disabled 1 Enabled	
15:19	PMT	Power Management Timer 0-31	The EBC makes a sleep request to the Clock and Power Management unit when PME=1 and the EBC has been idle for 32*PMT PerClk cycles.
20:21	PR	Pending Request Timer 00 -16 01 - 32 10 - 64 11 - 128	Number of PerClk cycles to wait for a retried OPB operation[1] to return before relinquishing external bus ownership back to the external master.
22:31		Reserved	

DCR 0x012 R/W

See *EBC Address Register (EBC0_CFGADDR)* on page 880.

This register is used to determine offsets for the indirectly-accessed EBC DCRs.

Figure 0-6. EBC Address Register (EBC0_CFGADDR)

0:26		Reserved
27:31	DCRA	DCR Address Offset

Figure 29-83. EBC Address Register (EBC0_CFGADDR)

0:26		Reserved
27:31	DCRA	DCR Address Offset

DCR 0x013 R/W

See *EBC Data Register (EBC0_DATA)* on page 880.

This register is used to access the indirectly-accessed EBC DCRs.

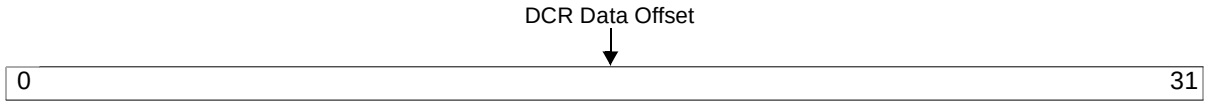


Figure 0-7. EBC Data Register (EBC0_CFGDATA)

0:31	DCRD	DCR Data Port

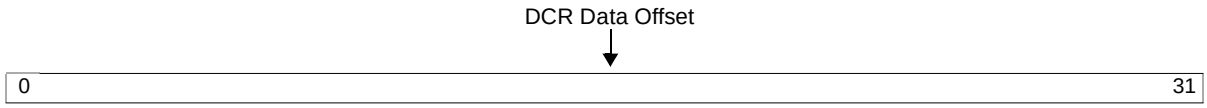


Figure 29-84. EBC Data Register (EBC0_CFGDATA)

0:31	DCRD	DCR Data Port

DCR Accessed using EBC0_CFGADDR; EBC0_CFGDATA; Offset 0x24 R/W

See *EBC Core ID Register (EBC0_CID)* on page 889.

This register is used to access the indirectly-accessed EBC DCRs.

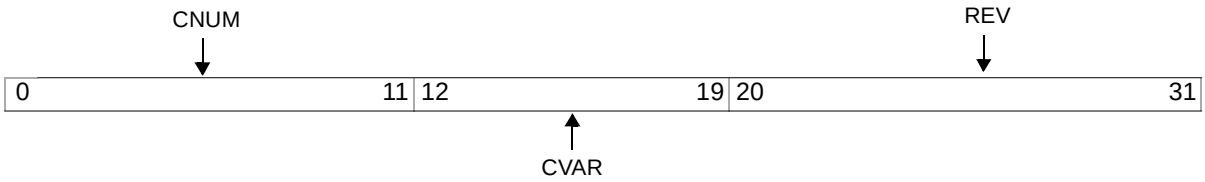


Figure 0-8. EBCO Core ID Register 0 (EBC0_CIDn)

0:11	CNUM	Core Number	‘324
12:19	CVAR	Core Variation	‘01’
20:31	REV	Revision Number	This value is the IBM SCCS revision number tracked and modified by the core designer.

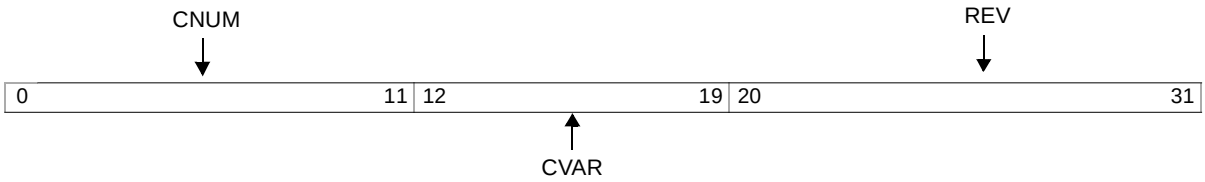


Figure 29-85. EBCO Core ID Register 0 (EBC0_CIDn)

0:11	CNUM	Core Number	‘324
12:19	CVAR	Core Variation	‘01’
20:31	REV	Revision Number	This value is the IBM SCCS revision number tracked and modified by the core designer.

DCR Accessed using EBM0_CFGADDR; EBM0_CFGDATA; Offset 0x02 R/W

See *EBMI Bus Error Address Register (EBM0_BEAR)* on page 848.

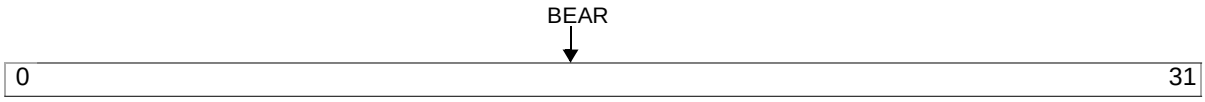


Figure 0-1. Peripheral Bus Error Address Register (EBM0_BEAR)

0:31	BEAR	Bus Error Address	The contents of this register are valid when any bit is set in the EBM0_BESR.

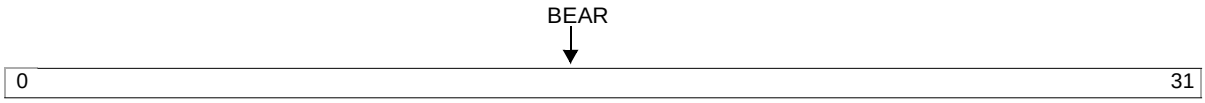


Figure 29-86. Peripheral Bus Error Address Register (EBM0_BEAR)

0:31	BEAR	Bus Error Address	The contents of this register are valid when any bit is set in the EBM0_BESR.

DCR Accessed using EBM0_CFGADDR; EBM0_CFGDATA; Offset 0x04 R/W

See *EBMI Bus Error Mask Register (EBM0_BEMR)* on page 849.

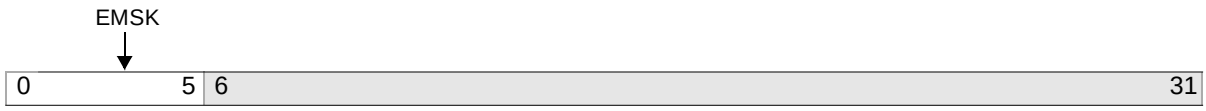


Figure 0-2. EBMI Bus Error Mask Register (EBM0_BEMR)

0:5	EMSK	Bus Error Mask 0 Reporting of this error is not masked 1 Reporting of this error is masked.
6:31		Reserved

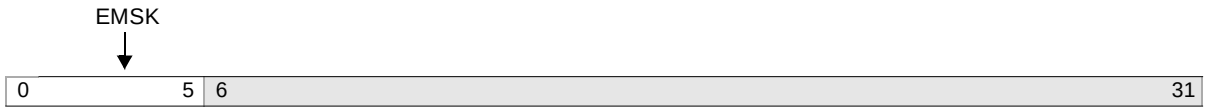


Figure 29-87. EBMI Bus Error Mask Register (EBM0_BEMR)

0:5	EMSK	Bus Error Mask 0 Reporting of this error is not masked 1 Reporting of this error is masked.
6:31		Reserved

DCR Accessed using EBM0_CFGADDR; EBM0_CFGDATA; Offset 0x03 R/W

See *EBMI Bus Error Status Register (EBM0_BESR)* on page 848.

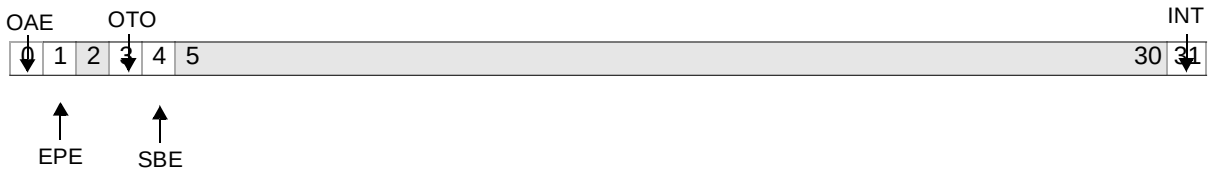


Figure 0-3. EBMI Bus Error Status Register (EBM0_BESR)

0	OAE	OPB Bus Ack Error 0 No ack error detected 1 Ack error detected	This bit is set when the EBMI detects an OPB error indication asserted during a data transfer on the OPB.
1	EPE	EXPB Bus Parity Error 0 No parity error detected 1 Parity error detected	This bit is set when the EBMI detects a parity error on incoming write data from the external master.0 The transfer with the parity error is not sent onto the OPB. Burst write operations must fill the buffer/terminate before they are sent onto the OPB. Consequently, none of the write data in the write buffer is forwarded to the OPB if a parity error is detected. All burst write data after the parity error is also discarded.
2		Reserved	
3	OTO	OPB Bus Timeout Error 0 No timeout detected 1 Timeout indication received when EBMI is reading or writing data on the OPB.	The current transfer on the OPB is terminated.
4	SBE	EXPB Special Cycle Error 0 No Special Cycle error detected. 1 External master has issued a non-Special Cycle operation when a Special Cycle operation was expected.	This error can only occur when using 16 or 8-bit external masters since the Special Cycle operation takes multiple transfers.
5:30			
31	INT	Interrupt Asserted 0 Interrupt line is not asserted. 1 Interrupt line is asserted to the chip internal interrupt controller.	This bit is a logical OR of all the other bits in the EBM0_BESR that are not masked in the EBM0_BEMR. This bit is not writable by software.

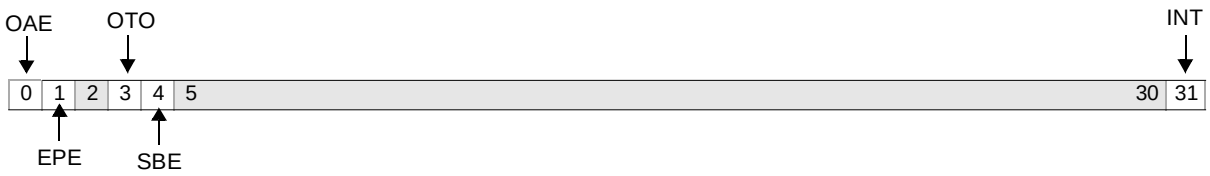


Figure 29-88. EBMI Bus Error Status Register (EBM0_BESR)

0	OAE	OPB Bus Ack Error 0 No ack error detected 1 Ack error detected	This bit is set when the EBMI detects an OPB error indication asserted during a data transfer on the OPB.
1	EPE	EXPB Bus Parity Error 0 No parity error detected 1 Parity error detected	This bit is set when the EBMI detects a parity error on incoming write data from the external master.0 The transfer with the parity error is not sent onto the OPB. Burst write operations must fill the buffer/terminate before they are sent onto the OPB. Consequently, none of the write data in the write buffer is forwarded to the OPB if a parity error is detected. All burst write data after the parity error is also discarded.
2		Reserved	
3	OTO	OPB Bus Timeout Error 0 No timeout detected 1 Timeout indication received when EBMI is reading or writing data on the OPB.	The current transfer on the OPB is terminated.
4	SBE	EXPB Special Cycle Error 0 No Special Cycle error detected. 1 External master has issued a non-Special Cycle operation when a Special Cycle operation was expected.	This error can only occur when using 16 or 8-bit external masters since the Special Cycle operation takes multiple transfers.
5:30			
31	INT	Interrupt Asserted 0 Interrupt line is not asserted. 1 Interrupt line is asserted to the chip internal interrupt controller.	This bit is a logical OR of all the other bits in the EBM0_BESR that are not masked in the EBM0_BEMR. This bit is not writable by software.

DCR Accessed using EBM0_CFGADDR; EBM0_CFGDATA; Offset 0x014 R/W

See *EBMI Configuration Address Register (EBM0_CFGADDR)* on page 845.

This register is used to determine offsets for the indirectly-accessed EBMI DCRs.

Figure 0-4. EBMI Configuration Address Register (EBM0_CFGADDR)

0:26		
27:31	DCRA	DCR Address Offset

Figure 29-89. EBMI Configuration Address Register (EBM0_CFGADDR)

0:26		
27:31	DCRA	DCR Address Offset

DCR Accessed using EBM0_CFGADDR; EBM0_CFGDATA; Offset 0x015 R/W

See *EBMI DCR Data Register (EBM0_CFGDATA)* on page 845.

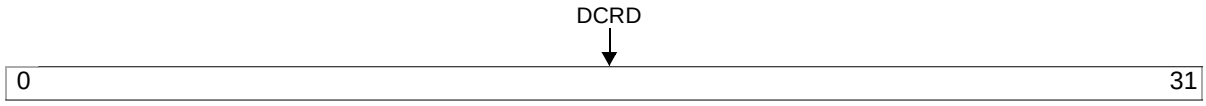


Figure 0-5. EBMI DCR Data Register (EBM0_CFGDATA)

0:31	DCRD	DCR Data Port

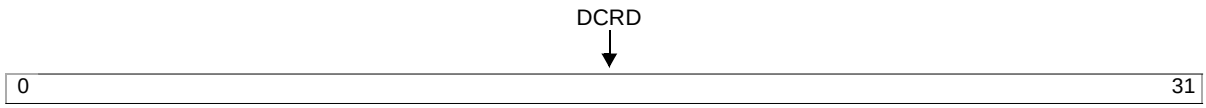


Figure 29-90. EBMI DCR Data Register (EBM0_CFGDATA)

0:31	DCRD	DCR Data Port

DCR Accessed using EBM0_CFGADDR; EBM0_CFGDATA; Offset 0x011 Read-only

See *EBMI Core ID Register (EBM0_CID)* on page 852.

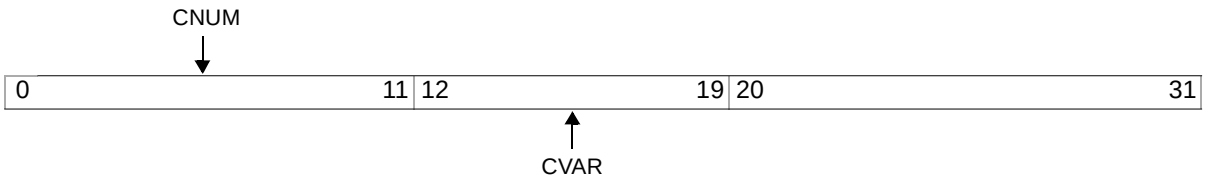


Figure 0-6. EBMI Core ID Register 0 (EBM0_CID)

0:11	CNUM	Core Number	'325'
12:19	CVAR	Core Variation	'01'
20:31	REV	Revision Number	This value is the IBM SCCS revision number tracked and modified by the core designer.

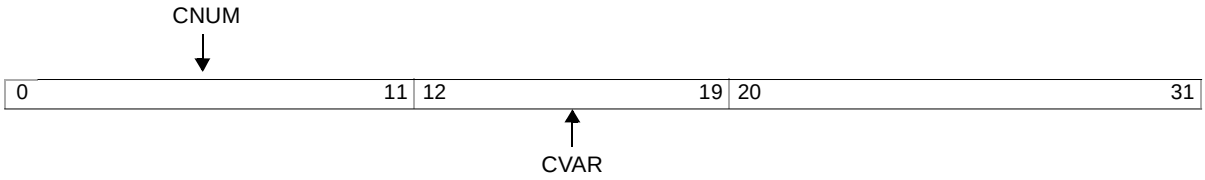


Figure 29-91. EBMI Core ID Register 0 (EBM0_CID)

0:11	CNUM	Core Number	'325'
12:19	CVAR	Core Variation	'01'
20:31	REV	Revision Number	This value is the IBM SCCS revision number tracked and modified by the core designer.

DCR Accessed using EBM0_CFGADDR; EBM0_CFGDATA; Offset 0x00 R/W

See *EBMI Control Register (EBM0_CTL)* on page 845.

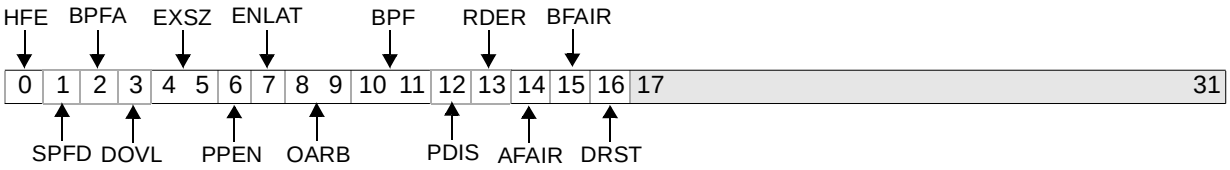


Figure 0-7. EBMI Control Register (EBM0_CTL)

0	HFE	Hold First Error 0 EBM0_BEAR and EBM0_BESR contain information from first error detected. 1 EBM0_BEAR and EBM0_BESR contain information from last error detected.	
1	SPFD	Single Beat Word Prefetch Disable 0 4-byte read on OPB for any single-beat EXPB read request. 1 1 to 4- byte read on OPB for any single-beat EXPB read request, according to EXPB PerWBE0:3 signals.	This bit must be set if the EBMI is to read only the bytes that are requested for a single beat read.
2	BPFA	Burst Prefetch Ahead 0 Data is read from OPB into the data buffer when the data is required. 1 Data is read from OPB into the data buffer before the data is required, while the 1st buffer is being emptied.	This bit should be set by software if the external master routinely does long or many sequential burst read operations longer than the burst prefetch size of the buffer. The EBMI will prefetch 1 buffer ahead to maintain EXPB bus bandwidth. This bit is ignored if the Burst Prefetch mode below is set to ‘10’ to disable the prefetch.
3	DOVL	Disable HoldA Arbitration Overlap 0 HoldAck asserted in response to an active HoldReq asserted by external master immediately. There is a possible 2-cycle delay if a Special Cycle operation is in progress from the previous tenure. 1 HoldAck is not asserted in response to an active HoldReq from the external master until all pending OPB operations and Special Cycle operations from a previous bus tenure are complete.	This bit should be left as 0 and the EBM0_FAIR register used to tune arbitration.
4:5	EXSZ	External Master Data Bus Size 00 8-bit data bus, PerData0:7 01 16-bit data bus, PerData0:15 10 32-bit data bus, PerData0:31 11 No external master	Software must initialize this register to indicate the size of the external master data bus.

6	PPEN	Enable second 32-byte buffer 0 Second buffer disabled. Use only 1 buffer. 1 Second buffer enabled. Ping/Pong enabled.	
7	ENLAT	Enable OPB Latency Counter 0 OPB Latency counter is disabled 1 OPB Latency counter is enabled	This bit enables the EBM0_LCNT register function.
8:9	OARB	OPB Arbitration Control Must be0b01	
10:11	BPF	Burst Prefetch 00 8-beat (32-byte) read 01 4-beat (16-byte) read 10 1-beat (4/2/1-byte) read (no prefetch, read byte size of master) 11 - Reserved, Unused.	For most applications, this should remain at '00.' If BPF=2'b10, the request will be forwarded exactly as received if SPFD=1. These bits control the number of bytes prefetched on a burst read from the external master.
12	PDIS	EXPB Parity Checking Disabled 0 Write data parity is checked on EXPB 1 Write data parity is not checked on EXPB	Field enabled parity checking is only for external master transfers. This bit must be set by software if the external master does not support parity on PerData.
13	RDER	Enable Read Early Indication 0 Burst read data returned to external master when burst read from OPB is complete. 1 Burst read data returned to external master beginning when first beat from OPB is complete.	This mode bit is used to reduce latency for read burst operations.
14	AFAIR	Arbitration Fairness Counter Disable	This bit disables the AFCNT in the EBM0_FAIR register.
15	BFAIR	BReq Fairness Counter Disable	This bit disables the BFCNT in the EBM0_FAIR register.
16	DRST	Disable Reset Valid for New Tenure	This bit disables the clearing of the read buffer valid bits at the start of a new external master bus tenure. This bit should be set if the external master does long burst reads but often has to give up the EXPB before receiving all the data because BusReq is asserted.
17:31		Reserved	

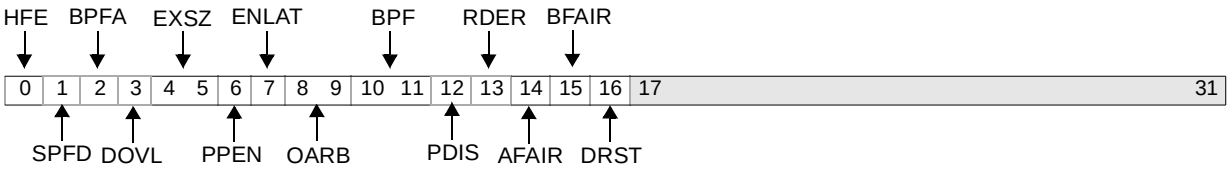


Figure 29-92. EBMI Control Register (EBM0_CTL)

0	HFE	Hold First Error 0 EBM0_BEAR and EBM0_BESR contain information from first error detected. 1 EBM0_BEAR and EBM0_BESR contain information from last error detected.	
1	SPFD	Single Beat Word Prefetch Disable 0 4-byte read on OPB for any single-beat EXPB read request. 1 1 to 4- byte read on OPB for any single-beat EXPB read request, according to EXPB PerWBE0:3 signals.	This bit must be set if the EBMI is to read only the bytes that are requested for a single beat read.
2	BPFA	Burst Prefetch Ahead 0 Data is read from OPB into the data buffer when the data is required. 1 Data is read from OPB into the data buffer before the data is required, while the 1st buffer is being emptied.	This bit should be set by software if the external master routinely does long or many sequential burst read operations longer than the burst prefetch size of the buffer. The EBMI will prefetch 1 buffer ahead to maintain EXPB bus bandwidth. This bit is ignored if the Burst Prefetch mode below is set to '10' to disable the prefetch.
3	DOVL	Disable HoldA Arbitration Overlap 0 HoldAck asserted in response to an active HoldReq asserted by external master immediately. There is a possible 2-cycle delay if a Special Cycle operation is in progress from the previous tenure. 1 HoldAck is not asserted in response to an active HoldReq from the external master until all pending OPB operations and Special Cycle operations from a previous bus tenure are complete.	This bit should be left as 0 and the EBM0_FAIR register used to tune arbitration.
4:5	EXSZ	External Master Data Bus Size 00 8-bit data bus, PerData0:7 01 16-bit data bus, PerData0:15 10 32-bit data bus, PerData0:31 11 No external master	Software must initialize this register to indicate the size of the external master data bus.
6	PPEN	Enable second 32-byte buffer 0 Second buffer disabled. Use only 1 buffer. 1 Second buffer enabled. Ping/Pong enabled.	
7	ENLAT	Enable OPB Latency Counter 0 OPB Latency counter is disabled 1 OPB Latency counter is enabled	This bit enables the EBM0_LCNT register function.
8:9	OARB	OPB Arbitration Control Must be0b01	

10:11	BPF	Burst Prefetch 00 8-beat (32-byte) read 01 4-beat (16-byte) read 10 1-beat (4/2/1-byte) read (no prefetch, read byte size of master) 11 - Reserved, Unused.	For most applications, this should remain at '00.' If BPF=2'b10, the request will be forwarded exactly as received if SPFD=1. These bits control the number of bytes prefetched on a burst read from the external master.
12	PDIS	EXPB Parity Checking Disabled 0 Write data parity is checked on EXPB 1 Write data parity is not checked on EXPB	Field enabled parity checking is only for external master transfers. This bit must be set by software if the external master does not support parity on PerData.
13	RDER	Enable Read Early Indication 0 Burst read data returned to external master when burst read from OPB is complete. 1 Burst read data returned to external master beginning when first beat from OPB is complete.	This mode bit is used to reduce latency for read burst operations.
14	AFAIR	Arbitration Fairness Counter Disable	This bit disables the AFCNT in the EBM0_FAIR register.
15	BFAIR	BReq Fairness Counter Disable	This bit disables the BFCNT in the EBM0_FAIR register.
16	DRST	Disable Reset Valid for New Tenure	This bit disables the clearing of the read buffer valid bits at the start of a new external master bus tenure. This bit should be set if the external master does long burst reads but often has to give up the EXPB before receiving all the data because BusReq is asserted.
17:31		Reserved	

DCR Accessed using EBM0_CFGADDR; EBM0_CFGDATA; Offset 0x08 R/W

See *EBMI Fairness Control Register (EBM0_FAIR)* on page 853.

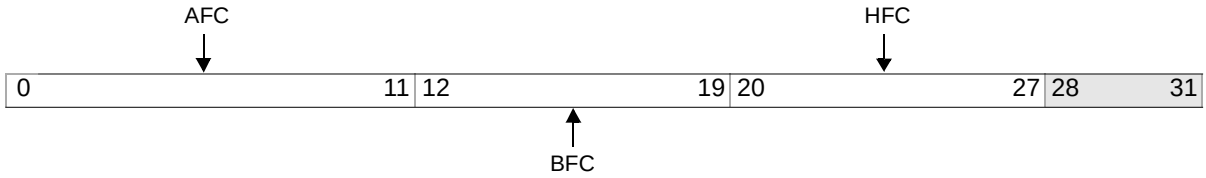


Figure 0-8. EBMI Fairness Control Register 0 (EBM0_FAIR)

0:11	AFC	Arbitration Fairness Count	These bits indicate the number of OPB clocks EBMI will wait before asserting a signal to stop PLB-to-OPB traffic when an external master request is pending. Under most conditions, these bits should be set to a high value to lower the priority of the external master. These bits are used to tune the priority of requests from the PLB to the OPB vs the priority of requests from the external master. If PLB-to-OPB traffic needs a relatively high priority, then a high value (for example x'F00') is written into this register by software.
12:19	BFC	BusReq Fairness Count	These bits indicate the number of PerClk cycles EBMI will wait before asserting BusReq to the external master. The BusReq fairness counter will begin decrementing when the EBCO has relinquished EXPB bus ownership. If BFC=0, BusReq may be asserted the same cycle as HoldAck. Under some conditions, the EBC may request ownership of the external bus and assert BusReq before the external master has asserted ExtReq. Setting the BFC bits to a non-zero value may allow the external master to make forward progress (if it is sensitive to this condition) by delaying BusReq until after the external master has asserted ExtReq.

20:27	HFC	HoldReq Fairness Count	These bits indicate the number of PerClk cycles EBMI will wait before forwarding HoldReq to the EBCO for the external master to gain bus ownership. This register is set to a high value to prioritize DMA transfers to and from the EBCO since DMA transfers are not affected by AFC.
28:31		Reserved	

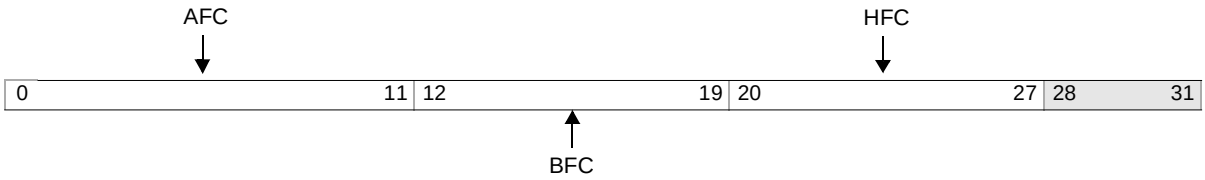


Figure 29-93. EBMI Fairness Control Register 0 (EBM0_FAIR)

0:11	AFC	Arbitration Fairness Count	These bits indicate the number of OPB clocks EBMI will wait before asserting a signal to stop PLB-to-OPB traffic when an external master request is pending. Under most conditions, these bits should be set to a high value to lower the priority of the external master. These bits are used to tune the priority of requests from the PLB to the OPB vs the priority of requests from the external master. If PLB-to-OPB traffic needs a relatively high priority, then a high value (for example x'F00') is written into this register by software.
12:19	BFC	BusReq Fairness Count	These bits indicate the number of PerClk cycles EBMI will wait before asserting BusReq to the external master. The BusReq fairness counter will begin decrementing when the EBCO has relinquished EXPB bus ownership. If BFC=0, BusReq may be asserted the same cycle as HoldAck. Under some conditions, the EBC may request ownership of the external bus and assert BusReq before the external master has asserted ExtReq. Setting the BFC bits to a non-zero value may allow the external master to make forward progress (if it is sensitive to this condition) by delaying BusReq until after the external master has asserted ExtReq.
20:27	HFC	HoldReq Fairness Count	These bits indicate the number of PerClk cycles EBMI will wait before forwarding HoldReq to the EBCO for the external master to gain bus ownership. This register is set to a high value to prioritize DMA transfers to and from the EBCO since DMA transfers are not affected by AFC.
28:31		Reserved	

DCR Accessed using EBM0_CFGADDR; EBM0_CFGDATA; Offset 0x01 R/W

See *EBMI OPB Latency Count Register (EBM0_LCNT)* on page 847.

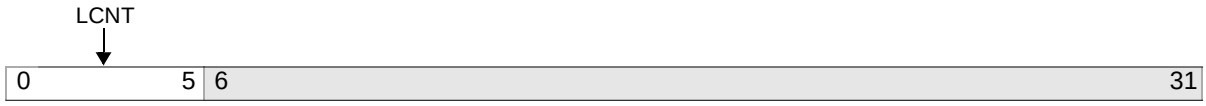


Figure 0-9. EBMI OPB Latency Count Register (EBM0_LCNT)

0:5	LCNT	Latency Counter This is the upper 6 bits of the initial load of a 10-bit OPB bus latency counter.
6:31		Reserved

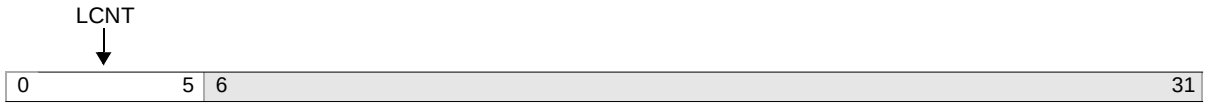


Figure 29-94. EBMI OPB Latency Count Register (EBM0_LCNT)

0:5	LCNT	Latency Counter This is the upper 6 bits of the initial load of a 10-bit OPB bus latency counter.
6:31		Reserved

DCR Accessed using EBM0_CFGADDR; EBM0_CFGDATA; Offset 0x07 R/W

See *EBMI Sleep Mode Register (EBM0_SLPMD)* on page 851.

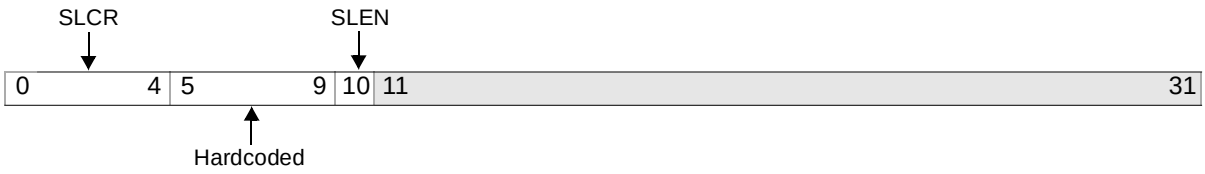


Figure 0-10. EBMI Sleep Mode Register (EBM0_SLPMD)

0:4	SLCR	Programmable timer values	
5:9		Hardcoded to 1s	
10	SLEN	Sleep mode enable bit 0 Sleep request not asserted 1 Sleep request asserted when sleep requirements are met	Software must set this bit for the EBMI to assert its sleep request and be put to sleep.
11:31		Reserved	

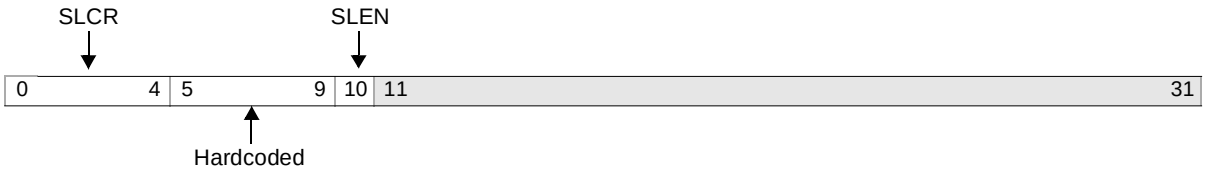


Figure 29-95. EBMI Sleep Mode Register (EBM0_SLPMD)

0:4	SLCR	Programmable timer values	
5:9		Hardcoded to 1s	
10	SLEN	Sleep mode enable bit 0 Sleep request not asserted 1 Sleep request asserted when sleep requirements are met	Software must set this bit for the EBMI to assert its sleep request and be put to sleep.
11:31		Reserved	

DCR Accessed using EBM0_CFGADDR; EBM0_CFGDATA; Offset 0x06 R/W

See *EBMI OPB Upper Address Mask Register (EBM0_UAM)* on page 850.

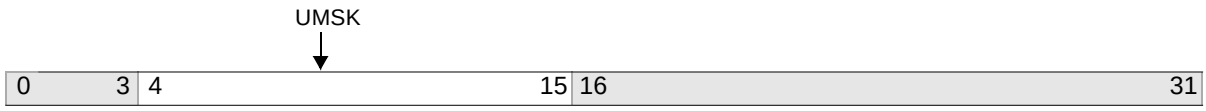


Figure 0-11. EBMI Upper Address Mask (EBM0_UAM)

0:3		Reserved	
4:15	UMSK	OPB Upper Address Mask	This register must be initialized by software iif the external master provides less than a complete 32-bit address.
16:31		Reserved	

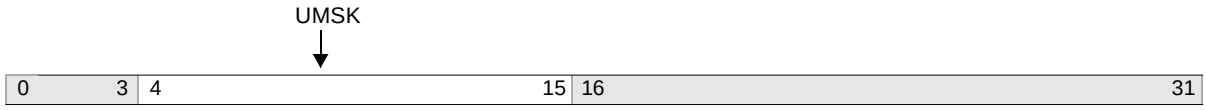


Figure 29-96. EBMI Upper Address Mask (EBM0_UAM)

0:3		Reserved	
4:15	UMSK	OPB Upper Address Mask	This register must be initialized by software iif the external master provides less than a complete 32-bit address.
16:31		Reserved	

DCR Accessed using EBM0_CFGADDR; EBM0_CFGDATA; Offset 0x05 R/W

See *EBMI OPB Upper Address Register (EBM0_UAR)* on page 850.

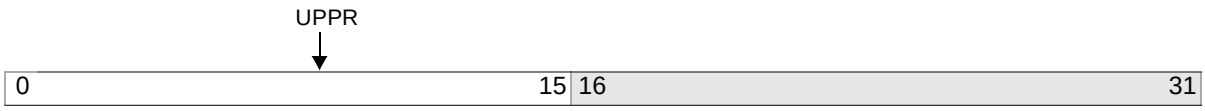


Figure 0-12. EBMI Upper Address Register (EBM0_UAR)

0:15	UPPR	OPB Upper Address	This register must be initialized by software.
16:31		Reserved	

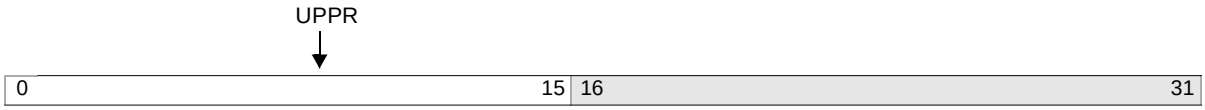


Figure 29-97. EBMI Upper Address Register (EBM0_UAR)

0:15	UPPR	OPB Upper Address	This register must be initialized by software.
16:31		Reserved	

MMIO 0x1 4000 0840–00x1 4000 084C (EMAC0), 0x1 4000 0940–00x1 4000 094C (EMAC1)

See *Group Address Hash Tables 1–4 (EMACx_GAHT1–EMACx_GAHT4)* on page 767.

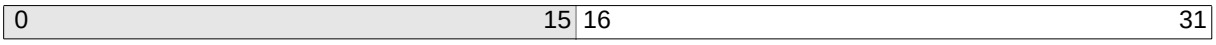


Figure 0-1. Group Address Hash Tables 1–4 (EMACx_GAHT1–EMACx_GAHT4)

0:15		Reserved
16:31		Group Address Hash Number

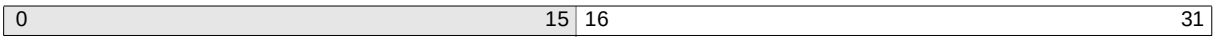


Figure 29-98. Group Address Hash Tables 1–4 (EMACx_GAHT1–EMACx_GAHT4)

0:15		Reserved
16:31		Group Address Hash Number

MMIO 0x1 4000081C (EMAC0), 0x1 4000091C (EMAC1)

See *Individual Address High (EMACx_IAHR)* on page 764.

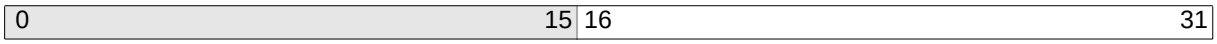


Figure 0-2. Individual Address High Register (EMACx_IAHR)

0:15		Reserved	
16:31		High-order halfword of the station unique individual address	This field contains bits 0:15 of the destination address (bit 0 is the most significant bit).

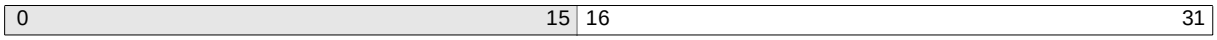


Figure 29-99. Individual Address High Register (EMACx_IAHR)

0:15		Reserved	
16:31		High-order halfword of the station unique individual address	This field contains bits 0:15 of the destination address (bit 0 is the most significant bit).

MMIO 0x1 4000 0830–0x1 4000 083C (EMAC0), 0x1 4000 0930–0x1 4000 093C (EMAC1)

See *Individual Address Hash Tables 1–4 (EMACx_IAHT1–EMACx_IAHT4)* on page 767.

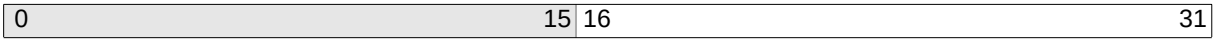


Figure 0-3. Individual Address Hash Tables 1–4 (EMACx_IAHT1–EMACx_IAHT4)

0:15		Reserved
16:31		Individual Address Hash Number

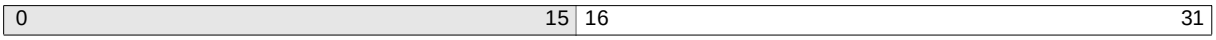


Figure 29-100. Individual Address Hash Tables 1–4 (EMACx_IAHT1–EMACx_IAHT4)

0:15		Reserved
16:31		Individual Address Hash Number

MMIO 0x1 4000 0820 (EMAC0), 0x1 4000 0920 (EMAC1)

See *Individual Address Low (EMACx_IALR)* on page 765.

Figure 0-4. Individual Address Low Register (EMACx_IALR)

0:31		Low-order bits of Receive Individual Address or Transmit Source Address

Figure 29-101. Individual Address Low Register (EMACx_IALR)

0:31		Low-order bits of Receive Individual Address or Transmit Source Address

MMIO 0x1 4000 0858 (EMAC0), 0x1 4000 0958 (EMAC1)

See *Inter-Packet Gap Value Register (EMACx_IPGVR)* on page 768.

Figure 0-5. inter-Packet Gap Value Register (EMACx_IPGVR)

0:25		Reserved
26:31		Inter-Packet Gap

Figure 29-102. inter-Packet Gap Value Register (EMACx_IPGVR)

0:25		Reserved
26:31		Inter-Packet Gap

MMIO 0x1 4000 0818 (EMAC0), 0x1 4000 0918 (EMAC1)

See *Interrupt Status Enable Register (EMACx_ISER)* on page 762.

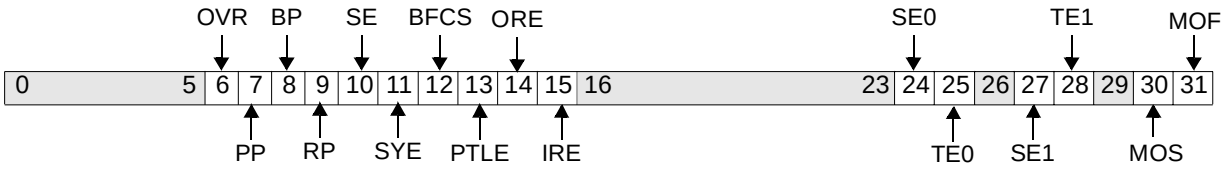


Figure 0-6. Interrupt Status Enable Register (EMACx_ISER)

0:5		Reserved
6	OVR	Overrun 0 Overrun error will not generate an interrupt. 1 Overrun error will generate an interrupt.
7	PP	Pause Packet 0 Received control pause packet will not generate an interrupt. 1 Received control pause packet will generate an interrupt.
8	BP	Bad Packet 0 Early termination on received packet will not generate an interrupt. 1 Early termination on received packet will generate an interrupt.
9	RP	Runt Packet 0 Received runt packet will not generate an interrupt. 1 Received runt packet will generate an interrupt.
10	SE	Short Event 0 Short event during receive will not generate an interrupt. 1 Short event during receive will generate an interrupt.
11	ALE	Alignment Error 0 Alignment error in received packet will not generate an interrupt. 1 Alignment error in received packet will generate an interrupt.
12	BFCS	Bad FCS 0 FCS error in received packet will not generate an interrupt. 1 FCS error in received packet will generate an interrupt.

13	PTLE	Packet Too Long Error 0 Oversized packets received will not generate an interrupt. 1 Oversized packet received will generate an interrupt.
14	ORE	Out Of Range Error 0 Out of range error on received packet will not generate an interrupt. 1 Out of range error on received packet will generate an interrupt.
15	IRE	In Range Error 0 In range error on received packet will not generate an interrupt. 1 In range error on received packet will generate an interrupt.
16:23		Reserved
24	SE0	SQE Error 0 0 SQE error on TX Channel 0 will not generate an interrupt. 1 SQE error on TX Channel 0 will generate an interrupt.
25	TE0	Transmit Error 0 0 TX error on TX Channel 0 will not generate an interrupt. 1 TX error on TX Channel 0 will generate an interrupt.
26		Reserved
27	SE1	SQE Error 1 0 SQE error on TX Channel 1 will not generate an interrupt. 1 SQE error on TX Channel 1 will generate an interrupt.
28	TE1	Transmit Error 1 0 TX error on TX Channel 1 will not generate an interrupt. 1 TX error on TX Channel 1 will generate an interrupt.
29		Reserved
30	MOS	MMA Operation Succeeded 0 Successful MMA Operation with a PHY will not generate an interrupt. 1 Successful MMA Operation with a PHY will generate an interrupt.

31	MOF	MMA Operation Failed 0 Unsuccessful MMA Operation with a PHY will not generate an interrupt. 1 Unsuccessful MMA Operation with a PHY will generate an interrupt.

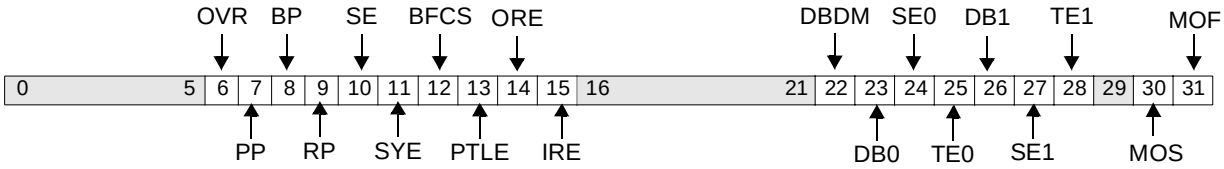


Figure 29-103. Interrupt Status Enable Register (EMACx_ISER)

0:5		Reserved
6	OVR	Overrun 0 Overrun error will not generate an interrupt. 1 Overrun error will generate an interrupt.
7	PP	Pause Packet 0 Received control pause packet will not generate an interrupt. 1 Received control pause packet will generate an interrupt.
8	BP	Bad Packet 0 Early termination on received packet will not generate an interrupt. 1 Early termination on received packet will generate an interrupt.
9	RP	Runt Packet 0 Received runt packet will not generate an interrupt. 1 Received runt packet will generate an interrupt.
10	SE	Short Event 0 Short event during receive will not generate an interrupt. 1 Short event during receive will generate an interrupt.
11	ALE	Alignment Error 0 Alignment error in received packet will not generate an interrupt. 1 Alignment error in received packet will generate an interrupt.
12	BFCS	Bad FCS 0 FCS error in received packet will not generate an interrupt. 1 FCS error in received packet will generate an interrupt.
13	PTLE	Packet Too Long Error 0 Oversized packets received will not generate an interrupt. 1 Oversized packet received will generate an interrupt.

14	ORE	Out Of Range Error 0 Out of range error on received packet will not generate an interrupt. 1 Out of range error on received packet will generate an interrupt.
15	IRE	In Range Error 0 In range error on received packet will not generate an interrupt. 1 In range error on received packet will generate an interrupt.
16:21		Reserved
22	DBDM	Dead Bit Dependent Mode 0 Dead bit dependent mode will not generate an interrupt 1 Dead bit dependent mode will generate an interrupt
23	DB0	Dead Bit 0 0 Dead bit 0 will not generate an interrupt 1 Dead bit 0 will generate an interrupt
24	SE0	SQE Error 0 0 SQE error on TX Channel 0 will not generate an interrupt. 1 SQE error on TX Channel 0 will generate an interrupt.
25	TE0	Transmit Error 0 0 TX error on TX Channel 0 will not generate an interrupt. 1 TX error on TX Channel 0 will generate an interrupt.
26	DB1	Dead Bit 1 0 Dead bit 1 will not generate an interrupt 1 Dead bit 1 will generate an interrupt
27	SE1	SQE Error 1 0 SQE error on TX Channel 1 will not generate an interrupt. 1 SQE error on TX Channel 1 will generate an interrupt.
28	TE1	Transmit Error 1 0 TX error on TX Channel 1 will not generate an interrupt. 1 TX error on TX Channel 1 will generate an interrupt.
29		Reserved
30	MOS	MMA Operation Succeeded 0 Successful MMA Operation with a PHY will not generate an interrupt. 1 Successful MMA Operation with a PHY will generate an interrupt.
31	MOF	MMA Operation Failed 0 Unsuccessful MMA Operation with a PHY will not generate an interrupt. 1 Unsuccessful MMA Operation with a PHY will generate an interrupt.

MMIO 0x1 4000 0814 (EMAC0), 0x1 4000 0914 (EMAC1)

See *Interrupt Status Register (EMACx_ISR)* on page 760.

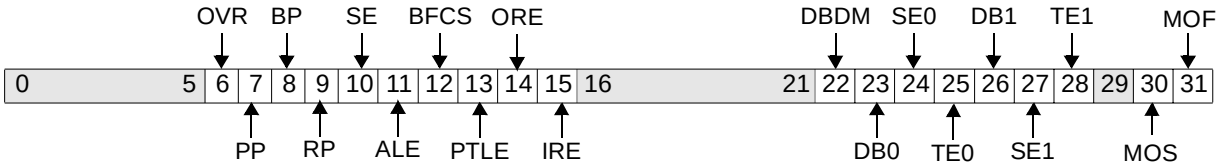


Figure 0-7. Interrupt Status Register (EMACx_ISR)

0:5		Reserved	
6	OVR	Overrun 0 No overrun error 1 Overrun error during reception of recent packet	
7	PP	Pause Packet 0 Received packet is not a control pause packet 1 Received packet is a control pause packet	
8	BP	Bad Packet 0 Receive operation OK 1 Early termination was initiated because of a packet error	
9	RP	Runt Packet 0 No Runt packets received 1 Runt packet received	Set when EMACx_RMR[RRP] = 1 and the duration of PHY_RX_DV signal was greater than ShortEventMaxTime constant and less than the collision window.
10	SE	Short Event 0 No short events 1 Duration of PHY_RX_DV signal less than ShortEventMaxTime constant	
11	ALE	Alignment Error 0 No alignment error in received packet 1 Alignment error in received packet	The packet contained an odd number of nibbles (4 bits).
12	BFCS	Bad FCS 0 No FCS error in received packet 1 Packet with an FCS error received	Set if EMACx_RMR[RFP] = 1.

13	PTLE	Packet Too Long Error 0 No oversized packets received 1 Oversized packet received	Set if EMACx_RMR[ROP] = 1 and the received packet length exceeded the maximum allowed value: • 1518 octets for standard packet (checked only if the length/type field of the transmitted packet contained length value) • 1522 octets for VLAN tagged packet (checked only if the length/type field of the transmitted packet contained length value and jumbo support is disabled)
14	ORE	Out Of Range Error 0 Received packet length field value OK 1 Received packet length field value greater than the maximum allowed LLC data size	Indicates that received packet has a length field value greater than the maximum allowed logical link control (LLC) data size (greater than 1500 and less than 1536).
15	IRE	In Range Error 0 Received packet does not contain an In Range Error 1 Received packet contains an In Range Error	
16:21		Reserved	
22	DBDM	Dead Bit Dependent Mode 0 No transmit error or SQE in dependent mode 1 Transmit error or SQE has occurred while in dependent mode	If EMACx_ISR[DBDM] = 1, EMAC does not request MAL service, even if EMACx_TMR0[GNPD] = 1. EMACx_ISR[DBDM] does not affect EMAC interrupt.
23	DB0	Dead Bit 0 0 No transmit error or SQE for TX Channel 0 while not in dependent mode 1 Transmit error or SQE has occurred for TX Channel 0 while not in dependent mode	If EMACx_ISR[DB0] = 1, EMAC does not request service for TX Channel 0 from MAL, even if EMACx_TMR0[GNP0] = 1. EMACx_ISR[DB0] does not affect EMAC interrupt.
24	SE0	Signal Quality Error 0 0 No SQEs on TX Channel 0 1 SQE test failure during transmission of a packet from TX Channel 0	Applicable only in half-duplex mode during 10 Mbps operations; 0 in all other modes.
25	TE0	Transmit Error 0 0 TX Channel 0 transmission OK 1 TX Channel 0 transmission aborted	EMAC aborts the transmitted packet if one of the following events takes place: • Late collision detection • Excessive collision detection • Excessive deferral • TX FIFO underrun • Loss of carrier sense

26	DB1	Dead Bit 1 0 No transmit error or SQE for TX Channel 1 while not in dependent mode 1 Transmit error or a SQE has occurred for TX Channel 1 while not in dependent mode	If this bit is set, EMAC does not request MAL service for TX Channel 1 even if EMACx_TMR1[GNP1] = 1. EMACx_ISR[DB1] does not affect EMAC interrupt.
27	SE1	Signal Quality Error 1 0 No SQE on TX Channel 1 1 SQE test failure during transmission of a packet from TX Channel 1	Applicable only in half-duplex mode during 10 Mbps operations; 0 in all other modes.
28	TE1	Transmit Error 1 0 TX Channel 1 transmission OK 1 TX Channel 1 transmission aborted	EMAC aborts the transmitted packet if one of the following events takes place: • Late collision detection • Excessive collision detection • Excessive deferral • TX FIFO underrun • Loss of carrier sense
29		Reserved	Always 0
30	MOS	MMA Operation Succeeded 0 MMA_CONTROL addressed on the OPB 1 PHY transfer valid	The device driver should poll assertion of EMACx_ISR[MOS] or EMACx_ISR[MOF] before issuing a new command or before using data read from the PHY.
31	MOF	MMA Operation Failed 0 MMA_CONTROL addressed on the OPB 1 PHY transfer *not* valid	The device driver should poll assertion of EMACx_ISR[MOF] or EMACx_ISR[MOS] before issuing a new command or before using data read from the PHY.

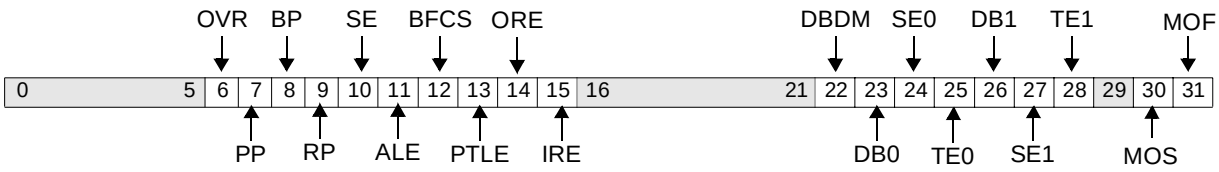


Figure 29-104. Interrupt Status Register (EMACx_ISR)

0:5		Reserved
6	OVR	Overrun 0 No overrun error 1 Overrun error during reception of recent packet
7	PP	Pause Packet 0 Received packet is not a control pause packet 1 Received packet is a control pause packet
8	BP	Bad Packet 0 Receive operation OK 1 Early termination was initiated because of a packet error

9	RP	Runt Packet 0 No Runt packets received 1 Runt packet received	Set when EMACx_RMR[RRP] = 1 and the duration of PHY_RX_DV signal was greater than Short-EventMaxTime constant and less than the collision window.
10	SE	Short Event 0 No short events 1 Duration of PHY_RX_DV signal less than ShortEventMaxTime constant	
11	ALE	Alignment Error 0 No alignment error in received packet 1 Alignment error in received packet	The packet contained an odd number of nibbles (4 bits).
12	BFCS	Bad FCS 0 No FCS error in received packet 1 Packet with an FCS error received	Set if EMACx_RMR[RFP] = 1.
13	PTLE	Packet Too Long Error 0 No oversized packets received 1 Oversized packet received	Set if EMACx_RMR[ROP] = 1 and the received packet length exceeded the maximum allowed value: • 1518 octets for standard packet (checked only if the length/type field of the transmitted packet contained length value) • 1522 octets for VLAN tagged packet (checked only if the length/type field of the transmitted packet contained length value and jumbo support is disabled)
14	ORE	Out Of Range Error 0 Received packet length field value OK 1 Received packet length field value greater than the maximum allowed LLC data size	Indicates that received packet has a length field value greater than the maximum allowed logical link control (LLC) data size (greater than 1500 and less than 1536).
15	IRE	In Range Error 0 Received packet does not contain an In Range Error 1 Received packet contains an In Range Error	
16:21		Reserved	
22	DBDM	Dead Bit Dependent Mode 0 No transmit error or SQE in dependent mode 1 Transmit error or SQE has occurred while in dependent mode	If EMACx_ISR[DBDM] = 1, EMAC does not request MAL service, even if EMACx_TMR0[GNPD] = 1. EMACx_ISR[DBDM] does not affect EMAC interrupt.
23	DB0	Dead Bit 0 0 No transmit error or SQE for TX Channel 0 while not in dependent mode 1 Transmit error or SQE has occurred for TX Channel 0 while not in dependent mode	If EMACx_ISR[DB0] = 1, EMAC does not request service for TX Channel 0 from MAL, even if EMACx_TMR0[GNP0] = 1. EMACx_ISR[DB0] does not affect EMAC interrupt.
24	SE0	Signal Quality Error 0 0 No SQEs on TX Channel 0 1 SQE test failure during transmission of a packet from TX Channel 0	Applicable only in half-duplex mode during 10 Mbps operations; 0 in all other modes.

25	TE0	Transmit Error 0 0 TX Channel 0 transmission OK 1 TX Channel 0 transmission aborted	EMAC aborts the transmitted packet if one of the following events takes place: • Late collision detection • Excessive collision detection • Excessive deferral • TX FIFO underrun • Loss of carrier sense
26	DB1	Dead Bit 1 0 No transmit error or SQE for TX Channel 1 while not in dependent mode 1 Transmit error or a SQE has occurred for TX Channel 1 while not in dependent mode	If this bit is set, EMAC does not request MAL service for TX Channel 1 even if EMACx_TMR1[GNP1] = 1. EMACx_ISR[DB1] does not affect EMAC interrupt.
27	SE1	Signal Quality Error 1 0 No SQE on TX Channel 1 1 SQE test failure during transmission of a packet from TX Channel 1	Applicable only in half-duplex mode during 10 Mbps operations; 0 in all other modes.
28	TE1	Transmit Error 1 0 TX Channel 1 transmission OK 1 TX Channel 1 transmission aborted	EMAC aborts the transmitted packet if one of the following events takes place: • Late collision detection • Excessive collision detection • Excessive deferral • TX FIFO underrun • Loss of carrier sense
29		Reserved	Always 0
30	MOS	MMA Operation Succeeded 0 MMA_CONTROL addressed on the OPB 1 PHY transfer valid	The device driver should poll assertion of EMACx_ISR[MOS] or EMACx_ISR[MOF] before issuing a new command or before using data read from the PHY.
31	MOF	MMA Operation Failed 0 MMA_CONTROL addressed on the OPB 1 PHY transfer *not* valid	The device driver should poll assertion of EMACx_ISR[MOF] or EMACx_ISR[MOS] before issuing a new command or before using data read from the PHY.

MMIO 0x1 4000 0850 (EMAC0), 0x1 4000 0950 (EMAC1)

See *Last Source Address High (EMACx_LSAH)* on page 767.

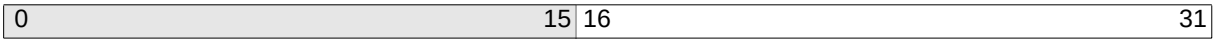


Figure 0-8. Last Source Address High Register (EMACx_LSAH)

0:15		Reserved
16:31		Last Source Address High-Order Halfword

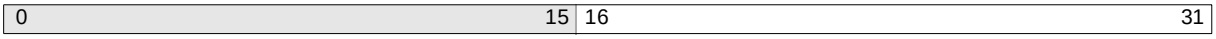


Figure 29-105. Last Source Address High Register (EMACx_LSAH)

0:15		Reserved
16:31		Last Source Address High-Order Halfword

MMIO 0x1 4000 0854 (EMAC0), 0x1 4000 0954 (EMAC1)

See *Last Source Address Low (EMACx_LSAL)* on page 768.

Figure 0-9. Last Source Address Low Register (EMACx_LSAL)

0:31		Last Source Address Low-Order Word

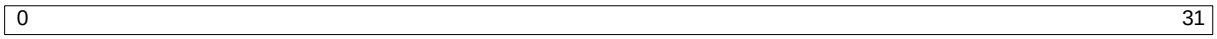


Figure 29-106. Last Source Address Low Register (EMACx_LSAL)

0:31		Last Source Address Low-Order Word

MMIO 0x1 4000 0800 (EMAC0), 0x1 4000 0900 (EMAC1)

See *Mode Register 0 (EMACx_MR0)* on page 754.

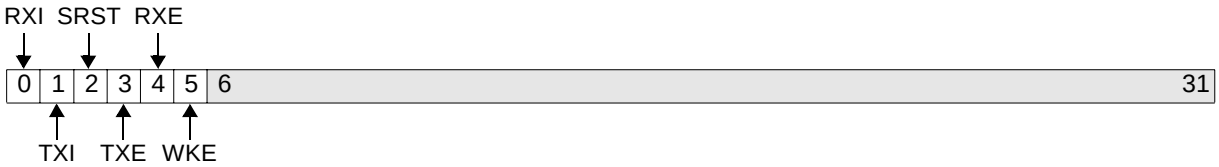


Figure 0-10. Mode Register 0 (EMACx_MR0)

0	RXI	Receive MAC Idle 0 RX MAC processing packet 1 RX MAC idle; RX packet processing complete	Read-only
1	TXI	Transmit MAC Idle 0 TX MAC processing packet 1 TX MAC idle; TX packet processing complete	Read-only
2	SRST	EMAC Software Reset 0 EMAC reset is complete 1 Reset the EMAC	Generates a general reset to EMAC through a software command. After setting this bit, EMAC hardware (registers, interface and internal state machines) returns to the power-on reset value. The software writes 1 to this bit in order to drive EMAC to the reset state. The bit is cleared by the hardware when the reset is completed. If EMACx_MR0[SRST] = 1, writing to any EMAC register, and reading any other bit in this register, is not supported.
3	TXE	Transmit MAC Enable 0 TX MAC is disabled 1 TX MAC is enabled	
4	RXE	Receive MAC Enable 0 RX MAC is disabled 1 RX MAC is enabled	
5	WKE	Wake-Up Enable 0 Incoming packets are not examined for wake-up packet 1 Examine incoming packets for wake-up packet	Software can change EMACx_MR0[WKE] only while EMACx_MR0[RXI] = 1 and EMACx_MR0[RXE] = 0.
6:31		Reserved	

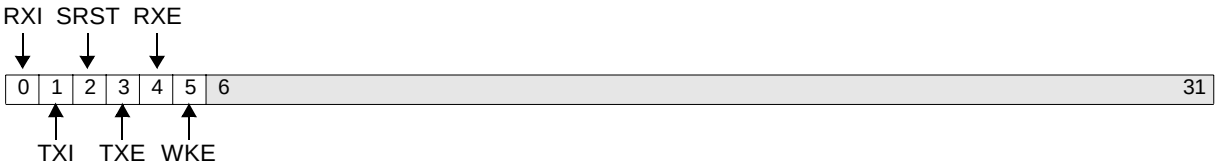


Figure 29-107. Mode Register 0 (EMACx_MR0)

0	RXI	Receive MAC Idle 0 RX MAC processing packet 1 RX MAC idle; RX packet processing complete	Read-only
1	TXI	Transmit MAC Idle 0 TX MAC processing packet 1 TX MAC idle; TX packet processing complete	Read-only
2	SRST	EMAC Software Reset 0 EMAC reset is complete 1 Reset the EMAC	Generates a general reset to EMAC through a software command. After setting this bit, EMAC hardware (registers, interface and internal state machines) returns to the power-on reset value. The software writes 1 to this bit in order to drive EMAC to the reset state. The bit is cleared by the hardware when the reset is completed. If EMACx_MR0[SRST] = 1, writing to any EMAC register, and reading any other bit in this register, is not supported.
3	TXE	Transmit MAC Enable 0 TX MAC is disabled 1 TX MAC is enabled	
4	RXE	Receive MAC Enable 0 RX MAC is disabled 1 RX MAC is enabled	
5	WKE	Wake-Up Enable 0 Incoming packets are not examined for wake-up packet 1 Examine incoming packets for wake-up packet	Software can change EMACx_MR0[WKE] only while EMACx_MR0[RXI] = 1 and EMACx_MR0[RXE] = 0.
6:31		Reserved	

MMIO 0x1 4000 0804 (EMAC0), 0x1 4000 0904 (EMAC1)

See *Mode Register 1 (EMACx_MR1)* on page 755.

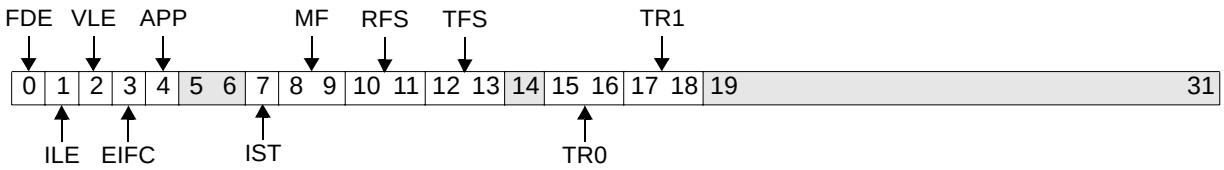


Figure 0-11. Mode Register 1 (EMACx_MR1)

0	FDE	Full-Duplex Enable 0 Disable simultaneous transmit and receive 1 Enable simultaneous transmit and receive	
1	ILE	Internal Loop-back Enable 0 No wrap back 1 Transmitted packets wrapped back to receive FIFO	Full Duplex must also be set (EMACx_MRI[FDE]=1).
2	VLE	VLAN Enable 0 Disable processing of VLAN Tags 1 Enable processing of VLAN Tags	
3	EIFC	Enable Integrated Flow Control 0 Disable integrated flow control mechanism 1 Enable integrated flow control mechanism	Refer to "Flow Control" on page 22-15 for more details. Set EMACx_MR1[EIFC] = 0 in half-duplex mode.
4	APP	Allow Pause Packet 0 Disables processing of incoming control (pause) packets 1 Enables processing of incoming control (pause) packets	
5:6		Reserved	Always zero
7	IST	Ignore SQE test 0 Wait for end of SQE test period before activation of valid signal 1 Do not wait for end of SQE test period before activation of valid signal	EMACx_MR1[IST] = 0 only during half-duplex operation on 10 Mbps media.
8:9	MF	Medium Frequency 00 10 Mbps (Ethernet mode) 01 100 Mbps (Fast Ethernet mode) 10 Reserved 11 Reserved	Defines the possible operational frequency on the MII interface.

10:11	RFS	Receive (RX) FIFO Size 00 512 bytes 01 1 KB 10 2 KB 11 4 KB	
12:13	TFS	Transmit (TX) FIFO Size 00 Reserved 01 1 KB 10 2 KB 11 Reserved	
14		Reserved	Always 0
15:16	TR0	Transmit Request 0 00 Single packet 01 Multiple packets 10 Dependent mode (bits 17:18 must also be programmed to 10) 11 Reserved	Defines the different modes for using transmit channel 0 of EMAC.
17:18	TR1	Transmit Request 1 00 Single packet 01 Multiple packets 10 Dependent mode (bits 15:16 must also be programmed to 10) 11 Reserved	Defines the different modes for using transmit channel 1 of EMAC.
19:31		Reserved	

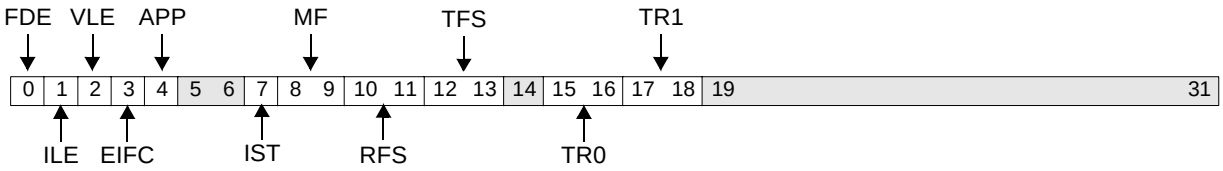


Figure 29-108. Mode Register 1 (EMACx_MR1)

0	FDE	Full-Duplex Enable 0 Disable simultaneous transmit and receive 1 Enable simultaneous transmit and receive	
1	ILE	Internal Loop-back Enable 0 No wrap back 1 Transmitted packets wrapped back to receive FIFO	Full Duplex must also be set (EMACx_MRI[FDE]=1).
2	VLE	VLAN Enable 0 Disable processing of VLAN Tags 1 Enable processing of VLAN Tags	
3	EIFC	Enable Integrated Flow Control 0 Disable integrated flow control mechanism 1 Enable integrated flow control mechanism	Refer to "Flow Control" on page -744 for more details. Set EMACx_MR1[EIFC] = 0 in half-duplex mode.

4	APP	Allow Pause Packet 0 Disables processing of incoming control (pause) packets 1 Enables processing of incoming control (pause) packets	
5:6		Reserved	Always zero
7	IST	Ignore SQE test 0 Wait for end of SQE test period before activation of valid signal 1 Do not wait for end of SQE test period before activation of valid signal	EMACx_MR1[IST] = 0 only during half-duplex operation on 10 Mbps media.
8:9	MF	Medium Frequency 00 10 Mbps (Ethernet mode) 01 100 Mbps (Fast Ethernet mode) 10 Reserved 11 Reserved	Defines the possible operational frequency on the MII interface.
10:11	RFS	Receive (RX) FIFO Size 00 512 bytes 01 1 KB 10 2 KB 11 4 KB	
12:13	TFS	Transmit (TX) FIFO Size 00 Reserved 01 1 KB 10 2 KB 11 Reserved	
14		Reserved	Always 0
15:16	TR0	Transmit Request 0 00 Single packet 01 Multiple packets 10 Dependent mode (bits 17:18 must also be programmed to 10) 11 Reserved	Defines the different modes for using transmit channel 0 of EMAC.
17:18	TR1	Transmit Request 1 00 Single packet 01 Multiple packets 10 Dependent mode (bits 15:16 must also be programmed to 10) 11 Reserved	Defines the different modes for using transmit channel 1 of EMAC.
19:31		Reserved	

MMIO 0x1 4000 086C (EMAC0), 0x1 4000 096C (EMAC1)

See *Number of Octets Received (EMACx_OCRX)* on page 772.

Figure 0-12. Number of Octets Received (EMACx_OCRX)

0:31	OCRX	Number of octets (bytes) received.

Figure 29-109. Number of Octets Received (EMACx_OCRX)

0:31	OCRX	Number of octets (bytes) received.

MMIO 0x1 4000 0868 (EMAC0), 0x1 4000 0968 (EMAC1)

See *Number of Octets Transmitted (EMACx_OCTX)* on page 771.

Figure 0-13. Number of Octets Transmitted (EMACx_OCTX)

0:31	OCTX	Number of octets (bytes) transmitted.

Figure 29-110. Number of Octets Transmitted (EMACx_OCTX)

0:31	OCTX	Number of octets (bytes) transmitted.

MMIO 0x1 4000 082C (EMAC0), 0x1 4000 092C (EMAC1)

See *Pause Timer Register (EMACx_PTR)* on page 766.

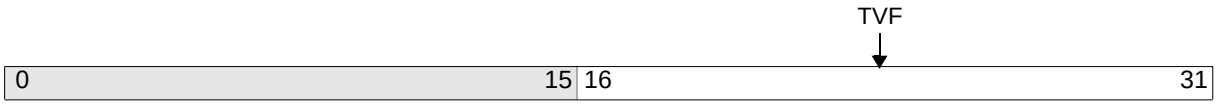


Figure 0-14. Pause Timer Register (EMACx_PTR)

0:15		Reserved
16:31	TVF	Timer Value Field

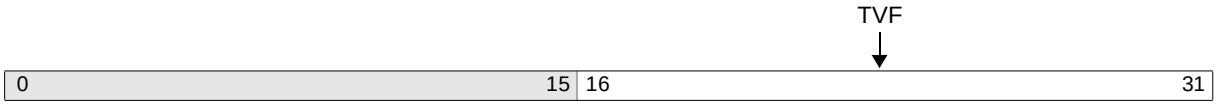


Figure 29-111. Pause Timer Register (EMACx_PTR)

0:15		Reserved
16:31	TVF	Timer Value Field

MMIO 0x1 4000 0810 (EMAC0), 0x1 4000 0910 (EMAC1)

See *Receive Mode Register (EMACx_RMR)* on page 759.

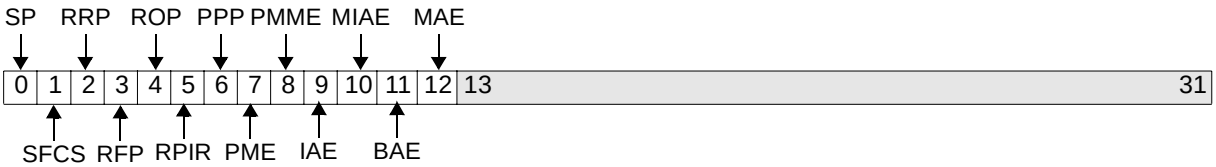


Figure 0-15. Receive Mode Register (EMACx_RMR)

0	SP	Strip Padding 0 Do not strip pad bytes from the received packet. 1 Strip pad/FCS bytes from the received packet.
1	SFCS	Strip FCS 0 Do not strip FCS bytes from the received packet. 1 Strip FCS bytes from the received packet.
2	RRP	Receive Runt Packets 0 Discard packets less than 64 bytes in length. 1 Receive packets less than 64 bytes in length.
3	RFP	Allow Receive Packets with a FCS Error 0 Discard packets containing a FCS error. 1 Receive packets containing a FCS error.
4	ROP	Receive Oversize Packet 0 Discard packets that activate Packet Is Too Long error. 1 Receive packets that activate Packet Is Too Long error.
5	RPIR	Receive Packets with In Range Error 0 Discard packets that activate In Range Error. 1 Receive packets that activate In Range Error.
6	PPP	Propagate Pause Packet 0 Do not propagate incoming pause packet to MAL; remove packet from FIFO. 1 Propagate incoming pause packet to MAL.
7	PME	Promiscuous Mode Enable 0 Do not enable promiscuous mode. 1 Accept all packets.

8	PMME	Promiscuous Multicast Mode Enable 0 Do not accept all multicast packets. 1 Accept all multicast packets.
9	IAE	Individual Address Enable 0 Do not compare address of received packets with content of individual address register. 1 Compare address of received packets with content of individual address register.
10	MIAE	Multiple Individual Address Enable 0 Do not compare address of received packets with hash table of individual addresses. 1 Compare address of received packets with hash table of individual addresses.
11	BAE	Broadcast Address Enable 0 Do not compare address of received packets with broadcast addresses. 1 Compare address of received packets with broadcast addresses.
12	MAE	Multicast Address Enable 0 Do not compare address of received packets with multicast addresses. 1 Compare address of received packets with multicast addresses.
13:31		Reserved

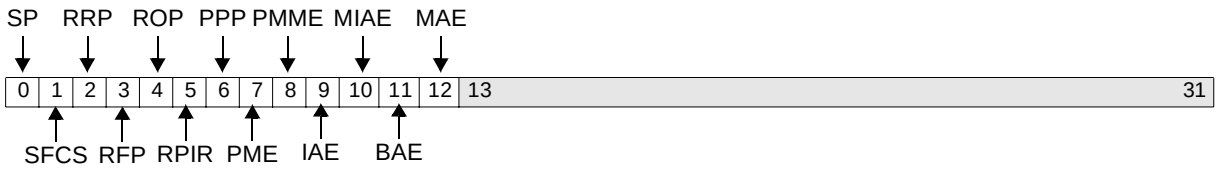


Figure 29-112. Receive Mode Register (EMACx_RMR)

0	SP	Strip Padding 0 Do not strip pad bytes from the received packet. 1 Strip pad/FCS bytes from the received packet.
1	SFCS	Strip FCS 0 Do not strip FCS bytes from the received packet. 1 Strip FCS bytes from the received packet.
2	RRP	Receive Runt Packets 0 Discard packets less than 64 bytes in length. 1 Receive packets less than 64 bytes in length.
3	RFP	Allow Receive Packets with a FCS Error 0 Discard packets containing a FCS error. 1 Receive packets containing a FCS error.

4	ROP	Receive Oversize Packet 0 Discard packets that activate Packet Is Too Long error. 1 Receive packets that activate Packet Is Too Long error.
5	RPIR	Receive Packets with In Range Error 0 Discard packets that activate In Range Error. 1 Receive packets that activate In Range Error.
6	PPP	Propagate Pause Packet 0 Do not propagate incoming pause packet to MAL; remove packet from FIFO. 1 Propagate incoming pause packet to MAL.
7	PME	Promiscuous Mode Enable 0 Do not enable promiscuous mode. 1 Accept all packets.
8	PMME	Promiscuous Multicast Mode Enable 0 Do not accept all multicast packets. 1 Accept all multicast packets.
9	IAE	Individual Address Enable 0 Do not compare address of received packets with content of individual address register. 1 Compare address of received packets with content of individual address register.
10	MIAE	Multiple Individual Address Enable 0 Do not compare address of received packets with hash table of individual addresses. 1 Compare address of received packets with hash table of individual addresses.
11	BAE	Broadcast Address Enable 0 Do not compare address of received packets with broadcast addresses. 1 Compare address of received packets with broadcast addresses.
12	MAE	Multicast Address Enable 0 Do not compare address of received packets with multicast addresses. 1 Compare address of received packets with multicast addresses.
13:31		Reserved

MMIO 0x1 4000 0864 (EMAC0), 0x1 4000 0964 (EMAC1)

See *Receive Low/High Water Mark Register (EMACx_RWMR)* on page 770.

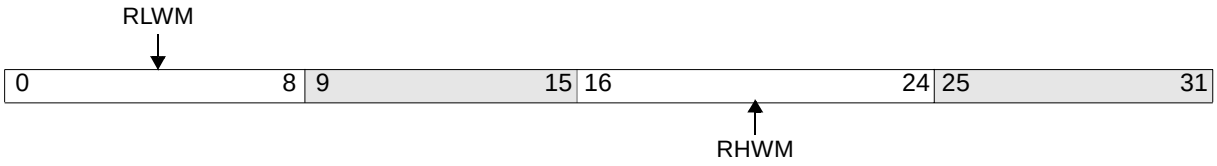


Figure 0-16. Receive Low/High Water Mark Register (EMACx_RWMR)

0:8	RLWM	Receive Low Water Mark
9:15		Reserved
16:24	RHWM	Receive High Water Mark
25:31		Reserved

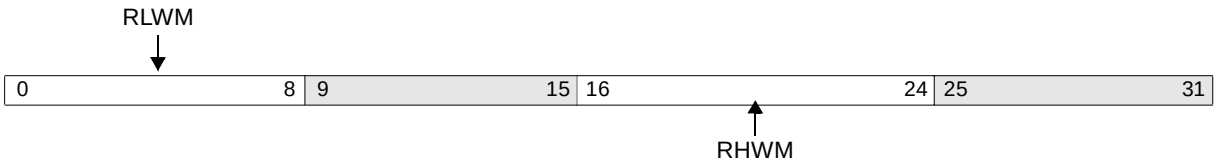


Figure 29-113. Receive Low/High Water Mark Register (EMACx_RWMR)

0:8	RLWM	Receive Low Water Mark
9:15		Reserved
16:24	RHWM	Receive High Water Mark
25:31		Reserved

MMIO 0x1 4000085C (EMAC0), 0x1 4000095C (EMAC1)

See *STA Control Register (EMACx_STACR)* on page 769.

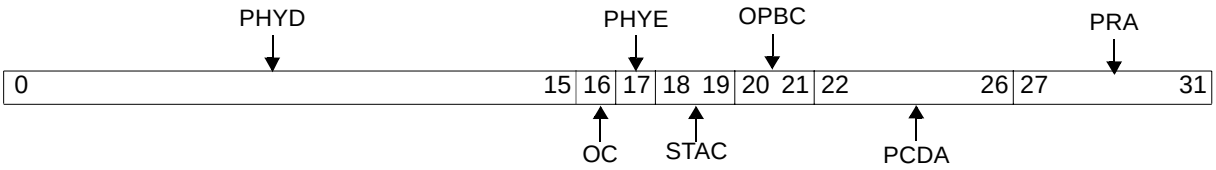


Figure 0-17. STA Control Register (EMACx_STACR)

0:15	PHYD	PHY data	Data to be sent to the PHY if the command is a write, or data is read from the PHY if the command is a read.
16	OC	Operation Complete 0 EMACx_STACR is addressed 1 PHY data transfer complete	
17	PHYE	PHY Error 0 Successful read transaction 1 Read transaction was not successful	EMACx_STACR[PHYE] = 0 when a read is successful.
18:19	STAC	STA Command 00 Reserved 01 Read 10 Write 11 Reserved	EMAC sets EMACx_STACR[STAC] = 0 when the command is completed.
20:21	OPBC	OPB Bus Clock Frequency 00 50 MHz 01 66 MHz 10 83 MHz 11 100 MHz	EMACx_STACR[OPBC] is used to generate the Management Data Clock (EMCMDClk). When the operational frequency differs from those in the list, then the next greater frequency should be chosen.
22:26	PCDA	PHY Command Destination Address	
27:31	PRA	PHY Register Address	

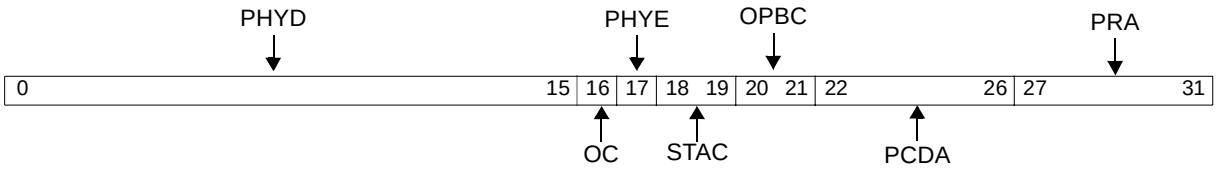


Figure 29-114. STA Control Register (EMACx_STACR)

0:15	PHYD	PHY data	Data to be sent to the PHY if the command is a write, or data is read from the PHY if the command is a read.
16	OC	Operation Complete 0 EMACx_STACR is addressed 1 PHY data transfer complete	
17	PHYE	PHY Error 0 Successful read transaction 1 Read transaction was not successful	EMACx_STACR[PHYE] = 0 when a read is successful.
18:19	STAC	STA Command 00 Reserved 01 Read 10 Write 11 Reserved	EMAC sets EMACx_STACR[STAC] = 0 when the command is completed.
20:21	OPBC	OPB Bus Clock Frequency 00 50 MHz 01 66 MHz 10 83 MHz 11 100 MHz	EMACx_STACR[OPBC] is used to generate the Management Data Clock (EMCMDCIk). When the operational frequency differs from those in the list, then the next greater frequency should be chosen.
22:26	PCDA	PHY Command Destination Address	
27:31	PRA	PHY Register Address	

MMIO 0x1 4000 0808 (EMAC0), 0x1 4000 0908 (EMAC1)

See *Transmit Mode Register 0 (EMACx_TMR0)* on page 756.

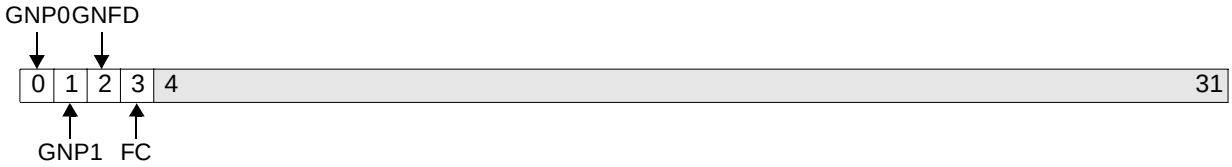


Figure 0-18. Transmit Mode Register 0 (EMACx_TMR0)

0	GNP0	Get New Packet 0 0 Writing 0 has no effect. 1 Packet ready for transmission on TX Channel 0	EMACx_TMR0[GNP0] = 0 if EMAC is programmed in dependent mode.
1	GNP1	Get New Packet 1 0 Writing 0 has no effect. 1 Packet ready for transmission on TX Channel 1	EMACx_TMR0[GNP1] = 0 if EMAC is programmed in dependent mode.
2	GNPD	Get New Packet for Dependent Mode 0 Writing 0 to this bit has no effect 1 Packet ready for transmission in dependent mode	EMACx_TMR0[GNPD] = 0 if EMAC is not programmed in dependent mode. EMACx_TMR0[GNPD] = 1 activates the EMAC transmit path in dependent mode.
3	FC	First Channel 0 Activate TX Channel 0 first when GNPD is 1 1 Activate TX Channel 1 first when GNPD is 1	EMACx_TMR0[FC] is only meaningful in dependent mode, after resetting EMACx_ISR[DBDM]. EMACx_TMR0[FC] = 0 if EMAC is not programmed in dependent mode.
4:31		Reserved	

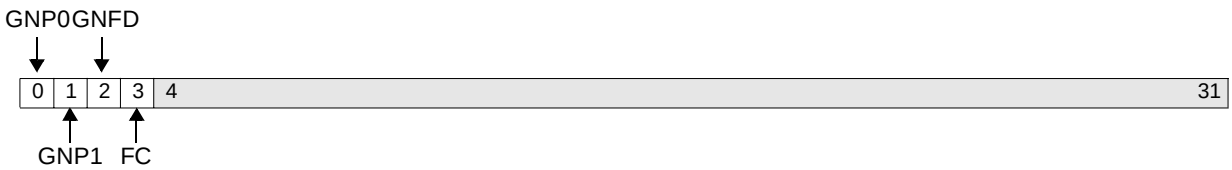


Figure 29-115. Transmit Mode Register 0 (EMACx_TMR0)

0	GNP0	Get New Packet 0 0 Writing 0 has no effect. 1 Packet ready for transmission on TX Channel 0	EMACx_TMR0[GNP0] = 0 if EMAC is programmed in dependent mode.
1	GNP1	Get New Packet 1 0 Writing 0 has no effect. 1 Packet ready for transmission on TX Channel 1	EMACx_TMR0[GNP1] = 0 if EMAC is programmed in dependent mode.

2	GNPD	Get New Packet for Dependent Mode 0 Writing 0 to this bit has no effect 1 Packet ready for transmission in dependent mode	EMACx_TMR0[GNPD] = 0 if EMAC is not programmed in dependent mode. EMACx_TMR0[GNPD] = 1 activates the EMAC transmit path in dependent mode.
3	FC	First Channel 0 Activate TX Channel 0 first when GNPD is 1 1 Activate TX Channel 1 first when GNPD is 1	EMACx_TMR0[FC] is only meaningful in dependent mode, after resetting EMACx_ISR[DBDM]. EMACx_TMR0[FC] = 0 if EMAC is not programmed in dependent mode.
4:31		Reserved	

MMIO 0x1 4000 080C (EMAC0), 0x1 4000 090C (EMAC1)

See *Transmit Mode Register 1 (EMACx_TMR1)* on page 758.

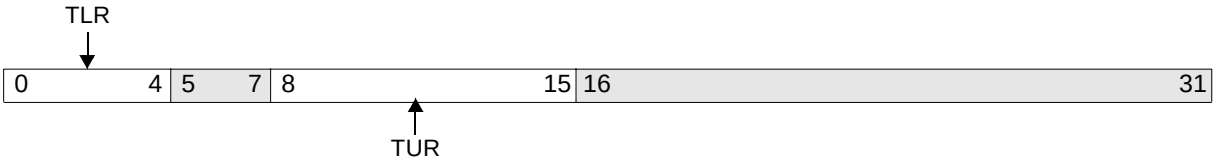


Figure 0-19. Transmit Mode Register 1 (EMACx_TMR1)

0:4	TLR	Transmit Low Request
5:7		Reserved
8:15	TUR	Transmit Urgent Request
16:31		Reserved

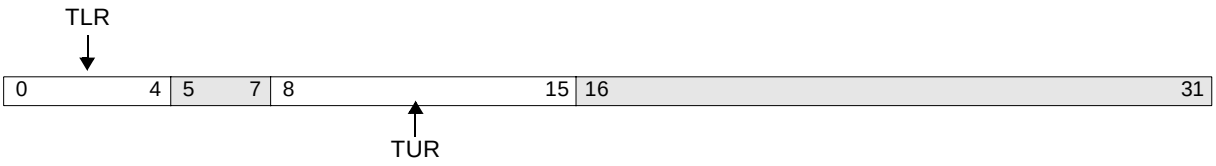


Figure 29-116. Transmit Mode Register 1 (EMACx_TMR1)

0:4	TLR	Transmit Low Request
5:7		Reserved
8:15	TUR	Transmit Urgent Request
16:31		Reserved

MMIO 0x1 4000 0860 (EMAC0), 0x1 4000 0960 (EMAC1)

See *Transmit Request Threshold Register (EMACx_TRTR)* on page 770.

Figure 0-20. Transmit Request Threshold Register (EMACx_TRTR)

0:4	TRT	Transmit Request Threshold The following number of bytes must be placed in the Transmit FIFO before initiating a transmit request. 00000 64 bytes 00001 128 bytes 00010 192 bytes 00011 256 bytes . . . 11111 2048 bytes
5:31		Reserved

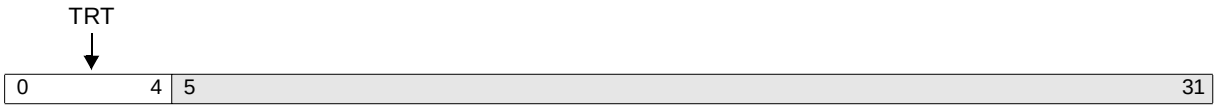


Figure 29-117. Transmit Request Threshold Register (EMACx_TRTR)

0:4	TRT	Transmit Request Threshold The following number of bytes must be placed in the Transmit FIFO before initiating a transmit request. 00000 64 bytes 00001 128 bytes 00010 192 bytes 00011 256 bytes . . . 11111 2048 bytes
5:31		Reserved

MMIO 0x1 4000 0828 (EMAC0), 0x1 4000 0928 (EMAC1)

See *VLAN TCI Register (EMACx_VTCI)* on page 766.

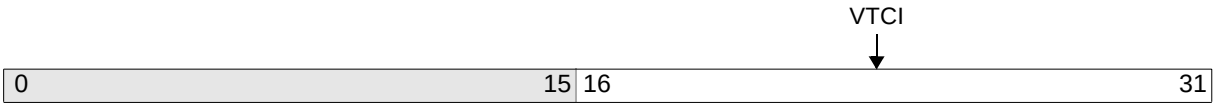


Figure 0-21. VLAN TCI Register (EMACx_VTCI)

0:15		Reserved
16:31	VTCI	VLAN TCI tag

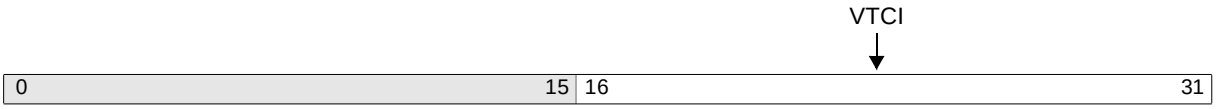


Figure 29-118. VLAN TCI Register (EMACx_VTCI)

0:15		Reserved
16:31	VTCI	VLAN TCI tag

MMIO 0x1 4000 0824 (EMAC0), 0x1 4000 0924 (EMAC1)

See *VLAN TPID Register (EMACx_VTPID)* on page 765.

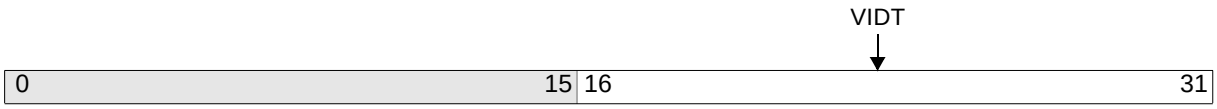


Figure 0-22. VLAN TPID Register (EMACx_VTPID)

0:15		Reserved
16:31	VIDT	VLAN ID tag

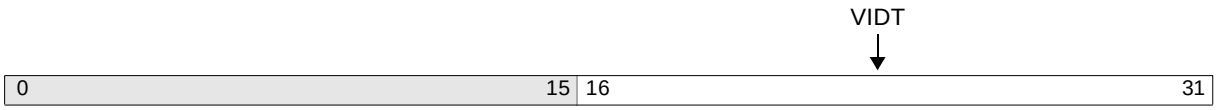


Figure 29-119. VLAN TPID Register (EMACx_VTPID)

0:15		Reserved
16:31	VIDT	VLAN ID tag

MMIO 0x1 40000 71C Read-Only

See *GPIO Input Register (GPIO0_IR)* on page 807.

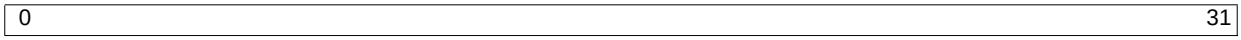


Figure 0-1. GPIO Input Register (GPIO0_IR)

0:31		GPIO register bits

Figure 29-120. GPIO Input Register (GPIO0_IR)

0:31		GPIO register bits

MMIO 0x1 4000 0718 R/W

See *GPIO Open Drain Register (GPIO0_ODR)* on page 806.

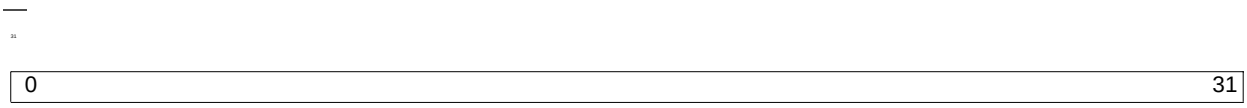


Figure 0-2. GPIO Open Drain Register (GPIO0_ODR)

0:31		GPIO0_ODR register bits

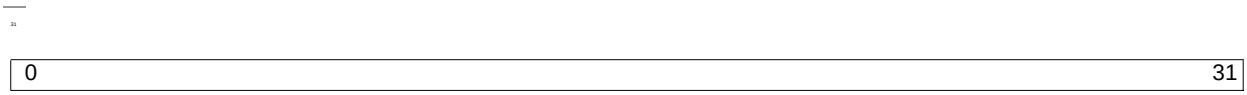


Figure 29-121. GPIO Open Drain Register (GPIO0_ODR)

0:31		GPIO0_ODR register bits

MMIO 0x1 4000 0700 R/W

See *GPIO Output Register (GPIO0_OR)* on page 805.

0	31

Figure 0-3. GPIO Output Register (GPIO0_OR)

0:31		GPIO0_OR register bits

0	31

Figure 29-122. GPIO Output Register (GPIO0_OR)

0:31		GPIO0_OR register bits

MMIO 0x1 40000704

See *GPIO Three-State Control Register (GPIO0_TCR)* on page 806.

0	31

Figure 0-4. GPIO Three-State Register (GPIO0_TCR)

0:31		GPIO0_TCR register bits

0	31

Figure 29-123. GPIO Three-State Register (GPIO0_TCR)

0:31		GPIO0_TCR register bits

MMIO 0x140000A80–0x140000A90 R/

See *GPT Compare Timer Registers (GPT0_COMP0 - GPT0_COMP4)* on page 403.

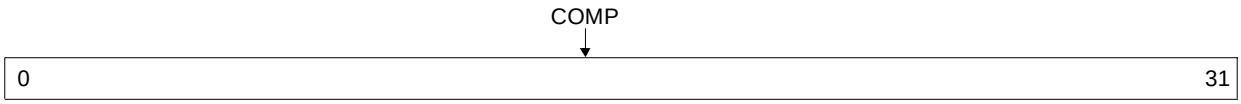


Figure 0-1. Compare Timer Register (GPT0_COMP0 - GPT0_COMP4)

0:31	COMP	Compare Timer

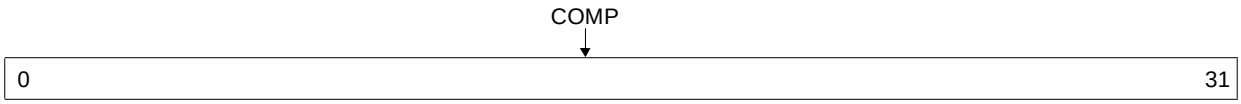


Figure 29-124. Compare Timer Register (GPT0_COMP0 - GPT0_COMP4)

0:31	COMP	Compare Timer

MMIO 0x140000A24 R/W

See *GPT Interrupt Enable Register (GPT0_IE)* on page 403.

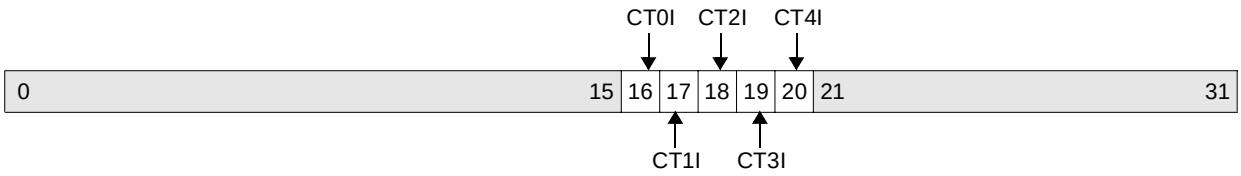


Figure 0-2. GPT Interrupt Enable Register (GPT0_IE)

0:15		Reserved
16	CT0I	Compare Timer 0 Interrupt Enable 0 Compare timer 0 interrupt enable disabled 1 Compare timer 0 interrupt enable enabled
17	CT1I	Compare Timer 1 Interrupt Enable 0 Compare timer 1 interrupt enable disabled 1 Compare timer 1 interrupt enable enabled
18	CT2I	Compare Timer 2 Interrupt Enable 0 Compare timer 2 interrupt enable disabled 1 Compare timer 2 interrupt enable enabled
19	CT3I	Compare Timer 3 Interrupt Enable 0 Compare timer 3 interrupt enable disabled 1 Compare timer 3 interrupt enable enabled
20	CT4I	Compare Timer 4 Interrupt Enable 0 Compare timer 4 interrupt enable disabled 1 Compare timer 4 interrupt enable enabled
21:31		Reserved

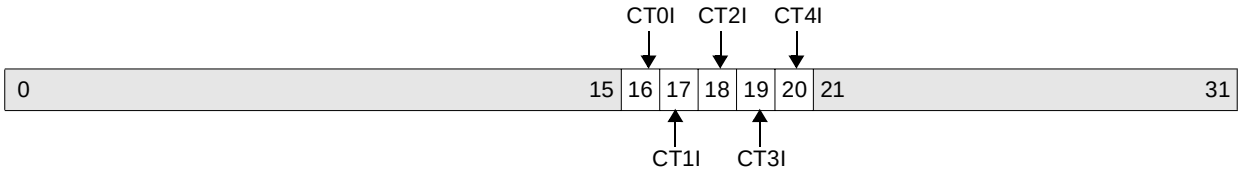


Figure 29-125. GPT Interrupt Enable Register (GPT0_IE)

0:15		Reserved
16	CT0I	Compare Timer 0 Interrupt Enable 0 Compare timer 0 interrupt enable disabled 1 Compare timer 0 interrupt enable enabled
17	CT1I	Compare Timer 1 Interrupt Enable 0 Compare timer 1 interrupt enable disabled 1 Compare timer 1 interrupt enable enabled
18	CT2I	Compare Timer 2 Interrupt Enable 0 Compare timer 2 interrupt enable disabled 1 Compare timer 2 interrupt enable enabled
19	CT3I	Compare Timer 3 Interrupt Enable 0 Compare timer 3 interrupt enable disabled 1 Compare timer 3 interrupt enable enabled
20	CT4I	Compare Timer 4 Interrupt Enable 0 Compare timer 4 interrupt enable disabled 1 Compare timer 4 interrupt enable enabled
21:31		Reserved

MMIO 0x140000A18 R/W

See *GPT Interrupt Mask Register (GPT0_IM)* on page 401.

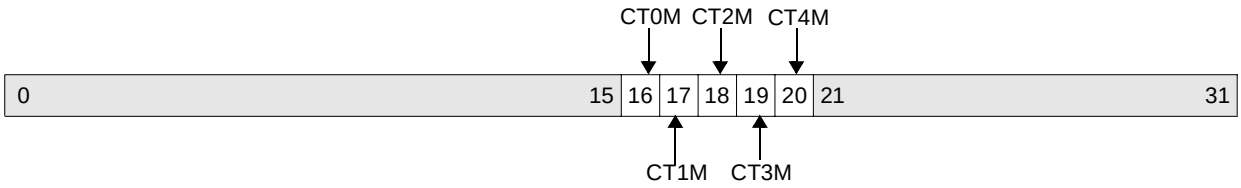


Figure 0-3. GPT Interrupt Mask Register (GPT0_IM)

0:15		Reserved
16	CT0M	Compare Timer 0 Interrupt Mask 0 Compare timer 0 interrupt mask disabled 1 Compare timer 0 interrupt mask enabled
17	CT1M	Compare Timer 1 Interrupt Mask 0 Compare timer 1 interrupt mask disabled 1 Compare timer 1 interrupt mask enabled
18	CT2M	Compare Timer 2 Interrupt Maskl 0 Compare timer 2 interrupt mask disabled 1 Compare timer 2 interrupt mask enabled
19	CT3M	Compare Timer 3 Interrupt Mask 0 Compare timer 3 interrupt mask disabled 1 Compare timer 3 interrupt mask enabled
20	CT4M	Compare Timer 4 Interrupt Mask 0 Compare timer 4 interrupt mask disabled 1 Compare timer 4 interrupt mask enabled
21:31		Reserved

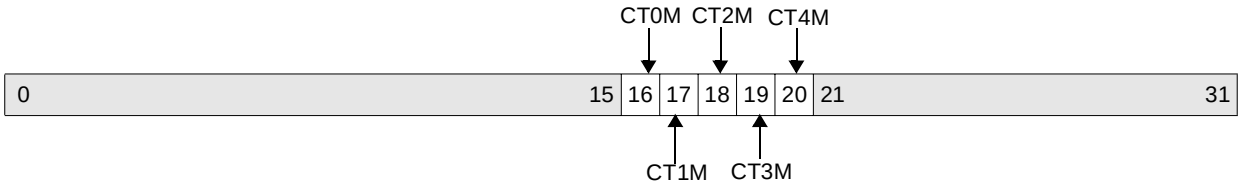


Figure 29-126. GPT Interrupt Mask Register (GPT0_IM)

0:15		Reserved
16	CT0M	Compare Timer 0 Interrupt Mask 0 Compare timer 0 interrupt mask disabled 1 Compare timer 0 interrupt mask enabled
17	CT1M	Compare Timer 1 Interrupt Mask 0 Compare timer 1 interrupt mask disabled 1 Compare timer 1 interrupt mask enabled
18	CT2M	Compare Timer 2 Interrupt Maskl 0 Compare timer 2 interrupt mask disabled 1 Compare timer 2 interrupt mask enabled
19	CT3M	Compare Timer 3 Interrupt Mask 0 Compare timer 3 interrupt mask disabled 1 Compare timer 3 interrupt mask enabled
20	CT4M	Compare Timer 4 Interrupt Mask 0 Compare timer 4 interrupt mask disabled 1 Compare timer 4 interrupt mask enabled
21:31		Reserved

MMIO 0x140000A1C R/W

See *GPT Interrupt Status Register (GPT0_ISS and GPT0_ISC)* on page 402.

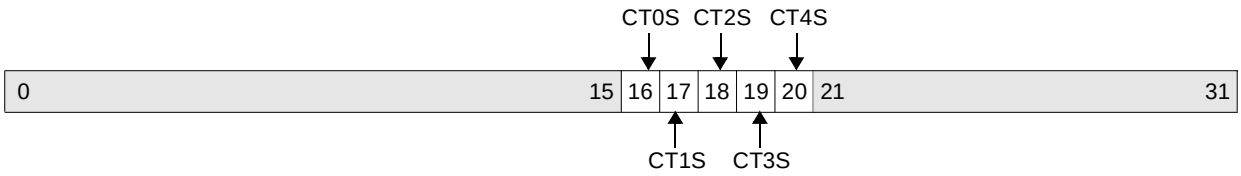


Figure 0-4. GPT Interrupt Status Register (GPT0_ISS and GPT0_ISC)

0:15		Reserved
16	CT0S	Compare Timer 0 Interrupt Status 0 Compare timer 0 interrupt status disabled 1 Compare timer 0 interrupt status enabled
17	CT1S	Compare Timer 1 Interrupt Status 0 Compare timer 1 interrupt status disabled 1 Compare timer 1 interrupt status enabled
18	CT2IS	Compare Timer 2 Interrupt Status 0 Compare timer 2 interrupt status disabled 1 Compare timer 2 interrupt status enabled
19	CT3S	Compare Timer 3 Interrupt Status 0 Compare timer 3 interrupt status disabled 1 Compare timer 3 interrupt status enabled
20	CT4S	Compare Timer 4 Interrupt Status 0 Compare timer 4 interrupt status disabled 1 Compare timer 4 interrupt status enabled
21:31		Reserved

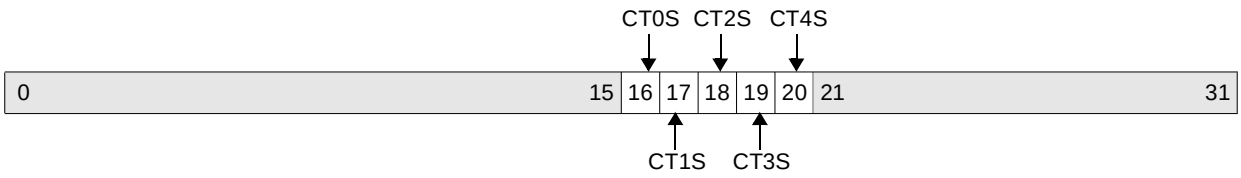


Figure 29-127. GPT Interrupt Status Register (GPT0_ISS and GPT0_ISC)

0:15		Reserved
16	CT0S	Compare Timer 0 Interrupt Status 0 Compare timer 0 interrupt status disabled 1 Compare timer 0 interrupt status enabled
17	CT1S	Compare Timer 1 Interrupt Status 0 Compare timer 1 interrupt status disabled 1 Compare timer 1 interrupt status enabled
18	CT2IS	Compare Timer 2 Interrupt Status 0 Compare timer 2 interrupt status disabled 1 Compare timer 2 interrupt status enabled
19	CT3S	Compare Timer 3 Interrupt Status 0 Compare timer 3 interrupt status disabled 1 Compare timer 3 interrupt status enabled
20	CT4S	Compare Timer 4 Interrupt Status 0 Compare timer 4 interrupt status disabled 1 Compare timer 4 interrupt status enabled
21:31		Reserved

MMIO 0x140000AC0–0x140000AD0 R/W

See *GPT Compare Mask Registers (GPT0_MASK0 - GPT0_MASK4)* on page 405.

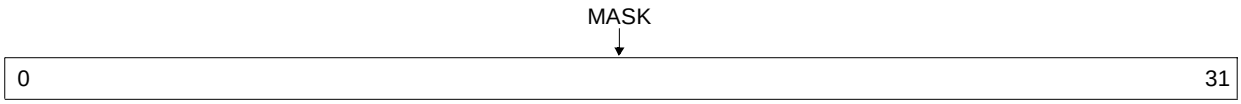


Figure 0-5. Compare Mask Register (GPT0_MASK0 - GPT0_MASK4)

0:31	MASK	Comparison Function 0 Comparison enabled 1 Comparison disabled	When set to 1, a valid comparison is assumed.

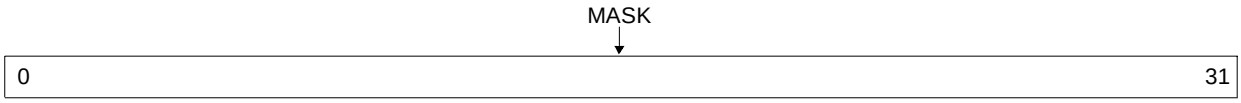


Figure 29-128. Compare Mask Register (GPT0_MASK0 - GPT0_MASK4)

0:31	MASK	Comparison Function 0 Comparison enabled 1 Comparison disabled	When set to 1, a valid comparison is assumed.

MMIO 0x140000A10 R/W

See *GPT Output Enable Register (GPT0_OE)* on page 399.

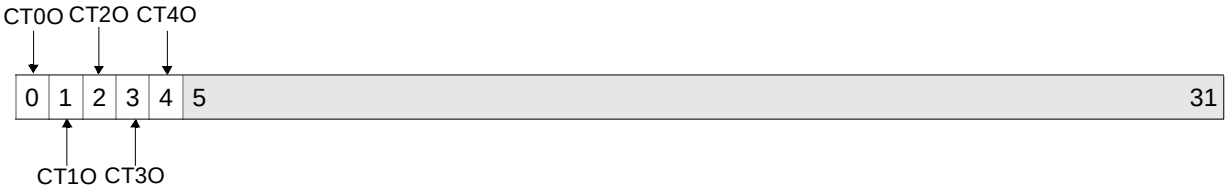


Figure 0-6. GPT Output Enable Register (GPT0_OE)

0	CT0O	Compare Timer 0 Output 0 Compare timer 0 output disabled 1 Compare timer 0 output enabled
1	CT1O	Compare Timer 1 Output 0 Compare timer 1 output disabled 1 Compare timer 1 output enabled
2	CT2O	Compare Timer 2 Output 0 Compare timer 2 output disabled 1 Compare timer 2 output enabled
3	CT3O	Compare Timer 3 Output 0 Compare timer 3 output disabled 1 Compare timer 3 output enabled
4	CT4O	Compare Timer 4 Output 0 Compare timer 4 output disabled 1 Compare timer 4 output enabled
5:31		Reserved

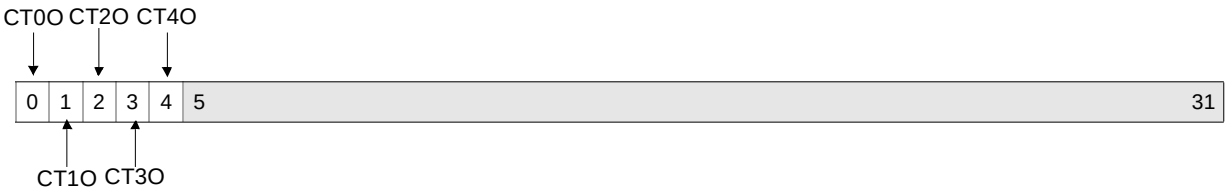


Figure 29-129. GPT Output Enable Register (GPT0_OE)

0	CT0O	Compare Timer 0 Output 0 Compare timer 0 output disabled 1 Compare timer 0 output enabled
1	CT1O	Compare Timer 1 Output 0 Compare timer 1 output disabled 1 Compare timer 1 output enabled

2	CT2O	Compare Timer 2 Output 0 Compare timer 2 output disabled 1 Compare timer 2 output enabled
3	CT3O	Compare Timer 3 Output 0 Compare timer 3 output disabled 1 Compare timer 3 output enabled
4	CT4O	Compare Timer 4 Output 0 Compare timer 4 output disabled 1 Compare timer 4 output enabled
5:31		Reserved

MMIO 0x140000A14 R/W

See *GPT Output Level Register (GPT0_OL)* on page 400.

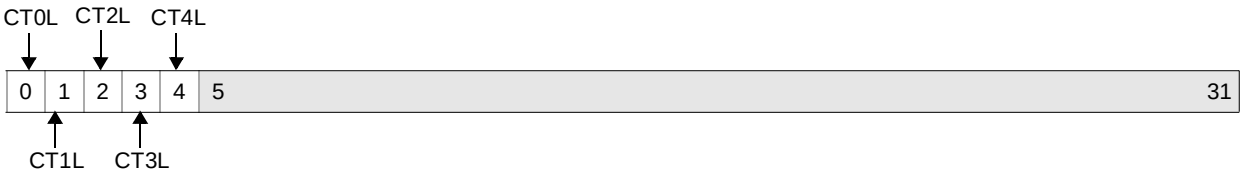


Figure 0-7. GPT Output Level Register (GPT0_OL)

0	CT0L	Compare Timer 0 Output Level 0 Compare timer 0 output level disabled 1 Compare timer 0 output level enabled
1	CT1L	Compare Timer 1 output level 0 Compare timer 1 output level disabled 1 Compare timer 1 output level enabled
2	CT2L	Compare Timer 2 output level 0 Compare timer 2 output level disabled 1 Compare timer 2 output level enabled
3	CT3L	Compare Timer 3 output level 0 Compare timer 3 output level disabled 1 Compare timer 3 output level enabled
4	CT4L	Compare Timer 4 output level 0 Compare timer 4 output level disabled 1 Compare timer 4 output level enabled
5:31		Reserved

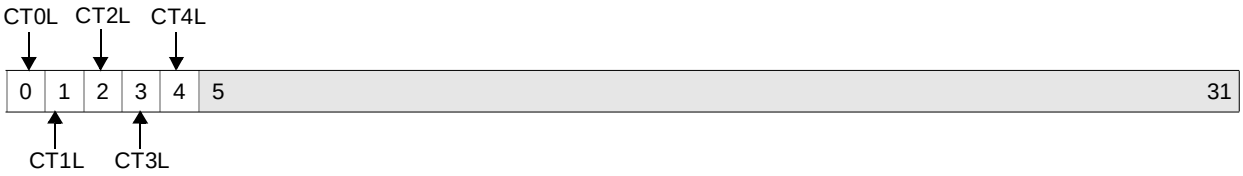


Figure 29-130. GPT Output Level Register (GPT0_OL)

0	CT0L	Compare Timer 0 Output Level 0 Compare timer 0 output level disabled 1 Compare timer 0 output level enabled
1	CT1L	Compare Timer 1 output level 0 Compare timer 1 output level disabled 1 Compare timer 1 output level enabled

2	CT2L	Compare Timer 2 output level 0 Compare timer 2 output level disabled 1 Compare timer 2 output level enabled
3	CT3L	Compare Timer 3 output level 0 Compare timer 3 output level disabled 1 Compare timer 3 output level enabled
4	CT4L	Compare Timer 4 output level 0 Compare timer 4 output level disabled 1 Compare timer 4 output level enabled
5:31		Reserved

MMIO 0x140000A00 R/W

See *GPT Time Base Counter Register (GPT0_TBC)* on page 398.

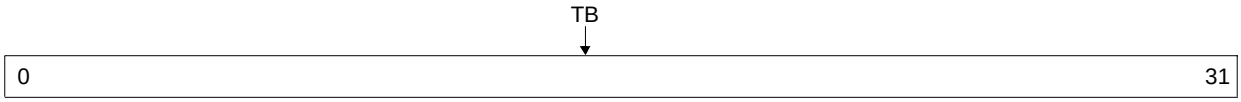


Figure 0-8. Time Base Counter Register (GPT0_TBC)

0:31	TB	Time Base

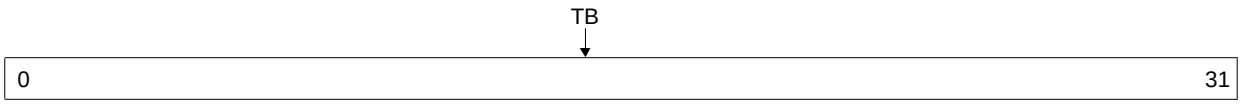


Figure 29-131. Time Base Counter Register (GPT0_TBC)

0:31	TB	Time Base

MMIO 0x1 4000040C IIC0, 0x1 4000050c IIC1 R/W

See *IICx Clock Divide Register (IICx_CLKDIV)* on page 823.

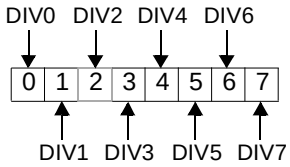


Figure 0-1. IICx Clock Divide Register (IICx_CLKDIV)

0	DIV0	Divisor bit 0
1	DIV1	Divisor bit 1
2	DIV2	Divisor bit 2
3	DIV3	Divisor bit 3
4	DIV4	Divisor bit 4
5	DIV5	Divisor bit 5
6	DIV6	Divisor bit 6
7	DIV7	Divisor bit 7

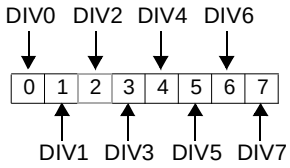


Figure 29-132. IICx Clock Divide Register (IICx_CLKDIV)

0	DIV0	Divisor bit 0
1	DIV1	Divisor bit 1
2	DIV2	Divisor bit 2
3	DIV3	Divisor bit 3
4	DIV4	Divisor bit 4
5	DIV5	Divisor bit 5
6	DIV6	Divisor bit 6
7	DIV7	Divisor bit 7

MMIO 0x1 40000406 IIC0, 0x1 40000506 IIC1 R/W

See *IICx Control Register (IICx_CNTL)* on page 814.

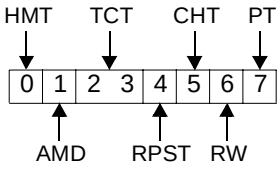


Figure 0-2. IICx Control Register (IICx_CNTL)

0	HMT	Halt Master Transfer 0 Normal transfer operation. 1 Issue Stop signal on the IIC bus as soon as possible to halt master transfer.	If no transfer is in progress, no action is taken. IICx_CNTL[PT] needs not be set. If IICx_MDCNTL[EINT] = 1, an interrupt is generated.
1	AMD	Addressing Mode 0 Use 7-bit addressing. 1 Use 10-bit addressing.	Does not affect slave transfers.
2:3	TCT	Transfer Count 00 Transfer one byte. 01 Transfer two bytes. 10 Transfer three bytes. 11 Transfer four bytes.	
4	RPST	Repeated Start 0 Normal start operation 1 Use repeated Start function to start transfer.	
5	CHT	Chain Transfer 0 Transfer is only or last transfer. 1 Transfer is one of a sequence of transfers (but not last in sequence).	Completion of a requested transfer causes a Stop condition to be generated on the IIC bus.
6	RW	Read/Write 0 Transfer is a write. 1 Transfer is a read.	
7	PT	Pending Transfer 0 Most recent requested transfer is complete. 1 Start transfer if bus is free.	

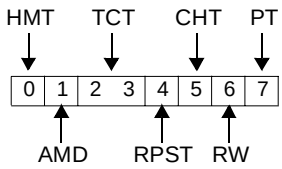


Figure 29-133. IICx Control Register (IICx_CNTL)

0	HMT	Halt Master Transfer 0 Normal transfer operation. 1 Issue Stop signal on the IIC bus as soon as possible to halt master transfer.	If no transfer is in progress, no action is taken. IICx_CNTL[PT] needs not be set. If IICx_MDCNTL[EINT] = 1, an interrupt is generated.
1	AMD	Addressing Mode 0 Use 7-bit addressing. 1 Use 10-bit addressing.	Does not affect slave transfers.
2:3	TCT	Transfer Count 00 Transfer one byte. 01 Transfer two bytes. 10 Transfer three bytes. 11 Transfer four bytes.	
4	RPST	Repeated Start 0 Normal start operation 1 Use repeated Start function to start transfer.	
5	CHT	Chain Transfer 0 Transfer is only or last transfer. 1 Transfer is one of a sequence of transfers (but not last in sequence).	Completion of a requested transfer causes a Stop condition to be generated on the IIC bus.
6	RW	Read/Write 0 Transfer is a write. 1 Transfer is a read.	
7	PT	Pending Transfer 0 Most recent requested transfer is complete. 1 Start transfer if bus is free.	

MMIO 0x1 40000410 IIC0, 0x1 40000510 IIC1 R/W

See *IICx Direct Control Register (IICx_DIRECTCNTL)* on page 828.

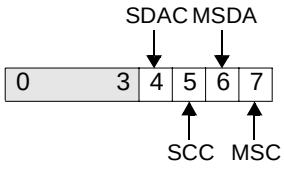


Figure 0-3. IICx Direct Control Register (IICx_DIRECTCNTL)

0:3		Reserved	
4	SDAC	IICSDA Output Control Directly controls the IICSDA output. 0 IICSDA is a logic 0 1 IICSDA is a logic 1	
5	SCC	IICSCL Output Control Directly controls the IICSCL output 0 IICSCL is a logic 0 1 IICSCL is a logic 1	
6	MSDA	Monitor IICSDA Used to monitor the IICSDA input 0 IICSDA is a logic 0 1 IICSDA is a logic 1	Read-only
7	MSC	Monitor IICSCL. Used to monitor the IICSCL input. 0 IICSCL is a logic 0 1 IICSCL is a logic 1	Read-only

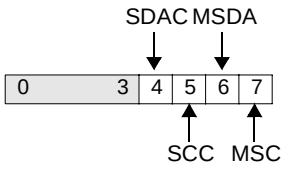


Figure 29-134. IICx Direct Control Register (IICx_DIRECTCNTL)

0:3		Reserved
4	SDAC	IICSDA Output Control Directly controls the IICSDA output. 0 IICSDA is a logic 0 1 IICSDA is a logic 1

5	SCC	IICSCL Output Control Directly controls the IICSCL output 0 IICSCL is a logic 0 1 IICSCL is a logic 1	
6	MSDA	Monitor IICSDA Used to monitor the IICSDA input 0 IICSDA is a logic 0 1 IICSDA is a logic 1	Read-only
7	MSC	Monitor IICSCL. Used to monitor the IICSCL input. 0 IICSCL is a logic 0 1 IICSCL is a logic 1	Read-only

MMIO 0x1 40000409 IIC0, 0x1 40000509 IIC1 R/W

See *IICx Extended Status Register (IICx_EXTSTS)* on page 819.

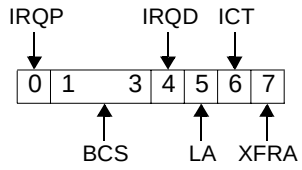


Figure 0-4. IICx Extended Status Register (IICx_EXTSTS)

0	IRQP	IRQ Pending 0 No IRQ is pending. 1 An IRQ is active, another IRQ is on-deck, and another interrupt-generating condition has occurred.	• IICx_EXTSTS[IRQP] might be set momentarily while an IRQ moves from the Pending to the On-deck state. • An interrupt remains pending, IICx_EXTSTS[IRQP]=1, until the current on-deck interrupt becomes active, IICx_STS[IRQD]=0 and IICx_STS[IRQA]=1. • Writing 1 to IICx_EXTSTS[IRQP] clears the field. • When the IIC interrupt is disabled, IICx_MDCNTL[IRQP] = 0, IICx_EXTSTS[IRQP] should be ignored.
1:3	BCS	Bus Control State 000 Unused; if this value is read an error occurred. 001 Slave-selected state; the IIC interface has detected and decoded a slave transfer request on the IIC bus. 010 Slave Transfer state; the IIC interface has detected but has not decoded a slave transfer request on the IIC bus. 011 Master Transfer state; entered after a master transfer request has started on the IIC bus. 100 Free Bus state; the bus is free and no transfer request is pending. 101 Busy Bus state; the bus is busy. 110 Unknown state; value after IIC reset. 111 Unused; if this value is read an error occurred.	Read-only.

4	IRQD	IRQ On-Deck 0 No IRQ is on-deck. 1 An interrupt is active, and another interrupt-generating condition has occurred.	• IICx_EXTSTS[IRQD] might be set momentarily while an IRQ moves from the On-deck to the Active state. • An interrupt remains on-deck, IICx_EXTSTS[IRQD = 1, until the current active interrupt is no longer active, IICx_STS[IRQA] = 0. • If IICx_EXTSTS[IRQP] = 1, IICx_EXTSTS[IRQD] is set on the next OPB clock. • Writing 1 to IICx_EXTSTS[IRQD] clears the field. • When the IIC interrupt is disabled, IICx_MDCNTL[IRQP]=0, IICx_EXTSTS[IRQD] should be ignored.
5	LA	Lost Arbitration 0 Normal operation. 1 Loss of arbitration has ended the requested master transfer.	• If arbitration is lost, any requested master transaction may have terminated prematurely. Read data may be incomplete and not all write data may have been written. • If arbitration is lost during a repeat start, the master may not own the IIC bus.
6	ICT	Incomplete Transfer 0 Normal operation. 1 Some of the bytes of the requested master transfer were not transferred.	For an incomplete transfer, read the transfer count, IICx_XFRCNT, to determine how bytes were transferred.
7	XFRA	Transfer Aborted 0 No transfer is pending, or transfer is in progress. 1 A requested master transfer was aborted by a NACK during the transfer of the address byte, or was aborted because arbitration was lost. Lost arbitration can be caused by the loss of data during the transfer of the second or subsequent data byte.	Transfer aborted. When set to a 1, a requested master transfer was aborted by a NOT acknowledge during the transfer of the address byte. It is also set to a 1 when a requested master transfer loses data. Lost arbitration can be caused by the loss of data during the transfer of the second or subsequent data byte.

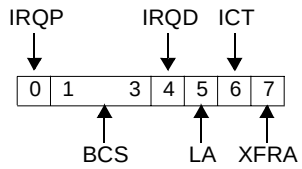


Figure 29-135. IICx Extended Status Register (IICx_EXTSTS)

0	IRQP	IRQ Pending 0 No IRQ is pending. 1 An IRQ is active, another IRQ is on-deck, and another interrupt-generating condition has occurred.	• IICx_EXTSTS[IRQP] might be set momentarily while an IRQ moves from the Pending to the On-deck state. • An interrupt remains pending, IICx_EXTSTS[IRQP]=1, until the current on-deck interrupt becomes active, IICx_STS[IRQD]=0 and IICx_STS[IRQA]=1. • Writing 1 to IICx_EXTSTS[IRQP] clears the field. • When the IIC interrupt is disabled, IICx_MDCNTL[IRQP] = 0, IICx_EXTSTS[IRQP] should be ignored.
1:3	BCS	Bus Control State 000 Unused; if this value is read an error occurred. 001 Slave-selected state; the IIC interface has detected and decoded a slave transfer request on the IIC bus. 010 Slave Transfer state; the IIC interface has detected but has not decoded a slave transfer request on the IIC bus. 011 Master Transfer state; entered after a master transfer request has started on the IIC bus. 100 Free Bus state; the bus is free and no transfer request is pending. 101 Busy Bus state; the bus is busy. 110 Unknown state; value after IIC reset. 111 Unused; if this value is read an error occurred.	Read-only.

4	IRQD	IRQ On-Deck 0 No IRQ is on-deck. 1 An interrupt is active, and another interrupt-generating condition has occurred.	• IICx_EXTSTS[IRQD] might be set momentarily while an IRQ moves from the On-deck to the Active state. • An interrupt remains on-deck, IICx_EXTSTS[IRQD = 1, until the current active interrupt is no longer active, IICx_STS[IRQA] = 0. • If IICx_EXTSTS[IRQP] = 1, IICx_EXTSTS[IRQD] is set on the next OPB clock. • Writing 1 to IICx_EXTSTS[IRQD] clears the field. • When the IIC interrupt is disabled, IICx_MDCNTL[IRQP]=0, IICx_EXTSTS[IRQD] should be ignored.
5	LA	Lost Arbitration 0 Normal operation. 1 Loss of arbitration has ended the requested master transfer.	• If arbitration is lost, any requested master transaction may have terminated prematurely. Read data may be incomplete and not all write data may have been written. • If arbitration is lost during a repeat start, the master may not own the IIC bus.
6	ICT	Incomplete Transfer 0 Normal operation. 1 Some of the bytes of the requested master transfer were not transferred.	For an incomplete transfer, read the transfer count, IICx_XFRCNT, to determine how bytes were transferred.
7	XFRA	Transfer Aborted 0 No transfer is pending, or transfer is in progress. 1 A requested master transfer was aborted by a NACK during the transfer of the address byte, or was aborted because arbitration was lost. Lost arbitration can be caused by the loss of data during the transfer of the second or subsequent data byte.	Transfer aborted. When set to a 1, a requested master transfer was aborted by a NOT acknowledge during the transfer of the address byte. It is also set to a 1 when a requested master transfer loses data. Lost arbitration can be caused by the loss of data during the transfer of the second or subsequent data byte.

MMIO 0x1 40000405 IIC0, 0x1 40000505 IIC1 R/W

See *IICx High Master Address Register (IICx_HMADR)* on page 814.

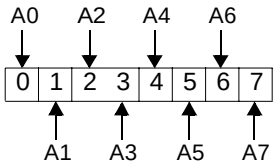


Figure 0-5. IICx High Master Address Register (IICx_HMADR)

0	A0	Address bit 0	1 for 10-bit addresses
1	A1	Address bit 1	1 for 10-bit addresses
2	A2	Address bit 2	1 for 10-bit addresses
3	A3	Address bit 3	1 for 10-bit addresses
4	A4	Address bit 4	0 for 10-bit addresses
5	A5	Address bit 5	MSb for 10-bit addresses
6	A6	Address bit 6	Next to MSb for 10-bit addresses
7	A7	Address bit 7	Don't care for 10-bit addresses

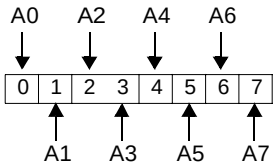


Figure 29-136. IICx High Master Address Register (IICx_HMADR)

0	A0	Address bit 0	1 for 10-bit addresses
1	A1	Address bit 1	1 for 10-bit addresses
2	A2	Address bit 2	1 for 10-bit addresses
3	A3	Address bit 3	1 for 10-bit addresses
4	A4	Address bit 4	0 for 10-bit addresses
5	A5	Address bit 5	MSb for 10-bit addresses
6	A6	Address bit 6	Next to MSb for 10-bit addresses
7	A7	Address bit 7	Don't care for 10-bit addresses

MMIO 0x1 4000040B IIC0, 0x1 4000050B IIC1 R/W

See *IICx High Slave Address Register (IICx_HSADR)* on page 822.

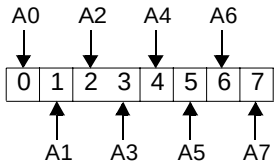


Figure 0-6. IICx High Slave Address Register (IICx_HSADR)

0	A0	Address bit 0	1 for 10-bit addresses
1	A1	Address bit 1	1 for 10-bit addresses
2	A2	Address bit 2	1 for 10-bit addresses
3	A3	Address bit 3	1 for 10-bit addresses
4	A4	Address bit 4	0 for 10-bit addresses
5	A5	Address bit 5	MSb for 10-bit addresses
6	A6	Address bit 6	Next to MSb for 10-bit addresses
7	A7	Address bit 7	Don't care for 10-bit addresses

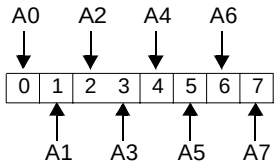


Figure 29-137. IICx High Slave Address Register (IICx_HSADR)

0	A0	Address bit 0	1 for 10-bit addresses
1	A1	Address bit 1	1 for 10-bit addresses
2	A2	Address bit 2	1 for 10-bit addresses
3	A3	Address bit 3	1 for 10-bit addresses
4	A4	Address bit 4	0 for 10-bit addresses
5	A5	Address bit 5	MSb for 10-bit addresses
6	A6	Address bit 6	Next to MSb for 10-bit addresses
7	A7	Address bit 7	Don't care for 10-bit addresses

MMIO 0x1 4000040D IIC0, 0x1 4000050D IIC1 R/W

See *IICx Interrupt Mask Register (IICx_INTRMSK)* on page 824.

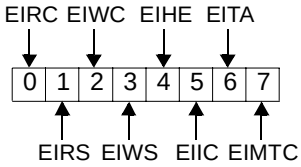


Figure 0-7. IICx Interrupt Mask Register (IICx_INTRMSK)

0	EIRC	Enable IRQ on Slave Read Complete 0 Disable 1 Enable	The interrupt is activated upon receipt of a Stop during a slave read on the IIC bus. IICx_XTCNTLSS[SRC] = 1 indicates a Slave Read Complete.
1	EIRS	Enable IRQ on Slave Read Needs Service 0 Disable 1 Enable	The interrupt is activated upon receipt of a slave read on the IIC bus and the slave buffer was empty or went empty and more data was requested on the IIC bus. **Note:** IICx_XTCNTLSS[SRS] = 1 indicates a Slave Read Needs Service.
2	EIWC	Enable IRQ on Slave Write Complete 0 Disable 1 Enable	The interrupt is activated upon receipt of a Stop during a slave write on the IIC bus. **Note:** IICx_XTCNTLSS[SWC] = 1 indicates a Slave Write Compete.
3	EIWS	Enable IRQ on Slave Write Needs Service 0 Disable 1 Enable	The interrupt is activated when the slave buffer becomes full during a slave write on the IIC bus. **Note:** IICx_XTCNTLSS[SWS] = 1 indicates a Slave Write Needs Service.
4	EIHE	Enable IRQ on Halt Executed 0 Disable 1 Enable	
5	EIIC	Enable IRQ on Incomplete Transfer 0 Disable 1 Enable	
6	EITA	Enable IRQ on Transfer Aborted 0 Disable 1 Enable	
7	EIMTC	Enable IRQ on Requested Master Transfer Complete 0 Disable 1 Enable	

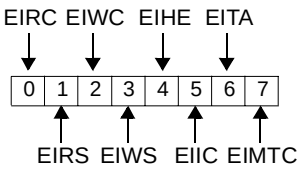


Figure 29-138. IICx Interrupt Mask Register (IICx_INTRMSK)

0	EIRC	Enable IRQ on Slave Read Complete 0 Disable 1 Enable	The interrupt is activated upon receipt of a Stop during a slave read on the IIC bus. IICx_XTCNTLSS[SRC] = 1 indicates a Slave Read Complete.
1	EIRS	Enable IRQ on Slave Read Needs Service 0 Disable 1 Enable	The interrupt is activated upon receipt of a slave read on the IIC bus and the slave buffer was empty or went empty and more data was requested on the IIC bus. **Note:** IICx_XTCNTLSS[SRS] = 1 indicates a Slave Read Needs Service.
2	EIWC	Enable IRQ on Slave Write Complete 0 Disable 1 Enable	The interrupt is activated upon receipt of a Stop during a slave write on the IIC bus. **Note:** IICx_XTCNTLSS[SWC] = 1 indicates a Slave Write Compete.
3	EIWS	Enable IRQ on Slave Write Needs Service 0 Disable 1 Enable	The interrupt is activated when the slave buffer becomes full during a slave write on the IIC bus. **Note:** IICx_XTCNTLSS[SWS] = 1 indicates a Slave Write Needs Service.
4	EIHE	Enable IRQ on Halt Executed 0 Disable 1 Enable	
5	EIIC	Enable IRQ on Incomplete Transfer 0 Disable 1 Enable	
6	EITA	Enable IRQ on Transfer Aborted 0 Disable 1 Enable	
7	EIMTC	Enable IRQ on Requested Master Transfer Complete 0 Disable 1 Enable	

MMIO 0x1 40000404 IIC0, 0x1 40000504 IIC1 R/W

See *IICx Low Master Address Register (IICx_LMADR)* on page 813.

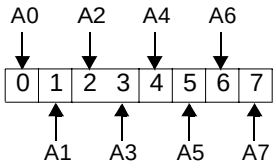


Figure 0-8. IICx Low Master Address Register (IICx_LMADR)

0	A0	Address bit 0	
1	A1	Address bit 1	
2	A2	Address bit 2	
3	A3	Address bit 3	
4	A4	Address bit 4	
5	A5	Address bit 5	
6	A6	Address bit 6	LSb for 7-bit addresses
7	A7	Address bit 7	LSb for 10-bit addresses; don't care for 7-bit addresses

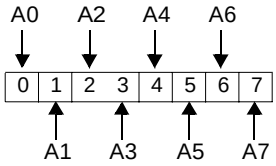


Figure 29-139. IICx Low Master Address Register (IICx_LMADR)

0	A0	Address bit 0	
1	A1	Address bit 1	
2	A2	Address bit 2	
3	A3	Address bit 3	
4	A4	Address bit 4	
5	A5	Address bit 5	
6	A6	Address bit 6	LSb for 7-bit addresses
7	A7	Address bit 7	LSb for 10-bit addresses; don't care for 7-bit addresses

MMIO 0x1 4000040A IIC0, 0x1 4000050A IIC1 R/W

See *IICx Low Slave Address Register (IICx_LSADR)* on page 822.

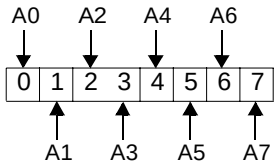


Figure 0-9. IICx Low Slave Address Register (IICx_LSADR)

0	A0	Address bit 0	
1	A1	Address bit 1	
2	A2	Address bit 2	
3	A3	Address bit 3	
4	A4	Address bit 4	
5	A5	Address bit 5	
6	A6	Address bit 6	LSb for 7-bit addresses
7	A7	Address bit 7	LSb for 10-bit addresses; don't care for 7-bit addresses

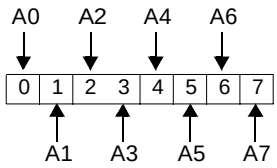


Figure 29-140. IICx Low Slave Address Register (IICx_LSADR)

0	A0	Address bit 0	
1	A1	Address bit 1	
2	A2	Address bit 2	
3	A3	Address bit 3	
4	A4	Address bit 4	
5	A5	Address bit 5	
6	A6	Address bit 6	LSb for 7-bit addresses
7	A7	Address bit 7	LSb for 10-bit addresses; don't care for 7-bit addresses

MMIO 0x1 40000400 IIC0, 0x1 40000500

See *IICx Master Data Buffer (IICx_MDBUF)* on page 811.

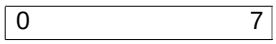


Figure 0-10. IICx Master Data Buffer (IICx_MDBUF)

0		Data bit
1		Data bit
2		Data bit
3		Data bit
4		Data bit
5		Data bit
6		Data bit
7		Data bit

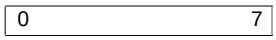


Figure 29-141. IICx Master Data Buffer (IICx_MDBUF)

0		Data bit
1		Data bit
2		Data bit
3		Data bit
4		Data bit
5		Data bit
6		Data bit
7		Data bit

MMIO 0x1 40000407 IIC0, 0x1 40000507 IIC1 R/W

See *IICx Mode Control Register (IICx_MDCNTL)* on page 816.

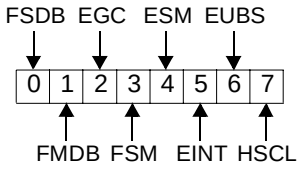


Figure 0-11. IICx Mode Control Register (IICx_MDCNTL)

0	FSDB	Flush Slave Data Buffer 0 Normal operation 1 Set slave data buffer to empty.	Cleared after buffer is emptied.
1	FMDB	Flush Master Data Buffer 0 Normal operation 1 Set master data buffer to empty.	Cleared after buffer is emptied.
2	EGC	Enable General Call 0 Ignore general call on IIC bus. 1 Respond to general call on IIC bus.	IICx_MDCNTL[ESM] overrides this field; if IICx_MDCNTL[ESM] = 1, a general call is ignored.
3	FSM	Fast/Standard Mode 0 IIC transfers run at 100 kHz (standard mode). 1 IIC transfers run at 400 kHz (fast mode).	
4	ESM	Enable Slave Mode 0 Slave transfers are ignored. 1 Slave transfers are enabled.	Program IICx_LSADR and IICx_HSADR before setting this field.
5	EINT	Enable Interrupt 0 Interrupts are disabled. 1 Enables interrupts for interrupts enabled in IICx_NTRMSK.	
6	EUBS	Exit Unknown IIC Bus State 0 Normal operation. 1 IIC bus control state machine exits unknown bus state, if in an unknown state.	If the IIC bus control state machine is in a known state, setting IICx_MDCNTL[EUBS] = 1 has no effect.
7	HSCL	Hold IIC Serial Clock Low 0 If slave is not ready, issue a NACK in response to slave transfer request. 1 If slave is not ready, hold the IICSCL signal low until slave is ready.	This field is used only when in slave mode.

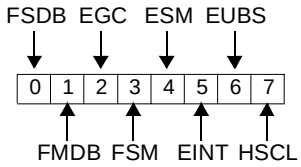


Figure 29-142. IICx Mode Control Register (IICx_MDCNTL)

0	FSDB	Flush Slave Data Buffer 0 Normal operation 1 Set slave data buffer to empty.	Cleared after buffer is emptied.
1	FMDB	Flush Master Data Buffer 0 Normal operation 1 Set master data buffer to empty.	Cleared after buffer is emptied.
2	EGC	Enable General Call 0 Ignore general call on IIC bus. 1 Respond to general call on IIC bus.	IICx_MDCNTL[ESM] overrides this field; if IICx_MDCNTL[ESM] = 1, a general call is ignored.
3	FSM	Fast/Standard Mode 0 IIC transfers run at 100 kHz (standard mode). 1 IIC transfers run at 400 kHz (fast mode).	
4	ESM	Enable Slave Mode 0 Slave transfers are ignored. 1 Slave transfers are enabled.	Program IICx_LSADR and IICx_HSADR before setting this field.
5	EINT	Enable Interrupt 0 Interrupts are disabled. 1 Enables interrupts for interrupts enabled in IICx_NTRMSK.	
6	EUBS	Exit Unknown IIC Bus State 0 Normal operation. 1 IIC bus control state machine exits unknown bus state, if in an unknown state.	If the IIC bus control state machine is in a known state, setting IICx_MDCNTL[EUBS] = 1 has no effect.
7	HSCL	Hold IIC Serial Clock Low 0 If slave is not ready, issue a NACK in response to slave transfer request. 1 If slave is not ready, hold the IICSCL signal low until slave is ready.	This field is used only when in slave mode.

MMIO 0x1 40000402 IIC0, 0x1 40000502 IIC1 R/W

See *IICx Slave Data Buffer (IICx_SDBUF)* on page 812.

Figure 0-12. IICx Slave Data Buffer (IICx_SDBUF)

0		Data bit
1		Data bit
2		Data bit
3		Data bit
4		Data bit
5		Data bit
6		Data bit
7		Data bit

Figure 29-143. IICx Slave Data Buffer (IICx_SDBUF)

0		Data bit
1		Data bit
2		Data bit
3		Data bit
4		Data bit
5		Data bit
6		Data bit
7		Data bit

MMIO 0x1 40000408 IIC0, 0x1 40000508 IIC1 R/W

See *IICx Status Register (IICx_STS)* on page 817.

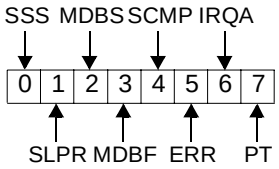


Figure 0-13. IICx Status Register (IICx_STS)

0	SSS	Slave Status Set 0 No slave operations are in progress. 1 Slave operation is in progress.	Read-only; this field is set when any of the following fields are set: IICx_XTCNTLSS[SRC, SRS, SWC, SWS].
1	SLPR	Sleep Request 0 Normal operation. 1 Sleep mode (CPC0_ER[IIC] = 1).	Read-only. The IIC interface is awakened when a start signal is detected on the IIC bus or when the CPC0_ER[IICx] is cleared.
2	MDBS	Master Data Buffer Status 0 Master data buffer is empty. 1 Master data buffer contains data.	Read-only.
3	MDBF	Master Data Buffer Full 0 Master data buffer is not full. 1 Master data buffer is full.	Read-only.
4	SCMP	Stop Complete 0 No request to halt transfer, or master data transfer, is complete. 1 Request to halt transfer, or master data transfer, is complete.	To clear IICx_STS[SCMP], set IICx_STS[SCMP] = 1.
5	ERR	Error 0 No error has occurred. 1 One of the following fields is set: IICx_EXTSTS[LA, ICT, XFRA] = 1.	Read-only.
6	IRQA	IRQ Active 0 No IIC interrupt has been sent to the universal interrupt controller (UIC). 1 An IIC interrupt has been sent to the UIC.	To clear IICx_STS[IRQA], set IICx_STS[IRQA] = 1. If IICx_MDCNTL[EINT] = 0, then IICx_STS[IRQA] is not set.
7	PT	Pending Transfer 0 No transfer is pending, or transfer is in progress. 1 Transfer is pending.	Read-only.

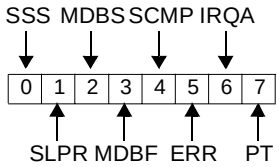


Figure 29-144. IICx Status Register (IICx_STS)

0	SSS	Slave Status Set 0 No slave operations are in progress. 1 Slave operation is in progress.	Read-only; this field is set when any of the following fields are set: IICx_XTCNTLSS[SRC, SRS, SWC, SWS].
1	SLPR	Sleep Request 0 Normal operation. 1 Sleep mode (CPC0_ER[IIC] = 1).	Read-only. The IIC interface is awakened when a start signal is detected on the IIC bus or when the CPC0_ER[IICx] is cleared.
2	MDBS	Master Data Buffer Status 0 Master data buffer is empty. 1 Master data buffer contains data.	Read-only.
3	MDBF	Master Data Buffer Full 0 Master data buffer is not full. 1 Master data buffer is full.	Read-only.
4	SCMP	Stop Complete 0 No request to halt transfer, or master data transfer, is complete. 1 Request to halt transfer, or master data transfer, is complete.	To clear IICx_STS[SCMP], set IICx_STS[SCMP] = 1.
5	ERR	Error 0 No error has occurred. 1 One of the following fields is set: IICx_EXTSTS[LA, ICT, XFRA] = 1.	Read-only.
6	IRQA	IRQ Active 0 No IIC interrupt has been sent to the universal interrupt controller (UIC). 1 An IIC interrupt has been sent to the UIC.	To clear IICx_STS[IRQA], set IICx_STS[IRQA] = 1. If IICx_MDCNTL[EINT] = 0, then IICx_STS[IRQA] is not set.
7	PT	Pending Transfer 0 No transfer is pending, or transfer is in progress. 1 Transfer is pending.	Read-only.

MMIO 0x1 4000040E IIC0 0x1 4000050E IIC1 R/W

See *IICx Transfer Count Register (IICx_XFRCNT)* on page 825.

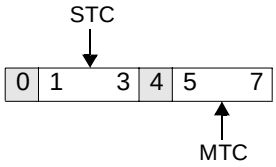


Figure 0-14. IICx Transfer Count Register (IICx_XFRCNT)

0		Reserved
1:3	STC	Slave Transfer Count 000 0 bytes transferred 001 1 byte transferred 010 2 bytes transferred 011 3 bytes transferred 100 4 bytes transferred 101 Reserved 110 Reserved 111 Reserved
4		Reserved
5:7	MTC	Master Transfer Count 000 0 bytes transferred 001 1 byte transferred 010 2 bytes transferred 011 3 bytes transferred 100 4 bytes transferred 101 Reserved 110 Reserved 111 Reserved

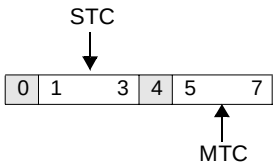


Figure 29-145. IICx Transfer Count Register (IICx_XFRCNT)

0		Reserved

1:3	STC	Slave Transfer Count 000 0 bytes transferred 001 1 byte transferred 010 2 bytes transferred 011 3 bytes transferred 100 4 bytes transferred 101 Reserved 110 Reserved 111 Reserved
4		Reserved
5:7	MTC	Master Transfer Count 000 0 bytes transferred 001 1 byte transferred 010 2 bytes transferred 011 3 bytes transferred 100 4 bytes transferred 101 Reserved 110 Reserved 111 Reserved

MMIO 0x1 4000040F IIC0, 0x1 4000050F IIC1 R/W

See *IICx Extended Control and Slave Status Register (IICx_XTCNTLSS)* on page 826.

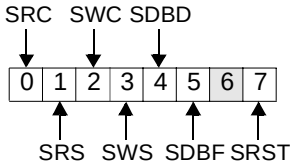


Figure 0-15. IICx Extended Control and Slave Status Register (IICx_XTCNTLSS)

0	SRC	Slave Read Complete 0 Normal operation, or IICx_MDCNTL[HSCL] = 0, IICx_SDBUF is empty, and a read operation is in progress. 1 A NACK or Stop condition was received over the IIC bus, or a repeated Start condition ended a read operation.	Check whether the read operation emptied IICx_SDBUF.
1	SRS	Slave Read Needs Service 0 Normal operation or slave read does not need service. 1 IICx_SDBUF is empty, and a read operation was requested on the IIC bus. The set condition may also indicate that IICx_SDBUF is empty due to a slave read and additional data is requested by the master.	1. If IICx_MDCNTL[HSCL]=0 and IICx_SDBUF contains no data, the slave will issue a NACK and IICx_XTCNTLSS[SRS] is set. 2. If ICx_MDCNTL[HSCL]=0, and IICx_SDBUF contains data, the slave will send the data. IICx_XTCNTLSS[SRS] is not set unless the master request additional data. 3. If IICx_MDCNTL[HSCL]=0, and IICx_SDBUF contains no data, the slave will hold IICSCL low to indicate the slave is busy. IICx_XTCNTLSS[SRS] is set until the IICx_SDBUF is filled. Once filled, IICSCL is released, IICx_XTCNTLSS[SRS] is cleared, and the slave sends the data. 4. If IICx_MDCNTL[HSCL]=1, and IICx_SDBUF contains data, the slave will send the data. IICx_XTCNTLSS[SRS] is not set unless the master requests additional data.
2	SWC	Slave Write Complete 0 Normal operation or slave write in progress. 1 A Stop signal was received during a write operation, or a repeated Start condition ended a write operation.	

3	SWS	Slave Write Needs Service 0 Normal operation or slave write does not need service. 1 IICx_SDBUF is full during a slave write.	1. If IICx_MDCNTL[HSCL] = 1 and IICx_SDBUF is full, the slave will hold IICSCL low to indicate the slave is busy. IICx_XTCNTLSS[SWS] is set until IICx_SDBUF is empty. Once empty, IICSCL is released, IICx_XTCNTLSS[SWS] is cleared, and the slave receives the data. 2. If IICx_MDCNTL[HSCL] = 0 and IICx_SDBUF is full, the slave will issue a NACK and IICx_XTCNTLSS[SWS] is set.
4	SDBD	Slave Data Buffer Has Data 0 IICx_SDBUF is empty 1 IICx_SDBUF contains data	Read-only
5	SDBF	Slave Data Buffer Full 0 IICx_SDBUF is not full 1 IICx_SDBUF is full	Read-only
6		Reserved	
7	SRST	Soft Reset 0 Normal operation 1 Soft reset	

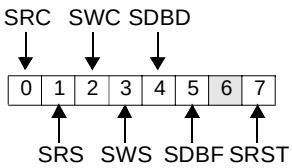


Figure 29-146. IICx Extended Control and Slave Status Register (IICx_XTCNTLSS)

0	SRC	Slave Read Complete 0 Normal operation, or IICx_MDCNTL[HSCL] = 0, IICx_SDBUF is empty, and a read operation is in progress. 1 A NACK or Stop condition was received over the IIC bus, or a repeated Start condition ended a read operation.	Check whether the read operation emptied IICx_SDBUF.

1	SRS	Slave Read Needs Service 0 Normal operation or slave read does not need service. 1 IICx_SDBUF is empty, and a read operation was requested on the IIC bus. The set condition may also indicate that IICx_SDBUF is empty due to a slave read and additional data is requested by the master.	1. If IICx_MDCNTL[HSCL]=0 and IICx_SDBUF contains no data, the slave will issue a NACK and IICx_XTCNTLSS[SRS] is set. 2. If ICx_MDCNTL[HSCL]=0, and IICx_SDBUF contains data, the slave will send the data. IICx_XTCNTLSS[SRS] is not set unless the master request additional data. 3. If IICx_MDCNTL[HSCL]=0, and IICx_SDBUF contains no data, the slave will hold IICSCL low to indicate the slave is busy. IICx_XTCNTLSS[SRS] is set until the IICx_SDBUF is filled. Once filled, IIC-SCL is released, IICx_XTCNTLSS[SRS] is cleared, and the slave sends the data. 4. If IICx_MDCNTL[HSCL]=1, and IICx_SDBUF contains data, the slave will send the data. IICx_XTCNTLSS[SRS] is not set unless the master requests additional data.
2	SWC	Slave Write Complete 0 Normal operation or slave write in progress. 1 A Stop signal was received during a write operation, or a repeated Start condition ended a write operation.	
3	SWS	Slave Write Needs Service 0 Normal operation or slave write does not need service. 1 IICx_SDBUF is full during a slave write.	1. If IICx_MDCNTL[HSCL] = 1 and IICx_SDBUF is full, the slave will hold IICSCL low to indicate the slave is busy. IICx_XTCNTLSS[SWS] is set until IICx_SDBUF is empty. Once empty, IICSCL is released, IICx_XTCNTLSS[SWS] is cleared, and the slave receives the data. 2. If IICx_MDCNTL[HSCL] = 0 and IICx_SDBUF is full, the slave will issue a NACK and IICx_XTCNTLSS[SWS] is set.
4	SDBD	Slave Data Buffer Has Data 0 IICx_SDBUF is empty 1 IICx_SDBUF contains data	Read-only
5	SDBF	Slave Data Buffer Full 0 IICx_SDBUF is not full 1 IICx_SDBUF is full	Read-only
6		Reserved	
7	SRST	Soft Reset 0 Normal operation 1 Soft reset	

DCR 0x180 Read/Write

See *MAL Configuration Register (MAL0_CFG)* on page 705.

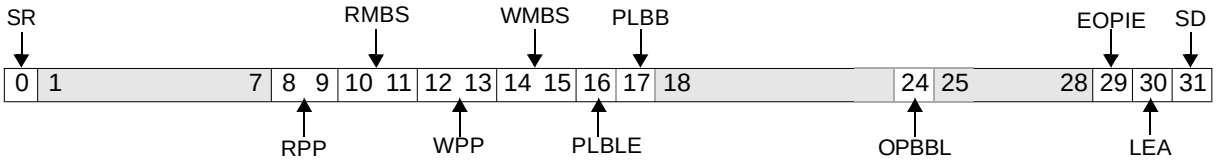


Figure 0-1. MAL Configuration Register (MAL0_CFG)

0	SR	MAL Software Reset 0 MAL reset is complete 1 Reset the MAL	Generates a general reset to MAL through a software command. After setting this bit, MAL hardware (registers, interface and internal state machines) returns to the power-on reset value. The software writes 1 to this bit in order to drive MAL to the reset state. The bit is cleared by the hardware when the reset is completed (one system clock).
1:7		Reserved	
8:9	RPP	Read PLB Priority 00 Lowest 01 10 11 Highest	Determines the priority of MAL requests on the PLB for read transactions.
10:11	RMBS	Read Max Burst Size 00 Max burst size of 4 01 Max burst size of 8 10 Max burst size of 16 11 Max burst size of 32 (recommended)	Maximum PLB burst size for read transactions. Determines the maximum data transfers (RdDAcks) per read burst transaction.
12:13	WPP	Write PLB Priority 00 Lowest 01 10 11 Highest	Determines the priority of MAL requests on the PLB for write transactions.
14:15	WMBS	Write Max Burst Size 00 Max burst size of 4 01 Max burst size of 8 10 Max burst size of 16 11 Max burst size of 32 (recommended)	Maximum PLB burst size for write transactions. Determines the maximum data transfers (WrDAcks) per write burst transaction.
16	PLBLE	PLB Lock Error 0 LOCKERROR signal not applied to the PLB slave 1 LOCKERROR signal applied to the PLB slave	When this bit is set, MAL applies the LOCKERROR signal to the PLB slave when it is the initiator during PLB transactions.

17	PLBB	PLB Burst 0 Burst transactions not allowed 1 Burst transactions allowed	When this bit is reset, MAL is not allowed to perform burst transactions.
18:23		Reserved	
24	OPBBL	OPB Bus Lock 0 OPB not locked 1 OPB locked	When this bit is set, MAL locks the OPB during data transfers to and from the COMMACs.
25:28		Reserved	
29	EOPIE	End of Packet Interrupt Enable 0 Generate interrupt on every end-of-packet only if the buffers I bit is set 1 Generate interrupt is on every end-of-packet	When this bit is set, an interrupt is generated on every end of packet (both transmit and receive). When clear, end of packet/buffer interrupt is generated only if the buffers I bit is set (1). Note: An interrupt is generated for every descriptor on which the I bit is set, regardless of the state of the EOPIE bit.
30	LEA	Locked Error Active 0 Handle errors in a non-locked mode 1 Handle errors in locked mode	Determines MAL's error handling mode. When this bit is set, MAL will handle errors in the locked mode, otherwise it will handle errors in a non-locked mode.
31	SD	MAL Scroll Descriptor 0 Do not scroll to the first descriptor of the next packet 1 Scroll to the first descriptor of the next packet	Determines whether or not MAL should scroll to the first descriptor of the next packet, following an early packet termination initiated by the related COMMAC. When set, Scrolling mode is active.

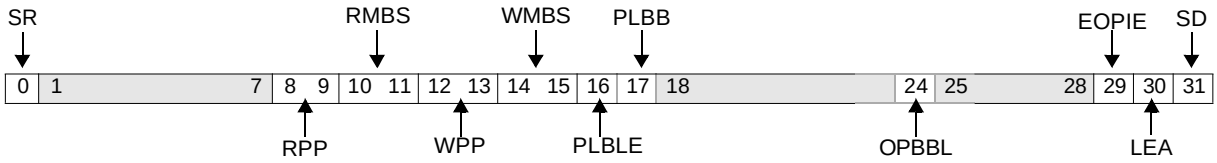


Figure 29-147. MAL Configuration Register (MAL0_CFG)

0	SR	MAL Software Reset 0 MAL reset is complete 1 Reset the MAL	Generates a general reset to MAL through a software command. After setting this bit, MAL hardware (registers, interface and internal state machines) returns to the power-on reset value. The software writes 1 to this bit in order to drive MAL to the reset state. The bit is cleared by the hardware when the reset is completed (one system clock).
1:7		Reserved	

8:9	RPP	Read PLB Priority 00 Lowest 01 10 11 Highest	Determines the priority of MAL requests on the PLB for read transactions.
10:11	RMBS	Read Max Burst Size 00 Max burst size of 4 01 Max burst size of 8 10 Max burst size of 16 11 Max burst size of 32 (recommended)	Maximum PLB burst size for read transactions. Determines the maximum data transfers (RdDAcks) per read burst transaction.
12:13	WPP	Write PLB Priority 00 Lowest 01 10 11 Highest	Determines the priority of MAL requests on the PLB for write transactions.
14:15	WMBS	Write Max Burst Size 00 Max burst size of 4 01 Max burst size of 8 10 Max burst size of 16 11 Max burst size of 32 (recommended)	Maximum PLB burst size for write transactions. Determines the maximum data transfers (WrDAcks) per write burst transaction.
16	PLBLE	PLB Lock Error 0 LOCKERROR signal not applied to the PLB slave 1 LOCKERROR signal applied to the PLB slave	When this bit is set, MAL applies the LOCKERROR signal to the PLB slave when it is the initiator during PLB transactions.
17	PLBB	PLB Burst 0 Burst transactions not allowed 1 Burst transactions allowed	When this bit is reset, MAL is not allowed to perform burst transactions.
18:23		Reserved	
24	OPBBL	OPB Bus Lock 0 OPB not locked 1 OPB locked	When this bit is set, MAL locks the OPB during data transfers to and from the COMMACs.
25:28		Reserved	
29	EOPIE	End of Packet Interrupt Enable 0 Generate interrupt on every end-of-packet only if the buffers I bit is set 1 Generate interrupt is on every end-of-packet	When this bit is set, an interrupt is generated on every end of packet (both transmit and receive). When clear, end of packet/buffer interrupt is generated only if the buffers I bit is set (1). Note: An interrupt is generated for every descriptor on which the I bit is set, regardless of the state of the EOPIE bit.
30	LEA	Locked Error Active 0 Handle errors in a non-locked mode 1 Handle errors in locked mode	Determines MAL's error handling mode. When this bit is set, MAL will handle errors in the locked mode, otherwise it will handle errors in a non-locked mode.
31	SD	MAL Scroll Descriptor 0 Do not scroll to the first descriptor of the next packet 1 Scroll to the first descriptor of the next packet	Determines whether or not MAL should scroll to the first descriptor of the next packet, following an early packet termination initiated by the related COMMAC. When set, Scrolling mode is active.

DCR 0x181 Read/Clear

See *MAL Error Status Register (MAL0_ESR)* on page 709.

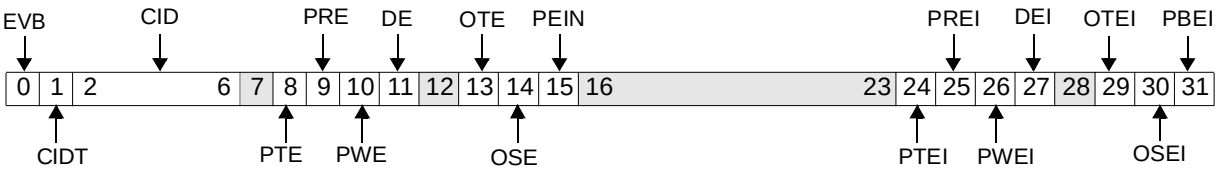


Figure 0-2. MAL Error Status Register (MAL0_ESR)

0	EVB	Error Valid Bit 0 Bit 1:15 are available for latching new error information. 1 Bits 1:15 contain last error. A new error cannot be latched.	When this bit is set, bits 1-6 include the ID of the erroneous channel (in case of OPB errors). Bits 11-15 indicate the type of error. In non-locked mode, the error indication describes the last error that had occurred. In locked mode, the error is the first one that had occurred after this bit was cleared. This bit is set when an error occurs and remains set until reset by the software. In locked mode, new errors cannot be latched in the error lock indication fields if this bit is set
1	CIDT	Channel ID Type 0 channel ID represents RX channel 1 channel ID represents TX channel	
2:6	CID	Channel ID Number	Indicates the number of the channel that caused the error. Note: An error on the PLB cannot be related to a channel. The error condition may be resolved by using the error information optionally locked in the PLB slave.
7		Reserved	
8	PTE	PLB Timeout Error 0 No error 1 PLB timeout error has occurred	Indicates the error is a PLB timeout
9	PRE	PLB Read Error 0 No error 1 PLB read error has occurred	
10	PWE	PLB Write Error 0 No error 1 PLB write error has occurred	
11	DE	Descriptor Error 0 No error 1 Non-valid descriptor	Indicates that the error is a non-valid descriptor, which is *not* the first descriptor in a TX packet.

12		Reserved	
13	OTE	OPB Timeout Error 0 No error 1 OPB timeout error has occurred	Indicates the error is an OPB timeout.
14	OSE	OPB Slave Error 0 No error 1 OPB slave error occurred	Indicates the error is an error indication asserted by an OPB slave.
15	PEIN	PLB Bus Error Indication 0 No error 1 PLB bus error occurred	When this bit is set, the detected error is a PLB error. There is no meaning to the Channel ID field in this case.
16:23		Reserved	
24	PTEI	PLB Timeout Error Interrupt 0 No error 1 PLB timeout error occurred	This bit is set following a PLB timeout error indication. Set condition for this bit generates a maskable interrupt.
25	PREI	PLB Read Error Interrupt 0 No error 1 PLB read error occurred	This bit is set following a PLB read error indication. Set condition for this bit generates a maskable interrupt.
26	PWEI	PLB Write Error Interrupt 0 No error 1 PLB write error occurred	This bit is set following a PLB write error indication. Set condition for this bit generates a maskable interrupt.
27	DEI	Descriptor Error Interrupt 0 No error 1 Descriptor data error recognized	A descriptor data error is recognized during access to the descriptor table. This error indication is asserted when a non-valid descriptor is accessed, which is *not* the first descriptor in a TX packet. Set condition for this bit generates a maskable interrupt.
28		Reserved	
29	OTEI	OPB Timeout Error Interrupt 0 No error 1 OPB time-out	This bit is set following an OPB time out error indication. Set condition for this bit generates a maskable interrupt.
30	OSEI	OPB Slave Error Interrupt 0 No error 1 OPB error from a slave	This bit is set following an OPB error indicated by the slave. Set condition for this bit generates a maskable interrupt.
31	PBEI	PLB Bus Error Interrupt 0 No error 1 PLB error indication	This bit is set following a PLB error indication (from the PLB slave). Set condition for this bit generates a maskable interrupt.

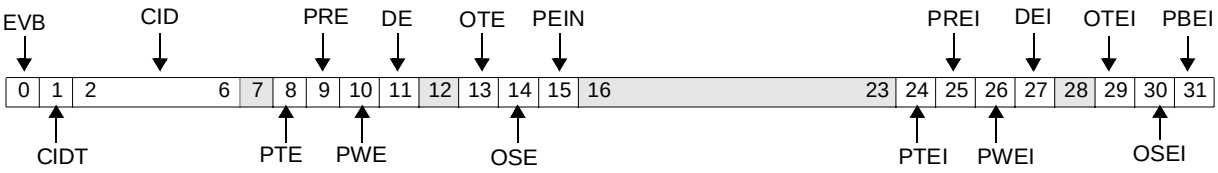


Figure 29-148. MAL Error Status Register (MAL0_ESR)

0	EVB	Error Valid Bit 0 Bit 1:15 are available for latching new error information. 1 Bits 1:15 contain last error. A new error cannot be latched.	When this bit is set, bits 1-6 include the ID of the erroneous channel (in case of OPB errors). Bits 11-15 indicate the type of error. In non-locked mode, the error indication describes the last error that had occurred. In locked mode, the error is the first one that had occurred after this bit was cleared. This bit is set when an error occurs and remains set until reset by the software. In locked mode, new errors cannot be latched in the error lock indication fields if this bit is set
1	CIDT	Channel ID Type 0 channel ID represents RX channel 1 channel ID represents TX channel	
2:6	CID	Channel ID Number	Indicates the number of the channel that caused the error. Note: An error on the PLB cannot be related to a channel. The error condition may be resolved by using the error information optionally locked in the PLB slave.
7		Reserved	
8	PTE	PLB Timeout Error 0 No error 1 PLB timeout error has occurred	Indicates the error is a PLB timeout
9	PRE	PLB Read Error 0 No error 1 PLB read error has occurred	
10	PWE	PLB Write Error 0 No error 1 PLB write error has occurred	
11	DE	Descriptor Error 0 No error 1 Non-valid descriptor	Indicates that the error is a non-valid descriptor, which is *not* the first descriptor in a TX packet.
12		Reserved	
13	OTE	OPB Timeout Error 0 No error 1 OPB timeout error has occurred	Indicates the error is an OPB timeout.
14	OSE	OPB Slave Error 0 No error 1 OPB slave error occurred	Indicates the error is an error indication asserted by an OPB slave.
15	PEIN	PLB Bus Error Indication 0 No error 1 PLB bus error occurred	When this bit is set, the detected error is a PLB error. There is no meaning to the Channel ID field in this case.
16:23		Reserved	

24	PTEI	PLB Timeout Error Interrupt 0 No error 1 PLB timeout error occurred	This bit is set following a PLB timeout error indication. Set condition for this bit generates a maskable interrupt.
25	PREI	PLB Read Error Interrupt 0 No error 1 PLB read error occurred	This bit is set following a PLB read error indication. Set condition for this bit generates a maskable interrupt.
26	PWEI	PLB Write Error Interrupt 0 No error 1 PLB write error occurred	This bit is set following a PLB write error indication. Set condition for this bit generates a maskable interrupt.
27	DEI	Descriptor Error Interrupt 0 No error 1 Descriptor data error recognized	A descriptor data error is recognized during access to the descriptor table. This error indication is asserted when a non-valid descriptor is accessed, which is *not* the first descriptor in a TX packet. Set condition for this bit generates a maskable interrupt.
28		Reserved	
29	OTEI	OPB Timeout Error Interrupt 0 No error 1 OPB time-out	This bit is set following an OPB time out error indication. Set condition for this bit generates a maskable interrupt.
30	OSEI	OPB Slave Error Interrupt 0 No error 1 OPB error from a slave	This bit is set following an OPB error indicated by the slave. Set condition for this bit generates a maskable interrupt.
31	PBEI	PLB Bus Error Interrupt 0 No error 1 PLB error indication	This bit is set following a PLB error indication (from the PLB slave). Set condition for this bit generates a maskable interrupt.

DCR 0x182 Read/Write

See *MAL Interrupt Enable Register (MAL0_IER)* on page 711.

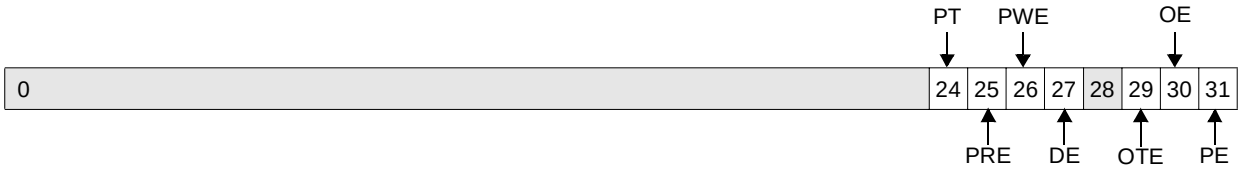


Figure 0-3. MAL Interrupt Enable Register (MAL0_IER)		
0:23		Reserved
24	PT	PLB Timeout Interrupt 0 PLB timeout interrupt disabled 1 PLB timeout interrupt enabled
25	PRE	PLB Read Error Interrupt 0 PLB read error interrupt disabled 1 PLB read error interrupt enabled
26	PWE	PLB Write Error Interrupt 0 PLB write error interrupt disabled 1 PLB write error interrupt enabled
27	DE	Descriptor Error Interrupt 0 Descriptor error interrupt disabled 1 Descriptior error interrupt enabled
28		Reserved
29	OTE	OPB Timeout Error Interrupt 0 OPB timeout error interrupt disabled 1 OPB timeout error interrupt enabled
30	OE	OPB Error Interrupt 0 OPB slave error interrupt disabled 0 OPB slave error interrupt enabled
31	PE	PLB Error Interrupt 0 PLB error interrupt disabled 1 PLB error interrupt enabled

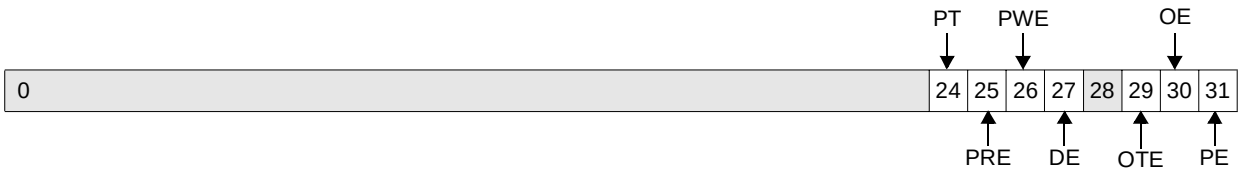


Figure 29-149. MAL Interrupt Enable Register (MAL0_IER)

0:23		Reserved
24	PT	PLB Timeout Interrupt 0 PLB timeout interrupt disabled 1 PLB timeout interrupt enabled
25	PRE	PLB Read Error Interrupt 0 PLB read error interrupt disabled 1 PLB read error interrupt enabled
26	PWE	PLB Write Error Interrupt 0 PLB write error interrupt disabled 1 PLB write error interrupt enabled
27	DE	Descriptor Error Interrupt 0 Descriptor error interrupt disabled 1 Descriptior error interrupt enabled
28		Reserved
29	OTE	OPB Timeout Error Interrupt 0 OPB timeout error interrupt disabled 1 OPB timeout error interrupt enabled
30	OE	OPB Error Interrupt 0 OPB slave error interrupt disabled 0 OPB slave error interrupt enabled
31	PE	PLB Error Interrupt 0 PLB error interrupt disabled 1 PLB error interrupt enabled

DCR 0x1E0 - 0x1E1 Read/Write

See *RX Channel Buffer Size Register (MAL0_RCBSx)* on page 716.

Figure 0-4. RX Channel Buffer Size Register (MAL0_RCBSx

0:23		Reserved
24:31		Receive Channel Buffer Size

Figure 29-150. RX Channel Buffer Size Register (MAL0_RCBSx

0:23		Reserved
24:31		Receive Channel Buffer Size

DCR 0x195 Read/Write

~~See "Descriptor Base Address Registers (MAL0_TXBADDR, MAL0_RXBADDR)" on page 19-35.~~See *Descriptor Base Address Registers (MAL0_TXBADDR, MAL0_RXBADDR)* on page 714.

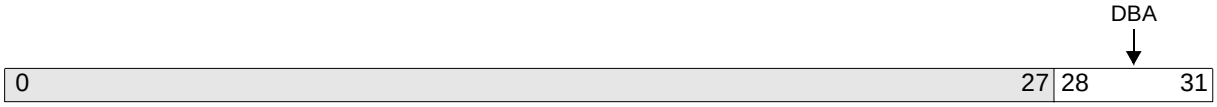


Figure 0-5. RX Descriptor Base Address Register (MAL0_RXBADDR)

0:27		Reserved
28:31	DBA	Descriptor Base Address

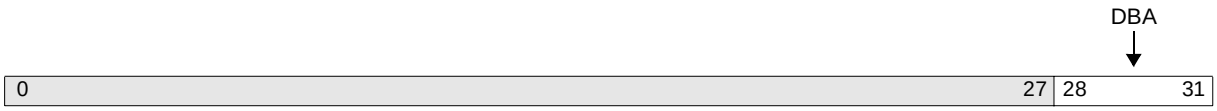


Figure 29-151. RX Descriptor Base Address Register (MAL0_RXBADDR)

0:27		Reserved
28:31	DBA	Descriptor Base Address

DCR 0x191 Read/Write

See *Channel Active Set and Reset Registers* on page 706.

Figure 0-6. RX Channel_Active Reset Register (MAL0_RXCARR)

0:1	RCAR	Receive Channel Active Reset	Each bit represents its related channel (bit 0 for channel 0, and so on). When 0 is written to the bit, channel operation is disabled. MAL0 has two receive channels.
2:31		Reserved	

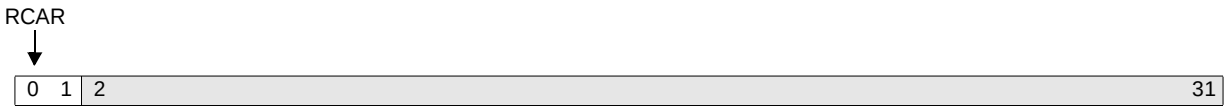


Figure 29-152. RX Channel_Active Reset Register (MAL0_RXCARR)

0:1	RCAR	Receive Channel Active Reset	Each bit represents its related channel (bit 0 for channel 0, and so on). When 0 is written to the bit, channel operation is disabled. MAL0 has two receive channels.
2:31		Reserved	

DCR 0x190 Read/Write

See *Channel Active Set and Reset Registers* on page 706.

Figure 0-7. RX Channel_Active Set Register (MAL0_RXCASR)

0:1	RCAS	Receive Channel Active Set	Each bit represents its related channel (bit 0 for channel 0, and so on). When 1 is written to the bit, channel operation is enabled. MAL0 has two receive channels.
2:31		Reserved	

Figure 29-153. RX Channel_Active Set Register (MAL0_RXCASR)

0:1	RCAS	Receive Channel Active Set	Each bit represents its related channel (bit 0 for channel 0, and so on). When 1 is written to the bit, channel operation is enabled. MAL0 has two receive channels.
2:31		Reserved	

DCR 0x1C0 MAL0_RXCTP0R, 0x1C1 MAL0RXCTP1R Read/Write

~~See "Channel Table Pointer Registers (MAL0_TXCTPxR, MAL0_RXCTPxR)" on page 20-34.~~See *Channel Table Pointer Registers (MAL0_TXCTPxR, MAL0_RXCTPxR)* on page 715.

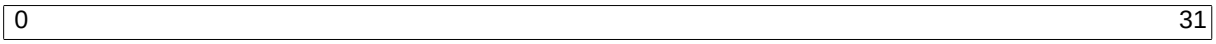


Figure 0-8. RX Channel Table Pointer Register (MAL0_RXCTPxR)

0:31		Channel Table Pointer	Pointer to the base address of the buffer descriptor table used by the channel. The value should point to a location in memory accommodating an aligned doubleword (the three least significant bits of the pointer must be 000). MAL0 has two receive channels.

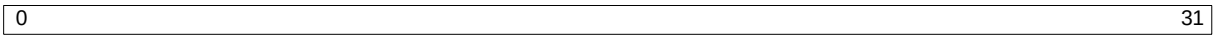


Figure 29-154. RX Channel Table Pointer Register (MAL0_RXCTPxR)

0:31		Channel Table Pointer	Pointer to the base address of the buffer descriptor table used by the channel. The value should point to a location in memory accommodating an aligned doubleword (the three least significant bits of the pointer must be 000). MAL0 has two receive channels.

DCR 0x193 Read/Clear

See *Descriptor Error Interrupt Registers (MAL0_TXDEIR, MAL0_RXDEIR)* on page 712.

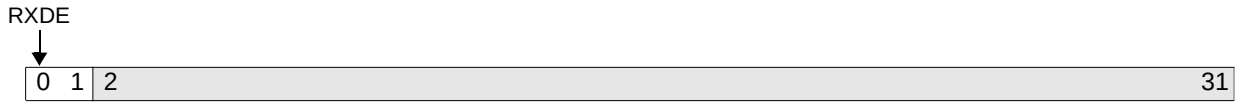


Figure 0-9. RX Descriptor Error Interrupt Register (MAL0_RXDEIR)

0:1	RXDE	Receive Descriptor Error Interrupt	Each bit represents its related channel (bit 0 for channel 0, and so on). When one or more bits are set, MAL RXDE interrupt is set. Writing 1 to a bit clears it. MAL0 has two receive channels.
2:31		Reserved	

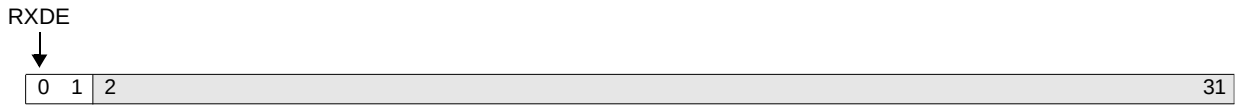


Figure 29-155. RX Descriptor Error Interrupt Register (MAL0_RXDEIR)

0:1	RXDE	Receive Descriptor Error Interrupt	Each bit represents its related channel (bit 0 for channel 0, and so on). When one or more bits are set, MAL RXDE interrupt is set. Writing 1 to a bit clears it. MAL0 has two receive channels.
2:31		Reserved	

DCR 0x192 Read/Clear

See *End of Buffer Interrupt Status Registers* on page 708.

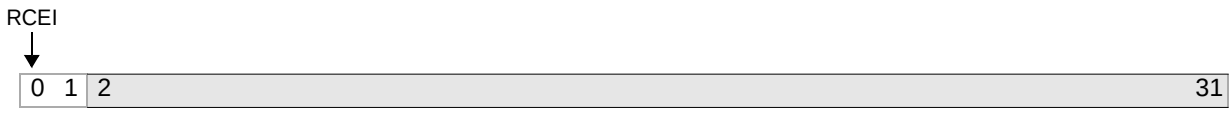


Figure 0-10. RX End of Buffer Interrupt Status Register (MAL0_RXEOBISR)

0:1	RCEI	Receive Channel End-of-Buffer Interrupt	Each bit represents its related channel (bit 0 for channel 0, and so on). Writing 1 to a bit clears it. MAL0 has two receive channels.
2:31		Reserved	

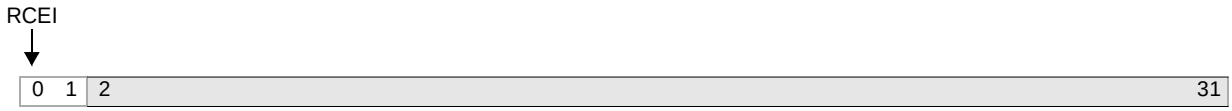


Figure 29-156. RX End of Buffer Interrupt Status Register (MAL0_RXEOBISR)

0:1	RCEI	Receive Channel End-of-Buffer Interrupt	Each bit represents its related channel (bit 0 for channel 0, and so on). Writing 1 to a bit clears it. MAL0 has two receive channels.
2:31		Reserved	

DCR 0x194 Read/Write

See *PLB Storage Attribute Registers (MAL0_TXTATTRR, MAL0_RXTATTRR)* on page 713.

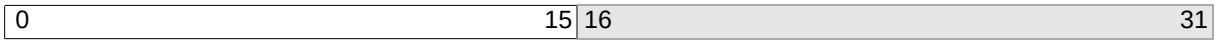


Figure 0-11. RX PLB Attribute Register (MAL0_RXTATTRR)

0:15		RX PLB Storage Attributes
16:31		Reserved

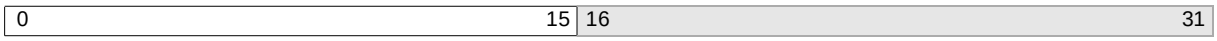


Figure 29-157. RX PLB Attribute Register (MAL0_RXTATTRR)

0:15		RX PLB Storage Attributes
16:31		Reserved

DCR 0x189 Read/Write

~~See "Descriptor Base Address Registers (MAL0_TXBADDR, MAL0_RXBADDR)" on page 19-35.~~See *Descriptor Base Address Registers (MAL0_TXBADDR, MAL0_RXBADDR)* on page 714.

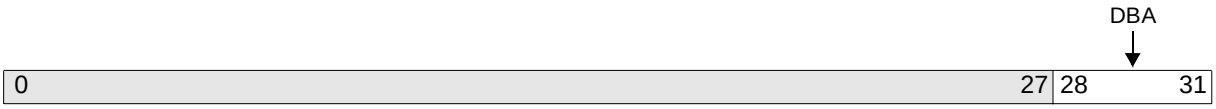


Figure 0-12. TX Descriptor Base Address Register (MAL0_TXBADDR)

0:27		Reserved
28:31	DBA	Descriptor Base Address

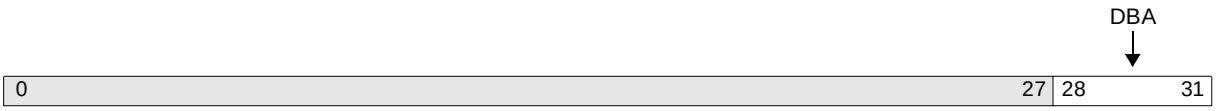


Figure 29-158. TX Descriptor Base Address Register (MAL0_TXBADDR)

0:27		Reserved
28:31	DBA	Descriptor Base Address

DCR 0x185 Read/Write

See *Channel Active Set and Reset Registers* on page 706.

Figure 0-13. TX Channel Active Reset Register (MAL0_TXCARR)

0:3	TCAR	Transmit Channel Active Reset	Each bit represents its related channel (bit 0 for channel 0, and so on). When 1 is written to the bit, channel operation is disabled. MAL0 has four transmit channels.
4:31		Reserved	

Figure 29-159. TX Channel Active Reset Register (MAL0_TXCARR)

0:3	TCAR	Transmit Channel Active Reset	Each bit represents its related channel (bit 0 for channel 0, and so on). When 1 is written to the bit, channel operation is disabled. MAL0 has four transmit channels.
4:31		Reserved	

DCR 0x184 Read/Write

See *Channel Active Set and Reset Registers* on page 706.

Figure 0-14. TX Channel_Active Set Register (MAL0_TXCASR)

0:3	TCAS	Transmit Channel Active Set	Each bit represents its related channel (bit 0 for channel 0, and so on). When 1 is written to the bit, channel operation is enabled. MAL0 has four transmit channels.
4:31		Reserved	

Figure 29-160. TX Channel_Active Set Register (MAL0_TXCASR)

0:3	TCAS	Transmit Channel Active Set	Each bit represents its related channel (bit 0 for channel 0, and so on). When 1 is written to the bit, channel operation is enabled. MAL0 has four transmit channels.
4:31		Reserved	

DCR 0x1A0–0x1A3 Read/Write

~~See "Channel Table Pointer Registers (MAL0_TXCTPxR, MAL0_RXCTPxR)" on page 20-34.~~See *Channel Table Pointer Registers (MAL0_TXCTPxR, MAL0_RXCTPxR)* on page 715.

Figure 0-15. TX Channel Table Pointer Register (MAL0_TXCTPxR)

0:31		Channel Table Pointer	Pointer to the base address of the buffer descriptor table used by the channel. The value entered should point to a location in memory accommodating an aligned doubleword (the three least significant bits of the pointer must be 000). MAL0 has four transmit channels.

Figure 29-161. TX Channel Table Pointer Register (MAL0_TXCTPxR)

0:31		Channel Table Pointer	Pointer to the base address of the buffer descriptor table used by the channel. The value entered should point to a location in memory accommodating an aligned doubleword (the three least significant bits of the pointer must be 000). MAL0 has four transmit channels.

DCR 0x187 Read/Clear

See *Descriptor Error Interrupt Registers (MAL0_TXDEIR, MAL0_RXDEIR)* on page 712.

Figure 0-16. TX Descriptor Error Interrupt Register (MAL0_TXDEIR)

0:3	TXDE	Transmit Descriptor Error Interrupt	Each bit represents its related channel (bit 0 for channel 0, and so on). When one or more bits are set to 1, MAL TXDE interrupt is set. Writing 1 to a bit clears it. MAL0 has four transmit channels.
4:31		Reserved	

Figure 29-162. TX Descriptor Error Interrupt Register (MAL0_TXDEIR)

0:3	TXDE	Transmit Descriptor Error Interrupt	Each bit represents its related channel (bit 0 for channel 0, and so on). When one or more bits are set to 1, MAL TXDE interrupt is set. Writing 1 to a bit clears it. MAL0 has four transmit channels.
4:31		Reserved	

DCR 0x186 Read/Clear

See *End of Buffer Interrupt Status Registers* on page 708.

Figure 0-17. TX End of Buffer Interrupt Status Register (MAL0_TXEOBISR)

0:3	TCEI	Transmit Channel End-of-Buffer Interrupt	Each bit represents its related channel (bit 0 for channel 0, and so on). Writing 1 to a bit clears it. MAL0 has four transmit channels.
4:31		Reserved	

Figure 29-163. TX End of Buffer Interrupt Status Register (MAL0_TXEOBISR)

0:3	TCEI	Transmit Channel End-of-Buffer Interrupt	Each bit represents its related channel (bit 0 for channel 0, and so on). Writing 1 to a bit clears it. MAL0 has four transmit channels.
4:31		Reserved	

DCR 0x188 Read/Write

See *PLB Storage Attribute Registers (MAL0_TXTATTRR, MAL0_RXTATTRR)* on page 713.

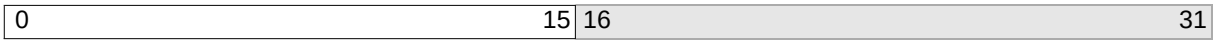


Figure 0-18. TX PLB Attribute Register (MAL0_TXTATTRR)

0:15		TX PLB Storage Attributes
16:31		Reserved

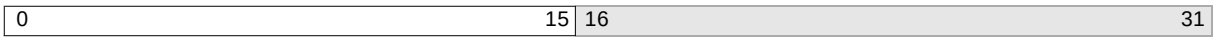


Figure 29-164. TX PLB Attribute Register (MAL0_TXTATTRR)

0:15		TX PLB Storage Attributes
16:31		Reserved

MMIO 0x140000601 Read/Write

See *OPB Arbiter Control Register (OPBA0_CR)* on page 111.

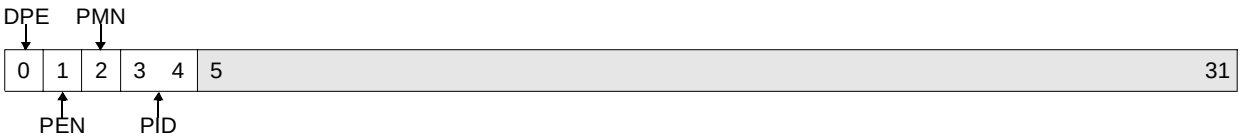


Figure 0-1. OPB Arbiter Control Register (OPBA0_CR)

0	DPE	Dynamic Priority Enable 0 Dynamic priority disabled 1 Dynamic priority enabled	
1	PEN	Park Enable 0 Park disabled 1 Park enabled	
2	PMN	Park on Master Not Last 0 Park on master last 1 Park on master not last	
3:4	PID	Parked Master ID 00 Master ID 0of OPB master device connected to M0_request and OPB_M0Grant 01 Master ID of OPB master device connected to M1_request and OPB_M1GrantReserved 10 Master ID 2of OPB master device connected to M2_request and OPB_M2Grant 11 Master ID of OPB master device connected to M3_request and OPB_M3GrantReserved	Master 0 is the PLB to OPB bridge controller; master 1 is the external bus master interface, and master 2 is the DMA controller. Master 3 is unused.
5:31		Reserved	

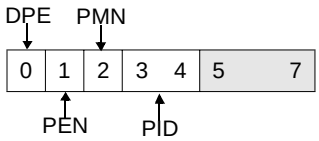


Figure 29-165. OPB Arbiter Control Register (OPBA0_CR)

0	DPE	Dynamic Priority Enable 0 Dynamic priority disabled 1 Dynamic priority enabled	
1	PEN	Park Enable 0 Park disabled 1 Park enabled	
2	PMN	Park on Master Not Last 0 Park on master last 1 Park on master not last	
3:4	PID	Parked Master ID 00 Master ID 0of OPB master device connected to M0_request and OPB_M0Grant 01 Master ID of OPB master device connected to M1_request and OPB_M1GrantReserved 10 Master ID 2of OPB master device connected to M2_request and OPB_M2Grant 11 Master ID of OPB master device connected to M3_request and OPB_M3GrantReserved	Master 0 is the PLB to OPB bridge controller; master 1 is the external bus master interface, and master 2 is the DMA controller. Master 3 is unused.
5:7		Reserved	

MMIO 0x140000600 Read/Write

See *OPB Arbiter Priority Register (OPBA0_PR)* on page 109.

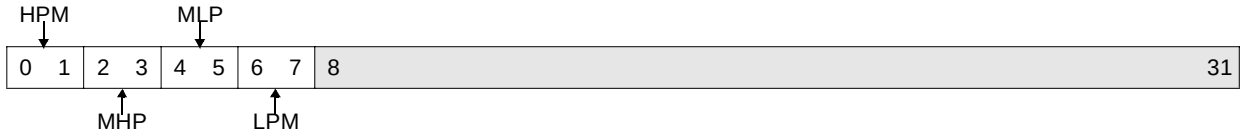


Figure 0-2. OPB Arbiter Priority Register (OPBA0_PR)

0:1	HPM	High Priority Master ID 00 Master ID 0of OPB master device connected to M0_request and OPB_M0Grant 01 Master ID of OPB master device connected to M1_request and OPB_M1GrantReserved 10 Master ID of OPB master device connected to M2_request and OPB_M2GrantReserved 11 Master ID of OPB master device connected to M3_request and OPB_M3GrantReserved	At reset, this priority is assigned to PLB to OPB bridge.
2:3	MHP	Medium High Priority master ID 00 Master ID of OPB master device connected to M0_request and OPB_M0Grant 01 Master ID of OPB master device connected to M1_request and OPB_M1Grant 10 Master ID of OPB master device connected to M2_request and OPB_M2Grant 11 Master ID of OPB master device connected to M3_request and OPB_M3Grant	At reset, this priority is assigned to the external bus master interface.

4:5	MLP	LowMedium Low Priority master ID 00 Master ID of OPB master device connected to M0_request and OPB_M0GrantReserved 01 Master ID of OPB master device connected to M1_request and OPB_M1GrantReserved 10 Master ID 2of OPB master device connected to M2_request and OPB_M2Grant 11 Master ID of OPB master device connected to M3_request and OPB_M3GrantReserved	At reset, this priority is assigned to DMA
6:7	LPM	Low Priority Master ID 00 Master ID of OPB master device connected to M0_request and OPB_M0Grant 01 Master ID of OPB master device connected to M1_request and OPB_M1Grant 10 Master ID of OPB master device connected to M2_request and OPB_M2Grant 11 Master ID of OPB master device connected to M3_request and OPB_M3Grant	
8:31		Reserved	

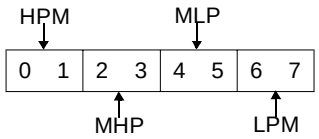


Figure 29-166. OPB Arbiter Priority Register (OPBA0_PR)

0:1	HPM	High Priority Master ID 00 Master ID 0of OPB master device connected to M0_request and OPB_M0Grant 01 Master ID of OPB master device connected to M1_request and OPB_M1GrantReserved 10 Master ID of OPB master device connected to M2_request and OPB_M2GrantReserved 11 Master ID of OPB master device connected to M3_request and OPB_M3GrantReserved	At reset, this priority is assigned to PLB to OPB bridge.
2:3	MHP	Medium High Priority master ID 00 Master ID of OPB master device connected to M0_request and OPB_M0Grant 01 Master ID of OPB master device connected to M1_request and OPB_M1Grant 10 Master ID of OPB master device connected to M2_request and OPB_M2Grant 11 Master ID of OPB master device connected to M3_request and OPB_M3Grant	At reset, this priority is assigned to the external bus master interface.
4:5	MLP	Medium Low Priority master ID 00 Master ID of OPB master device connected to M0_request and OPB_M0GrantReserved 01 Master ID of OPB master device connected to M1_request and OPB_M1GrantReserved 10 Master ID 2of OPB master device connected to M2_request and OPB_M2Grant 11 Master ID of OPB master device connected to M3_request and OPB_M3GrantReserved	At reset, this priority is assigned to DMA
6:7	LPM	Low Priority Master ID 00 Master ID of OPB master device connected to M0_request and OPB_M0Grant 01 Master ID of OPB master device connected to M1_request and OPB_M1Grant 10 Master ID of OPB master device connected to M2_request and OPB_M2Grant 11 Master ID of OPB master device connected to M3_request and OPB_M3Grant	

DCR 0x0A8 Read/Write

See *OPB to PLB Bridge Control Register (OPB0_BCTRL)* on page 112.

Figure 0-1. OPB to PLB Bridge Control Register (OPB0_CTRL)

0:1	PRI	PLB Priority Bits. Determines the priority of PLB requests. Reset to value on PGM_priority inputs	These bits determine the priority of PLB requests. They are directly connected to the PLB_priority(0:1) bits during a PLB request. During reset, these bits are set to the value present on the PGM_priority input (set by the system designer).
2:31		Reserved	

Figure 29-167. OPB to PLB Bridge Control Register (OPB0_BCTRL)

0:1	PRI	PLB Priority Bits. Determines the priority of PLB requests. Reset to value on PGM_priority inputs	These bits determine the priority of PLB requests. They are directly connected to the PLB_priority(0:1) bits during a PLB request. During reset, these bits are set to the value present on the PGM_priority input (set by the system designer).
2:31		Reserved	

DCR 0x0AB Read-Only

See *OPB to PLB Bridge Error Address Register High (OPB0_BEARH)* on page 114.

Figure 0-2. OPB to PLB Bridge Error Address Register High (OPB0_BEARH)

0:27		Reserved
28:31	10'h0B3	Upper address of bus error

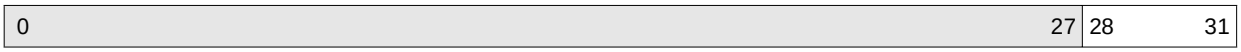


Figure 29-168. OPB to PLB Bridge Error Address Register High (OPB0_BEARH)

0:27		Reserved
28:31	10'h0B3	Upper address of bus error

DCR 0x0AA Read-Only

See *OPB to PLB Bridge Error Address Register Low (OPB0_BEARL)* on page 113.

0	31

Figure 0-3. OPB to PLB Bridge Error Address Register Low (OPB0_BEARL)

0:31	0x0B2	Lower address of bus error

0	31

Figure 29-169. OPB to PLB Bridge Error Address Register Low (OPB0_BEARL)

0:31	0x0B2	Lower address of bus error

DCR 0x0A9 Read-Only

See *OPB to PLB Bridge Status Register (OPB0_BSTAT)* on page 112.

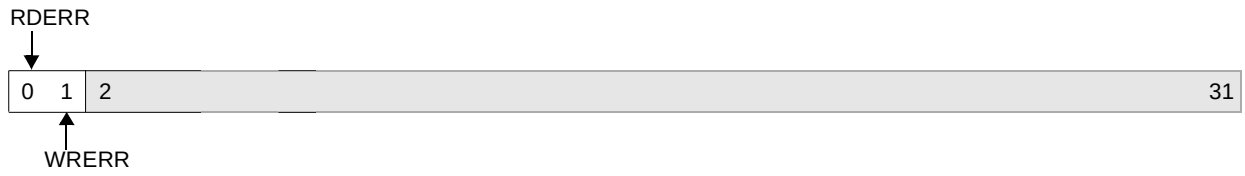


Figure 0-4. OPB to PLB Bridge Status Register (OPB0_STAT)

0	RDERR	Read Error Bit Reset to 0	This error bit is set whenever a PLB slave device reports a read error to the bridge. The error is additionally reported to the OPB master through BGI_errAck. The OPB to PLB bridge status register read error bit should be polled to detect errors, and the SEAR and SEGR of the PLB slave consulted for details of the error condition.
1	WRERR	Write Error Bit Reset to 0	This error bit is set whenever a PLB slave device reports a write error to the bridge. For posted single writes and errors occurring after the deassertion of OPB_seqAddr on burst writes, no error is reported to the OPB master through BGI_errAck. An error is reported to the OPB master through BGI_errAck for a previous write error if OPB_seqAddr is still asserted (but note that the error does not correspond to the transfer during which it is reported).
2:31		Reserved	

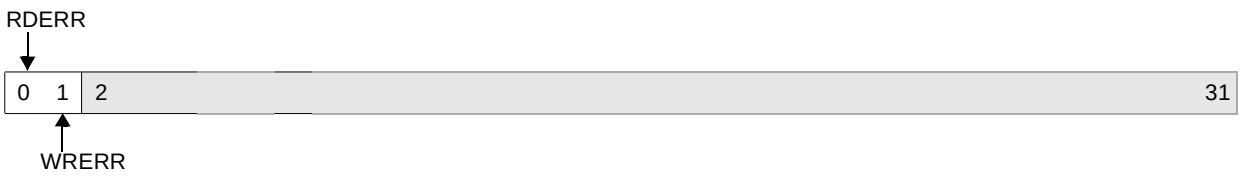


Figure 29-170. OPB to PLB Bridge Status Register (OPB0_BSTAT)

0	RDERR	Read Error Bit Reset to 0	This error bit is set whenever a PLB slave device reports a read error to the bridge. The error is additionally reported to the OPB master through BGI_errAck. The OPB to PLB bridge status register read error bit should be polled to detect errors, and the SEAR and SEGR of the PLB slave consulted for details of the error condition.
1	WRERR	Write Error Bit Reset to 0	This error bit is set whenever a PLB slave device reports a write error to the bridge. For posted single writes and errors occurring after the deassertion of OPB_seqAddr on burst writes, no error is reported to the OPB master through BGI_errAck. An error is reported to the OPB master through BGI_errAck for a previous write error if OPB_seqAddr is still asserted (but note that the error does not correspond to the transfer during which it is reported).
2:31		Reserved	

DCR 0x0AC Read-Only

See *OPB to PLB Bridge Error Address Register Low (OPB0_BEARL)* on page 113.

Figure 0-5. OPB to PLB Bridge Revision ID Register (OPB0_REVID)

0:27		Reserved
28:31	0x0B2	This value indicates what revision level this core is. The revision ID for this core is 1.

Figure 29-171. OPB to PLB Bridge Revision ID Register (OPB0_REVID)

0:27		Reserved
28:31	0x0B2	This value indicates what revision level this core is. The revision ID for this core is 1.

MMIO 0x2 0EC80014 Read/Write (PLB), 0x17-0x14 Read/Write (PCI-X)

~~See "PCI-X BAR0 High Register (PCIX0_BAR0H)" on page 18-50.~~See *PCI-X BAR0 High Register (PCIX0_BAR0H)* on page 582.

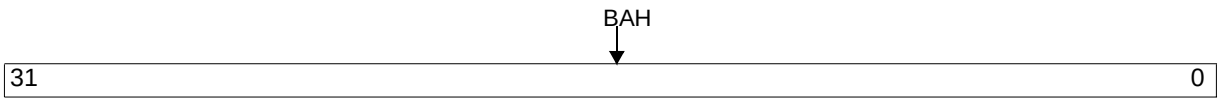


Figure 0-1. PCI-X BAR0 High Register (PCIX0_BAR0H)

31:0	BAH	Base Address High	Defines the upper 32 bits of PCI memory space that is mapped to PLB.

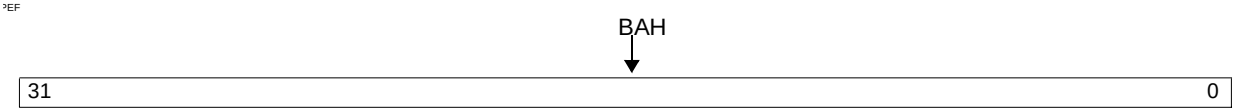


Figure 29-172. PCI-X BAR0 High Register (PCIX0_BAR0H)

31:0	BAH	Base Address High	Defines the upper 32 bits of PCI memory space that is mapped to PLB.

MMIO 0x2 0EC80010 Read/Write (PLB), 0x13-0x10 Read/Write (PCI-X)

~~See "PCI-X BAR0 Low Register (PCIX0_BAR0L)" on page 18-49.~~See *PCI-X BAR0 Low Register (PCIX0_BAR0L)* on page 581.

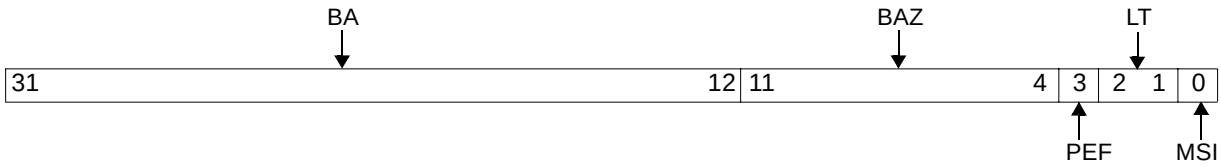


Figure 0-2. PCI-X BAR0 Low Register (PCIX0_BAR0L)

31:12	BA	Base Address	These bits determine the lower 32 bits of the PCI Memory address space where this region is located. Only corresponding bits that are 1 in the size field of the PIM0 Size/Attribute Register are writable. Bits that are 0 in the size field of the PIM0 Size/Attribute Register cause the corresponding Base Address bits to be always 0.
11:4	BAZ	Base Address - always zero	These bits are always 0 since the minimum size of this range is 4 KB.
3	PEF	Prefetchable	This bit determines if the region is prefetchable. Its value is determined by the PIM0 Size/Attribute, Prefetch Enable bit.
2:1	LT	Location Type	These bits are always 10 to indicate that the memory space can be located anywhere in the 64-bit address space.
0	MSI	Memory Space Indicator	This bit is always 0 to indicate memory space (rather than I/O).

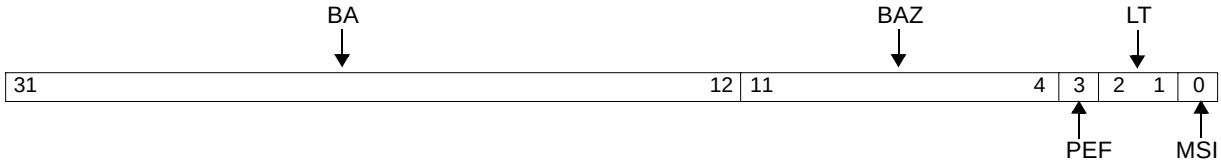


Figure 29-173. PCI-X BAR0 Low Register (PCIX0_BAR0L)

31:12	BA	Base Address	These bits determine the lower 32 bits of the PCI Memory address space where this region is located. Only corresponding bits that are 1 in the size field of the PIM0 Size/Attribute Register are writable. Bits that are 0 in the size field of the PIM0 Size/Attribute Register cause the corresponding Base Address bits to be always 0.
11:4	BAZ	Base Address - always zero	These bits are always 0 since the minimum size of this range is 4 KB.

3	PEF	Prefetchable	This bit determines if the region is prefetchable. Its value is determined by the PIM0 Size/Attribute , Prefetch Enable bit.
2:1	LT	Location Type	These bits are always 10 to indicate that the memory space can be located anywhere in the 64-bit address space.
0	MSI	Memory Space Indicator	This bit is always 0 to indicate memory space (rather than I/O).

MMIO 0x2 0EC80018 Read/Write (PLB), 0x1B-0x18 Read/Write (PCI-X)

~~See "PCI-X BAR1 Register (PCIX0_BAR1)" on page 18-51.~~See *PCI-X BAR1 Register (PCIX0_BAR1)* on page 583.

Figure 0-3. PCI-X BAR1 Register (PCIX0_BAR1)

31:8	BA	Base Address	These bits determine the PCI I/O address space where this region is located.
7:2	BAZ	Base Address always zero	These bits are always 0 since the minimum size of this range is 256 bytes.
1		Reserved	Returns zero when read.
0	MSI	Memory Space Indicator	This bit is always 1 to indicate I/O space (rather than memory).

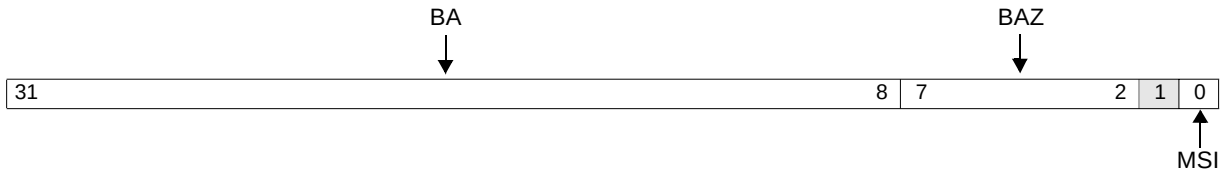


Figure 29-174. PCI-X BAR1 Register (PCIX0_BAR1)

31:8	BA	Base Address	These bits determine the PCI I/O address space where this region is located.
7:2	BAZ	Base Address always zero	These bits are always 0 since the minimum size of this range is 256 bytes.
1		Reserved	Returns zero when read.
0	MSI	Memory Space Indicator	This bit is always 1 to indicate I/O space (rather than memory).

MMIO 0x2 0EC80020 Read/Write (PLB), 0x23-0x20 Read/Write (PCI-X)

~~See "PCI-X BAR2 High Register (PCIX0_BAR2H)" on page 18-53.~~See *PCI-X BAR2 High Register (PCIX0_BAR2H)* on page 585.

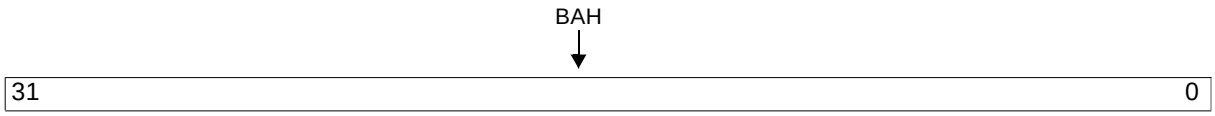


Figure 0-4. PCI-X BAR2 High Register (PCIX0_BAR2H)

31:0	BAH	Base Address High	Defines the upper 32 bits of PCI memory space that is mapped to PLB.

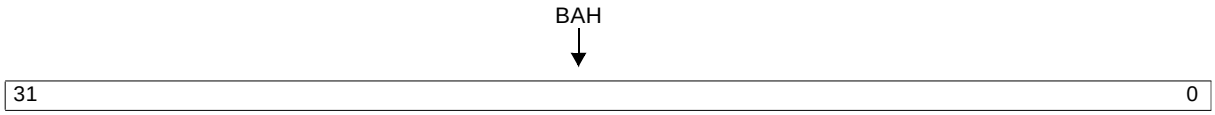


Figure 29-175. PCI-X BAR2 High Register (PCIX0_BAR2H)

31:0	BAH	Base Address High	Defines the upper 32 bits of PCI memory space that is mapped to PLB.

MMIO 0x2 0EC8001CRead/Write (PLB), 0x1F-0x1C Read/Write (PCI-X)

~~See "PCI-X BAR2 Low Register (PCIX0_BAR2L)" on page 18-52.~~See *PCI-X BAR2 Low Register (PCIX0_BAR2L)* on page 584.

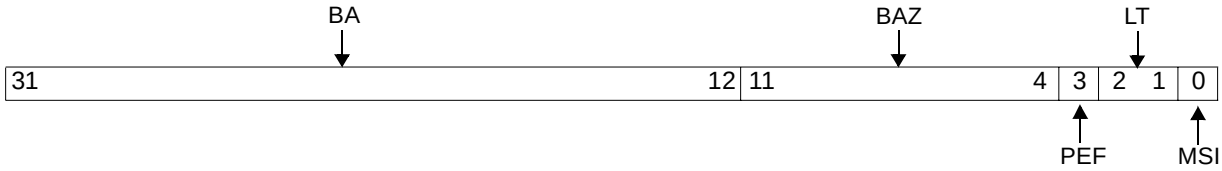


Figure 0-5. PCI-X BAR2 Low Register (PCIX0_BAR2L)

31:12	BA	Base Address	These bits determine the lower 32-bits of the PCI Memory address space where this region is located. Only corresponding bits that are 1 in the size field of the PIM0 Size/Attribute Register are writable. Bits that are 0 in the size field of the PIM2 Size/Attribute Register cause the corresponding Base Address bits to be always 0.
11:4	BAZ	Base Address - always zero	These bits are always 0 since the minimum size of this range is 4 KB.
3	PEF	Prefetchable	This bit determines if the region is prefetchable. Its value is determine by the PIM2 Size/Attribute, Prefetch Enable bit.
2:1	LT	Location Type	These bits are always 10 to indicate that the memory space can be located anywhere in the 64-bit address space.
0	MSI	Memory Space Indicator	This bit is always 0 to indicate memory space (rather than I/O).

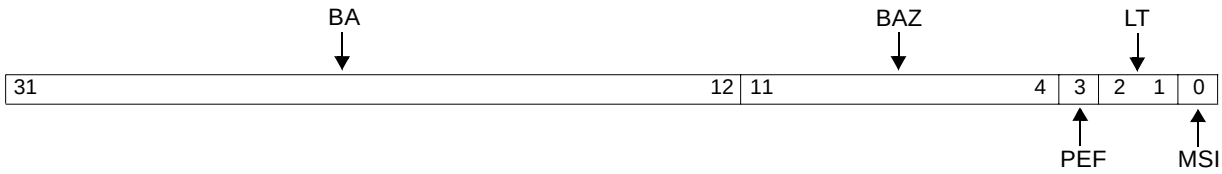


Figure 29-176. PCI-X BAR2 Low Register (PCIX0_BAR2L)

31:12	BA	Base Address	These bits determine the lower 32-bits of the PCI Memory address space where this region is located. Only corresponding bits that are 1 in the size field of the PIM2 Size/Attribute Register are writable. Bits that are 0 in the size field of the PIM2 Size/Attribute Register cause the corresponding Base Address bits to be always 0.
11:4	BAZ	Base Address - always zero	These bits are always 0 since the minimum size of this range is 4 KB.

3	PEF	Prefetchable	This bit determines if the region is prefetchable. Its value is determine by the PIM2 Size/Attribute , Prefetch Enable bit.
2:1	LT	Location Type	These bits are always 10 to indicate that the memory space can be located anywhere in the 64-bit address space.
0	MSI	Memory Space Indicator	This bit is always 0 to indicate memory space (rather than I/O).

MMIO 0x2 0EC8000F Read-Only (PLB), 0x0F Read-Only (PCI-X)

~~See "PCI-X Built-in Self Test (BIST) Control Register (PCIX0_BIST)" on page 18-48.~~See *PCI-X Built-in Self Test (BIST) Control Register (PCIX0_BIST)* on page 580.

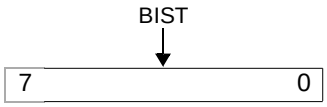


Figure 0-6. PCI-X BIST Control Register (PCIX0_BIST)

7:0	BIST	PCI Built-in-Self Test (BIST) Control

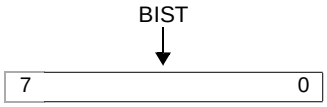


Figure 29-177. PCI-X BIST Control Register (PCIX0_BIST)

7:0	BIST	PCI Built-in-Self Test (BIST) Control

MMIO 0x2 0EC80040Read/Write (PLB), 0x43-0x40 Read/Write (PCI-X)

~~See "PCI-X Bridge Options 1 (PCIX0_BRDGOPT1)" on page 18-62.~~See *PCI-X Bridge Options 1 (PCIX0_BRDGOPT1)* on page 594.

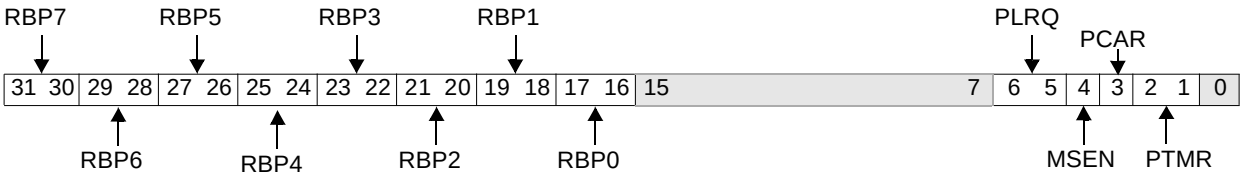


Figure 0-7. PCI-X Bridge Options 1 (PCIX0_BRDGOPT1)

31:30	RBP7		Must be set 00.
29:28	RBP6	Outbund Read Buffer Mapping for PLB master 6 DMA	Must be set to 01.
27:26	RBP5		Must be set to 00.
25:24	RBP4		Must be set to 00.
23:22	RBP3		Must be set to 00.
21:20	RBP2		Must be set to 00.
19:18	RBP1		Must be set to 00.
17:16	RBP0		Must be set to 00.
15:8		Reserved	These bits are always zero.
7		Reserved	This bit must be set to zero.
6:5	PLRQ	PLB Request Priority	This bit controls how the bridge PLB master controls the PLB arbitration priority for all PLB accesses. 11b: highest 10b: next highest 01b: next highest 00b: lowest
4	MSEN	Inbound MSI Enable	This bit enables Inbound MSI. This is typically only used in Host-Bridge Mode. When high, the Inbound MSI logic drives TBD to UIC.
3	PCAR	PCI Arbiter Park Mode	This bit defines how the internal PCI arbiter will handle bus parking. A value of 0 means that the arbiter will park on requester 0 (the bridge PCI master). A value of 1 means that the arbiter will park on the last master granted the bus. This bit must not be changed while PCI masters are active.

2:1	PTMR	PCI Target Memory Read Command interpretation	This field allows PLB-PCIX Bridge to be forced to treat a PCI memory read as a memory read multiple or memory read line with respect to the burst size implied by these read commands. This is for masters that use memory read for multiple beat bursts. 00 Memory read 01 Memory read line 10 Memory read multiple 11 Reserved
0		Reserved	Must be set to 0.

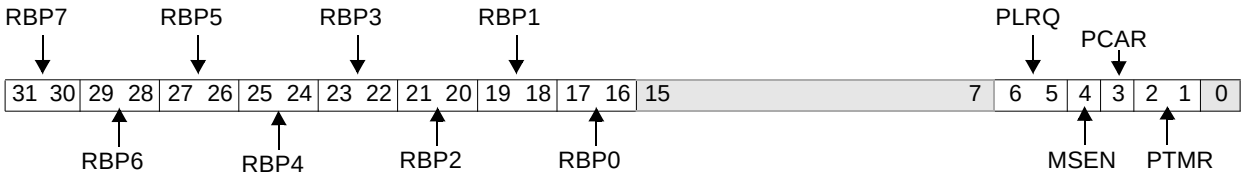


Figure 29-178. PCI-X Bridge Options 1 (PCIX0_BRDGOPT1)

31:30	RBP7		Must be set 00.
29:28	RBP6	Outbund Read Buffer Mapping for PLB master 6 DMA	Must be set to 01.
27:26	RBP5		Must be set to 00.
25:24	RBP4		Must be set to 00.
23:22	RBP3		Must be set to 00.
21:20	RBP2		Must be set to 00.
19:18	RBP1		Must be set to 00.
17:16	RBP0		Must be set to 00.
15:8		Reserved	These bits are always zero.
7		Reserved	This bit must be set to zero.
6:5	PLRQ	PLB Request Priority	This bit controls how the PLB-PCIX Bridge PLB master controls the PLB arbitration priority for all PLB accesses. 11b: highest 10b: next highest 01b: next highest 00b: lowest
4	MSEN	Inbound MSI Enable	This bit enables Inbound MSI. This is typically only used in Host-Bridge Mode. When high, the Inbound MSI logic drives TBD to UIC.
3	PCAR	PCI Arbiter Park Mode	This bit defines how the internal PCI arbiter will handle bus parking. A value of 0 means that the arbiter will park on requester 0 (the bridge PCI master). A value of 1 means that the arbiter will park on the last master granted the bus. This bit must not be changed while PCI masters are active.

2:1	PTMR	PCI Target Memory Read Command interpretation	This field allows PLB-PCIX Bridge to be forced to treat a PCI memory read as a memory read multiple or memory read line with respect to the burst size implied by these read commands. This is for masters that use memory read for multiple beat bursts. 00 Memory read 01 Memory read line 10 Memory read multiple 11 Reserved
0		Reserved	Must be set to 0.

MMIO 0x2 0EC80044 Read/Write (PLB), 0x47-0x44 Read/Write (PCI-X)

~~See "PCI-XBridge Options 2 (PCIX0_BRDGOPT2)" on page 18-64.~~See *PCI-X Bridge Options 2 (PCIX0_BRDGOPT2)* on page 596.

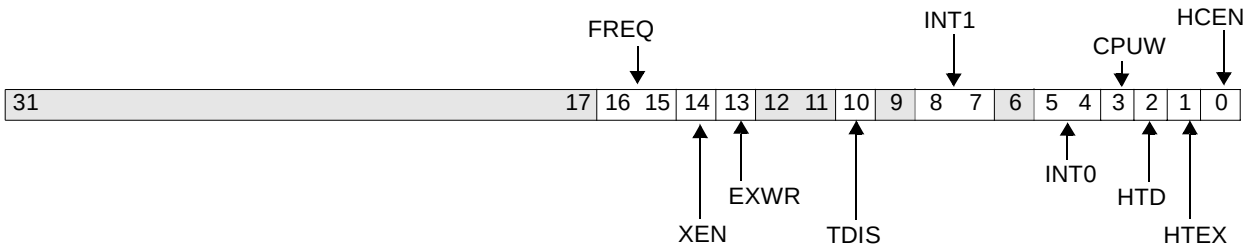


Figure 0-8. PCI-X Bridge Options 2 (PCIX0_BRDGOPT2)

31:17		Reserved	Always read as zero.
16:15	FREQ	PCI Frequency Range	Read only value that reports the PCI frequency range: For PCI-X: 00b = 100-133 MHz 01b = 66-100 MHz 10b = 50-66 MHz 11b = reserved. For PCI-Conv: 00b = reserved 01b = reserved 10b = 33-66 MHz 11b = 0-33 MHz.
14	XEN	PCI-X Mode Enabled	Read only value that reports if X mode was selected.
13	EXWR	External Write to PCI Command Interrupt	When the PCI Command Register is written from the PCI side (inbound config write), this bit is set. It causes the TBD interrupt, which goes to the local CPU, to assert. Software may also set or clear this bit normally.
12:11		Reserved	Must be set to 0.
10	TDIS	PCI Discard Timer Disable	When 1, PLB-PCIX Bridge never discards delayed read data.
9		Reserved	Must be set to 0.
8:7	INT1	PCI Interrupt Source 1	Reset to 01. Do not change.
6		Reserved	Must be set to 0.
5:4	INT0	PCI Interrupt Source 0	Reset to 00. Do not change.

3	CPUW	Local CPU wait	This bit controls the value of the PCI local CPU wait (PCWE) strapping bit, which prevents the local CPU from running when this bit is 1. By causing this bit to be 1 at reset, the host PCI master is given time to initialize the internal register set of the PLB-PCIX Bridge before the local CPU boots. This is typically used to set up boot code, when the local CPU is not booting from local ROM.
2	HTD	Host Configuration Enable Timer Disable	If this bit is set, the Host Configuration Enable timer never expires. See Host Configuration Enable bit below.
1	HTEX	Host Configuration Enable Timer Expired	This bit is set if the Host Configuration Enable timer expired. See Host Configuration Enable bit below.
0	HCEN	Host Configuration Enable	This bit controls host PCI access to the PCI configuration registers. If this bit is a 0, all host attempts to access PCI configuration registers (internal registers) of the PLB-PCIX Bridge are retried. By causing this bit to be 0 at reset, the local CPU is given time to initialize the internal register set before the host sees it. The reset value of this bit is determined by the PCI host configuration enable (PHCE) strapping bit. If PCI host configuration enable (PHCE) strapping bit is tied to a 1 (high), this bit has a value of 1 after reset. If PCI host configuration enable (PHCE) strapping bit is tied to a 0 (low), this bit has a value of 0 after reset. If this bit is cleared following reset, and the Host Configuration Enable Timer Disable bit is cleared, and if this bit does not get set by the local CPU within 2^{24} PLB clocks (126 ms at 133 MHz), then it is set automatically, and the Host Configuration Enable Timer Expired bit is set in this register. **Note:** As per the PCI Specification, Version 2.2, as little as 500 ms is allowed for configuration to be set up. Thus, if PLB is run below 34 MHz, this timer will not expire before this time period.

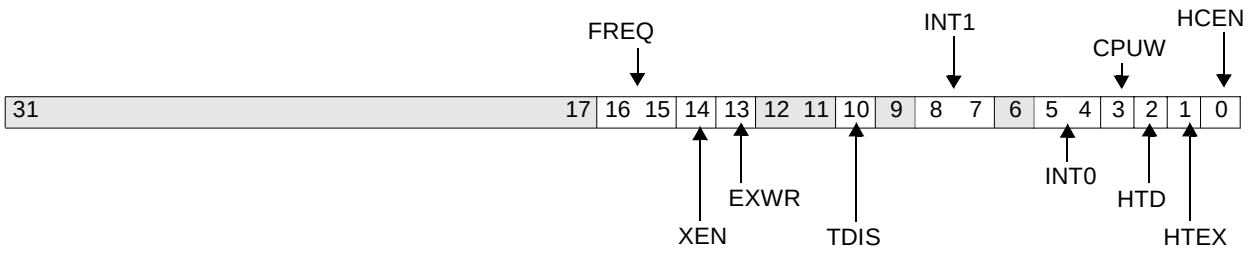


Figure 29-179. PCI-X Bridge Options 2 (PCIX0_BRDGOPT2)

31:17		Reserved	Always read as zero.
16:15	FREQ	PCI Frequency Range	Read only value that reports the PCI frequency range: For PCI-X: 00b = 100-133 MHz 01b = 66-100 MHz 10b = 50-66 MHz 11b = reserved. For PCI-Conv: 00b = reserved 01b = reserved 10b = 33-66 MHz 11b = 0-33 MHz.
14	XEN	PCI-X Mode Enabled	Read only value that reports if X mode was selected.
13	EXWR	External Write to PCI Command Interrupt	When the PCI Command Register is written from the PCI side (inbound config write), this bit is set. It causes the TBD interrupt, which goes to the local CPU, to assert. Software may also set or clear this bit normally.
12:11		Reserved	Must be set to 0.
10	TDIS	PCI Discard Timer Disable	When 1, PLB-PCIX Bridge never discards delayed read data.
9		Reserved	Must be set to 0.
8:7	INT1	PCI Interrupt Source 1	Reset to 01. Do not change.
6		Reserved	Must be set to 0.
5:4	INT0	PCI Interrupt Source 0	Resets to 00. Setting this field to 01 will mask the generation of outbound interrupts, both via INTA and MSI.
3	CPUW	Local CPU wait	This bit controls the value of the PCI local CPU wait (PCWE) strapping bit, which prevents the local CPU from running when this bit is 1. By causing this bit to be 1 at reset, the host PCI master is given time to initialize the internal register set of the PLB-PCIX Bridge before the local CPU boots. This is typically used to set up boot code, when the local CPU is not booting from local ROM.
2	HTD	Host Configuration Enable Timer Disable	If this bit is set, the Host Configuration Enable timer never expires. See Host Configuration Enable bit below.
1	HTEX	Host Configuration Enable Timer Expired	This bit is set if the Host Configuration Enable timer expired. See Host Configuration Enable bit below.

0	HCEN	Host Configuration Enable	This bit controls host PCI access to the PCI configuration registers. If this bit is a 0, all host attempts to access PCI configuration registers (internal registers) of the PLB-PCIX Bridge are retried. By causing this bit to be 0 at reset, the local CPU is given time to initialize the internal register set before the host sees it. The reset value of this bit is determined by the PCI host configuration enable (PHCE) strapping bit. If PCI host configuration enable (PHCE) strapping bit is tied to a 1 (high), this bit has a value of 1 after reset. If PCI host configuration enable (PHCE) strapping bit is tied to a 0 (low), this bit has a value of 0 after reset. If this bit is cleared following reset, and the Host Configuration Enable Timer Disable bit is cleared, and if this bit does not get set by the local CPU within 2^{24} PLB clocks (126 ms at 133 MHz), then it is set automatically, and the Host Configuration Enable Timer Expired bit is set in this register. **Note:** As per the PCI Specification, Version 2.2, as little as 500 ms is allowed for configuration to be set up. Thus, if PLB is run below 34 MHz, this timer will not expire before this time period.

MMIO 0x2 0EC8000C Read/Write (PLB), 0x0C Read/Write (PCI-X)

~~See "PCI-X Cache Line Size Register (PCIX0_CACHELS)" on page 18-45.~~See *PCI-X Cache Line Size Register (PCIX0_CACHELS)* on page 577.

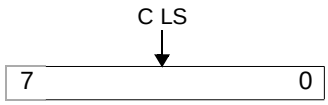


Figure 0-9. PCI-X Cache Line Size Register (PCIX0_CACHELS)

7:0	CLS	PCI Cache Line Size

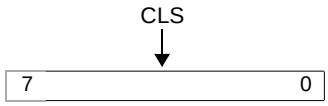


Figure 29-180. PCI-X Cache Line Size Register (PCIX0_CACHELS)

7:0	CLS	PCI Cache Line Size

MMIO 0x2 0EC80034 Read-Only (PLB), 0x34 Read-Onlt (PCI-X)

~~See "PCI-X Capabilities Pointer (PCIX0_CAP)" on page 18-57.~~See *PCI-X Capabilities Pointer (PCIX0_CAP)* on page 589.

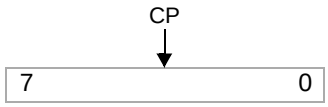


Figure 0-10. PCI-X Capabilities Pointer (PCIX0_CAP)

7:0	CP	PCI Capabilities Pointer

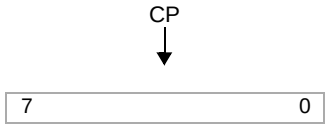


Figure 29-181. PCI-X Capabilities Pointer (PCIX0_CAP)

7:0	CP	PCI Capabilities Pointer

MMIO 0x2 0EC80009 Read/Write (PLB), 0x0B-0x09 Read-Only (PCI)-X)

~~See "PCI-X Class Register (PCIX0_CLS)" on page 18-44.~~See *PCI-X Class Register (PCIX0_CLS)* on page 576.

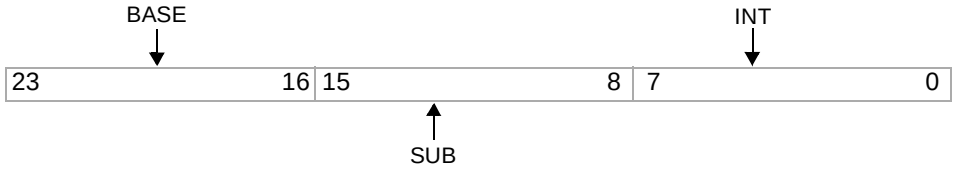


Figure 0-11. PCI-X Class Register (PCIX0_CLS)

23:16	BASE	Base Class	Reset to 0x06, which indicates bridge device.
15:8	SUB	Subclass	Reset to 00, which indicates host bridge.
7:0	INT	Interface Class	Reset to 00.

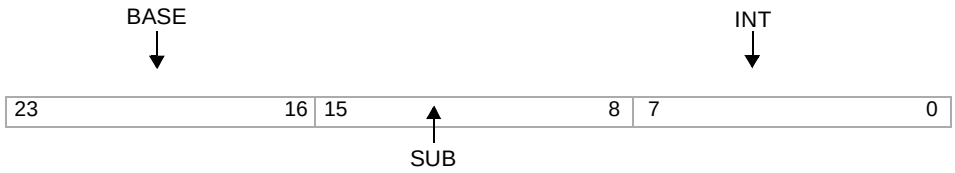


Figure 29-182. PCI-X Class Register (PCIX0_CLS)

23:16	BASE	Base Class	Reset to 0x06, which indicates bridge device.
15:8	SUB	Subclass	Reset to 00, which indicates host bridge.
7:0	INT	Interface Class	Reset to 00.

MMIO 0x2 0EC80004 R/W (PLB) 0x05-0x04 Read-Only (PCI-X)

~~See "PCI-X Command Register (PCIX0_CMD)" on page 18-39.~~See *PCI-X Command Register (PCIX0_CMD)* on page 572.

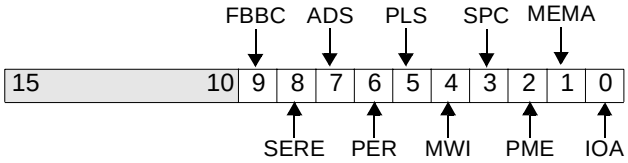


Figure 0-12. PCI-X Command Register (PCIX0_CMD)

15:10		Reserved	These bits are reserved, and return zeros when read.
9	FBBC	Fast Back-to-Back Capable Read-only; returns 0 when read.	Fast back-to-back write enable. This bit enables PCI masters to perform fast back-to-back transactions to different devices. The PLB-PCIX Bridge does not perform fast back-to-back transactions, therefore, this bit is read-only and returns zero when read.
8	SERE	SERR# Enable	Enable PCI_SERR#. Enables driving PCI_SERR# to report the detection of an error out to the PCI bus. If disabled, PCI_SERR# is never asserted regardless of the detection of any error.
7	ADS	Address Stepping	Address stepping wait states. PLB-PCIX bridge does not address step (except when generating a Configuration Type 0 cycle), therefore, this bit is read-only and returns zero when read.
6	PER	Parity Error Response	Parity error response. This bit enables the following types of PCI bus parity errors: PCI data bus parity errors while PCI is master PCI data bus parity errors while PCI is target PCI address bus parity errors When parity error response is disabled (set to 0), detection of these errors is masked and PERR# is not asserted, although parity is still generated.
5	PLS	Palette Snooping	Enable special palette snooping. The PLB-PCIX Bridge macro is not a VGA device, therefore, this bit is read-only and returns zero when read.
4	MWI	Memory Write and Invalidate	Enable memory write and invalidate command support. When high, PLB-PCIX Bridge is allowed to generate MWI transactions. Only used in PCI-Conv mode. This bit is 0 at reset.
3	SPC	Special Cycle	Enable special cycle operations. PLB-PCIX Bridge never monitors special cycles, therefore, this bit is read-only and returns zero when read.

2	PME	PCI Master Enable	Enables PLB-PCIX Bridge to master cycles on the PCI bus. When this bit is 0, PLB-PCI Bridge only responds as a PLB slave to accesses to the internal register set. However, the PLB-PCIX Bridge can still master transfers for split completions and to complete outbound writes that were posted prior to setting this bit to 0.
1	MEMA	Memory Access	Controls response of PLB-PCIX Bridge as a PCI memory target. A value of "1" enables PLB-PCIX Bridge to respond as a target. This bit is 0 (disabled) at reset.
0	IOA	I/O Access	Controls response of PLB-PCIX Bridge as a PCI I/O target. A value of 1 enables PLB-PCIX Bridge to respond as a target. This bit is 0 (disabled) at reset.

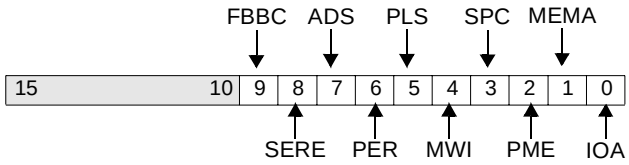


Figure 29-183. PCI-X Command Register (PCIX0_CMD)

15:10		Reserved	These bits are reserved, and return zeros when read.
9	FBBC	Fast Back-to-Back Capable Read-only; returns 0 when read.	Fast back-to-back write enable. This bit enables PCI masters to perform fast back-to-back transactions to different devices. The PLB-PCIX Bridge does not perform fast back-to-back transactions, therefore, this bit is read-only and returns zero when read.
8	SERE	SERR# Enable	Enable PCI_SERR#. Enables driving PCI_SERR# to report the detection of an error out to the PCI bus. If disabled, PCI_SERR# is never asserted regardless of the detection of any error.
7	ADS	Address Stepping	Address stepping wait states. PLB-PCIX bridge does not address step (except when generating a Configuration Type 0 cycle), therefore, this bit is read-only and returns zero when read.
6	PER	Parity Error Response	Parity error response. This bit enables the following types of PCI bus parity errors: PCI data bus parity errors while PCI is master PCI data bus parity errors while PCI is target PCI address bus parity errors When parity error response is disabled (set to 0), detection of these errors is masked and PERR# is not asserted, although parity is still generated.
5	PLS	Palette Snooping	Enable special palette snooping. The PLB-PCIX Bridge macro is not a VGA device, therefore, this bit is read-only and returns zero when read.
4	MWI	Memory Write and Invalidate	Enable memory write and invalidate command support. When high, PLB-PCIX Bridge is allowed to generate MWI transactions. Only used in PCI-Conv mode. This bit is 0 at reset.
3	SPC	Special Cycle	Enable special cycle operations. PLB-PCIX Bridge never monitors special cycles, therefore, this bit is read-only and returns zero when read.

2	PME	PCI Master Enable	Enables PLB-PCIX Bridge to master cycles on the PCI bus. When this bit is 0, PLB-PCI Bridge only responds as a PLB slave to accesses to the internal register set. However, the PLB-PCIX Bridge can still master transfers for split completions and to complete out-bound writes that were posted prior to setting this bit to 0.
1	MEMA	Memory Access	Controls response of PLB-PCIX Bridge as a PCI memory target. A value of "1" enables PLB-PCIX Bridge to respond as a target. This bit is 0 (disabled) at reset.
0	IOA	I/O Access	Controls response of PLB-PCIX Bridge as a PCI I/O target. A value of 1 enables PLB-PCIX Bridge to respond as a target. This bit is 0 (disabled) at reset.

MMIO 0x2 0EC80002 Read/Write (PLB), 0x03-0x02 Read-Only (PCI-X)

~~See "PCI-X Device ID Register (PCIX0_DEVID)" on page 18-38.~~See *PCI-X Device ID Register (PCIX0_DEVID)* on page 571.

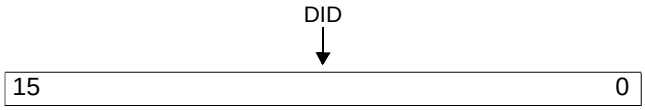


Figure 0-13. PCI-X Device ID Register (PCIX0_DEVID)

15:0	DID	Device ID

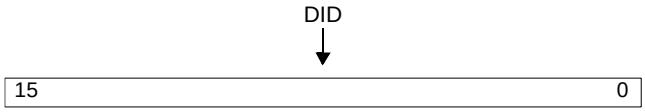


Figure 29-184. PCI-X Device ID Register (PCIX0_DEVID)

15:0	DID	Device ID

MMIO 0x2 0EC80030 Read/Write (PLB), 0x33-0x30 Read/Write (PCI-X)

~~See "PCI-X Expansion ROM Base Address Register (PCIX0_EROMBA)" on page 18-56.~~See *PCI-X Expansion ROM Base Address Register (PCIX0_EROMBA)* on page 588.

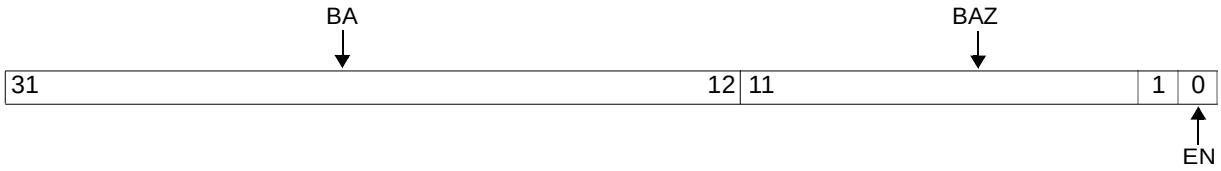


Figure 0-14. PCI-X Expansion ROM Base Address Register (PCIX0_EROMBA)

31:12	BA	Base Address	These bits determine the PCI I/O address space where this region is located.
11:1	BAZ	Base Address - always zero	These bits are always 0 since the minimum size of this range is 4 KB.
0	EN	Enable	When 1, the expansion ROM address space of the PLB_PCIX Bridge is enabled and the PIM2 (BAR2) address space is disabled. When 0, the expansion ROM address space is disabled and the PIM2 (BAR2) address space is enabled.

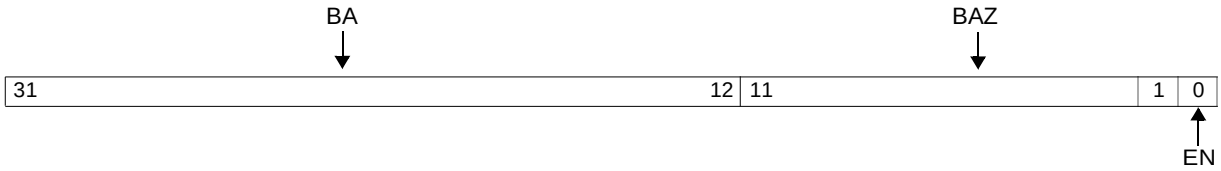


Figure 29-185. PCI-X Expansion ROM Base Address Register (PCIX0_EROMBA)

31:12	BA	Base Address	These bits determine the PCI I/O address space where this region is located.
11:1	BAZ	Base Address - always zero	These bits are always 0 since the minimum size of this range is 4 KB.
0	EN	Enable	When 1, the expansion ROM address space of the PLB-PCIX Bridge is enabled and the BAR2 address space is disabled. When 0, the expansion ROM address space is disabled and the BAR2 address space is enabled, translated through PIM2.

MMIO 0x2 0EC80050 Read/Write (PLB), 0x53-0x50 Read/Write (PCI-X)

~~See "PCI-X Error Enable (PCIX0_ERREN)" on page 18-66.~~See *PCI-X Error Enable (PCIX0_ERREN)* on page 598.

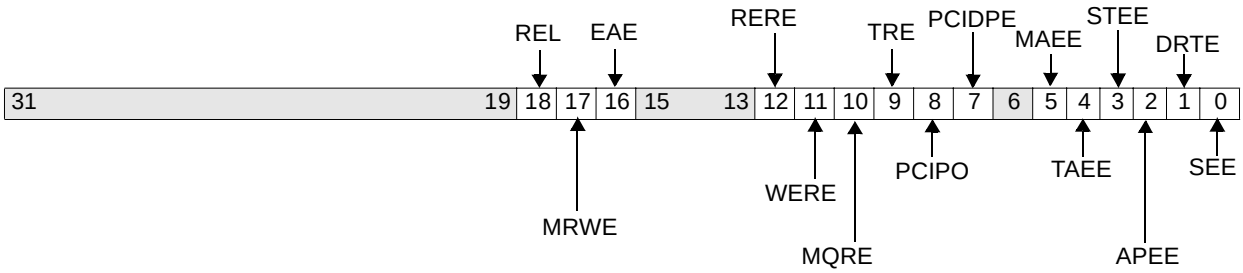


Figure 0-15. PCI-X Error Enable (PCIX0_ERREN)

31:19		Reserved	Always read as zero.
18	REL	Route Error Locally	When set, errors are routed locally using ERROR_INT. Assertion of PLB MERR is not affected by this bit. When cleared, errors are routed to the PCI using PCI_SERR#.
17	MRWE	MRdErr/MWrErr Assertion Enable	This bit enables the assertion of PLB Read MERR or PLB Write MERR when the PLB-PCIX Bridge detects an error.
16	EAE	ERROR_INT Assertion Enable	This bit enables the assertion of ERROR_INT when the PLB-PCIX Bridge detects an error.
13		Reserved	Always read as zero.
12	RERE	MRdErr Receive Enable	This bit enables the detection of PLB Read MERR when the PLB-PCIX Bridge is the PLB master. If this bit is set, the PLB-PCIX Bridge may drive ERROR_INT to the local CPU or PCI_SERR# to the PCI bus.
11	WERE	MWrErr Receive Enable	This bit enables the detection of PLB Write MERR when PLB-PCIX Bridge is the PLB master. If this bit is set, the PLB-PCIX Bridge may drive ERROR_INT to the local CPU or PCI_SERR# to the PCI bus.
10	MQRE	Mirq Receive Enable	This bit enables the detection of MIRQ when PLB-PCIX Bridge is the PLB master. If this bit is set, the PLB-PCIX Bridge may drive ERROR_INT to the local CPU or PCI_SERR# to the PCI bus.
9	TRE	Timeout Receive Enable	This bit enables the detection of Timeout when PLB-PCIX Bridge is the PLB master. If this bit is set, the PLB-PCIX Bridge may drive ERROR_INT to the local CPU or PCI_SERR# to the PCI bus.
8	PO	PCI Parity Option	This bit should normally be left 1. It may be set to 0 to mask all types of PCI parity error checking. This bit is 1 after reset.

7	IDPE	PCI Inbound Write Data Parity Error Enable	This bit enables the detection of PCI data parity errors on inbound writes. **Note:** The Detected Parity Error bit of the PCI Status Register and the masking of PCI PERR# is not affected by the above, but the assertion of ERROR_INT and PCI_SERR# is.
6		Reserved	Always read as zero.
5	MAEE	Master Abort Error Enable	This bit enables the detection of master aborts as an error condition, when PLB-PCIX Bridge is the PCI master. If this bit is set, PLB-PCIX Bridge may drive PLB Read MERR or PLB Write MERR on the PLB or PCI_SERR# on the PCI bus.
4	TAEE	Target Abort Error Enable	This bit enables the detection of a target abort received while PLB-PCIX Bridge is the PCI master to be detected as an error condition. If this bit is set, the PLB-PCIX Bridge may drive ERROR_INT to the local CPU or PCI_SERR# on the PCI bus.
3	STEE	Split Transaction Error Enable	This bit enables the detection of split transaction errors (PCI-X only), while the PLB-PCIX Bridge is the PCI master. If this bit is set, the PLB-PCIX Bridge may drive ERROR_INT to the local CPU or PCI_SERR# to the PCI bus.
2	APEE	Addr/Attrib Parity Error Enable	This bit enables the detection of Address or Attribute (PCI-X only) parity errors while the PLB-PCIX Bridge is the PCI target (including split response). If this bit is set, the PLB-PCIX Bridge may drive PCI_SERR# on the PCI bus.
1	DRTE	Delayed Read Discard Timer Expired Error Enable	This bit enables the detection of Delayed Read Discard Timer expiration as an error. If this bit is set, the PLB-PCIX Bridge may drive the ERROR_INT signal to the local CPU or PCI_SERR# on the PCI bus.
0	SEE	PCI_SERR# Received as Error Enable	This bit enables the detection of PCI_SERR# as an error. If this bit is set, the PLB-PCIX Bridge may drive the ERROR_INT signal to the local CPU. **Note:** The PLB-PCIX Bridge may be receiving this error, while at the same time, it is driving PCI_SERR# in response to some other error.

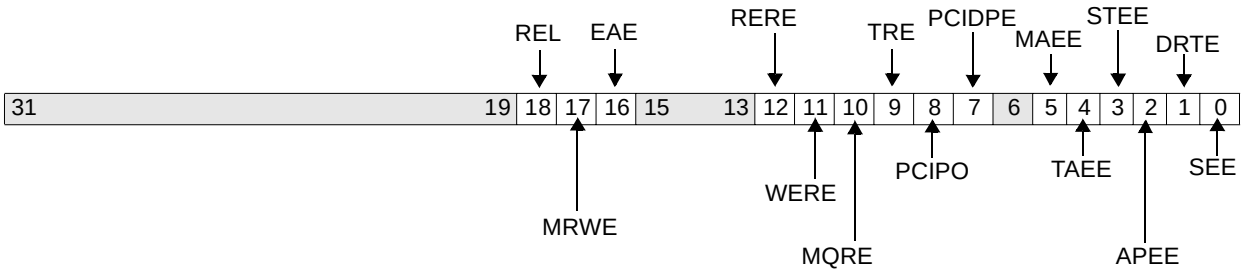


Figure 29-186. PCI-X Error Enable (PCIX0_ERREN)

31:19		Reserved	Always read as zero.
18	REL	Route Error Locally	When set, errors are routed locally using ERROR_INT. Assertion of PLB MERR is not affected by this bit. When cleared, errors are routed to the PCI using PCI_SERR#.
17	MRWE	MRdErr/MWrErr Assertion Enable	This bit enables the assertion of PLB Read MERR or PLB Write MERR when the PLB-PCIX Bridge detects an error.
16	EAE	ERROR_INT Assertion Enable	This bit enables the assertion of ERROR_INT when the PLB-PCIX Bridge detects an error.
13		Reserved	Always read as zero.
12	RERE	MRdErr Receive Enable	This bit enables the detection of PLB Read MERR when the PLB-PCIX Bridge is the PLB master. If this bit is set, the PLB-PCIX Bridge may drive ERROR_INT to the local CPU or PCI_SERR# to the PCI bus.
11	WERE	MWrErr Receive Enable	This bit enables the detection of PLB Write MERR when PLB-PCIX Bridge is the PLB master. If this bit is set, the PLB-PCIX Bridge may drive ERROR_INT to the local CPU or PCI_SERR# to the PCI bus.
10	MQRE	Mirq Receive Enable	This bit enables the detection of MIRQ when PLB-PCIX Bridge is the PLB master. If this bit is set, the PLB-PCIX Bridge may drive ERROR_INT to the local CPU or PCI_SERR# to the PCI bus.
9	TRE	Timeout Receive Enable	This bit enables the detection of Timeout when PLB-PCIX Bridge is the PLB master. If this bit is set, the PLB-PCIX Bridge may drive ERROR_INT to the local CPU or PCI_SERR# to the PCI bus.
8	PO	PCI Parity Option	This bit should normally be left 1. It may be set to 0 to mask all types of PCI parity error checking. This bit is 1 after reset.
7	IDPE	PCI Inbound Write Data Parity Error Enable	This bit enables the detection of PCI data parity errors on inbound writes. **Note:** The Detected Parity Error bit of the PCI Status Register and the masking of PCI PERR# is not affected by the above, but the assertion of ERROR_INT and PCI_SERR# is.
6		Reserved	Always read as zero.
5	MAEE	Master Abort Error Enable	This bit enables the detection of master aborts as an error condition, when PLB-PCIX Bridge is the PCI master. If this bit is set, PLB-PCIX Bridge may drive PLB Read MERR or PLB Write MERR on the PLB or PCI_SERR# on the PCI bus.
4	TAEE	Target Abort Error Enable	This bit enables the detection of a target abort received while PLB-PCIX Bridge is the PCI master to be detected as an error condition. If this bit is set, the PLB-PCIX Bridge may drive ERROR_INT to the local CPU or PCI_SERR# on the PCI bus.

3	STEE	Split Transaction Error Enable	This bit enables the detection of split transaction errors (PCI-X only), while the PLB-PCIX Bridge is the PCI master. If this bit is set, the PLB-PCIX Bridge may drive ERROR_INT to the local CPU or PCI_SERR# to the PCI bus.
2	APEE	Addr/Attrib Parity Error Enable	This bit enables the detection of Address or Attribute (PCI-X only) parity errors while the PLB-PCIX Bridge is the PCI target (including split response). If this bit is set, the PLB-PCIX Bridge may drive PCI_SERR# on the PCI bus.
1	DRTE	Delayed Read Discard Timer Expired Error Enable	This bit enables the detection of Delayed Read Discard Timer expiration as an error. If this bit is set, the PLB-PCIX Bridge may drive the ERROR_INT signal to the local CPU or PCI_SERR# on the PCI bus.
0	SEE	PCI_SERR# Received as Error Enable	This bit enables the detection of PCI_SERR# as an error. If this bit is set, the PLB-PCIX Bridge may drive the ERROR_INT signal to the local CPU. **Note:** The PLB-PCIX Bridge may be receiving this error, while at the same time, it is driving PCI_SERR# in response to some other error.

MMIO 0x2 0EC80054 Read/Write (PLB), 057-054 Read/Write (PCI-X)

~~See "PCI-X Error Status (PCIX0_ERRSTS)" on page 18-68.~~See *PCI-X Error Status (PCIX0_ERRSTS)* on page 600.

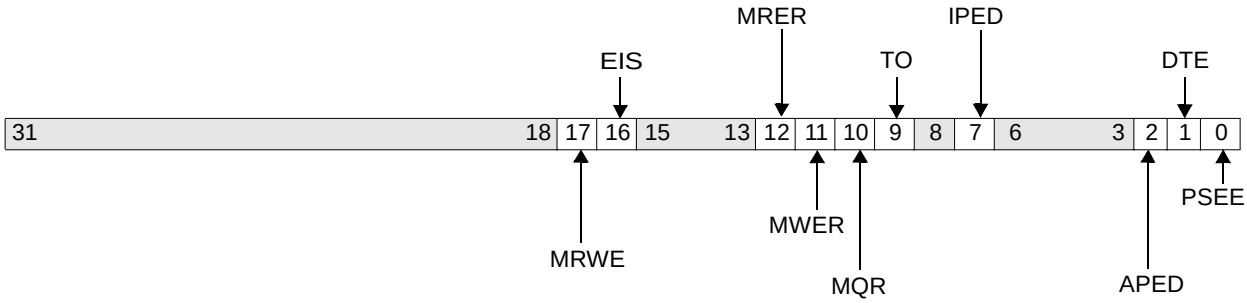


Figure 0-16. PCI-X Error Status (PCIX0_ERRSTS)

31:18		Reserved	Always read as zero.
17	MRWE	MRdErr or MWrErr Signaled	This bit is set whenever an error occurs that causes PLB-PCIX Bridge to assert PLB Read MERR or PLB Write MERR.
16	EIS	ERROR_INT Signaled	This bit is set whenever an error occurs that causes PLB-PCIX Bridge to assert ERROR_INT.
15:13		Reserved	Always read as zero.
12	MRER	MRdErr Received	This bit is set when PLB-PCIX Bridge is a PLB master and receives PLB Read MERR.
11	MWER	MWrErr Received	This bit is set when PLB-PCIX Bridge is a PLB master and receives PLB Write MERR.
10	MQR	Mirq Received	This bit is set when PLB-PCIX Bridge is a PLB master and receives MIRQ.
9	TO	Timeout	This bit is set when PLB-PCIX Bridge is a PLB master and receives Timeout.
8		Reserved	Always read as zero.
7	IPED	PCI Inbound Write Data Parity Error Detected	This bit is set when a PCI inbound write data parity error is detected and the PCI Inbound Write Data Parity Error Enable bit of the Error Enable Register is set.
6		Reserved	Always read as zero.
5		Reserved	Always read as zero. **Note:** The status bit for the detection of master aborts is in the PCI Status Register.
4		Reserved	Always read as zero. **Note:** The status bit for the detection of target aborts is in the PCI Status Register.

3		Reserved	Always read as zero. **Note:** The split completion error is indicated in the PCI-X Status Register.
2	APED	Addr/Attrib Parity Error Detected	This bit is set when the PLB-PCIX Bridge is the target on the PCI bus and detects an address or attribute parity error and the Address/Attribute Parity Error Enable bit of the Error Enable Register is set and the Parity Error Response bit of the PCI Command Register is set.
1	DTE	Delayed Read Discard Timer Expired	This bit is set when the PLB-PCIX Bridge detects that the Discard Timer Expires bit (PCI-Conv Only) and the Delayed Read Discard Timer Expired Error Enable bit of the Error Enable Register is set.
0	PSEE	PCI_SERR# Received	This bit is set when PCI_SERR# is asserted and the PCI_SERR# Receive as Error Enable bit of the Error Enable Register is set.

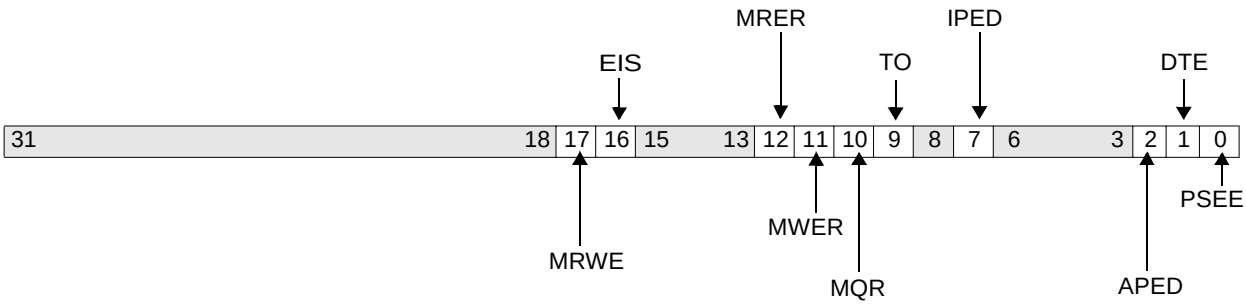


Figure 29-187. PCI-X Error Status (PCIX0_ERRSTS)

31:18		Reserved	Always read as zero.
17	MRWE	MRdErr or MWrErr Signaled	This bit is set whenever an error occurs that causes PLB-PCIX Bridge to assert PLB Read MERR or PLB Write MERR.
16	EIS	ERROR_INT Signaled	This bit is set whenever an error occurs that causes PLB-PCIX Bridge to assert ERROR_INT.
15:13		Reserved	Always read as zero.
12	MRER	MRdErr Received	This bit is set when PLB-PCIX Bridge is a PLB master and receives PLB Read MERR.
11	MWER	MWrErr Received	This bit is set when PLB-PCIX Bridge is a PLB master and receives PLB Write MERR.
10	MQR	Mirq Received	This bit is set when PLB-PCIX Bridge is a PLB master and receives MIRQ.
9	TO	Timeout	This bit is set when PLB-PCIX Bridge is a PLB master and receives Timeout.
8		Reserved	Always read as zero.
7	IPED	PCI Inbound Write Data Parity Error Detected	This bit is set when a PCI inbound write data parity error is detected and the PCI Inbound Write Data Parity Error Enable bit of the Error Enable Register is set.
6		Reserved	Always read as zero.

5		Reserved	Always read as zero. **Note:** The status bit for the detection of master aborts is in the PCI Status Register.
4		Reserved	Always read as zero. **Note:** The status bit for the detection of target aborts is in the PCI Status Register.
3		Reserved	Always read as zero. **Note:** The split completion error is indicated in the PCI-X Status Register.
2	APED	Addr/Attrib Parity Error Detected	This bit is set when the PLB-PCIX Bridge is the target on the PCI bus and detects an address or attribute parity error and the Address/Attribute Parity Error Enable bit of the Error Enable Register is set and the Parity Error Response bit of the PCI Command Register is set.
1	DTE	Delayed Read Discard Timer Expired	This bit is set when the PLB-PCIX Bridge detects that the Discard Timer Expires bit (PCI-Conv Only) and the Delayed Read Discard Timer Expired Error Enable bit of the Error Enable Register is set.
0	PSEE	PCI_SERR# Received	This bit is set when PCI_SERR# is asserted and the PCI_SERR# Receive as Error Enable bit of the Error Enable Register is set.

MMIO 0x2 0EC8000E Read-Only (PLB), 0x0E Read-Only (PCI-X)

~~See "PCI-X Header Type Register (PCIX0_HDTYPE)" on page 18-47.~~See *PCI-X Header Type Register (PCIX0_HDTYPE)* on page 579.

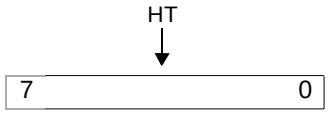


Figure 0-17. PCI-X Header Type Register (PCIX0_HDTYPE)

7:0	HT	PCI Header Type

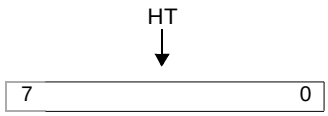


Figure 29-188. PCI-X Header Type Register (PCIX0_HDTYPE)

7:0	HT	PCI Header Type

MMIO 0x2 0EC801F8 Write-Only (PLB), (PCIX0_BAR0H || PCIX0_BAR0L) + (0xFB-0xF8) Write-Only (PCI-X)

~~See "PCI-X Inbound Message MSI (PCIX0_IM)" on page 18-119.~~See *PCI-X Inbound Message MSI (PCIX0_IM)* on page 649.

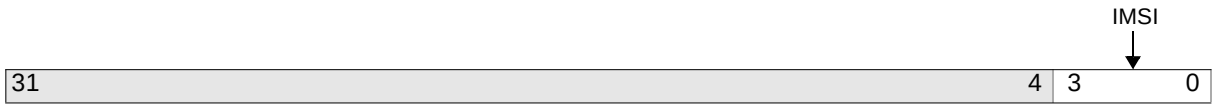


Figure 0-18. PCI-X Inbound Message MSI (PCIX0_IM)

31:4		Reserved	Returns zero when read.
3:0	IMSI	MSI In	Inbound MSIs write this register. The data value written causes the corresponding pci_msi_int0:12 to go active to the interrupt controller.

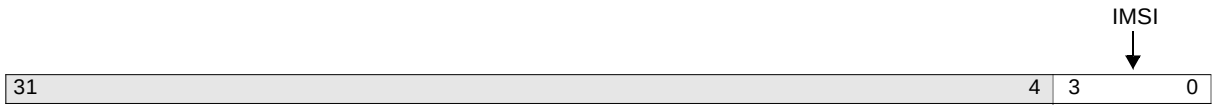


Figure 29-189. PCI-X Inbound Message MSI (PCIX0_IM)

31:4		Reserved	Returns zero when read.
3:0	IMSI	MSI In	Inbound MSIs write this register. The data value written causes the corresponding pci_msi_int0:12 to go active to the interrupt controller.

MMIO 0x2 0EC8003C Read/Write (PLB), 0x3C Read/Write (PCI-X)

~~See "PCI-X Interrupt Line Register (PCIX0_INTLN)" on page 18-58.~~See *PCI-X Interrupt Line Register (PCIX0_INTLN)* on page 590.

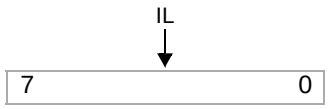


Figure 0-19. PCI-X Interrupt Line Register (PCIX0_INTLN)

7:0	IL	PCI Interrupt Line

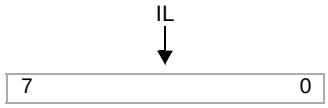


Figure 29-190. PCI-X Interrupt Line Register (PCIX0_INTLN)

7:0	IL	PCI Interrupt Line

MMIO 0x2 0EC8003D Read-Only (PLB), 0x3D read-Only (PCI-X)

~~See "PCI-X Interrupt Pin Register (PCIX0_INTPN)" on page 18-59.~~See *PCI-X Interrupt Pin Register (PCIX0_INTPN)* on page 591.

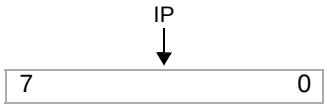


Figure 0-20. PCI-X Interrupt Pin Register (PCIX0_INTPN)

7:0	IP	PCI Interrupt Pin

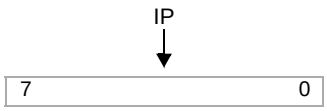


Figure 29-191. PCI-X Interrupt Pin Register (PCIX0_INTPN)

7:0	IP	PCI Interrupt Pin

MMIO 0x2 0EC8000D Read/Write (PLB) 0x0D Read/Write (PCI-X)

~~See "PCI-X Latency Timer Register (PCIX0_LATTIM)" on page 18-46.~~See *PCI-X Latency Timer Register (PCIX0_LATTIM)* on page 578.

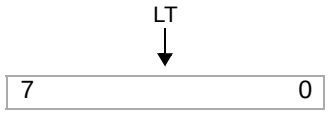


Figure 0-21. PCI-X Latency Timer Register (PCIX0_LATTIM)

7:0	LT	PCI Latency Timer

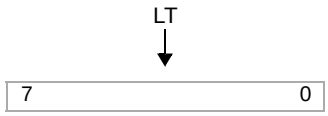


Figure 29-192. PCI-X Latency Timer Register (PCIX0_LATTIM)

7:0	LT	PCI Latency Timer

MMIO 0x2 0EC8003F Read-Only (PLB), 0x3F Read-Only (PCI-X)

~~See "PCI-X Maximum Latency Register (PCIX0_MAXLTNCY)" on page 18-61.~~See *PCI-X Maximum Latency Register (PCIX0_MAXLTNCY)* on page 593.

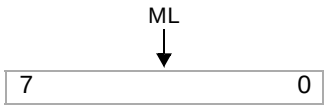


Figure 0-22. PCI-X Maximum Latency Register (PCIX0_MAXLTNCY)

7:0	ML	PCI Maximum Latency

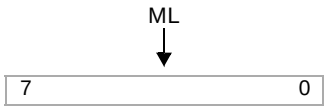


Figure 29-193. PCI-X Maximum Latency Register (PCIX0_MAXLTNCY)

7:0	ML	PCI Maximum Latency

MMIO 0x2 0EC8003E Read-Only (PLB), 0x3E Read-Only (PCI-X)

~~See "PCI-X MIN_GNT Register (PCIX0_MINGNT)" on page 18-60.~~See *PCI-X MIN_GNT Register (PCIX0_MINGNT)* on page 592.

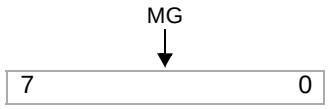


Figure 0-23. PCI-X Minimum Grant Register (PCIX0_MINGNT)

7:0	MG	PCI Minimum Grant

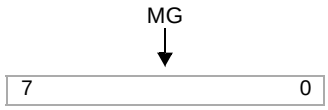


Figure 29-194. PCI-X Minimum Grant Register (PCIX0_MINGNT)

7:0	MG	PCI Minimum Grant

MMIO 0x2 0EC80104 RW (PLB), (PCIX0_BAR0H || PCIX0_BAR0L) + (0x07-0x04) RW (PCI-X)

~~See "PCI-X Message In High (PCIX0_MSGIH)" on page 18-116.~~See *PCI-X Message In High (PCIX0_MSGIH)* on page 647.

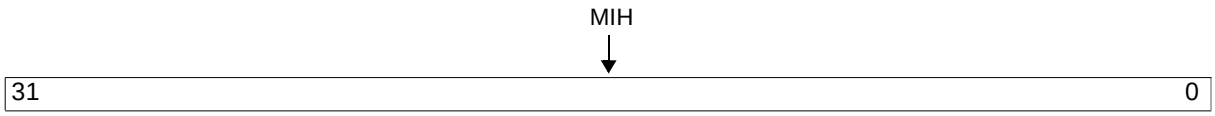


Figure 0-24. PCI-X Message In High (PCIX0_MSGIH)

31:0	MIH	Message In High	This holds the upper 32 bits of a 64-bit inbound message.

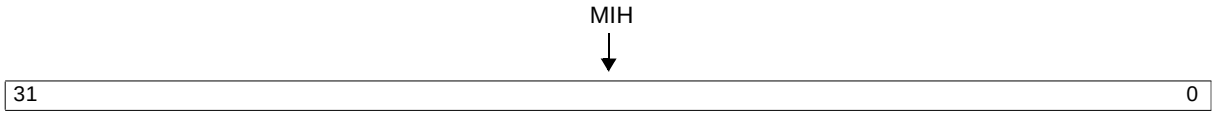


Figure 29-195. PCI-X Message In High (PCIX0_MSGIH)

31:0	MIH	Message In High	This holds the upper 32 bits of a 64-bit inbound message.

MMIO 0x2 0EC80100 R/W (PLB), (PCIX0_BAR0H || PCIX0_BAR0L) + (0x03-0x00) R/W (PCI-X)

~~See "PCI-X Message In Low (PCIX0_MSGIL)" on page 18-115.~~See *PCI-X Message In Low (PCIX0_MSGIL)* on page 647.

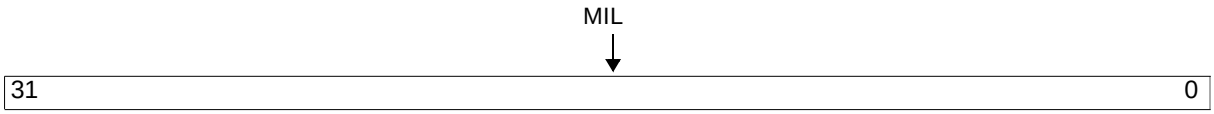


Figure 0-25. PCI-X Message In Low (PCIX0_MSGIL)

31:0	MIL	Message In Low	This holds an inbound message. If bit 0 is high, then an inbound interrupt is generated.

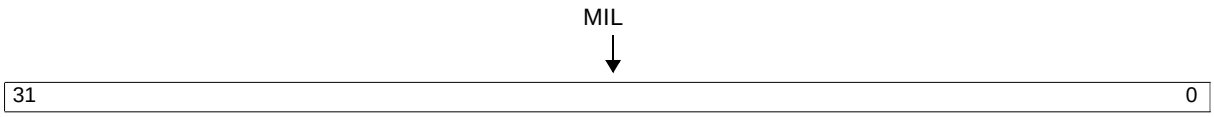


Figure 29-196. PCI-X Message In Low (PCIX0_MSGIL)

31:0	MIL	Message In Low	This holds an inbound message. If bit 0 is high, then an inbound interrupt is generated.

MMIO 0x2 0EC8010C RW (PLB), (PCIX0_BAR0H || PCIX0_BAR0L) + (0x0F-0x0C RW (PCI-X)

~~See "PCI-X Message Out High (PCIX0_MSGOH)" on page 18-118.~~See *PCI-X Message Out High (PCIX0_MSGOH)* on page 648.

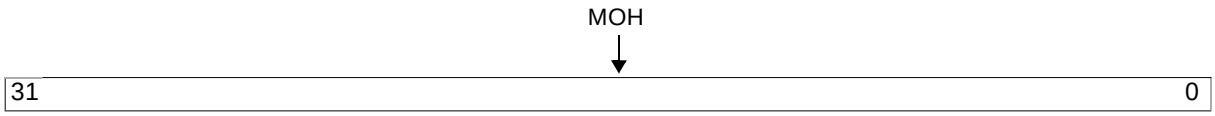


Figure 0-26. PCI-X Message Out High (PCIX0_MSGOH)

31:0	MOH	Message Out High	This holds the upper 32 bits of a 64-bit outbound message.

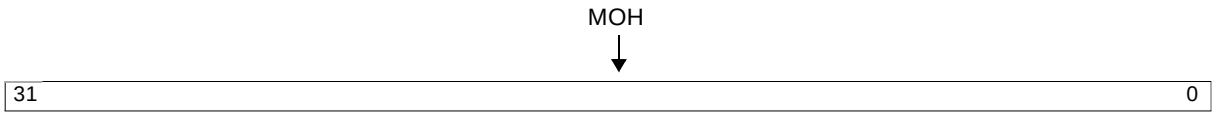


Figure 29-197. PCI-X Message Out High (PCIX0_MSGOH)

31:0	MOH	Message Out High	This holds the upper 32 bits of a 64-bit outbound message.

MMIO 0x2 0EC80108 RW (PLB), (PCIX0_BAR0H || PCIX0_BAR0L) + (0x0B-0x08) RW (PCI-X)

~~See "PCI-X Message Out Low (PCIX0_MSGOL)" on page 18-117.~~See *PCI-X Message Out Low (PCIX0_MSGOL)* on page 648.

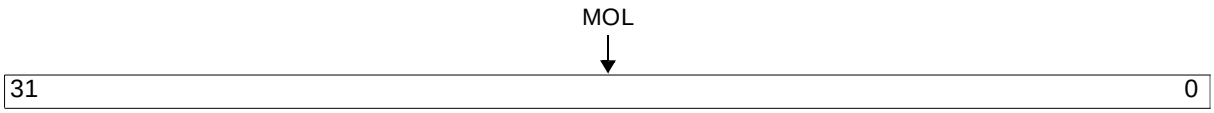


Figure 0-27. PCI-X Message Out Low (PCIX0_MSGOL)

31:0	MOL	Message Out Low	This holds an outbound message. If bit 0 is high, then an outbound interrupt is generated.

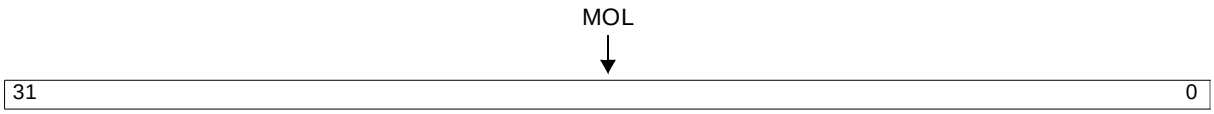


Figure 29-198. PCI-X Message Out Low (PCIX0_MSGOL)

31:0	MOL	Message Out Low	This holds an outbound message. If bit 0 is high, then an outbound interrupt is generated.

MMIO 0x2 0EC800C0 Read-Only (PLB), 0xC0 read-Only (PCI-X)

~~See "PCI-X MSI Capability Identifier (PCIX0_OMCAPID)" on page 18-93.~~See *PCI-X MSI Capability Identifier (PCIX0_OMCAPID)* on page 625.

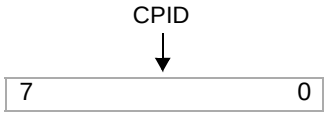


Figure 0-28. PCI-X MSI Capabilities Identifier (PCIX0_OMCAPID)

7:0	CPID	Message Signaled Interrupts (MSI) Capabilities Identifier

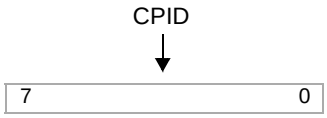


Figure 29-199. PCI-X MSI Capabilities Identifier (PCIX0_OMCAPID)

7:0	CPID	Message Signaled Interrupts (MSI) Capabilities Identifier

MMIO 0x2 0EC800C4 Read/Write (PLB), 0xC7-0xC4 Read/Write (PCI-X)

~~See "PCI-X Message Address Register (PCIX0_OMMA)" on page 18-96.~~See *PCI-X Message Address Register (PCIX0_OMMA)* on page 628.

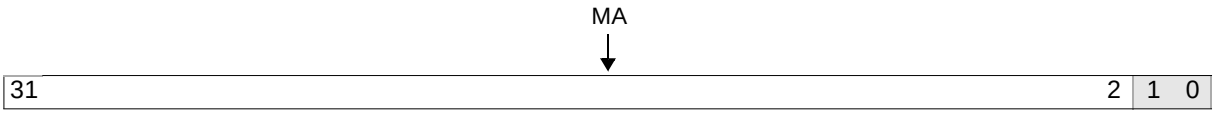


Figure 0-29. PCI-X Message Address Register (PCIX0_OMMA)

31:2	MA	Message Address	Indicates system specific message address. If the Message Enable bit (bit 0 of the Message Control Register) is set, the contents of this register specify the doubleword aligned address pci_ad[31:2] for the MSI memory write transaction. pci_ad[1:0] are driven to zero during the address phase. This field is read/write.
1:0		Reserved	Returns zero when read.

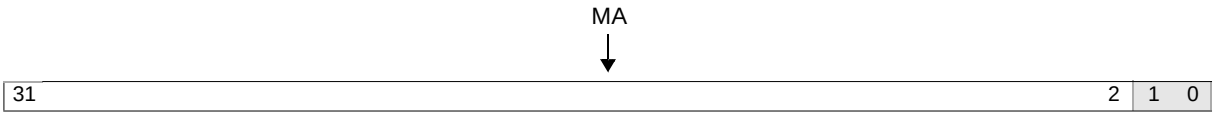


Figure 29-200. PCI-X Message Address Register (PCIX0_OMMA)

31:2	MA	Message Address	Indicates system specific message address. If the Message Enable bit (bit 0 of the Message Control Register) is set, the contents of this register specify the doubleword aligned address pci_ad[31:2] for the MSI memory write transaction. pci_ad[1:0] are driven to zero during the address phase. This field is read/write.
1:0		Reserved	Returns zero when read.

MMIO 0x2 0EC800C2 Read/Write (PLB), 0xC3-0xC2 Read/Write (PCI-X)

~~See "PCI-X Message Control Register (PCIX0_OMMC)" on page 18-95.~~See *PCI-X Message Control Register (PCIX0_OMMC)* on page 627.

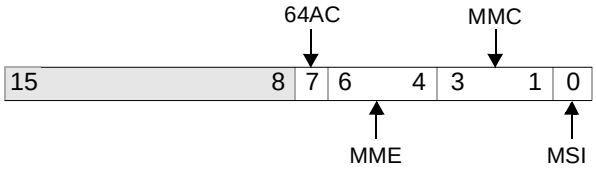


Figure 0-30. PCI-X Message Control Register (PCIX0_OMMC)

15:8		Reserved	Returns zero when read.
7	64AC	64-bit Address Capable	This bit is hardwired to a 1 and indicates that PLB-PCIX Bridge is capable of generating a 64-bit message address.
6:4	MME	Multiple Message Enable	System software writes to this field to indicate the number of allocated messages for PLB-PCIX Bridge. The number of allocated messages is aligned to a power of 2. Bit 6:5 of this field are hardwired to "0" and only bit 4 is writable. This field's state after reset is 000b.
3:1	MMC	Multiple Message Capable	System software reads this field to determine the number of messages requested by PLB-PCIX Bridge. The PLB-PCIX Bridge requests two messages and is indicated by hardwiring this field to a value of 001b.
0	MSI	MSI Enable 1 MSI enabled 0 MSI disabled	This read/write bit is used by the system to enable or disable MSI capability. This bit state after reset is 0 (MSI is disabled after reset).

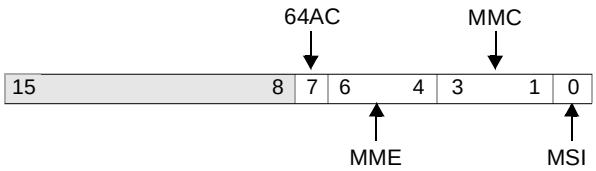


Figure 29-201. PCI-X Message Control Register (PCIX0_OMMC)

15:8		Reserved	Returns zero when read.
7	64AC	64-bit Address Capable	This bit is hardwired to a 1 and indicates that PLB-PCIX Bridge is capable of generating a 64-bit message address.

6:4	MME	Multiple Message Enable	System software writes to this field to indicate the number of allocated messages for PLB-PCIX Bridge. The number of allocated messages is aligned to a power of 2. Bit 6:5 of this field are hardwired to "0" and only bit 4 is writable. This field's state after reset is 000b.
3:1	MMC	Multiple Message Capable	System software reads this field to determine the number of messages requested by PLB-PCIX Bridge. The PLB-PCIX Bridge requests two messages and is indicated by hardwiring this field to a value of 001b.
0	MSI	MSI Enable 1 MSI enabled 0 MSI disabled	This read/write bit is used by the system to enable or disable MSI capability. This bit state after reset is 0 (MSI is disabled after reset).

MMIO 0x2 0EC800CC Read/Write (PLB), 0xCD-0xCC Read/Write (PCI-X)

~~See "PCI-X Message Data Register (PCIX0_OMMDATA)" on page 18-98.~~See *PCI-X Message Data Register (PCIX0_OMMDATA)* on page 630.

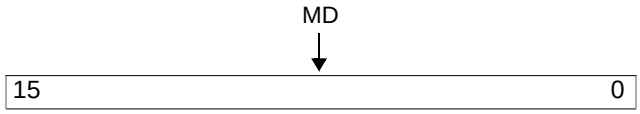


Figure 0-31. PCI-X Message Data Register (PCIX0_OMMDATA)

15:0	MD	Message Data	System-specified message. This field is written by the system. When PLB-PCIX Bridge issues an MSI message, it writes this value to the previously defined message address. If PLB-PCIX Bridge is allocated two messages by the system, the least significant bit of this field determines which message is being reported. If PLB-PCIX Bridge is allocated only one message by the system, then PLB-PCIX Bridge cannot modify this data on a MSI write.

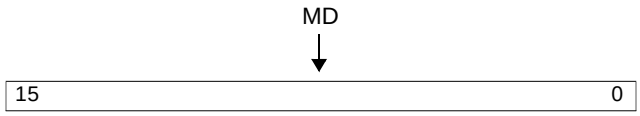


Figure 29-202. PCI-X Message Data Register (PCIX0_OMMDATA)

15:0	MD	Message Data	System-specified message. This field is written by the system. When PLB-PCIX Bridge issues an MSI message, it writes this value to the previously defined message address. If PLB-PCIX Bridge is allocated two messages by the system, the least significant bit of this field determines which message is being reported. If PLB-PCIX Bridge is allocated only one message by the system, then PLB-PCIX Bridge cannot modify this data on a MSI write.

MMIO 0x2 0EC800CE Read/Write (PLB), 0xCE Read/Write (PCI-X)

~~See "PCI-X Message End of Interrupt Register (PCIX0_OMMEOI)" on page 18-99.~~See *PCI-X Message End of Interrupt Register (PCIX0_OMMEOI)* on page 631.

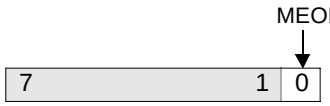


Figure 0-32. PCI-X Message End of Interrupt Register (PCIX0_OMMEOI)

7:1		Reserved	Returns zero when read.
0	MEOI	Message End of Interrupt	Following the generation of an outbound MSI, the outbound MSI unit must be rearmed before another outbound MSI can be generated. Writing a 1 to this bit rearms the MSI function. This bit automatically clears itself (always reads zero). EOI is required only when using MSI.

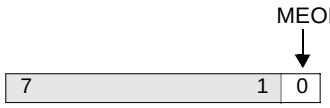


Figure 29-203. PCI-X Message End of Interrupt Register (PCIX0_OMMEOI)

7:1		Reserved	Returns zero when read.
0	MEOI	Message End of Interrupt	Following the generation of an outbound MSI, the outbound MSI unit must be rearmed before another outbound MSI can be generated. Writing a 1 to this bit rearms the MSI function. This bit automatically clears itself (always reads zero). EOI is required only when using MSI.

MMIO 0x2 0EC800C8 Read/Write (PLB), 0xCB-0XC8 Read/Write (PCI-X)

~~See "PCI-X Message Upper Address Register (PCIX0_OMMUA)" on page 18-97.~~See *PCI-X Message Upper Address Register (PCIX0_OMMUA)* on page 629.

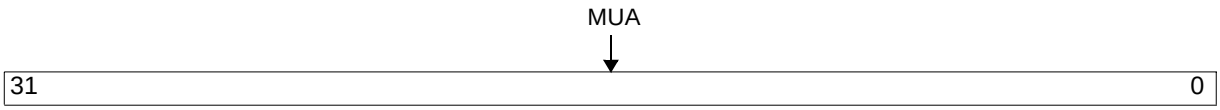


Figure 0-33. PCI-X Message Upper Address Register (PCIX0_OMMUA)

31:0	MUA	Message Upper Address	PLB-PCIX Bridge supports a 64-bit message address (bit 7 in Message Control Register is set to 1). The contents of this register specify the upper 32 bits of a 64-bit message address pci_ad[63:32]. If the contents of this register are zero, PLB-PCIX Bridge uses the 32-bit address specified by the Message Address Register. This field is read/write and is configured by the system.

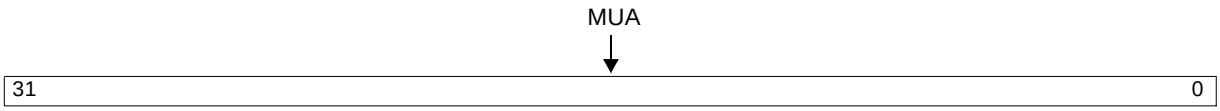


Figure 29-204. PCI-X Message Upper Address Register (PCIX0_OMMUA)

31:0	MUA	Message Upper Address	PLB-PCIX Bridge supports a 64-bit message address (bit 7 in Message Control Register is set to 1). The contents of this register specify the upper 32 bits of a 64-bit message address pci_ad[63:32]. If the contents of this register are zero, PLB-PCIX Bridge uses the 32-bit address specified by the Message Address Register. This field is read/write and is configured by the system.

MMIO 0x2 0EC800C1 Read/Write (PLB), 0xC1 Read/Write (PCI-X)

~~See "PCI-X Next Item Pointer (PCIX0_OMNIPTR)" on page 18-94.~~See *PCI-X Next Item Pointer (PCIX0_OMNIPTR)* on page 626.

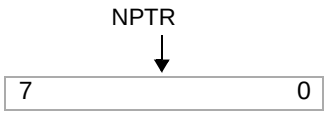


Figure 0-34. PCI-X MSI Next Item Pointer (PCIX0_OMNIPTR)

7:0	NPTR	Message Signaled Interrupts (MSI) Next Item Pointer

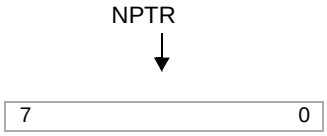


Figure 29-205. PCI-X MSI Next Item Pointer (PCIX0_OMNIPTR)

7:0	NPTR	Message Signaled Interrupts (MSI) Next Item Pointer

MMIO 0x2 0EC800DC Read-Only (PLB), 0xDC Read-Only (PCI-X)

~~See "PCI-X Capability Identifier (PCIX0_PCIXCAPID)" on page 18-107.~~See *PCI-X Capability Identifier (PCIX0_PCIXCAPID)* on page 638.

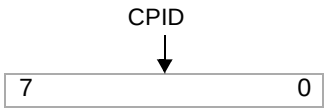


Figure 0-35. PCI-X Capability Identifier (PCIX0_PCIXCAPID)

7:0	CPID	Capability Identifier

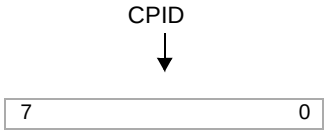


Figure 29-206. PCI-X Capability Identifier (PCIX0_PCIXCAPID)

7:0	CPID	Capability Identifier

MMIO 0x2 0EC800E8 Read-Only (PLB), 0xEB-0xE8 Read-Only (PCI-X)

~~See "PCI-X Internal Core Device ID Register (PCIX0_PCIXCID)" on page 18-113.~~See *PCI-X Internal Core Device ID Register (PCIX0_PCIXCID)* on page 645.

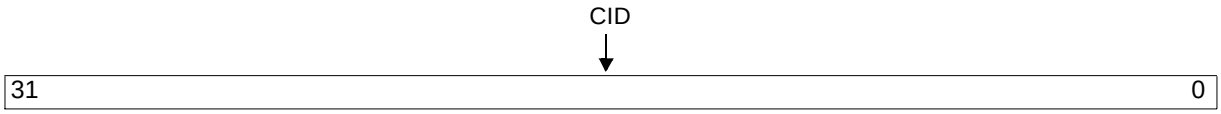


Figure 0-36. PCI-X Internal Core Device ID Register (PCIX0_PCIXCID)

31:0	CID	Internal Core Device ID	Read only.

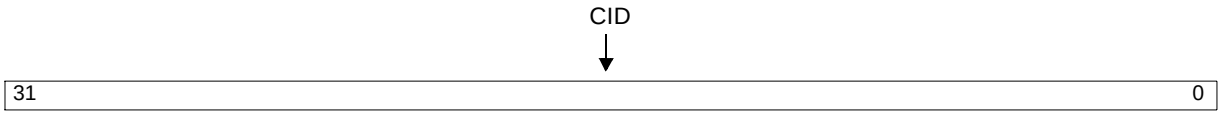


Figure 29-207. PCI-X Internal Core Device ID Register (PCIX0_PCIXCID)

31:0	CID	Internal Core Device ID	Read only.

MMIO 0x2 0EC800DE Read/Write (PLB), 0cDF-0xDE Read/Write (PCI-X)

~~See "PCI-X Command Register (PCIX0_PCIXCMD)" on page 18-109.~~See *PCI-X Command Register (PCIX0_PCIXCMD)* on page 641.

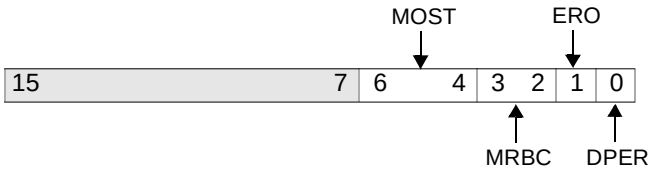


Figure 0-37. PCI-X Command Register (PCIX0_PCIXCMD)

15:7		Reserved	Returns zero when read.
6:4	MOST	Maximum Outstanding Split Transactions	This field indicates the number of outstanding split transactions allowed. If it is written to value less than the DMOS field of the PCIX0_PCIXSTS register, the PLB-PCIX Bridge limits the number of outstanding split transactions accordingly.
3:2	MRBC	Maximum Memory-Read Byte Count	This field defaults to 00 to indicate that the PLB-PCIX Bridge can request a maximum byte count of 512 bytes in a single memory read transaction. If the MRBC field of PCIX0_PCIXSTS register is not 0, this field can be programmed by the system to 01 or 10 to allow the PLB-PCIX Bridge to request 1K or 2K bytes, respectively. If it is programmed to 11, the PLB-PCIX Bridge will behave as if it is set to 10.
1	ERO	Enable Relaxed Ordering	This bit is hardwired to a 0, indicating that the PLB-PCIX Bridge never sets the Relaxed Ordering attribute bit.
0	DPER	Data Parity Error Recovery Enable	This bit is writable but has no effect.

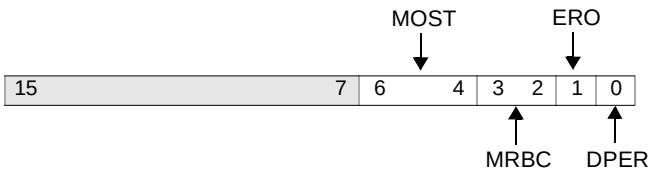


Figure 29-208. PCI-X Command Register (PCIX0_PCIXCMD)

15:7		Reserved	Returns zero when read.

6:4	MOST	Maximum Outstanding Split Transactions	This field indicates the number of outstanding split transactions allowed. If it is written to value less than the DMOS field of the PCIX0_PCIXSTS register, the PLB-PCIX Bridge limits the number of outstanding split transactions accordingly.
3:2	MRBC	Maximum Memory-Read Byte Count	This field defaults to 00 to indicate that the PLB-PCIX Bridge can request a maximum byte count of 512 bytes in a single memory read transaction. The system will not set this field higher than 00 because 512 bytes is the designed maximum memory read byte count.
1	ERO	Enable Relaxed Ordering	This bit is hardwired to a 0, indicating that the PLB-PCIX Bridge never sets the Relaxed Ordering attribute bit.
0	DPER	Data Parity Error Recovery Enable	This bit is writable but has no effect.

MMIO 0x2 0EC800E4 Read/Write (PLB), 0xE7-0xE4 Read/Write (PCI-X)

~~See "PCI-X Internal Debug Register (PCIX0_PCIXIDR)" on page 18-112.~~See *PCI-X Internal Debug Register (PCIX0_PCIXIDR)* on page 644.

Figure 0-38. PCI-X Internal Debug Register (PCIX0_PCIXIDR)

31:0		Reserved

Figure 29-209. PCI-X Internal Debug Register (PCIX0_PCIXIDR)

31:0		Reserved

MMIO 0x2 0EC800DD Read-Only (PLB), 0xDD Read-Only (PCI-X)

~~See "PCI-X Next Item Pointer Register (PCIX0_PCIXNIPTR)" on page 18-108.~~See *PCI-X Next Item Pointer Register (PCIX0_PCIXNIPTR)* on page 640.

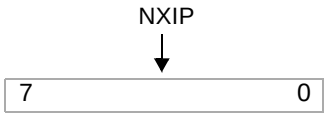


Figure 0-39. PCI-X Next Item Pointer Register (PCIX0_PCIXNIPTR)

7:0	NXIP	Next Item Pointer

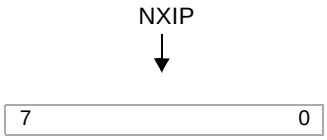


Figure 29-210. PCI-X Next Item Pointer Register (PCIX0_PCIXNIPTR)

7:0	NXIP	Next Item Pointer

MMIO 0x2 0EC800EC Read-Only (PLB), 0xEF-0xEC Read-Only (PCI-X)

~~See "PCI-X Internal Core Revision ID Register (PCIX0_PCIXRID)" on page 18-114.~~See *PCI-X Internal Core Revision ID Register (PCIX0_PCIXRID)* on page 646.

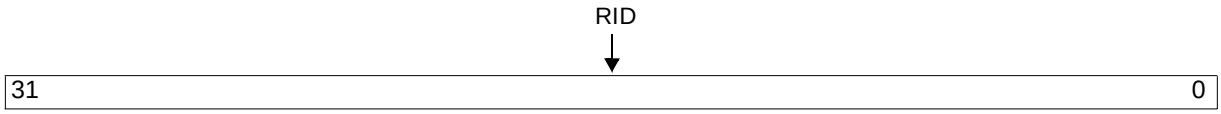


Figure 0-40. PCI-X Internal Core Revision ID Register (PCIX0_PCIXRID)

31:0	RID	Internal Core Revision ID	Read only.

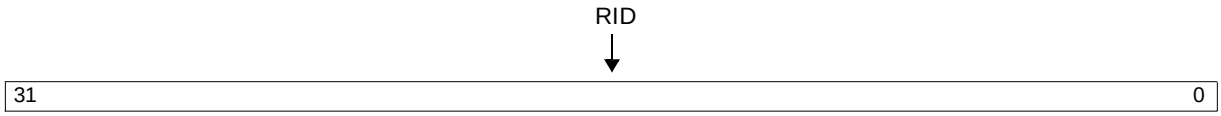


Figure 29-211. PCI-X Internal Core Revision ID Register (PCIX0_PCIXRID)

31:0	RID	Internal Core Revision ID	Read only.

MMIO 0x2 0EC800E0 Read/Write (PLB), 0xE3-0xE0 Read/Write (PCI-X)

~~See "PCI-X Status Register (PCIX0_PCIXSTS)" on page 18-110.~~See *PCI-X Status Register (PCIX0_PCIXSTS)* on page 642.

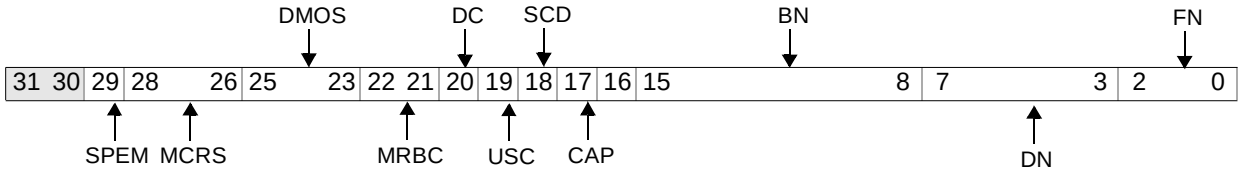


Figure 0-41. PCI-X Status Register (PCIX0_PCIXSTS)

31:30		Reserved	Returns zero when read.
29	SPEM	Received Split Completion Error Message	This bit is set if a split completion error message is received. Writing a value of 1 to this bit clears this bit.
28:26	MCRS	Designed Max Cumulative Read Size	This field is hardwired to 001b to indicate that the maximum cumulative outbound read size is less than or equal to 2 KB (it is 1.6 KB).
25:23	DMOS	Designed Max Outstanding Splits	This field indicates the maximum number of split transactions that the PLB-PCIX Bridge can have outstanding. Its reset value depends on the number of special purpose outbound read buffers present. The reset value is 010b, indicating that the PLB-PCIX Bridge can have three outstanding split completions.
22:21	MRBC	Designed Max Memory Read Byte Count	This field is hardwired to 00b to indicate that the maximum read byte count that the PLB-PCIX Bridge generates on outbound reads is 512 bytes.
20	DC	Device Complexity	This bit is hardwired to a 0 indicating that the PLB-PCIX Bridge is a simple device. (It does not handle LOCK# and never retries or disconnects split completions).
19	USC	Unexpected Split Completion	This bit is set if an unexpected split completion with Initiator Bus Number and Initiator Number of the PLB-PCIX Bridge is received. This bit is hardwired to 0, because this error is not checked.
18	SCD	Split Completion Discarded	This bit is set if the PLB-PCIX Bridge discards a split completion because the requestor would not accept it. This bit is hardwired to 0, because PLB-PCIX Bridge never discards a split completion.

17	CAP	133 MHz Capable	This bit indicates 133 MHz capability. This bit is hardwired to 1.
16	DEV	64-bit Device	This bit indicates the size of the data bus of the chip containing the PLB-PCIX Bridge. 1 means the bus is 64 bits. Its value is equal to the PCI initialize Req64 enable (PR64E) strapping bit.
15:8	BN	Bus Number	This field indicates the number of the bus segment for the PLB-PCIX Bridge containing this function. These bits are read/writable from the PLB side, but read only from the PCI side.
7:3	DN	Device Number	This field indicates the number of the device for the PLB-PCIX Bridge containing this function. These bits are read/writable from the PLB side, but read only from the PCI side.
2:0	FN	Function Number	This field indicates the number of the function for the PLB-PCIX Bridge containing this function. These bits are hardwired to 000b.

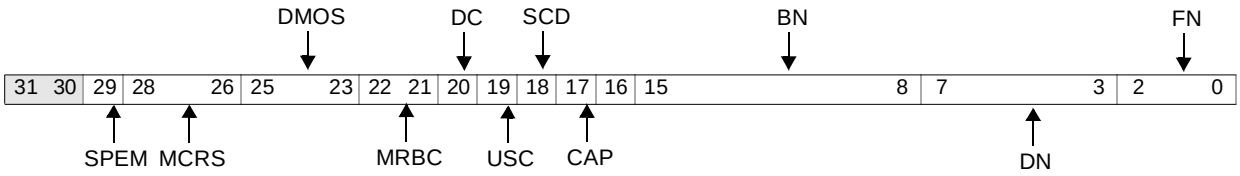


Figure 29-212. PCI-X Status Register (PCIX0_PCIXSTS)

31:30		Reserved	Returns zero when read.
29	SPEM	Received Split Completion Error Message	This bit is set if a split completion error message is received. Writing a value of 1 to this bit clears this bit.
28:26	MCRS	Designed Max Cumulative Read Size	This field is hardwired to 001b to indicate that the maximum cumulative outbound read size is less than or equal to 2 KB (it is 1.6 KB).
25:23	DMOS	Designed Max Outstanding Splits	This field indicates the maximum number of split transactions that the PLB-PCIX Bridge can have outstanding. Its reset value depends on the number of special purpose outbound read buffers present. The reset value is 010b, indicating that the PLB-PCIX Bridge can have three outstanding split completions.
22:21	MRBC	Designed Max Memory Read Byte Count	This field is hardwired to 00b to indicate that the maximum read byte count that the PLB-PCIX Bridge generates on outbound reads is 512 bytes.

20	DC	Device Complexity	This bit is hardwired to a 0 indicating that the PLB-PCIX Bridge is a simple device. (It does not handle LOCK# and never retries or disconnects split completions).
19	USC	Unexpected Split Completion	This bit is set if an unexpected split completion with Initiator Bus Number and Initiator Number of the PLB-PCIX Bridge is received. This bit is hardwired to 0, because this error is not checked.
18	SCD	Split Completion Discarded	This bit is set if the PLB-PCIX Bridge discards a split completion because the requestor would not accept it. This bit is hardwired to 0, because PLB-PCIX Bridge never discards a split completion.
17	CAP	133 MHz Capable	This bit indicates 133 MHz capability. This bit is hardwired to 1.
16	DEV	64-bit Device	This bit indicates the size of the data bus of the chip containing the PLB-PCIX Bridge. 1 means the bus is 64 bits. Its value is equal to the PCI initialize Req64 enable (PR64E) strapping bit.
15:8	BN	Bus Number	This field indicates the number of the bus segment for the PLB-PCIX Bridge containing this function. These bits are read/writable from the PLB side, but read only from the PCI side.
7:3	DN	Device Number	This field indicates the number of the device for the PLB-PCIX Bridge containing this function. These bits are read/writable from the PLB side, but read only from the PCI side.
2:0	FN	Function Number	This field indicates the number of the function for the PLB-PCIX Bridge containing this function. These bits are hardwired to 000b.

MMIO 0x2 0EC800A0 Read/Write (PLB), 0xA3-0xA0 Read/Write (PCI-X)

~~See "PCI-X PIM0 Local High Address (PCIX0_PIM0LAH)" on page 18-86.~~See *PCI-X PIM0 Local High Address (PCIX0_PIM0LAH)* on page 618.

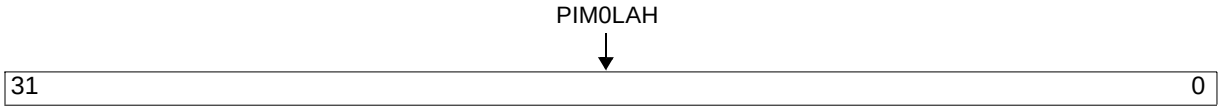


Figure 0-42. PCI-X PIM 0 Local High Address (PCIX0_PIM0LAH)

31:0	PIM0LAH	Local Address High	Defines the upper 32-bits of the starting address of range 0 in PLB space that is mapped to PCI memory. All bits are writable.

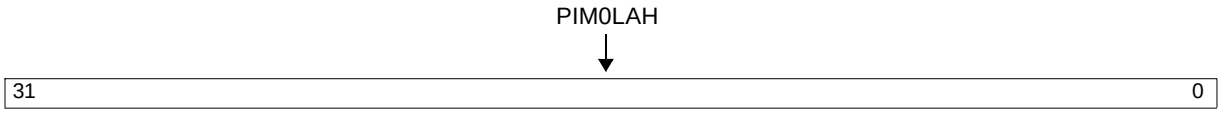


Figure 29-213. PCI-X PIM 0 Local High Address (PCIX0_PIM0LAH)

31:0	PIM0LAH	Local Address High	Defines the upper 32-bits of the starting address of range 0 in PLB space that is mapped to PCI memory.

MMIO 0x2 0EC8007C Read/Write (PLB), 0x9f-0x9C Read/Write (PCI-X)

~~See "PCI-X PIM0 Local Low Address (PCIX0_PIM0LAL)" on page 18-85.~~See *PCI-X PIM0 Local Low Address (PCIX0_PIM0LAL)* on page 616.

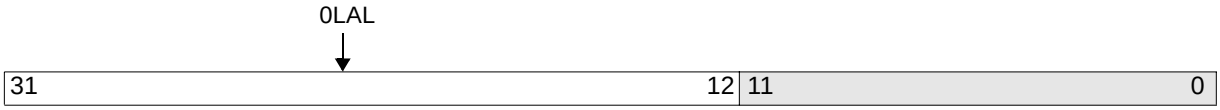


Figure 0-43. PIM 0 Local Low Address (PCIL0_PIM0LAL)

31:12	0LAL	Local Address Low	Defines the starting address of range 0 in PLB space that is mapped to PCI memory. Only the bits that are 1 in the PIM 0 Size/Attribute Register are actually passed to the PLB address.
11:0		Reserved	Returns zero when read.

Figure 29-214. PCI-X PIM 0 Local Low Address (PCIX0_PIM0LAL)

31:12	0LAL	Local Address Low	Defines the starting address of range 0 in PLB space that is mapped to PCI memory. Only the bits that are 1 in the PIM 0 Size/Attribute Register are actually passed to the PLB address.
11:0		Reserved	Returns zero when read.

MMIO 0x2 0EC80098 Read/Write (PLB), 0x9B-0x98 Read/Write (PCI-X)

~~See "PCI-X PIM0 Size/Attribute Register (PCIX0_PIM0SA)" on page 18-84.~~See *PCI-X PIM0 Size/Attribute Register (PCIX0_PIM0SA)* on page 616.

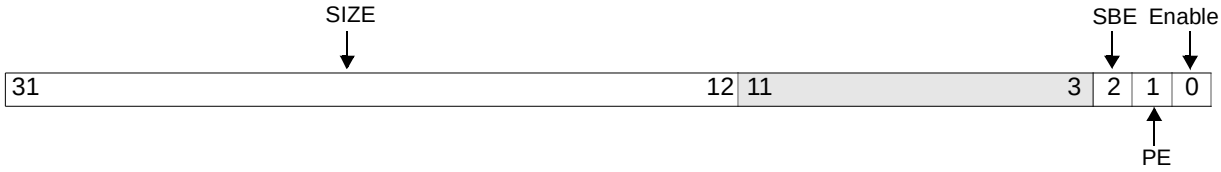


Figure 0-44. PCI-X PIM 0 Size/Attribute Register (PCIX0_PIM0SA)

31:12	SIZE	Size of PIM0	The size of the PIM0 can range from 4K bytes to 4GB and must be a power of 2. To determine the value to program in this field, use the following process: 1. Represent the desired size with a 32-bit number. Use only one bit set in that number since it is a power of two. 2. Set all the bits to the left of the set bit. 3. Store bits [31:12] of the resulting number in this field. For example, if the desired PIM0 size is 16M, start with 16x1024x1024 =0100_0000h. Set all the bits to the left of the set bit (FF00_0000h). Bits [31:12] of that number are FF00_0h; therefore, FF00_0h should be stored in this field.
11:3		Reserved	Always read as zero.
2	SBE	Split BAR0 Enable	This bit enables splitting the BAR0-specified region into two ranges. When this bit is low, the entire BAR0 region is mapped to one region of PLB space, as determined by PIM0 Local Address registers. When this bit is high, the BAR0 range is divided into two regions: the first 1 KB of the BAR0 range is mapped to either a PLB region as determined by the PIM1 Local Address registers, or to internal registers for using Simple Message Passing and/or Inbound MSI. The second region is the remainder of the BAR0 range and is mapped to the PLB address space as determined by the PIM0 Local Address Register. If this bit is low, the entire BAR0 address space is mapped to PLB address space as determined by the PIM0 Local Address Register. This bit is 0 after reset.
1	PE	Prefetch Enable	This bit determines whether this region is prefetchable. The value of this bit is also readable in BAR0 low, bit 3.
0	Enable	Enable	If set, enables the PCI BAR0 registers and decoders.

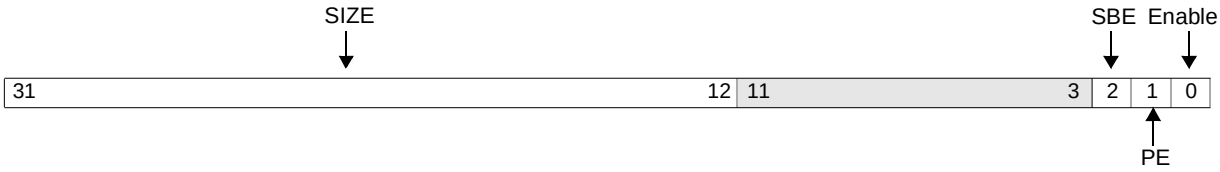


Figure 29-215. PCI-X PIM 0 Size/Attribute Register (PCIX0_PIM0SA)

31:12	SIZE	Size of PIM0	The size of the PIM0 can range from 4K bytes to 4GB and must be a power of 2. To determine the value to program in this field, use the following process: 1. Represent the desired size with a 32-bit number. Use only one bit set in that number since it is a power of two. 2. Set all the bits to the left of the set bit. 3. Store bits [31:12] of the resulting number in this field. For example, if the desired PIM0 size is 16M, start with 16x1024x1024 =0100_0000h. Set all the bits to the left of the set bit (FF00_0000h). Bits [31:12] of that number are FF00_0h; therefore, FF00_0h should be stored in this field.
11:3		Reserved	Always read as zero.
2	SBE	Split BAR0 Enable	This bit enables splitting the BAR0-specified region into two ranges. When this bit is low, the entire BAR0 region is mapped to one region of PLB space, as determined by PIM0 Local Address registers. When this bit is high, the BAR0 range is divided into two regions: the first 1 KB of the BAR0 range is mapped to either a PLB region as determined by the PIM1 Local Address registers, or to internal registers for using Simple Message Passing and/or Inbound MSI. The second region is the remainder of the BAR0 range and is mapped to the PLB address space as determined by the PIM0 Local Address Register. If this bit is low, the entire BAR0 address space is mapped to PLB address space as determined by the PIM0 Local Address Register. This bit is 0 after reset.
1	PE	Prefetch Enable	This bit determines whether this region is prefetchable. The value of this bit is also readable in BAR0 low, bit 3.
0	Enable	Enable	If set, enables the PCI BAR0 registers and decoders.

MMIO 0x2 0EC800AC Read/Write (PLB), 0xAF-0xAC Read/Write (PCI-X)

~~See "PCI-X PIM0 Local High Address (PCIX0_PIM0LAH)" on page 18-86.~~See *PCI-X PIM0 Local High Address (PCIX0_PIM0LAH)* on page 618.

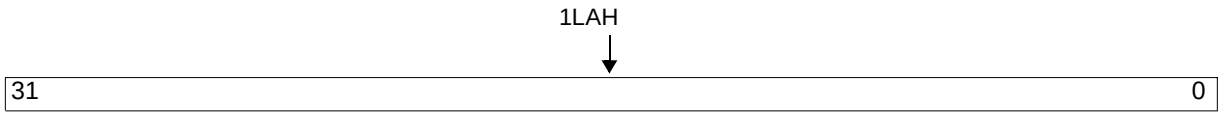


Figure 0-45. PCI-X PIM 1 Local High Address (PCIX0_PIM1LAH)

31:0	1LAH	Local Address High	Defines the upper 32 bits of the starting address of range 1 in PLB space that is mapped to PCI memory. All bits are writable.

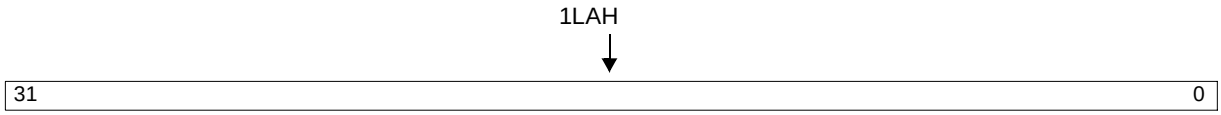


Figure 29-216. PCI-X PIM 1 Local High Address (PCIX0_PIM1LAH)

31:0	1LAH	Local Address High	Defines the upper 32 bits of the starting address of range 1 in PLB space that is mapped to PCI memory.

MMIO 0x2 0EC800A8 Read/Write (PLB), 0xAB-0xA8 Read/Write (PCI-X)

~~See "PCI-X PIM1 Local Low Address (PCIX0_PIM1LAL)" on page 18-88.~~See *PCI-X PIM1 Local Low Address (PCIX0_PIM1LAL)* on page 620.

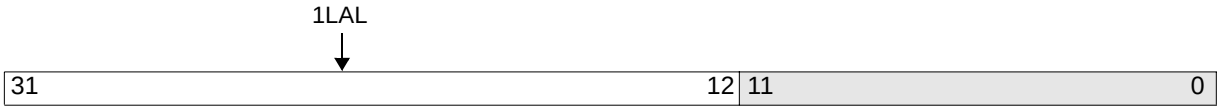


Figure 0-46. PCI-X PIM 1 Local Low Address (PCIX0_PIM1LAL)

31:12	1LAL	Local Address Low	Defines the starting address of range 1 in PLB space that is mapped to PCI memory. Only the bits that are "1" in the PIM 1 Size/Attribute Register are actually passed to the PLB address. Only bits 31:12 are writable.
11:0		Reserved	Returns zero when read.

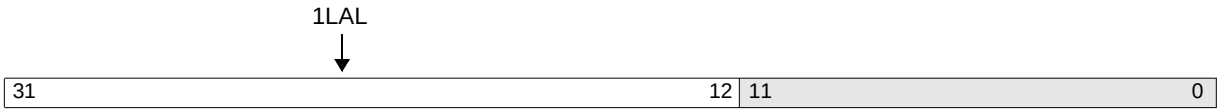


Figure 29-217. PCI-X PIM 1 Local Low Address (PCIX0_PIM1LAL)

31:12	1LAL	Local Address Low	Defines the starting address of range 1 in PLB space that is mapped to PCI memory. Only the bits that are "1" in the PIM 1 Size/Attribute Register are actually passed to the PLB address. Only bits 31:12 are writable.
11:0		Reserved	Returns zero when read.

MMIO 0x2 0EC800A4 Read/Write (PLB), 0xA7-0xA4 Read/Write (PCI-X)

~~See "PCI-X PIM1 Size/Attribute Register (PCIX0_PIM1SA)" on page 18-87.~~See *PCI-X PIM1 Size/Attribute Register (PCIX0_PIM1SA)* on page 619.

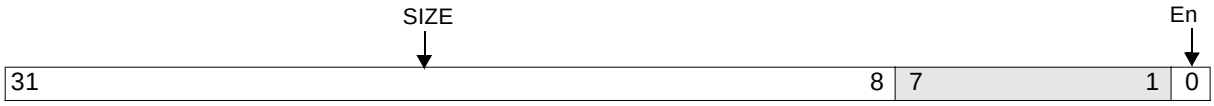


Figure 0-47. PCI-X PIM 1 Size/Attribute Register (PCIX0_PIM1SA)

31:8	SIZE	Size	SIZE defines the size of the region of PCI I/O space that is mapped to local (PLB) space through PIM 1. The size for the PIM1 is hardcoded to 256 bytes; therefore, read to these bits returns all ones. This size has no effect on the PIM0 1 KB region.
7:1		Reserved	Returns zero when read.
0	En	Enable	If set, enables the PCI BAR1 registers and decoder.

Figure 29-218. PCI-X PIM 1 Size/Attribute Register (PCIX0_PIM1SA)

31:8	SIZE	Size	SIZE defines the size of the region of PCI I/O space that is mapped to local (PLB) space through PIM 1. The size for the PIM1 is hardcoded to 256 bytes; therefore, read to these bits returns all ones. This size has no effect on the PIM0 1 KB region.
7:1		Reserved	Returns zero when read.
0	En	Enable	If set, enables the PCI BAR1 registers and decoder.

MMIO 0x2 0EC800B8 Read/Write (PLB), 0xBB-0xB8 Read/Write (PCI-X)

~~See "PCI-X PIM2 Local Address High (PCIX0_PIM2LAH)" on page 18-92.~~See *PCI-X PIM2 Local Address High (PCIX0_PIM2LAH)* on page 624.

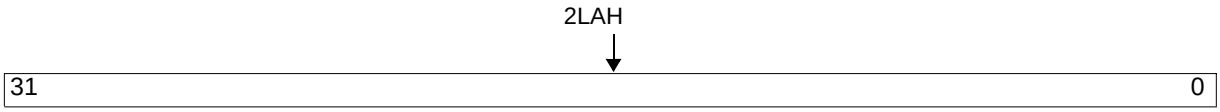


Figure 0-48. PCI-X PIM 2 Local Address High (PCIX0_PIM2LAH)

31:0	2LAH	Local Address High	Defines the upper 32 bits of the starting address of range 2 in PLB space that is mapped to PCI memory. All bits are writable.

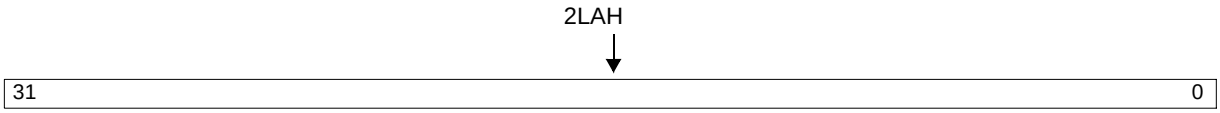


Figure 29-219. PCI-X PIM 2 Local Address High (PCIX0_PIM2LAH)

31:0	2LAH	Local Address High	Defines the upper 32 bits of the starting address of range 2 in PLB space that is mapped to PCI memory.

MMIO 0x2 0EC800B4 Read/Write (PLB), 0xB7-0xB4 Read/Write (PCI-X)

~~See "PCI-X PIM2 Local Low Address (PCIX0_PIM2LAL)" on page 18-91.~~See *PCI-X PIM2 Local Low Address (PCIX0_PIM2LAL)* on page 623.

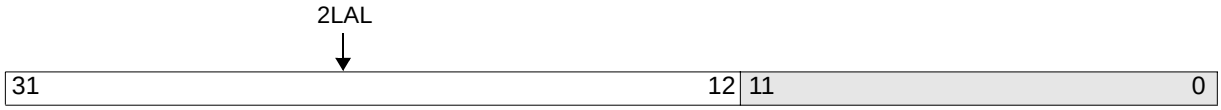


Figure 0-49. PCI-X PIM 2 Local Low Address (PCIX0_PIM2LAL)

31:12	2LAL	Local Address Low	Defines the starting address of range 2 in PLB space that is mapped to PCI memory. Only the bits that are 1 in the PIM 2 Size/Attribute Register are actually passed to the PLB address. Only bits 31:12 are writable.
11:0		Reserved	Returns zero when read.

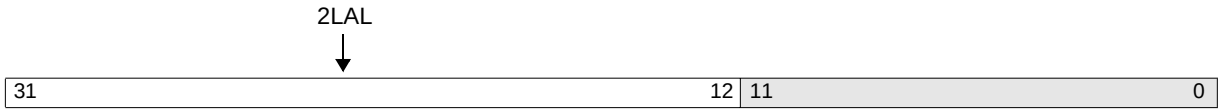


Figure 29-220. PCI-X PIM 2 Local Low Address (PCIX0_PIM2LAL)

31:12	2LAL	Local Address Low	Defines the starting address of range 2 in PLB space that is mapped to PCI memory. Only the bits that are 1 in the PIM 2 Size/Attribute Register are actually passed to the PLB address. Only bits 31:12 are writable.
11:0		Reserved	Returns zero when read.

MMIO 0x2 0EC800B0 Read/Write (PLB), 0xB3-0xB0 Read/Write (PCI-X)

~~See "PCI-X PIM2 Size/Attribute Register (PCIX0_PIM2SA)" on page 18-90.~~See *PCI-X PIM2 Size/Attribute Register (PCIX0_PIM2SA)* on page 622.

Figure 0-50. PCI-XPIM 2 Size/Attribute Register (PCIX0_PIM2SA)

31:12	SIZE	Size of PIM2	The size of the PIM2 can range from 4K to 4G and must be a power of 2. To determine the value to program in this field, use the following process: 1. Represent the desired size with a 32-bit number. Use only one bit set in that number since it is a power of two. 2 Set all the bits to the left of the set bit. 3. Store bits [31:12] of the resulting number in this field. For example, if the desired PIM0 size is 16M, start with 16x1024x1024 = 0100_0000h. Set all the bits to the left of the set bit (FF00_0000h). Bits [31:12] of that number are FF00_0h; therefore, FF00_0h should be stored in this field.
11:3		Reserved	Returns zero when read.
2	PIM2E	Big PIM2 Enable	This bit causes PIM2 to have a fixed map. When enabled, the map is PCI 8000_0000_0000_0000 to FFFF_FFFF_FFFF_FFFFh is mapped to PLB 8000_0000_0000_0000 to FFFF_FFFF_FFFF_FFFFh (no translation). When enabled, BAR2, PIM2 Size, and PIM2 Local Address are ignored (don't cares). The bit is 0 after reset.
1	PE	Prefetch Enable	This bit determines if this region is prefetchable. The value of this bit is also readable in BAR2 low, bit 3.
0	Enable	Enable	If set, enables the PCI PIM2 BAR Register and the Expansion ROM BAR Register and the decoder.

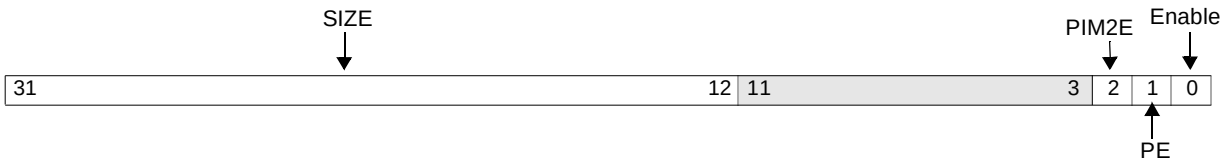


Figure 29-221. PCI-XPIM 2 Size/Attribute Register (PCIX0_PIM2SA)

31:12	SIZE	Size of PIM2	The size of the PIM2 can range from 4K to 4G and must be a power of 2. To determine the value to program in this field, use the following process: 1. Represent the desired size with a 32-bit number. Use only one bit set in that number since it is a power of two. 2 Set all the bits to the left of the set bit. 3. Store bits [31:12] of the resulting number in this field. For example, if the desired PIM0 size is 16M, start with 16x1024x1024 = 0100_0000h. Set all the bits to the left of the set bit (FF00_0000h). Bits [31:12] of that number are FF00_0h; therefore, FF00_0h should be stored in this field.
11:3		Reserved	Returns zero when read.
2	PIM2E	Big PIM2 Enable	This bit causes PIM2 to have a fixed map. When enabled, the map is PCI 8000_0000_0000_0000 to FFFF_FFFF_FFFF_FFFFh is mapped to PLB 8000_0000_0000_0000 to FFFF_FFFF_FFFF_FFFFh (no translation). When enabled, BAR2, PIM2 Size, and PIM2 Local Address are ignored (don't cares). The bit is 0 after reset.
1	PE	Prefetch Enable	This bit determines if this region is prefetchable. The value of this bit is also readable in BAR2 low, bit 3.
0	Enable	Enable	If set, enables the PCI PIM2 BAR Register and the Expansion ROM BAR Register and the decoder.

MMIO 0x2 0EC80060 Read-Only (PLB), 0x63-0x60 Read-Only (PCI-X)

~~See "PCI-X PLB Slave Error Address High (PCIX0_PLBEARH)" on page 18-72.~~See *PCI-X PLB Slave Error Address High (PCIX0_PLBEARH)* on page 604.

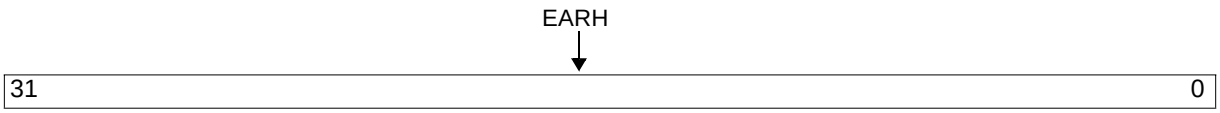


Figure 0-51. PCI-X PLB Slave Error Address High (PCIX0_PLBEARH)

31:0	EARH	PLB Slave Error Address High

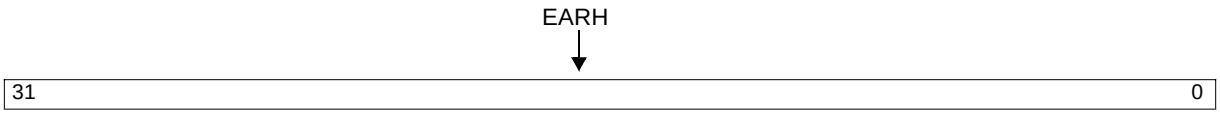


Figure 29-222. PCI-X PLB Slave Error Address High (PCIX0_PLBEARH)

31:0	EARH	PLB Slave Error Address High

MMIO 0x2 0EC8005C Read-Only PLB, 0x5f-0x5C Read-Only (PCI-X)

~~See "PCI-X PLB Slave Error Address Low (PCIX0_PLBBEARL)" on page 18-71.~~See *PCI-X PLB Slave Error Address Low (PCIX0_PLBBEARL)* on page 603.

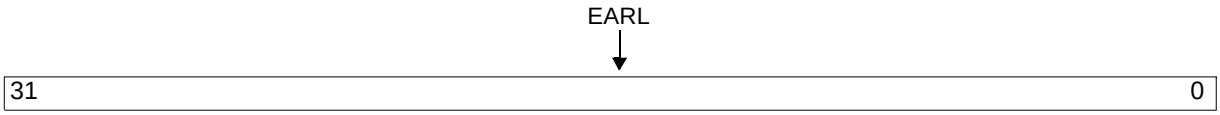


Figure 0-52. PCI-X PLB Slave Error Address Low (PCIX0_PLBEARL)

31:0	EARL	PLB Slave Error Address Low

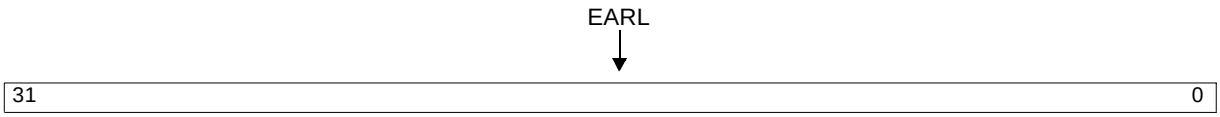


Figure 29-223. PCI-X PLB Slave Error Address Low (PCIX0_PLBEARL)

31:0	EARL	PLB Slave Error Address Low

MMIO 0x2 0EC80058 Read-Only (PLB), 0x5b-0x58 (PCI-X)

~~See "PCI-X PLB Slave Error Attribute Register (PCIX0_PLBBESR)" on page 18-70.~~See *PCI-X PLB Slave Error Attribute Register (PCIX0_PLBBESR)* on page 602.

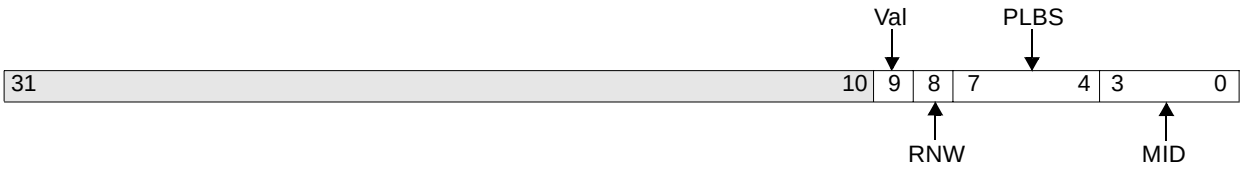


Figure 0-53. PCI-X PLB Slave Error Attribute Register (PCIX0_PLBBESR)

31:10		Reserved	Always read as zero.
9	Val	Valid	Indicates if bits 8:0 are valid (This bit is only meaningful when an outbound error is indicated in the PCIX0_ERRSTS register). **Note:** It is possible for a parity error to be detected too late for the PLB information to be saved.
8	RNW	RNW	RNW from the PLB master
7:4	PLBS	size	Size from the PLB master
3:0	MID	masterID	MasterID of the PLB master

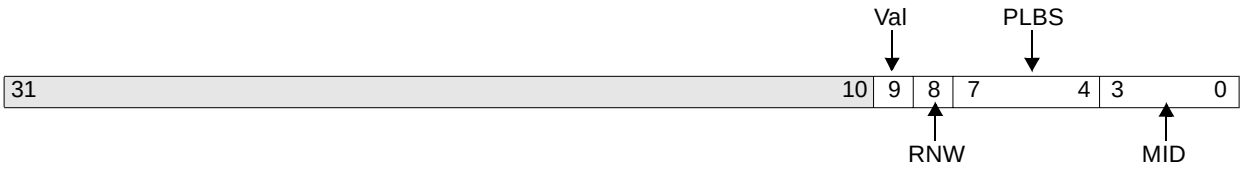


Figure 29-224. PCI-X PLB Slave Error Attribute Register (PCIX0_PLBBESR)

31:10		Reserved	Always read as zero.
9	Val	Valid	Indicates if bits 8:0 are valid (This bit is only meaningful when an outbound error is indicated in the PCIX0_ERRSTS register). **Note:** It is possible for a parity error to be detected too late for the PLB information to be saved.
8	RNW	RNW	RNW from the PLB master
7:4	PLBS	size	Size from the PLB master
3:0	MID	masterID	MasterID of the PLB master

MMIO 0x2 0EC800D2 Read-Only (PLB), 0xD3-0xD2 Read-Only (PCI-X)

~~See "PCI-X Power Management Capabilities Register (PCIX0_PMC)" on page 18-102.~~See *PCI-X Power Management Capabilities Register (PCIX0_PMC)* on page 634.

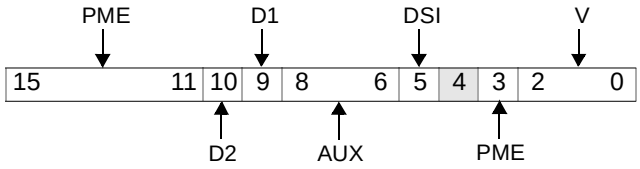


Figure 0-54. PCI-X Power Management Capabilities Register (PCIX0_PMC)

15:11	PME	PME_Support	The PLB-PCIXBridge does not support PME#; therefore, all the bits are hardwired to 0.
10	D2	D2_Support	This bit determines if the D2 Power Management State is supported. PLB-PCIX Bridge does not support the D2 Power Management State; therefore, this bit is hardwired to a 0.
9	D1	D1_Support	This bit determines if the D1 Power Management State is supported. PLB-PCIX Bridge supports the D1 Power Management State; therefore, this bit is hardwired to 1.
8:6	AUX	Aux_Current	PLB-PCIX Bridge does not support PME#; therefore, this field is unsupported. All the bits are hardwired to 0.
5	DSI	Device Specific Initialization (DSI)	The DSI bit indicates whether special initialization of this function is required (beyond the standard PCI configuration header) before the generic class device driver is able to use it. This bit is hardwired to 0.
4		Reserved	Returns zero when read.
3	PME	PME Clock	This bit is hardwired to 0 indicating that the function does not support PME# generation in any state.
2:0	V	Version	Returns a value of 010b on reads, indicating that this function complies with the *PCI Bus Power Management Interface Specification,* Version 1.1.

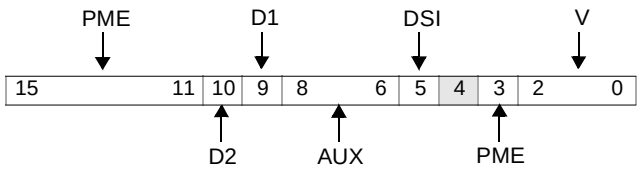


Figure 29-225. PCI-X Power Management Capabilities Register (PCIX0_PMC)

15:11	PME	PME_Support	The PLB-PCIX Bridge does not support PME#; therefore, all the bits are hardwired to 0.

10	D2	D2_Support	This bit determines if the D2 Power Management State is supported. PLB-PCIX Bridge does not support the D2 Power Management State; therefore, this bit is hardwired to a 0.
9	D1	D1_Support	This bit determines if the D1 Power Management State is supported. PLB-PCIX Bridge supports the D1 Power Management State; therefore, this bitis hardwired to 1.
8:6	AUX	Aux_Current	PLB-PCIX Bridge does not support PME#; therefore, this field is unsupported. All the bits are hardwired to 0.
5	DSI	Device Specific Initialization (DSI)	The DSI bit indicates whether special initialization of this function is required (beyond the standard PCI configuration header) before the generic class device driver is able to use it. This bit is hardwired to 0.
4		Reserved	Returns zero when read.
3	PME	PME Clock	This bit is hardwired to 0 indicating that the function does not support PME# generation in any state.
2:0	V	Version	Returns a value of 010b on reads, indicating that this function complies with the *PCI Bus Power Management Interface Specification,* Version 1.1.

MMIO 0x2 0EC800D0 Read/Write (PLB), 0xD0 Read-Only (PCI-X)

~~See "PCI-X Power Management Capability Identifier (PCIX0_PMCAPID)" on page 18-100.~~See *PCI-X Power Management Capability Identifier (PCIX0_PMCAPID)* on page 632.

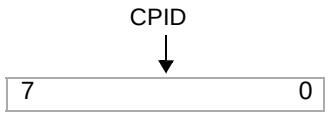


Figure 0-55. PCI-X Power Management Capabilities Identifier (PCIX0_PMCAPID)

7:0	CPID	Capabilities Identifier	When read by system software as 01h indicates that PLB-PCIX Bridge supports power management and the data structure currently being pointed to is the PCI power management capability structure.

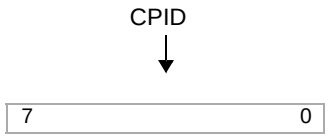


Figure 29-226. PCI-X Power Management Capabilities Identifier (PCIX0_PMCAPID)

7:0	CPID	Capabilities Identifier	When read by system software as 01h indicates that PLB-PCIX Bridge supports power management and the data structure currently being pointed to is the PCI power management capability structure.

MMIO 0x2 0EC800D6 Read-Only (PLB), 0xD6 Read-Only (PCI-X)

~~See "PCI-X PMCSR PCI-to-PLB-PCIX Bridge Support Extensions (PCIX0_PMCSRBSE)" on page 18-104.~~See *PCI-X PMCSR PCI-to-PLB-PCIX Bridge Support Extensions (PCIX0_PMCSRBSE)* on page 636.

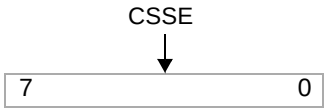


Figure 0-56. PCI-X PMCSR PCI-to-PLB-PCIX Bridge Support Extensions (PCIX0_PMCSRBSE)

7:0	CSSE	Power Management Control/Status PCI-to-PLB-PCIX Bridge Support Extensions.	Required for all PCI-to-PCI bridges. Always read as zero.

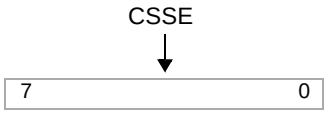


Figure 29-227. PCI-X PMCSR PCI-to-PLB-PCIX Bridge Support Extensions (PCIX0_PMCSRBSE)

7:0	CSSE	Power Management Control/Status PCI-to-PLB-PCIX Bridge Support Extensions.	Required for all PCI-to-PCI bridges. Always read as zero.

MMIO 0x2 0EC800D4 Read/Write (PLB), 0xD5-0xD4 Read/Write (PCI-X)

~~See "PCI-X Power Management Control/Status Register (PCIX0_PMCSR)" on page 18-103.~~See *PCI-X Power Management Control/Status Register (PCIX0_PMCSR)* on page 635.

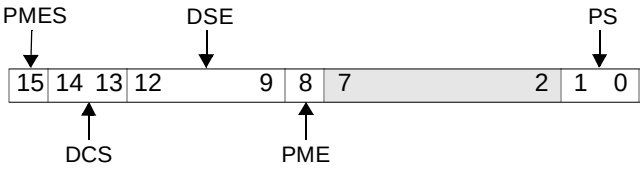


Figure 0-57. PCI-X Power Management Control/Status Register (PCIX0_PMCSR)

15	PMES	PME_Status	The PLB-PCIX Bridge does not support PME#; therefore, this bit is hardwired to a 0.
14:13	DSC	Data_Scale	The PLB-PCIX Bridge does not support the Data Register; therefore, these bits are hardwired to 00.
12:9	DSE	Data_Select	The PLB-PCIX Bridge does not support the Data Register; therefore, these bits are hardwired to 0000.
8	PME	PME_En	PLB-PCIX Bridge does not support PME# generation; therefore, this bit is hardwired to a 0.
7:2		Reserved	Retunrs zero when read.
1:0	PS	PowerState 00b - D0 01b - D1 10b - D2 11b - D3-Hot	This 2-bit R/W field is used both to determine the current power state of a function and to set the function into a new power state. If software attempts to write an unsupported, optional state to this field, the value of this field does not change. Writing this register may cause the PMSC_RR to change.

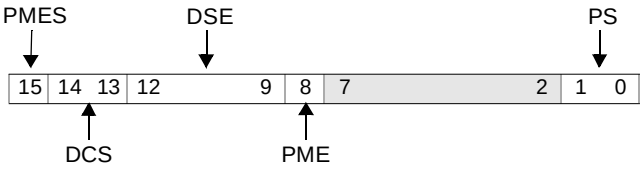


Figure 29-228. PCI-X Power Management Control/Status Register (PCIX0_PMCSR)

15	PMES	PME_Status	The PLB-PCIX Bridge does not support PME#; therefore, this bit is hardwired to a 0.
14:13	DSC	Data_Scale	The PLB-PCIX Bridge does not support the Data Register; therefore, these bits are hardwired to 00.

12:9	DSE	Data_Select	The PLB-PCIX Bridge does not support the Data Register; therefore, these bits are hardwired to 0000.
8	PME	PME_En	PLB-PCIX Bridge does not support PME# generation; therefore, this bit is hardwired to a 0.
7:2		Reserved	Retunrs zero when read.
1:0	PS	PowerState 00b - D0 01b - D1 10b - D2 11b - D3-Hot	This 2-bit R/W field is used both to determine the current power state of a function and to set the function into a new power state. If software attempts to write an unsupported, optional state to this field, the value of this field does not change. Writing this register may cause the PMSC_RR to change.

MMIO 0x2 0EC800D7 Read-Only (PLB), 0xD7 Read-Only (PCI-X)

~~See "PCI-X Power Management Data (PCIX0_PMDATA)" on page 18-105.~~See *PCI-X Power Management Data (PCIX0_PMDATA)* on page 637.

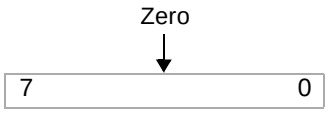


Figure 0-58. PCI-X Power Management Data (PCIX0_PMDATA)

7:0	Zero	Optional register	Returns zero when read.

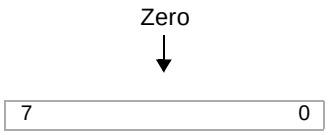


Figure 29-229. PCI-X Power Management Data (PCIX0_PMDATA)

7:0	Zero	Optional register	Returns zero when read.

MMIO 0x2 0EC800D1 Read-Write (PLB), 0xD1 Read-Only (PCI-X)

~~See "PCI-X Power Management Next Item Pointer (PCIX0_PMNIPTR)" on page 18-101.~~See *PCI-X Power Management Next Item Pointer (PCIX0_PMNIPTR)* on page 633.

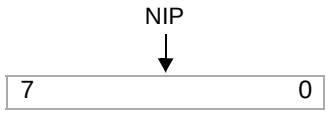


Figure 0-59. PCI-X Power Management Next Item Pointer (PCIX0_PMNIPTR)

7:0	NIP	Next Item Pointer	Describes the location of the next item in the function's capability list. The Next Item Pointer defaults to pointing to the capability structure for PCI-X located at DCh in the configuration space. This can be overwritten with 00h if support for PCI-X is not desired.

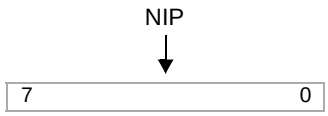


Figure 29-230. PCI-X Power Management Next Item Pointer (PCIX0_PMNIPTR)

7:0	NIP	Next Item Pointer	Describes the location of the next item in the function's capability list. The Next Item Pointer defaults to pointing to the capability structure for PCI-X located at DCh in the configuration space. This can be overwritten with 00h if support for PCI-X is not desired.

MMIO 0x2 0EC800D8 Read/Write (PLB), 0xD8 Read/Write (PCI-X)

~~See "PCI-X Power Management State Change Request Register (PCIX0_PMSCRR)" on page 18-106.~~See *PCI-X Power Management State Change Request Register (PCIX0_PMSCRR)* on page 638.

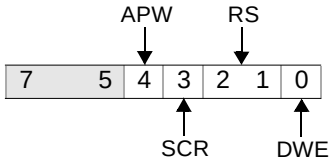


Figure 0-60. PCI-X Power Management State Change Request Register (PCIX0_PMSCRR)

7:5		Reserved	Returns zero when read.
4	APW	Accept PMCSR Write	The local processor sets this bit when it is ready to change the power management state. The bit is cleared when the host configuration write to the PMCSR is accepted. This bit is always a 1 if the Delayed Write Enable bit (bit 0) is a 0. (It is possible for the local processor to write a 0 to this bit).
3	SCR	State Change Request	The PLB-PCIX Bridge sets this bit when there is a host write to the PMCS for a power management state change request. This drives an interrupt to the local processor informing it of a state change request. The local processor must simultaneously clear this bit and set the Accept PMCSR Write bit when it is ready to change the state. After clearing this bit, new request will not be detected until the outstanding delayed write is accepted. (It is possible for the local processor to write a 1 to this bit.)
2:1	RS	Requested State	This field indicates the new power management state requested using the delayed host write to the PMCSR.
0	DWE	Delayed Write Enable 1 - Delayed write 0 - Immediate write	When 1, any configuration write to the PMCSR is completed as a delayed write. All writes to PMCSR are retried until the local processor sets the Accept PMCSR Write bit (bit 4). When 0, any configuration write to the PMCSR is completed immediately. This bit is a don't care if a host write to PMCSR requests a state change from D3hot to D0. The default state is 0 to prevent bus hang if the PM_Int service routine is not ready to go.

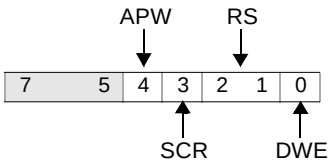


Figure 29-231. PCI-X Power Management State Change Request Register (PCIX0_PMSCRR)

7:5		Reserved	Returns zero when read.
4	APW	Accept PMCSR Write	The local processor sets this bit when it is ready to change the power management state. The bit is cleared when the host configuration write to the PMCSR is accepted. This bit is always a 1 if the Delayed Write Enable bit (bit 0) is a 0. (It is possible for the local processor to write a 0 to this bit).
3	SCR	State Change Request	The PLB-PCIX Bridge sets this bit when there is a host write to the PMCS for a power management state change request. This drives an interrupt to the local processor informing it of a state change request. The local processor must simultaneously clear this bit and set the Accept PMCSR Write bit when it is ready to change the state. After clearing this bit, new request will not be detected until the outstanding delayed write is accepted. (It is possible for the local processor to write a 1 to this bit.)
2:1	RS	Requested State	This field indicates the new power management state requested using the delayed host write to the PMCSR.
0	DWE	Delayed Write Enable 1 - Delayed write 0 - Immediate write	When 1, any configuration write to the PMCSR is completed as a delayed write. All writes to PMCSR are retried until the local processor sets the Accept PMCSR Write bit (bit 4). When 0, any configuration write to the PMCSR is completed immediately. This bit is a don't care if a host write to PMCSR requests a state change from D3hot to D0. The default state is 0 to prevent bus hang if the PM_Int service routine is not ready to go.

MMIO 0x2 0EC8006C Read/Write (PLB), 0x6F-0x6C Read/Write (PCI-X)

~~See "PCI-X POM 0 Local High Address (PCIX0_POM0LAH)" on page 18-74.~~See *PCI-X POM 0 Local High Address (PCIX0_POM0LAH)* on page 606.

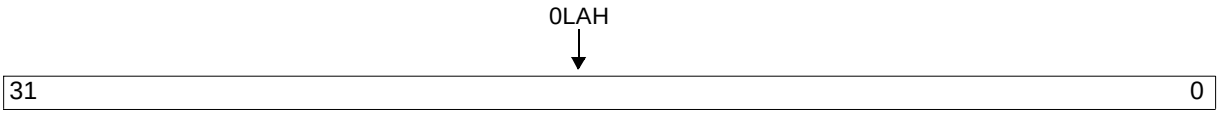


Figure 0-61. PCI-X POM 0 Local High Address (PCIX0_POM0LAH)

31:0	0LAH	Local Address High	Defines the upper 32 bits of the starting address of range 0 in PLB space that is mapped to PCI memory. All bits are writable.

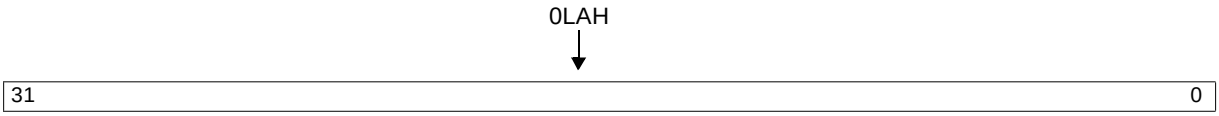


Figure 29-232. PCI-X POM 0 Local High Address (PCIX0_POM0LAH)

31:0	0LAH	Local Address High	Defines the upper 32 bits of the starting address of range 0 in PLB space that is mapped to PCI memory. All bits are writable.

MMIO 0x2 0EC80068 Read/Write (PLB), 0x6B-0x68 Read/Write (PCI-X)

~~See "PCI-X POM 0 Local Low Address (PCIX0_POM0LAL)" on page 18-73.~~See *PCI-X POM 0 Local Low Address (PCIX0_POM0LAL)* on page 605.

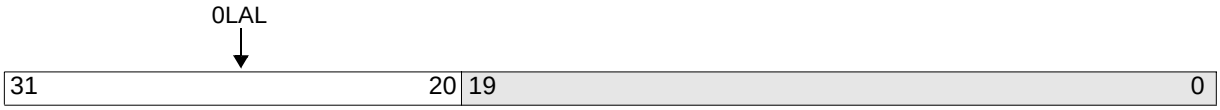


Figure 0-62. PCI-X POM 0 Local Low Address (PCIX0_POM0LAL)

31:20	0LAL	Local Address Low	Defines the starting address of range 0 in PLB space that is mapped to PCI memory. Only the bits that are 1 in the POM 0 size are actually used to determine the starting address, all other bits are don't cares.
19:0		Reserved	Returns zero when read.

Figure 29-233. PCI-X POM 0 Local Low Address (PCIX0_POM0LAL)

31:20	0LAL	Local Address Low	Defines the starting address of range 0 in PLB space that is mapped to PCI memory. Only the bits that are 1 in the POM 0 size are actually used to determine the starting address, all other bits are don't cares.
19:0		Reserved	Returns zero when read.

MMIO 0x2 0EC80078 Read/Write (PLB), 0x7B-0x78 Read/Write (PCI-X)

~~See "PCI-X POM 0 PCI Address High (PCIX0_POM0PCIAH)" on page 18-77.~~See *PCI-X POM 0 PCI Address High (PCIX0_POM0PCIAH)* on page 609.

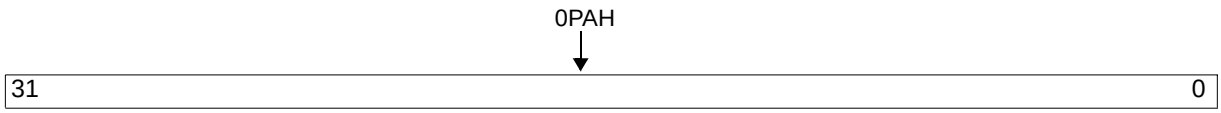


Figure 0-63. PCI-X POM 0 PCI Address High (PCIX0_POM0PCIAH)

31:0	0PAH	PCI High Address

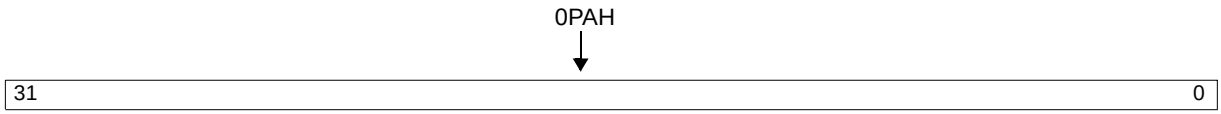


Figure 29-234. PCI-X POM 0 PCI Address High (PCIX0_POM0PCIAH)

31:0	0PAH	PCI High Address

MMIO 0x2 0EC80074 Read/Write (PLB), 0x77-0x74 Read/Write (PCI-X)

~~See "PCI-X POM 0 PCI Address Low (PCIX0_POM0PCIAL)" on page 18-76.~~See *PCI-X POM 0 PCI Address Low (PCIX0_POM0PCIAL)* on page 608.

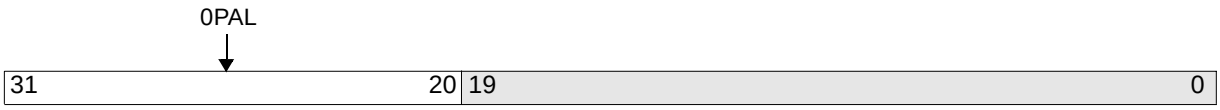


Figure 0-64. PCI-X POM 0 PCI Address Low (PCIX0_POM0PCIAL)

31:20	0PAL	PCI Low Address	
19:0		Reserved	Always read as zero.

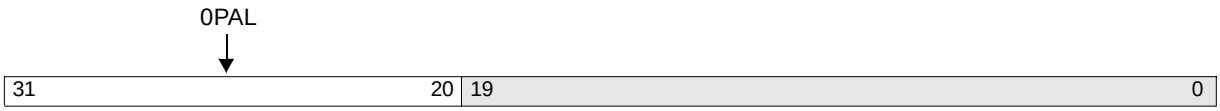


Figure 29-235. PCI-X POM 0 PCI Address Low (PCIX0_POM0PCIAL)

31:20	0PAL	PCI Low Address	
19:0		Reserved	Always read as zero.

MMIO 0x2 0EC80080 Read/Write (PLB), 0x83-0x80 Read/Write (PCI-X)

~~See "PCI-X POM 1 Local Address High (PCIX0_POM1LAH)" on page 18-79.~~See *PCI-X POM 1 Local Address High (PCIX0_POM1LAH)* on page 611.

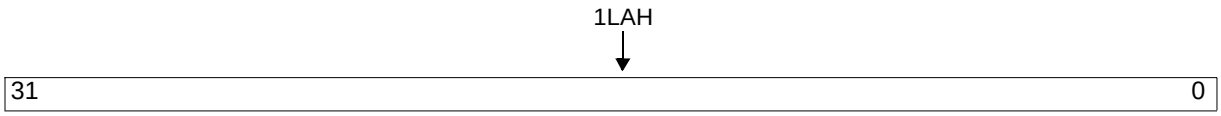


Figure 0-65. PCI-X POM 1 Local Address High (PCIX0_POM1LAH)

31:0	1LAH	Local Address High	Defines the starting address of range 1 in PLB space that is mapped to PCI memory. Only the bits that are 1 in the POM 1 Size are actually used to determine the starting address. All other bits are don't cares.

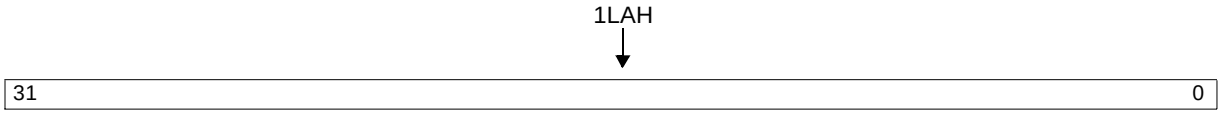


Figure 29-236. PCI-X POM 1 Local Address High (PCIX0_POM1LAH)

31:0	1LAH	Local Address High	Defines the starting address of range 1 in PLB space that is mapped to PCI memory. Only the bits that are 1 in the POM 1 Size are actually used to determine the starting address. All other bits are don't cares.

MMIO 0x2 0EC80070 Read/Write (PLB), 0x73-0x70 Read/Write (PCI-X)

~~See "PCI-X POM 0 Size/Attribute Register (PCIX0_POM0SA)" on page 18-75.~~See *PCI-X POM 0 Size/Attribute Register (PCIX0_POM0SA)* on page 607.

Figure 0-66. PCI-X POM 0 Size/Attribute Register (PCIX0_POM0SA)

31:20	SIZE	Size of POM0.	The size of the POM0 can range from 1M to 4G and must be a power of 2. To determine the value to program in this field, use the following process: 1. Represent the desired size with a 32-bit number. Use only one bit set in that number since it is a power of two. Use all zeros to represent 4 GB. 2 Set all the bits to the left of the set bit. 3. Store bits [31:20] of the resulting number in this field. For example, if the desired POM0 size is 16M, start with 16x1024x1024 = 0100_0000h. Set all the bits to the left of the set bit (FF00_0000h). Bits [31:20] of that number are FF0h; therefore, FF0h should be stored in this field.
19:1		Reserved	Always read as zero.
0	En	Enable Mapping 0 Disable 1 Enable	This bit determines if range 0 is enabled to map PLB space to PCI memory space. **Note:** The POM 0 Local Address, POM 0 PCI Low Address, and POM 0 PCI High Address must be initialized before enabling. This field has a value of 0 after reset.

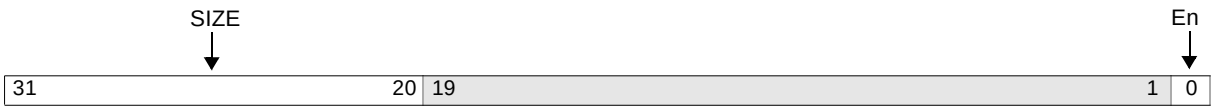


Figure 29-237. PCI-X POM 0 Size/Attribute Register (PCIX0_POM0SA)

31:20	SIZE	Size of POM0.	The size of the POM0 can range from 1M to 4G and must be a power of 2. To determine the value to program in this field, use the following process: 1. Represent the desired size with a 32-bit number. Use only one bit set in that number since it is a power of two. Use all zeros to represent 4 GB. 2 Set all the bits to the left of the set bit. 3. Store bits [31:20] of the resulting number in this field. For example, if the desired POM0 size is 16M, start with 16x1024x1024 = 0100_0000h. Set all the bits to the left of the set bit (FF00_0000h). Bits [31:20] of that number are FF0h; therefore, FF0h should be stored in this field.
19:1		Reserved	Always read as zero.
0	En	Enable Mapping 0 Disable 1 Enable	This bit determines if range 0 is enabled to map PLB space to PCI memory space. **Note:** The POM 0 Local Address, POM 0 PCI Low Address, and POM 0 PCI High Address must be initialized before enabling. This field has a value of 0 after reset.

MMIO 0x2 0EC8007C Read/Write (PLB), 0x7F-0x7C Read/Write (PCI-X)

~~See "PCI-X POM 1 Local Address Low (PCIX0_POM1LAL)" on page 18-78.~~See *PCI-X POM 1 Local Address Low (PCIX0_POM1LAL)* on page 610.

Figure 0-67. PCI-X POM 1 Local Address Low (PCIX0_POM1LAL)

31:20	1LAL	Local Address Low	Defines the starting address of range 1 in PLB space that is mapped to PCI memory.
19:0		Reserved	Returns zero when read.

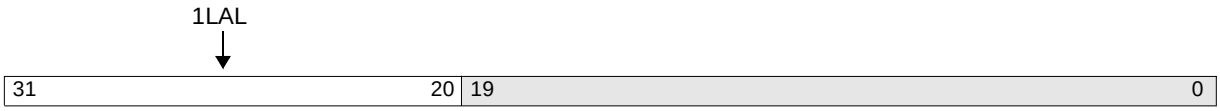


Figure 29-238. PCI-X POM 1 Local Address Low (PCIX0_POM1LAL)

31:20	1LAL	Local Address Low	Defines the starting address of range 1 in PLB space that is mapped to PCI memory.
19:0		Reserved	Returns zero when read.

MMIO 0x2 0EC80084 Read/Write (PLB), 0x87-0x84 Read/Write (PCI-X)

~~See "PCI-X POM 1 Size/Attribute Register (PCIX0_POM1SA)" on page 18-80.~~See *PCI-X POM 1 Size/Attribute Register (PCIX0_POM1SA)* on page 612.

Figure 0-68. PCI-X POM 1 Size/Attribute Register (PCIX0_POM1SA)

31:20	SIZE	Size of POM 1	The size of the POM 1 can range from 1M to 4G and must be a power of 2. To determine the value to program in this field, use the following process: 1. Represent the desired size with a 32-bit number. Use only one bit set in that number since it is a power of two. 2 Set all the bits to the left of the set bit. 3. Store bits [31:20] of the resulting number in this field. For example, if the desired POM 1 size is 16M, start with 16x1024x1024 = 0100_0000h. Set all the bits to the left of the set bit (FF00_0000h). Bits [31:20] of that number are FF0h; therefore, FF0h should be stored in this field.
19:1		Reserved	Always read as zero.
0	En	Enable mapping 1 Enable 0 Disable	This bit determines if range 1 is enabled to map PLB space to PCI memory space. **Note:** The POM 1 Local Address, POM 1 PCI Low Address, and POM 1 PCI High Address must be initialized before enabling. This field has a value of 0 after reset.

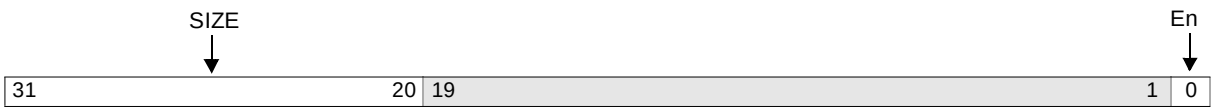


Figure 29-239. PCI-X POM 1 Size/Attribute Register (PCIX0_POM1SA)

31:20	SIZE	Size of POM 1	The size of the POM 1 can range from 1M to 4G and must be a power of 2. To determine the value to program in this field, use the following process: 1. Represent the desired size with a 32-bit number. Use only one bit set in that number since it is a power of two. Use all zeros to represent 4 GB. 2 Set all the bits to the left of the set bit. 3. Store bits [31:20] of the resulting number in this field. For example, if the desired POM 1 size is 16M, start with 16x1024x1024 = 0100_0000h. Set all the bits to the left of the set bit (FF00_0000h). Bits [31:20] of that number are FF0h; therefore, FF0h should be stored in this field.
19:1		Reserved	Always read as zero.
0	En	Enable mapping 1 Enable 0 Disable	This bit determines if range 1 is enabled to map PLB space to PCI memory space. **Note:** The POM 1 Local Address, POM 1 PCI Low Address, and POM 1 PCI High Address must be initialized before enabling. This field has a value of 0 after reset.

MMIO 0x2 0EC8008C Read/Write (PLB), 0x8F-0x8C Read/Write (PCI-X)

~~See "PCI-X POM 1 PCI Address High (PCIX0_POM1PCIAH)" on page 18-82.~~See *PCI-X POM 1 PCI Address High (PCIX0_POM1PCIAH)* on page 614.

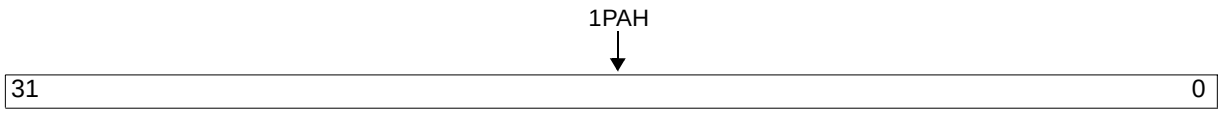


Figure 0-69. PCI-X POM 1 PCI Address High (PCIX0_POM1PCIAH)

31:0	1PAH	PCI Address High

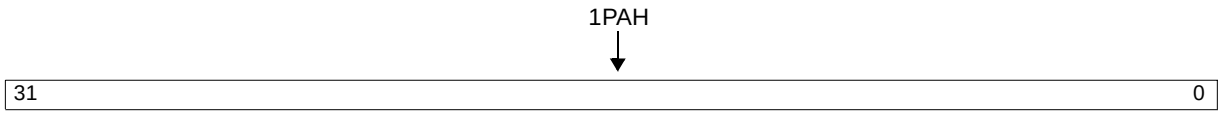


Figure 29-240. PCI-X POM 1 PCI Address High (PCIX0_POM1PCIAH)

31:0	1PAH	PCI Address High

MMIO 0x2 0EC80088 Read/Write (PLB), 0x8b-0x88 Read/Write (PCI-X)

~~See "PCI-X POM 1 PCI Address Low (PCIX0_POM1PCIAL)" on page 18-81.~~See *PCI-X POM 1 PCI Address Low (PCIX0_POM1PCIAL)* on page 613.

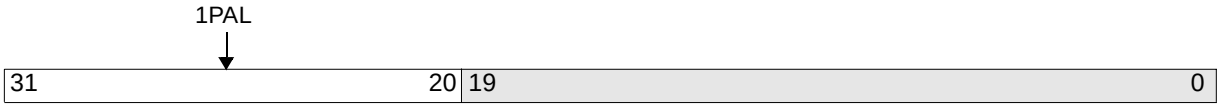


Figure 0-70. PCI-X POM 1 PCI Address Low (PCIX0_POM1PCIAL)

31:20	1PAL	PCI Address Low	
19:0		Reserved	Returns zero when read.

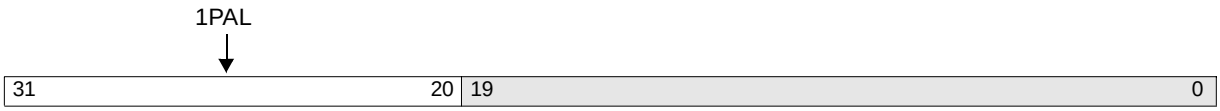


Figure 29-241. PCI-X POM 1 PCI Address Low (PCIX0_POM1PCIAL)

31:20	1PAL	PCI Address Low	
19:0		Reserved	Returns zero when read.

MMIO 0x2 0EC80090 Read/Write (PLB), 0x93-0x90 Read/Write (PCI-X)

~~See "PCI-X POM 2 Size/Attribute Register (PCIX0_POM2SA)" on page 18-83.~~See *PCI-X POM 2 Size/Attribute Register (PCIX0_POM2SA)* on page 615.

Figure 0-71. PCI-X POM 2 Size/Attribute Register (PCIX0_POM2SA)

31:1		Reserved	Always read as zero.
0	En	Enables mapping 1 Enable 0 Disable	This bit determines if range 2 is enabled to map PLB space to PCI memory space. A value of 1 enables the mapping.

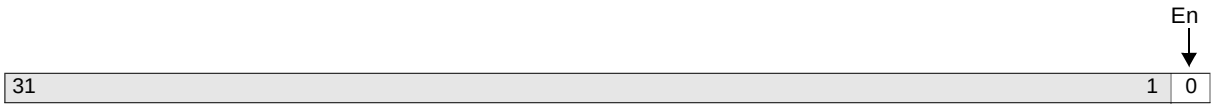


Figure 29-242. PCI-X POM 2 Size/Attribute Register (PCIX0_POM2SA)

31:1		Reserved	Always read as zero.
0	En	Enables mapping 1 Enable 0 Disable	This bit determines if range 2 is enabled to map PLB space to PCI memory space. A value of 1 enables the mapping.

MMIO 0x2 0EC80008 Read/Write (PLB), 0x08 Read-Only (PCI-X)

~~See "PCI-X Revision ID Register (PCIC0_REVID)" on page 18-43.~~See *PCI-X Revision ID Register (PCIX0_REVID)* on page 575.

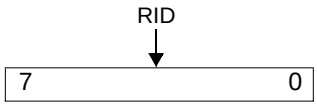


Figure 0-72. PCI-X Revision ID Register (PCIX0_REVID)

7:0	RID	Revision ID	Revision level of device.

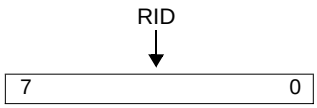


Figure 29-243. PCI-X Revision ID Register (PCIX0_REVID)

7:0	RID	Revision ID	Revision level of device.

MMIO 0x2 0EC8002E Read/Write (PLB), 0x2F-0x2E Read-Only (PCI-X)

~~See "PCI-X Subsystem ID Register (PCIX0_SBSYSID)" on page 18-55.~~See *PCI-X Subsystem ID Register (PCIX0_SBSYSID)* on page 587.

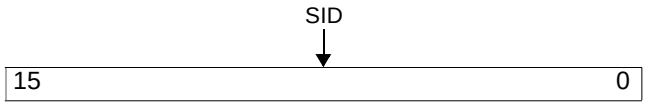


Figure 0-73. PCI-X Subsystem ID Register (PCIX0_SBSYSID)

15:0	SID	PCI Subsystem ID

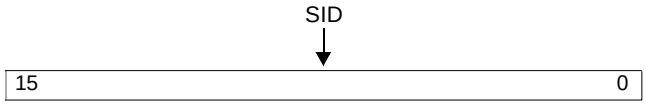


Figure 29-244. PCI-X Subsystem ID Register (PCIX0_SBSYSID)

15:0	SID	PCI Subsystem ID

MMIO 0x2 0EC8002C Read/Write (PLB), 0x2D-0x2C Read-Only (PCI-X)

~~See "PCI-X Subsystem Vendor ID Register (PCIX0_SBSYSVID)" on page 18-54.~~See *PCI-X Subsystem Vendor ID Register (PCIX0_SBSYSVID)* on page 586.

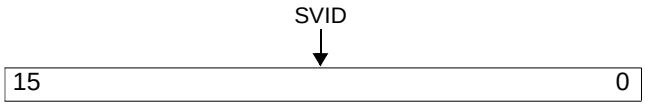


Figure 0-74. PCI-X Subsystem Vendor ID Register (PCIX0_SBSYSVID)

15:0	SVID	PCI Subsystem Vendor ID

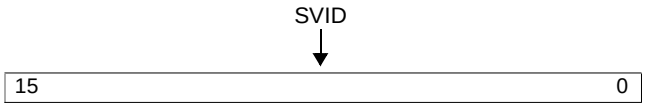


Figure 29-245. PCI-X Subsystem Vendor ID Register (PCIX0_SBSYSVID)

15:0	SVID	PCI Subsystem Vendor ID

MMIO 0x2 0EC80006 Read/Write (PLB), 0x07-0x06 Read/Write (PCI-X)

~~See "PCI-X Status Register (PCIX0_STATUS)" on page 18-41.~~See *PCI-X Status Register (PCIX0_STATUS)* on page 573.

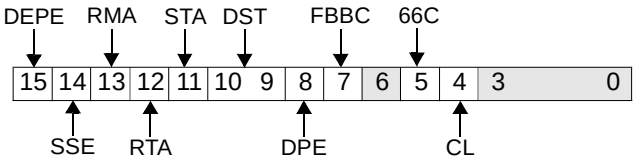


Figure 0-75. PCI-X Status Register (PCIX0_STATUS)

15	DPE	Detected Parity Error Write 1 to clear.	PLB-PCIX Bridge sets this bit whenever it detects a PCI bus parity error, regardless of the setting of any enable bits, that is, this bit is non-maskable. This bit is set when the following occurs: PCI address bus parity error detected when PLB-PCIX Bridge is a target. PCI data bus parity error detected when a PCI master writes to PLB memory (PLB-PCIX Bridge is the target). PCI data bus parity error detected when PLB-PCIX Bridge masters a PCI read cycle. Writing a 1 to this bit resets it to 0.
14	SSE	Signaled System Error Write 1 to clear.	Signaled system error. PLB-PCIX Bridge sets this bit if it asserts PCI_SERR#. Writing a 1 to this bit resets it to 0.
13	RMA	Received Master Abort Write 1 to clear.	This bit is set whenever PLB-PCIX Bridge terminates a PCI cycle for which it is the master with master abort (except configuration and special cycles) and the Master Abort Error Enable bit of the Error Enable Register is set. Writing a 1 to this bit resets it to 0.
12	RTA	Received Target Abort Write 1 to clear.	PLB-PCIX Bridge sets this bit whenever a PCI cycle for which it is the master is terminated with target abort and the Target Abort Error Enable bit of the Error Enable Register is set. Writing a 1 to this bit resets it to 0.
11	STA	Signaled Target Abort Write 1 to clear.	PLB-PCIX Bridge never signals a target abort. This bit is always 0.
10:9	DST	$\overline{\text{PCIDevSel}}$ Response Timing Read-only.	PCI_DEVSEL# response timing. PLB-PCIX Bridge asserts PCI_DEVSEL# on the second clock (also known as medium response time) after PCI_FRAME# is asserted by a PCI master attempting to access memory on the PLB side of the bridge. These bits are read-only and always return 01b when read.

8	DPE	Data Parity Error Detected Write 1 to clear.	This bit is set when the following two conditions are met: PLB-PCIX Bridge detects a data parity error (PCI_PERR# asserted) when it is the master on a PCI read cycle, or it is the master when it samples PCI_PERR# asserted on a PCI write cycle. The Parity Error Response bit (bit 6 of the PCI Command Register) is set. Writing a 1 to this bit resets it to 0.
7	FBBC	Fast Back-to-Back Capable Read-only; returns zero when read.	Indicates that the PCI target is capable of accepting fast back-to-back transactions when the transactions are not to the same agent. PLB-PCIX Bridge target does accept this type of fast back-to-back transaction, therefore, this bit is read-only and is 1 when in PCI-Conv mode. When in PCI-X mode, it is always 0.
6		Reserved	This bit is hardcoded to zero. This bit was the UDF support bit in PCI 2.1.
5	66C	66 MHz Capable	Indicates that the device is able to run at 66 MHz. Hardcoded to 1.
4	CL	Capabilities List	Indicates whether or not this device implements the pointer for a new capabilities linked list at offset 34h. This bit is read-only and returns 1 when read, indicating that the value read at offset 34h is a pointer in configuration space to a linked list of new capabilities.
3:0		Reserved	These bits are reserved, and return zeros when read.

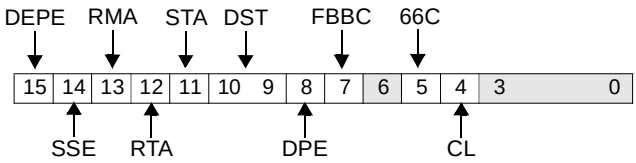


Figure 29-246. PCI-X Status Register (PCIX0_STATUS)

15	DPE	Detected Parity Error Write 1 to clear.	PLB-PCIX Bridge sets this bit whenever it detects a PCI bus parity error, regardless of the setting of any enable bits, that is, this bit is non-maskable. This bit is set when the following occurs: PCI address bus parity error detected when PLB-PCIX Bridge is a target. PCI data bus parity error detected when a PCI master writes to PLB memory (PLB-PCIX Bridge is the target). PCI data bus parity error detected when PLB-PCIX Bridge masters a PCI read cycle. Writing a 1 to this bit resets it to 0.
14	SSE	Signaled System Error Write 1 to clear.	Signaled system error. PLB-PCIX Bridge sets this bit if it asserts PCI_SERR#. Writing a 1 to this bit resets it to 0.
13	RMA	Received Master Abort Write 1 to clear.	This bit is set whenever PLB-PCIX Bridge terminates a PCI cycle for which it is the master with master abort (except configuration and special cycles) and the Master Abort Error Enable bit of the Error Enable Register is set. Writing a 1 to this bit resets it to 0.

12	RTA	Received Target Abort Write 1 to clear.	PLB-PCIX Bridge sets this bit whenever a PCI cycle for which it is the master is terminated with target abort and the Target Abort Error Enable bit of the Error Enable Register is set. Writing a 1 to this bit resets it to 0.
11	STA	Signaled Target Abort Write 1 to clear.	PLB-PCIX Bridge never signals a target abort. This bit is always 0.
10:9	DST	PCIDevSel Response Timing Read-only.	PCI_DEVSEL# response timing. PLB-PCIX Bridge asserts PCI_DEVSEL# on the second clock (also known as medium response time) after PCI_FRAME# is asserted by a PCI master attempting to access memory on the PLB side of the bridge. These bits are read-only and always return 01b when read.
8	DPE	Data Parity Error Detected Write 1 to clear.	This bit is set when the following two conditions are met: PLB-PCIX Bridge detects a data parity error (PCI_PERR# asserted) when it is the master on a PCI read cycle, or it is the master when it samples PCI_PERR# asserted on a PCI write cycle. The Parity Error Response bit (bit 6 of the PCI Command Register) is set. Writing a 1 to this bit resets it to 0.
7	FBBC	Fast Back-to-Back Capable Read-only; returns zero when read.	Indicates that the PCI target is capable of accepting fast back-to-back transactions when the transactions are not to the same agent. PLB-PCIX Bridge target does accept this type of fast back-to-back transaction, therefore, this bit is read-only and is 1 when in PCI-Conv mode. When in PCI-X mode, it is always 0.
6		Reserved	This bit is hardcoded to zero. This bit was the UDF support bit in PCI 2.1.
5	66C	66 MHz Capable	Indicates that the device is able to run at 66 MHz. Hardcoded to 1.
4	CL	Capabilities List	Indicates whether or not this device implements the pointer for a new capabilities linked list at offset 34h. This bit is read-only and returns 1 when read, indicating that the value read at offset 34h is a pointer in configuration space to a linked list of new capabilities.
3:0		Reserved	These bits are reserved, and return zeros when read.

MMIO 0x2 0EC80000 Read/Write (PLB), 0x01-0x00 Read-Only (PCI-X)

~~See "PCI-X Vendor ID Register (PCIX0_VENDID)" on page 18-37.~~See *PCI-X Vendor ID Register (PCIX0_VENDID)* on page 570.

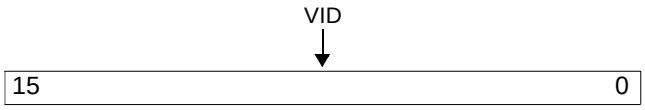


Figure 0-76. PCI-X Vendor ID Register (PCIX0_VENDID)

15:0	VID	Vendor ID

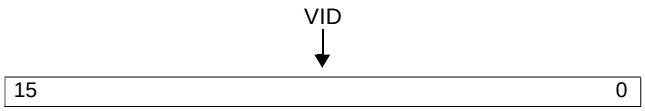


Figure 29-247. PCI-X Vendor ID Register (PCIX0_VENDID)

15:0	VID	Vendor ID

DCR 0x083 ReadWrite

~~See "PLB Arbiter Control Register (PLB0_ACR)" on page 2-7.~~See *PLB Arbiter Control Register (PLB0_ACR)* on page 97

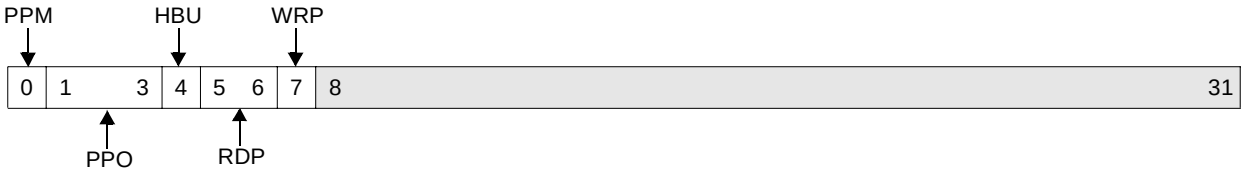


Figure 0-1. PLB Arbiter Control Register (PLB0_ACR)

0	PPM	PLB Priority Mode 0 Fixed 1 Fair	
1:3	PPO	PLB Priority Order 000 Masters 0, 1, 2, 3, 5, 6, 7 001 Masters 1, 2, 3, 5, 6, 7, 0 010 Masters 2, 3, 5, 6, 7, 0, 1 011 Masters 3, 5, 6, 7, 0, 1, 2 100 Masters 5, 6, 7, 0, 1, 2, 3 101 Masters 5, 6, 7, 0, 1, 2, 3, 110 Masters 6, 7, 0, 1, 2, 3, 5 111 Masters 7, 0, 1, 2, 3, 5, 6	The PLB priority order (PPO) will remain a constant value whenPLB priority mode (PPM) bit 0 is set to 0 (fixed). However, the PLB priority order (PPO) bits 1:3 are constantly changing when PLB priority mode (PPM) bit 0 is set to 1 (fair)
4	HBU	High Bus Utilization 0 Disabled 1 Enabled	If read and write pipelining are disabled this feature has no effect on arbiter operation.
5:6	RDP	Read Pipeline Control 00 Read pipelining disabled 01 2 Deep read pipe 10 3 Deep read pipe 11 4 Deep read pipe	
7	WRP	Write Pipeline Control 0 Write Pipeline Disabled 1 2 Deep Write Pipe	
8:31		Reserved	

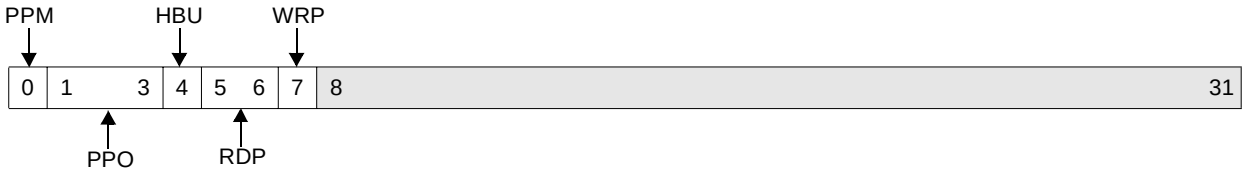


Figure 29-248. PLB Arbiter Control Register (PLB0_ACR)

0	PPM	PLB Priority Mode 0 Fixed 1 Fair	
1:3	PPO	PLB Priority Order 000 Masters 0, 1, 2, 3, 5, 6, 7 001 Masters 1, 2, 3, 5, 6, 7, 0 010 Masters 2, 3, 5, 6, 7, 0, 1 011 Masters 3, 5, 6, 7, 0, 1, 2 100 Masters 5, 6, 7, 0, 1, 2, 3 101 Masters 5, 6, 7, 0, 1, 2, 3, 110 Masters 6, 7, 0, 1, 2, 3, 5 111 Masters 7, 0, 1, 2, 3, 5, 6	The PLB priority order (PPO) will remain a constant value whenPLB priority mode (PPM) bit 0 is set to 0 (fixed). However, the PLB priority order (PPO) bits 1:3 are constantly changing when PLB priority mode (PPM) bit 0 is set to 1 (fair)
4	HBU	High Bus Utilization 0 Disabled 1 Enabled	If read and write pipelining are disabled this feature has no effect on arbiter operation.
5:6	RDP	Read Pipeline Control 00 Read pipelining disabled 01 2 Deep read pipe 10 3 Deep read pipe 11 4 Deep read pipe	
7	WRP	Write Pipeline Control 0 Write Pipeline Disabled 1 2 Deep Write Pipe	
8:31		Reserved	

DCR 0x087 Read/Write

~~See "PLB Error Address Register High (PLB0_BEARH)" on page 2-11.~~See *PLB Error Address Register High (PLB0_BEARH)* on page 101

0	31

Figure 0-2. PLB Error Address Register (PLB0_BEARH)

0:31		Upper address of bus timeout error

0	31

Figure 29-249. PLB Error Address Register (PLB0_BEARH)

0:31		Upper address of bus timeout error

DCR 0x086 Read/Write

See *PLB Error Address Register (PLB0_BEARL)* on page 100.

0	31

Figure 0-3. PLB Error Address Register (PLB0_BEARL)

0:31		Lower address of bus timeout error

0	31

Figure 29-250. PLB Error Address Register (PLB0_BEARL)

0:31		Lower address of bus timeout error

DCR 0x084 Read/Clear

~~See "PLB Error Status Register (PLB0_BESR)" on page 2-8.~~See *PLB Error Status Register (PLB0_BESR)* on page 98

–

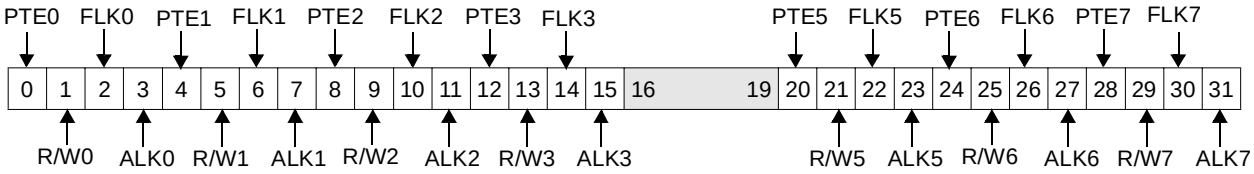


Figure 0-4. PLB Error Status Register (PLB0_BESR)

0	PTE0	Master 0 PLB Timeout Error Status 0 No master 0 timeout error 1 Master 0 timeout error	Master 0 is the read-only instruction cache unit (ICU)
1	R/W0	Master 0 Read/Write Status 0 Master 0 error operation was a write 1 Master 0 ICU error operation was a read	
2	FLK0	Master 0 PLB0_BESR Field Lock 0 Master 0 PLB0_BESR field is unlocked 1 Master 0 field is locked	
3	ALK0	Master 0 PLB0_BEAR Address Lock 0 Master 0 PLB0_BEAR is unlocked 1 Master 0 PLB0_BEAR is locked	
4	PTE1	Master 1 PLB Timeout Error Status 0 No master 1 timeout error 1 Master 1 timeout error	Master 1 is the read-only data cache unit (DCU)
5	R/W1	Master 1 Read/Write Status 0 Master 1 error operation was a write 1 Master 1 error operation was a read	
6	FLK1	Master 1PLB0_BESR Field Lock 0 Master 1 PLB0_BESR field is unlocked 1 Master 1 PLB0_BESR field is locked	
7	ALK1	Master 1 PLB0_BEAR Address Lock 0 Master 1 PLB0_BEAR is unlocked 1 Master 1 PLB0_BEAR is locked	
8	PTE2	Master 2 PLB Timeout Error Status 0 No master 2 timeout error 1 Master 2 timeout error	Master 2 is the write-only data cache unit (DCU)
9	R/W2	Master 2 Read/Write Status 0 Master 2 error operation was a write 1 Master 2 error operation was a read	
10	FLK2	Master 2 PLB0_BESR Field Lock 0 Master 2 PLB0_BESR field is unlocked 1 Master 2 PLB0_BESR field is locked	

11	ALK2	Master 2 PLB0_BEAR Address Lock 0 Master 2 PLB0_BEAR is unlocked 1 Master 2 PLB0_BEAR is locked	
12	PTE3	Master 3 PLB Timeout Error Status 0 No Master 3 timeout error 1 Master 3 timeout error	Master 3 is the PCIX bridge controller
13	R/W3	Master 3 Read/Write Status 0 Master 3 error operation was a write 1 Master 3 error operation was a read	
14	FLK3	Master 3 PLB0_BESR Field Lock 0 Master 3 PLB0_BESR field is unlocked 1 Master 3 PLB0_BESR field is locked	
15	ALK3	Master 3 PLB0_BEAR Address Lock 0 Master 3 PLB0_BEAR is unlocked 1 Master 3 PLB0_BEAR is locked	
16:19		Reserved	
20	PTE5	Master 5 PLB Timeout Error Status 0 No master 5 timeout error 1 Master 5 timeout error	Master 5 is the MAL controller
21	R/W5	Master 5 Read/Write Status 0 Master 5 error operation was a write 1 Master 5 error operation was a read	
22	FLK5	Master 5 PLB0_BESR Field Lock 0 Master 5 PLB0_BESR field is unlocked 1 Master 5 PLB0_BESR field is locked	
23	ALK5	Master 5 PLB0_BEAR Address Lock 0 Master 5 PLB0_BEAR is unlocked 1 Master 5 PLB0_BEAR is locked	
24	PTE6	Master 6 PLB Timeout Error Status 0 No master 6 timeout error 1 Master 6 timeout error	Master 6 is the DMA controller
25	R/W6	Master 6 Read/Write Status 0 Master 6 error operation was a write 1 Master 6 error operation was a read	
26	FLK6	Master 6 PLB0_BESR Field Lock 0 Master 6 PLB0_BESR field is unlocked 1 Master 6 PLB0_BESR field is locked	
27	ALK6	Master 6 PLB0_BEAR Address Lock 0 Master 6 PLB0_BEAR is unlocked 1 Master 6 PLB0_BEAR is locked	
28	PTE7	Master 7 PLB Timeout Error Status 0 No Master 7 timeout error 1 Master 7 timeout error	Master 7 is the OPB to PLB bridge controller

29	R/W7	Master 7 Read/Write Status 0 Master 7 error operation was a write 1 Master 7 error operation was a read
30	FLK7	Master 7 PLB0_BESR Field Lock 0 Master 7 PLB0_BESR field is unlocked 1 Master 7 PLB0_BESR field is locked
31	ALK7	Master 7 PLB0_BEAR Address Lock 0 Master 7 PLB0_BEAR is unlocked 1 Master 7 PLB0_BEAR is locked

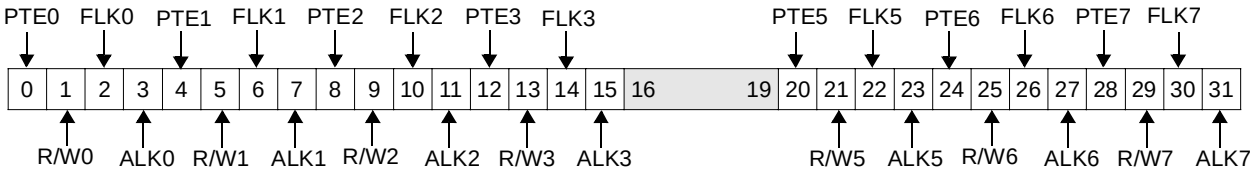


Figure 29-251. PLB Error Status Register (PLB0_BESR)

0	PTE0	Master 0 PLB Timeout Error Status 0 No master 0 timeout error 1 Master 0 timeout error	Master 0 is the read-only instruction cache unit (ICU)
1	R/W0	Master 0 Read/Write Status 0 Master 0 error operation was a write 1 Master 0 ICU error operation was a read	
2	FLK0	Master 0 PLB0_BESR Field Lock 0 Master 0 PLB0_BESR field is unlocked 1 Master 0 field is locked	
3	ALK0	Master 0 PLB0_BEAR Address Lock 0 Master 0 PLB0_BEAR is unlocked 1 Master 0 PLB0_BEAR is locked	
4	PTE1	Master 1 PLB Timeout Error Status 0 No master 1 timeout error 1 Master 1 timeout error	Master 1 is the read-only data cache unit (DCU)
5	R/W1	Master 1 Read/Write Status 0 Master 1 error operation was a write 1 Master 1 error operation was a read	
6	FLK1	Master 1PLB0_BESR Field Lock 0 Master 1 PLB0_BESR field is unlocked 1 Master 1 PLB0_BESR field is locked	
7	ALK1	Master 1 PLB0_BEAR Address Lock 0 Master 1 PLB0_BEAR is unlocked 1 Master 1 PLB0_BEAR is locked	
8	PTE2	Master 2 PLB Timeout Error Status 0 No master 2 timeout error 1 Master 2 timeout error	Master 2 is the write-only data cache unit (DCU)
9	R/W2	Master 2 Read/Write Status 0 Master 2 error operation was a write 1 Master 2 error operation was a read	

10	FLK2	Master 2 PLB0_BESR Field Lock 0 Master 2 PLB0_BESR field is unlocked 1 Master 2 PLB0_BESR field is locked	
11	ALK2	Master 2 PLB0_BEAR Address Lock 0 Master 2 PLB0_BEAR is unlocked 1 Master 2 PLB0_BEAR is locked	
12	PTE3	Master 3 PLB Timeout Error Status 0 No Master 3 timeout error 1 Master 3 timeout error	Master 3 is the PCIX bridge controller
13	R/W3	Master 3 Read/Write Status 0 Master 3 error operation was a write 1 Master 3 error operation was a read	
14	FLK3	Master 3 PLB0_BESR Field Lock 0 Master 3 PLB0_BESR field is unlocked 1 Master 3 PLB0_BESR field is locked	
15	ALK3	Master 3 PLB0_BEAR Address Lock 0 Master 3 PLB0_BEAR is unlocked 1 Master 3 PLB0_BEAR is locked	
16:19		Reserved	
20	PTE5	Master 5 PLB Timeout Error Status 0 No master 5 timeout error 1 Master 5 timeout error	Master 5 is the MAL controller
21	R/W5	Master 5 Read/Write Status 0 Master 5 error operation was a write 1 Master 5 error operation was a read	
22	FLK5	Master 5 PLB0_BESR Field Lock 0 Master 5 PLB0_BESR field is unlocked 1 Master 5 PLB0_BESR field is locked	
23	ALK5	Master 5 PLB0_BEAR Address Lock 0 Master 5 PLB0_BEAR is unlocked 1 Master 5 PLB0_BEAR is locked	
24	PTE6	Master 6 PLB Timeout Error Status 0 No master 6 timeout error 1 Master 6 timeout error	Master 6 is the DMA controller
25	R/W6	Master 6 Read/Write Status 0 Master 6 error operation was a write 1 Master 6 error operation was a read	
26	FLK6	Master 6 PLB0_BESR Field Lock 0 Master 6 PLB0_BESR field is unlocked 1 Master 6 PLB0_BESR field is locked	
27	ALK6	Master 6 PLB0_BEAR Address Lock 0 Master 6 PLB0_BEAR is unlocked 1 Master 6 PLB0_BEAR is locked	
28	PTE7	Master 7 PLB Timeout Error Status 0 No Master 7 timeout error 1 Master 7 timeout error	Master 7 is the OPB to PLB bridge controller
29	R/W7	Master 7 Read/Write Status 0 Master 7 error operation was a write 1 Master 7 error operation was a read	

30	FLK7	Master 7 PLB0_BESR Field Lock 0 Master 7 PLB0_BESR field is unlocked 1 Master 7 PLB0_BESR field is locked
31	ALK7	Master 7 PLB0_BEAR Address Lock 0 Master 7 PLB0_BEAR is unlocked 1 Master 7 PLB0_BEAR is locked

DCR 0x082 Read-Only

~~See "PLB Revision ID Register (PLB0_REVID)" on page 2-7.~~See *PLB Revision ID Register (PLB0_REVID)* on page 97

Figure 0-5. PLB Revision ID Register (PLB0_REVID)

0:31	RevId	This value indicates what revision level this core is. PLB Arbiter core Revision ID = 0x(C27E5031)

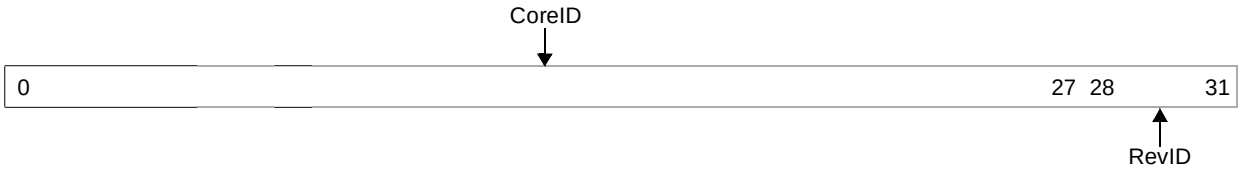


Figure 29-252. PLB Arbiter Revision ID Register (PLB0_REVID)

0:27	CoreID	Core ID	Corresponds to the technology.
28:31	RevId	Revision ID	Corresponds to the revision level.

DCR 0x093 Read-Only

See *PLB to OPB Bridge Error Address Register High (POB0_BEARH)* on page 105.

Figure 0-1. PLB to OPB Bridge Error Address Register High (POB0_BEARH)

0:27		Reserved
28:31	10'h0B3	Upper address of bus error

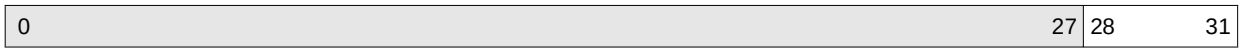


Figure 29-253. PLB to OPB Bridge Error Address Register High (POB0_BEARH)

0:27		Reserved
28:31	10'h0B3	Upper address of bus error

DCR 0x092 Read-Only

See *PLB to OPB Bridge Error Address Register Low (POB0_BEARL)* on page 104.

Figure 0-2. PLB to OPB Bridge Error Address Register Low (POB0_BEARL)

0:31	0x0B2	Lower address of bus error

Figure 29-254. PLB to OPB Bridge Error Address Register Low (POB0_BEARL)

0:31	0x0B2	Lower address of bus error

DCR 0x090 Read/Clear

See *PLB to OPB Bridge Error Status Register 0 (POB0_BESR0)* on page 102.

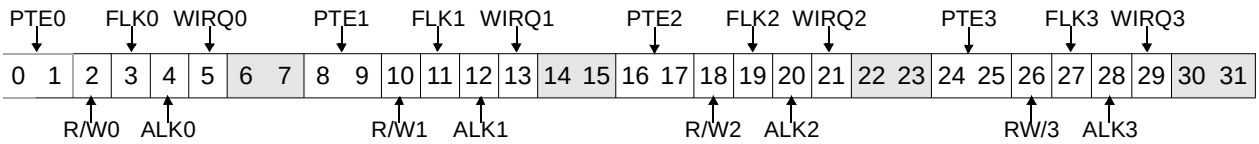


Figure 0-3. PLB to OPB Bridge Error Status Register 0 (POB0_BESR0)

0:1	PTE0	PLB Timeout Error Status Master 0 00 No master 0 error occurred 01 Master 0 timeout error occurred 10 Master 0 slave error occurred 11 Reserved	Master 0 is the read-only instruction cache unit (ICU)
2	R/W0	Read Write Status Master 0 0 Master 0 error operation is a read 1 Master 0 error operation is a write	
3	FLK0	POB0_BESR0 Field Lock Master 0 0 Master 0 POB0_BESR0 field is unlocked 1 Master 0 POB0_BESR0 field is locked	
4	ALK0	POB0_BEAR Address Lock Master 0 0 Master 0 POB0_BEAR address is unlocked 1 Master 0 POB0_BEAR address is locked	
5	WIRQ0	Write Error Interrupt Master 0 0 No write error detected - master 0 interrupt request is inactive 1 Write error detected - master 0 interrupt request is active	
6:7		Reserved	
8:9	PTE1	PLB Timeout Error Status Master 1 00 No master 1 error occurred 01 Master 1 timeout error occurred 10 Master 1 slave error occurred 11 Reserved	Master 1 is the read-only data cache unit (DCU)
10	R/W1	Read/Write Status Master 1 0 Master 1 error operation is a read 1 Master 1 error operation is a write	
11	FLK1	POB0_BESR0 Field Lock Master 1 0 Master 1 POB0_BESR0 field is unlocked 1 Master 1 POB0_BESR0 field is locked	
12	ALK1	POB0_BEAR Address Lock Master 1 0 Master 1 POB0_BEAR address is unlocked 1 Master 1 POB0_BEAR address is locked	

13	WIRQ1	Write Error Interrupt Master 1 0 No write error detected - master 1 interrupt request is inactive 1 Write error detected - master 1 interrupt request is active	
14:15		Reserved	
16:17	PTE2	PLB Timeout Error Status Master 2 00 No master 2 error occurred 01 Master 2 timeout error occurred 10 Master 2 slave error occurred 11 Reserved	Master 2 is the write-only data cache unit (DCU)
18	R/W2	Read/Write Status Master 2 0 Master 2 error operation is a read 1 Master 2 error operation is a write	
19	FLK2	POB0_BESR0 Field Lock Master 2 0 Master 2 POB0_BESR0 field is unlocked 1 Master 2 POB0_BESR0 field is locked	
20	ALK2	POB0_BEAR Address Lock Master 2 0 Master 2 POB0_BEAR address is unlocked 1 Master 2 POB0_BEAR address is locked	
21	WIRQ2	Write Error Interrupt Master 2 0 No write error detected - master 2 interrupt request is inactive 1 Write error detected - master 2 interrupt request is active	
22:23		Reserved	
24:25	PTE3	PLB Timeout Error Status Master 3 00 No master 3 error occurred 01 Master 3 timeout error occurred 10 Master 3 slave error occurred 11 Reserved	Master 3 is the PCIX bridge controller
25	R/W3	Read/Write Status Master 3 0 Master 3 error operation is a read 1 Master 3 error operation is a write	
26	FLK3	POB0_BESR0 Field Lock Master 3 0 Master 3 POB0_BESR0 field is unlocked 1 Master 3 POB0_BESR0 field is locked	
27	ALK3	POB0_BEAR Address Lock Master 3 0 Master 3 POB0_BEAR address is unlocked 1 Master 3 POB0_BEAR address is locked	
29	WIRQ3	Write Error Interrupt Master 3 0 No write error detected - master 3 interrupt request is inactive 1 Write error detected - master 3 interrupt request is active	
30:31		Reserved	

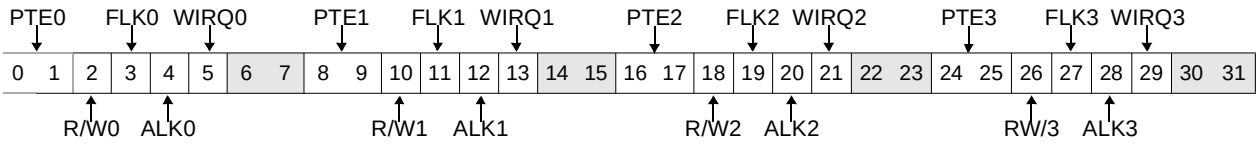


Figure 29-255. PLB to OPB Bridge Error Status Register 0 (POB0_BESR0)

0:1	PTE0	PLB Timeout Error Status Master 0 00 No master 0 error occurred 01 Master 0 timeout error occurred 10 Master 0 slave error occurred 11 Reserved	Master 0 is the read-only instruction cache unit (ICU)
2	R/W0	Read Write Status Master 0 0 Master 0 error operation is a write 1 Master 0 error operation is a read	
3	FLK0	POB0_BESR0 Field Lock Master 0 0 Master 0 POB0_BESR0 field is unlocked 1 Master 0 POB0_BESR0 field is locked	
4	ALK0	POB0_BEAR Address Lock Master 0 0 Master 0 POB0_BEAR address is unlocked 1 Master 0 POB0_BEAR address is locked	
5	WIRQ0	Write Error Interrupt Master 0 0 No write error detected - master 0 interrupt request is inactive 1 Write error detected - master 0 interrupt request is active	
6:7		Reserved	
8:9	PTE1	PLB Timeout Error Status Master 1 00 No master 1 error occurred 01 Master 1 timeout error occurred 10 Master 1 slave error occurred 11 Reserved	Master 1 is the read-only data cache unit (DCU)
10	R/W1	Read/Write Status Master 1 0 Master 1 error operation is a write 1 Master 1 error operation is a read	
11	FLK1	POB0_BESR0 Field Lock Master 1 0 Master 1 POB0_BESR0 field is unlocked 1 Master 1 POB0_BESR0 field is locked	
12	ALK1	POB0_BEAR Address Lock Master 1 0 Master 1 POB0_BEAR address is unlocked 1 Master 1 POB0_BEAR address is locked	
13	WIRQ1	Write Error Interrupt Master 1 0 No write error detected - master 1 interrupt request is inactive 1 Write error detected - master 1 interrupt request is active	
14:15		Reserved	

16:17	PTE2	PLB Timeout Error Status Master 2 00 No master 2 error occurred 01 Master 2 timeout error occurred 10 Master 2 slave error occurred 11 Reserved	Master 2 is the write-only data cache unit (DCU)
18	R/W2	Read/Write Status Master 2 0 Master 2 error operation is a write 1 Master 2 error operation is a read	
19	FLK2	POB0_BESR0 Field Lock Master 2 0 Master 2 POB0_BESR0 field is unlocked 1 Master 2 POB0_BESR0 field is locked	
20	ALK2	POB0_BEAR Address Lock Master 2 0 Master 2 POB0_BEAR address is unlocked 1 Master 2 POB0_BEAR address is locked	
21	WIRQ2	Write Error Interrupt Master 2 0 No write error detected - master 2 interrupt request is inactive 1 Write error detected - master 2 interrupt request is active	
22:23		Reserved	
24:25	PTE3	PLB Timeout Error Status Master 3 00 No master 3 error occurred 01 Master 3 timeout error occurred 10 Master 3 slave error occurred 11 Reserved	Master 3 is the PCIX bridge controller
26	R/W3	Read/Write Status Master 3 0 Master 3 error operation is a write 1 Master 3 error operation is a read	
27	FLK3	POB0_BESR0 Field Lock Master 3 0 Master 3 POB0_BESR0 field is unlocked 1 Master 3 POB0_BESR0 field is locked	
28	ALK3	POB0_BEAR Address Lock Master 3 0 Master 3 POB0_BEAR address is unlocked 1 Master 3 POB0_BEAR address is locked	
29	WIRQ3	Write Error Interrupt Master 3 0 No write error detected - master 3 interrupt request is inactive 1 Write error detected - master 3 interrupt request is active	
30:31		Reserved	

DCR 0x094 Read/Clear

See *PLB to OPB Bridge Error Status Register 1(POB0_BESR1)* on page 105.

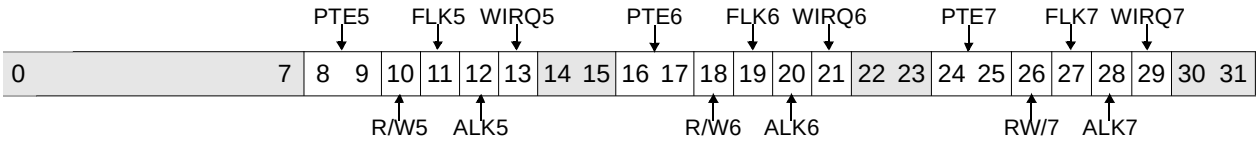


Figure 0-4. PLB to OPB Bridge Error Status Register 1 (POB0_BESR1)

0:7		Reserved	
8:9	PTE5	PLB Timeout Error Status Master 5 00 No master 5 error occurred 01 Master 5 timeout error occurred 10 Master 5 slave error occurred 11 Reserved	Master 5 is the MAL controller
10	R/W5	Read/Write Status Master 5 0 Master 5 error operation is a read 1 Master 5 error operation is a write	
11	FLK5	POB0_BESR0 Field Lock Master 5 0 Master 5 POB0_BESR0 field is unlocked 1 Master 5 POB0_BESR0 field is locked	
12	ALK5	POB0_BEAR Address Lock Master 5 0 Master 5 POB0_BEAR address is unlocked 1 Master 5 POB0_BEAR address is locked	
13	WIRQ5	Write Error Interrupt Master 5 0 No write error detected - master 5 interrupt request is inactive 1 Write error detected - master 5 interrupt request is active	
14:15		Reserved	
16:17	PTE6	PLB Timeout Error Status Master 6 00 No master 6 error occurred 01 Master 6 timeout error occurred 10 Master 6 slave error occurred 11 Reserved	Master 6 is the DMA controller
18	R/W6	Read/Write Status Master 6 0 Master 6 error operation is a read 1 Master 6 error operation is a write	
19	FLK6	POB0_BESR0 Field Lock Master 6 0 Master 6 POB0_BESR0 field is unlocked 1 Master 6 POB0_BESR0 field is locked	

20	ALK6	POB0_BEAR Address Lock Master 6 0 Master 6 POB0_BEAR address is unlocked 1 Master 6 POB0_BEAR address is locked	
21	WIRQ6	Write Error Interrupt Master 6 0 No write error detected - master 6 interrupt request is inactive 1 Write error detected - master 6 interrupt request is active	
22:23		Reserved	
24:25	PTE7	PLB Timeout Error Status Master 7 00 No master 7 error occurred 01 Master 7 timeout error occurred 10 Master 7 slave error occurred 11 Reserved	Master 7 is the OPB to PLB bridge controller
25	R/W7	Read/Write Status Master 7 0 Master 7 error operation is a read 1 Master 7 error operation is a write	
26	FLK7	POB0_BESR0 Field Lock Master 7 0 Master 7 POB0_BESR0 field is unlocked 1 Master 7 POB0_BESR0 field is locked	
27	ALK7	POB0_BEAR Address Lock Master 7 0 Master 7 POB0_BEAR address is unlocked 1 Master 7 POB0_BEAR address is locked	
29	WIRQ7	Write Error Interrupt Master 7 0 No write error detected - master 7 interrupt request is inactive 1 Write error detected - master 7 interrupt request is active	
30:31		Reserved	

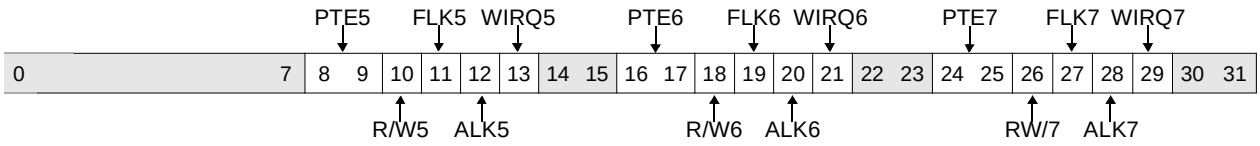


Figure 29-256. PLB to OPB Bridge Error Status Register 1 (POB0_BESR1)

0:7		Reserved	
8:9	PTE5	PLB Timeout Error Status Master 5 00 No master 5 error occurred 01 Master 5 timeout error occurred 10 Master 5 slave error occurred 11 Reserved	Master 5 is the MAL controller

10	R/W5	Read/Write Status Master 5 0 Master 5 error operation is a write 1 Master 5 error operation is a read
11	FLK5	POB0_BESR1 Field Lock Master 5 0 Master 5 POB0_BESR1 field is unlocked 1 Master 5 POB0_BESR1 field is locked
12	ALK5	POB0_BEAR Address Lock Master 5 0 Master 5 POB0_BEAR address is unlocked 1 Master 5 POB0_BEAR address is locked
13	WIRQ5	Write Error Interrupt Master 5 0 No write error detected - master 5 interrupt request is inactive 1 Write error detected - master 5 interrupt request is active
14:15		Reserved
16:17	PTE6	PLB Timeout Error Status Master 6 Master 6 is the DMA controller 00 No master 6 error occurred 01 Master 6 timeout error occurred 10 Master 6 slave error occurred 11 Reserved
18	R/W6	Read/Write Status Master 6 0 Master 6 error operation is a write 1 Master 6 error operation is a read
19	FLK6	POB0_BESR1 Field Lock Master 6 0 Master 6 POB0_BESR1 field is unlocked 1 Master 6 POB0_BESR1 field is locked
20	ALK6	POB0_BEAR Address Lock Master 6 0 Master 6 POB0_BEAR address is unlocked 1 Master 6 POB0_BEAR address is locked
21	WIRQ6	Write Error Interrupt Master 6 0 No write error detected - master 6 interrupt request is inactive 1 Write error detected - master 6 interrupt request is active
22:23		Reserved
24:25	PTE7	PLB Timeout Error Status Master 7 Master 7 is the OPB to PLB bridge controller 00 No master 7 error occurred 01 Master 7 timeout error occurred 10 Master 7 slave error occurred 11 Reserved
26	R/W7	Read/Write Status Master 7 0 Master 7 error operation is a write 1 Master 7 error operation is a read
27	FLK7	POB0_BESR1 Field Lock Master 7 0 Master 7 POB0_BESR1 field is unlocked 1 Master 7 POB0_BESR1 field is locked
28	ALK7	POB0_BEAR Address Lock Master 7 0 Master 7 POB0_BEAR address is unlocked 1 Master 7 POB0_BEAR address is locked

29	WIRQ7	Write Error Interrupt Master 7 0 No write error detected - master 7 interrupt request is inactive 1 Write error detected - master 7 interrupt request is active
30:31		Reserved

DCR 0x096 Read/Clear

See *PLB to OPB Bridge Configuration Register (POB0_CONFG)* on page 107.

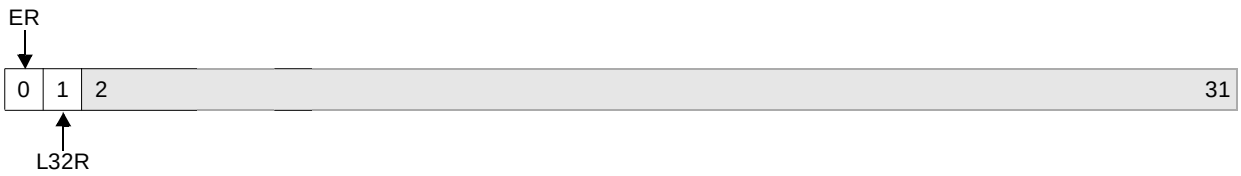


Figure 0-5. PLB to OPB Bridge Configuration Register (POB0_CONFG)

0	ER	Enable rearbitration 0 PLB to OPB bridge rearbitration is enabled 1 PLB to OPB bridge rearbitration is disabled
1	L32R	Line 32 bit read 0 PLB to OPB bridge reads line operations at requested size of master 1 PLB to OPB bridge reads line operations at 32 bit slave size regardless of requesting master size
2:31		Reserved

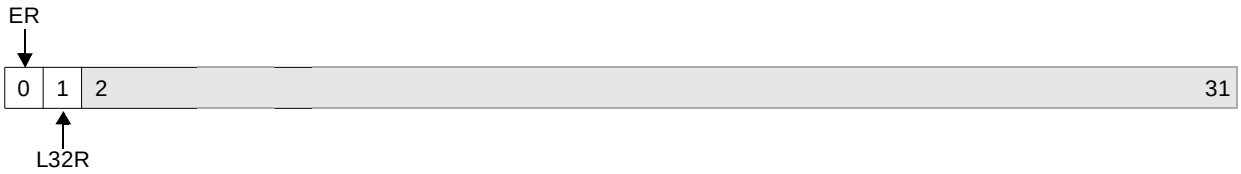


Figure 29-257. PLB to OPB Bridge Configuration Register (POB0_CONFG)

0	ER	Enable rearbitration 0 PLB to OPB bridge rearbitration is enabled 1 PLB to OPB bridge rearbitration is disabled
1	L32R	Line 32 bit read 0 PLB to OPB bridge reads line operations at requested size of master 1 PLB to OPB bridge reads line operations at 32 bit slave size regardless of requesting master size
2:31		Reserved

DCR 0x098 Read/Clear

See *PLB to OPB Bridge Burst Latency Timer (POB0_LATENCY)* on page 107.

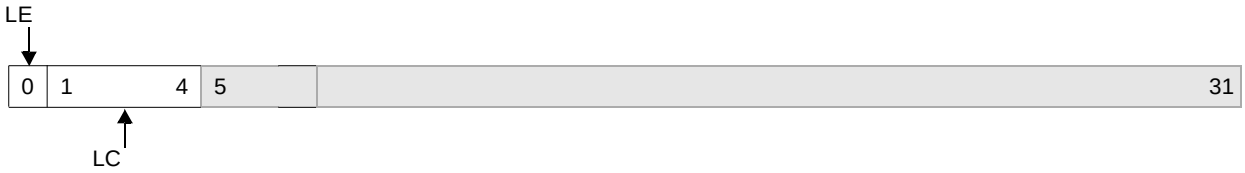


Figure 0-6. PLB to OPB Bridge Burst Latency Timer Register (POB0_LATENCY)

0	LE	Latency Enable 0 Latency timer disabled 1 Latency timer enabled
1:4	LC	Latency Count 0000 - Minimum count value. PLB to OPB bridge will count 16 OPB_xferAcks during a read or write burst sequence and if OPB_pendReq is sampled high at the end of the count - then the burst sequence is terminated. 1111 - Maximum count value. PLB to OPB bridge will count 16 blocks of 16 OPB_xferAcks, (or 256 xferAcks) during a read or write burst sequence, and if OPB_pendReq is sampled high at the end of count - then the burst sequence is terminated.
5:31		Reserved

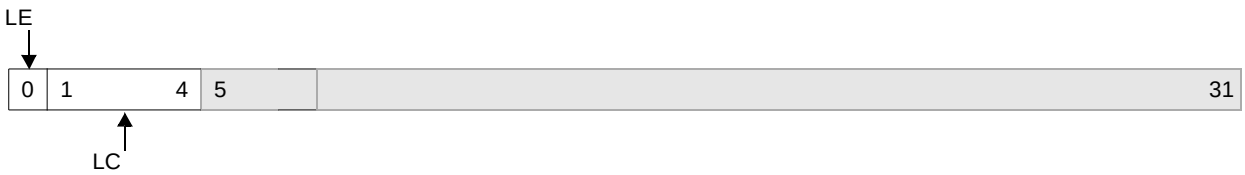


Figure 29-258. PLB to OPB Bridge Burst Latency Timer Register (POB0_LATENCY)

0	LE	Latency Enable 0 Latency timer disabled 1 Latency timer enabled
1:4	LC	Latency Count 0000 - Minimum count value. PLB to OPB bridge will count 16 OPB_xferAcks during a read or write burst sequence and if OPB_pendReq is sampled high at the end of the count - then the burst sequence is terminated. 1111 - Maximum count value. PLB to OPB bridge will count 16 blocks of 16 OPB_xferAcks, (or 256 xferAcks) during a read or write burst sequence, and if OPB_pendReq is sampled high at the end of count - then the burst sequence is terminated.
5:31		Reserved

DCR 0x09A Read/Clear

See *PLB to OPB Bridge Revision ID (POB0_REVID)* on page 108.

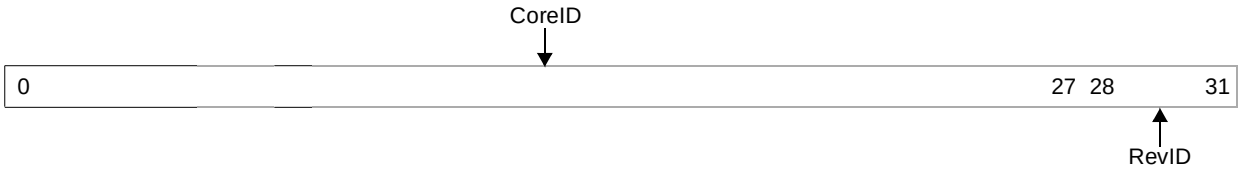


Figure 0-7. PLB to OPB Bridge Revision ID Register (POB0_REVID)

0:27	CoreID	Core ID value indicates what technology and what type of core this is. The core ID value for this core is C27E502.
28:31	RevId	This value indicates what revision level this core is. The revision ID for this core is 1.

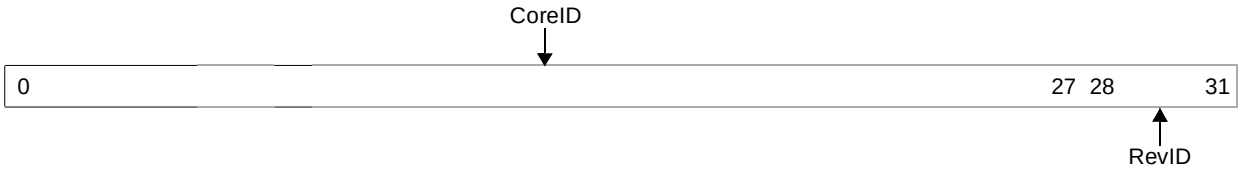


Figure 29-259. PLB to OPB Bridge Revision ID Register (POB0_REVID)

0:27	CoreID	Core ID	Corresponds to the technology.
28:31	RevId	Revision ID	Corresponds to the revision level.

DCR Accessed using PPM0_CFGADDR; PPM0_CFGDATA; Offset 0x23 R/W

See *Cycle Counter Register (PPM0_CCR)* on page 125.

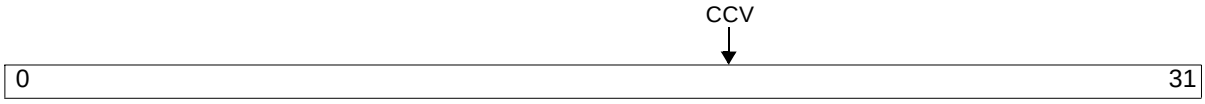


Figure 0-1. Cycle Control Register (PPM0_CCR)

0:31	CCV	32-bit Clock Count Value	This value can range from 0x0h to 0xFFFFFFFFh. Its contents are decremented by 1 (for every PLB clock) if the enable bit in the CR register is active. This counter will stop decrementing when a value of 0x0h reached.

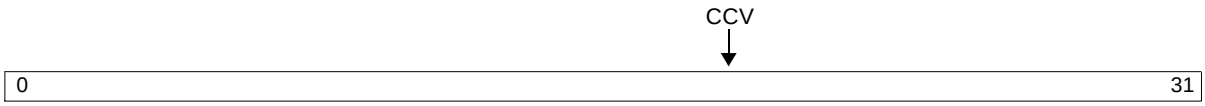


Figure 29-260. Cycle Control Register (PPM0_CCR)

0:31	CCV	32-bit Clock Count Value	This value can range from 0x0h to 0xFFFFFFFFh. Its contents are decremented by 1 (for every PLB clock) if the enable bit in the CR register is active. This counter will stop decrementing when a value of 0x0h reached.

DCR Offset 0x016 R/W

See *PPM Configuration Address Register (PPM0_CFGADDR)* on page 120.

Figure 0-2. PPM Configuration Address Register (PPM0_CFGADDR)

0:23			
24:31	DCRA	8-bit PPM Register Offset Value	This value can range from 0x000000 to 0x7F. Its contents are used as the indirect DCR address for accessing a PPM register.

Figure 29-261. PPM Configuration Address Register (PPM0_CFGADDR)

0:23			
24:31	DCRA	8-bit PPM Register Offset Value	This value can range from 0x000000 to 0x7F. Its contents are used as the indirect DCR address for accessing a PPM register.

DCR 0x017 R/W

See *PPM Configuration Data Register (PPM0_CFGDATA)* on page 121.

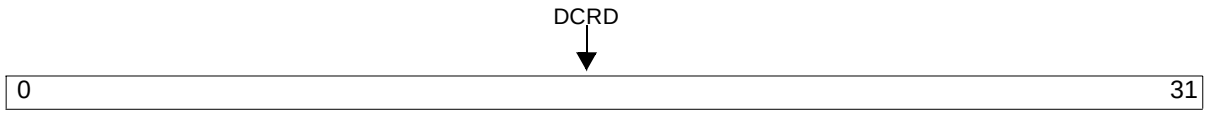


Figure 0-3. PPM Configuration Data Register (PPM0_CFGDATA)

0:31	DCRD	32-bit Data Value	This value can range from 0x00000000 to 0xFFFFFFFF. Its contents contain the value of the PPM register as indicated by the PPM_CFGADDR register contents.

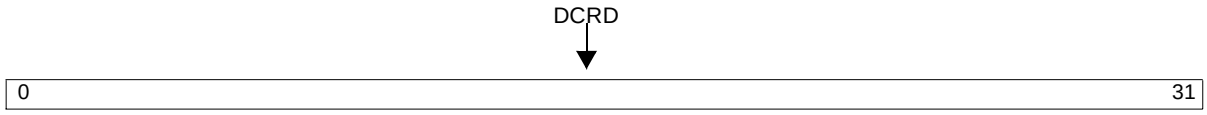


Figure 29-262. PPM Configuration Data Register (PPM0_CFGDATA)

0:31	DCRD	32-bit Data Value	This value can range from 0x00000000 to 0xFFFFFFFF. Its contents contain the value of the PPM register as indicated by the PPM_CFGADDR register contents.

DCR Accessed using PPM0_CFGADDR; PPM0_CFGDATA; Offset 0x23 R/W

See *Control Register (PPM0_CR)* on page 123.

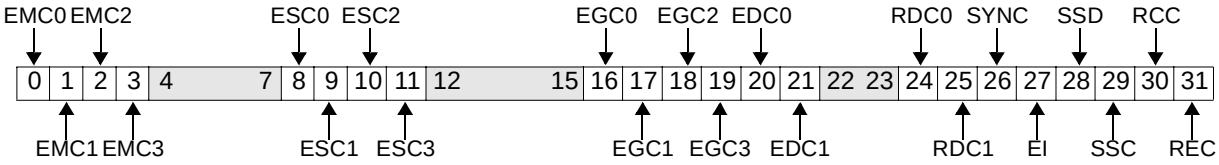


Figure 0-4. Control Register (PPM0_CR)

0	EMC0	Enable Master Counter 0 0 Disable master counter 0. 1 Enable master counter 0.
1	EMC1	Enable Master Counter 1 0 Disable master counter 1. 1 Enable master counter 1.
2	EMC2	Enable Master Counter 2 0 Disable master counter 2. 1 Enable master counter 2.
3	EMC3	Enable Master Counter 3 0 Disable master counter 3. 1 Enable master counter 3.
4:7		Reserved
8	ESC0	Enable Slave Counter 0 0 Disable slave counter 0. 1 Enable slave counter 0.
9	ESC1	Enable Slave Counter 1 0 Disable slave counter 1. 1 Enable slave counter 1.
10	ESC2	Enable Slave Counter 2 0 Disable slave counter 2. 1 Enable slave counter 2.
11	ESC3	Enable Slave Counter 3 0 Disable slave counter 3. 1 Enable slave counter 3.
12:15		Reserved
16	EGC0	Enable Generic Counter 0 0 Disable generic counter 0. 1 Enable generic counter 0.
17	EGC1	Enable Generic Counter 1 0 Disable generic counter 1. 1 Enable generic counter 1.

18	EGC2	Enable Generic Counter 2 0 Disable generic counter 2. 1 Enable generic counter 2.
19	EGC3	Enable Generic Counter 3 0 Disable generic counter 3. 1 Enable generic counter 3.
20	EDC0	Enable Duration Counter Set 0 0 Disable duration counter 0. 1 Enable duration counter 0.
21	EDC1	Enable Duration Counter Set 1 0 Disable duration counter 1. 1 Enable duration counter 1.
22:23		Reserved
24	RDC0	Reset Event Counters 0 No action. 1 Reset all DC0 counters.
25	RDC1	Reset Event Counters 0 No action. 1 Reset all DC1 counters.
26	SYNC	Sync enable 0 Disable sync-out functionality. 1 Enable sync-out functionality.
27	EI	Enable Interrupts 0 Disable interrupts. 1 Enable interrupts
28	SSD	Start/Stop Duration Events 0 Stop all counters programmed to operate in duration mode. 1 Start all counters programmed to operate in duration mode.
29	SSC	Start/Stop Cycle Counter 0 Stop decrementing cycle counter. 1 Start decrementing cycle counter.
30	RCC	Reset Cycle Counters 0 No action. 1 Reset cycle counter.
31	REC	Reset Event Counters 0 No action. 1 Reset all MCn, SCn, and GCn counters.

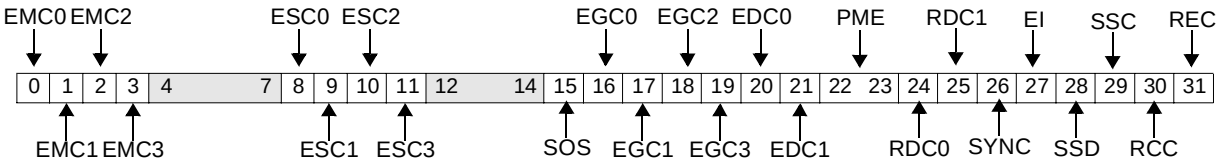


Figure 29-263. Control Register (PPM0_CR)

0	EMC0	Enable Master Counter 0 0 Disable master counter 0. 1 Enable master counter 0.
1	EMC1	Enable Master Counter 1 0 Disable master counter 1. 1 Enable master counter 1.
2	EMC2	Enable Master Counter 2 0 Disable master counter 2 1 Enable master counter 2
3	EMC3	Enable Master Counter 3 0 Disable master counter 3 1 Enable master counter 3
4:7		Reserved
8	ESC0	Enable Slave Counter 0 0 Disable slave counter 0 1 Enable slave counter 0
9	ESC1	Enable Slave Counter 1 0 Disable slave counter 1 1 Enable slave counter 1
10	ESC2	Enable Slave Counter 2 0 Disable slave counter 2 1 Enable slave counter 2
11	ESC3	Enable Slave Counter 3 0 Disable slave counter 3 1 Enable slave counter 3
12:14		Reserved
15	SOS	Synchout Selector 0 Drives PPM_synchOutSSC signal with value of SSC bit and PPM_synchOutSSD signal with value of SSD bit. 1 Asserts PPM_synchOutSSC signal when master counter 0, slave counter 0, generic counter 0, or cycle counter has overflow/underrun its value. Asserts PPM_synchOurSSD signal when duration counter 0 occurrence total counter, duration counter 1 occurrence total counter, duration counter o total value counter, and duration counter 1 total value counter has overrun their contents. **Note:** The associated interrupts must be enabled for this feature to work.
16	EGC0	Enable Generic Counter 0 0 Disable generic counter 0 1 Enable generic counter 0
17	EGC1	Enable Generic Counter 1 0 Disable generic counter 1 1 Enable generic counter 1
18	EGC2	Enable Generic Counter 2 0 Disable generic counter 2. 1 Enable generic counter 2.

19	EGC3	Enable Generic Counter 3 0 Disable generic counter 3. 1 Enable generic counter 3.
20	EDC0	Enable Duration Counter Set 0 0 Disable duration counter 0. 1 Enable duration counter 0.
21	EDC1	Enable Duration Counter Set 1 0 Disable duration counter 1. 1 Enable duration counter 1.
22:23	PME	Power Mode Enable 00 Normal mode 01 Low power mode 10 Ultra low power mode 11 Ultra low power mode
24	RDC0	Reset Event Counters 0 No action. 1 Reset all DC0 counters.
25	RDC1	Reset Event Counters 0 No action. 1 Reset all DC1 counters.
26	SYNC	SyncIn enable 0 Disable syncIn functionality. 1 Enable syncIn functionality.
27	EI	Enable Interrupts 0 Disable interrupts. 1 Enable interrupts
28	SSD	Start/Stop Duration Events 0 Stop all counters 1 Start all counters
29	SSC	Start/Stop Cycle Counter 0 Stop decrementing cycle counter. 1 Start decrementing cycle counter.
30	RCC	Reset Cycle Counters 0 No action. 1 Reset cycle counter.
31	REC	Reset Event Counters 0 No action. 1 Reset all MCn, SCn, and GCn counters.

DCR Accessed using PPM0_CFGADDR; PPM0_CFGDATA; Offset 0x35, 0x36 R/W

See *Duration Counter Minimum Register 0:1 (PPM0_DCMNR0-PPM0_DCMNR1)* on page 137.

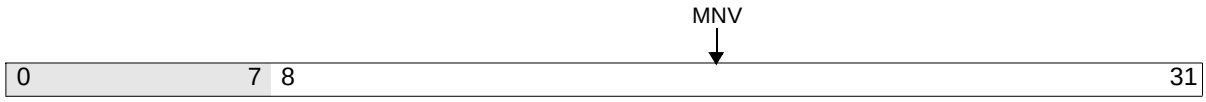


Figure 0-5. Duration Counter Minimum Registers 0:1 (PPM0_DCMNR0-PPM0_DCMNR1)

0:7		Reserved	
8:31	MNV	Least Min value	This value can range from 0x000000 to 0xFFFFFF. Its contents are the least minimum duration value calculated via the selection in its corresponding Duration Selection register. This register can only be updated if its corresponding enable bit in the CR register is active

Figure 29-264. Duration Counter Minimum Registers 0:1 (PPM0_DCMNR0-PPM0_DCMNR1)

0:7		Reserved	
8:31	MNV	Least Min value	This value can range from 0x000000 to 0xFFFFFF. Its contents are the least minimum duration value calculated via the selection in its corresponding Duration Selection register. This register can only be updated if its corresponding enable bit in the CR register is active

DCR Accessed using PPM0_CFGADDR; PPM0_CFGDATA; Offset 0x33, 0x34 R/W

See *Duration Counter Maximum Register 0:1 (PPM0_DCMXR0-PPM0_DCMXR1)* on page 136.

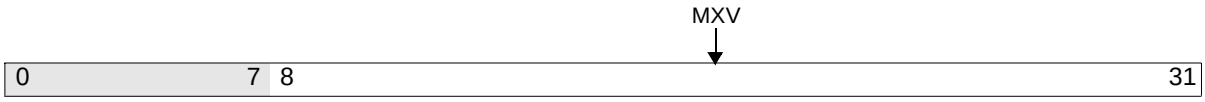


Figure 0-6. Duration Counter Maximum Registers 0:1 (PPM0_DCMXR0-PPM0_DCMXR1)

0:7		Reserved	
8:31	MXV	Highest Max value	This value can range from 0x0 to 0xFFFFFF. Its contents are the highest maximum duration value calculated via the selection in its corresponding Duration Selection register. This register can only be updated if its corresponding enable bit in the CR register is active

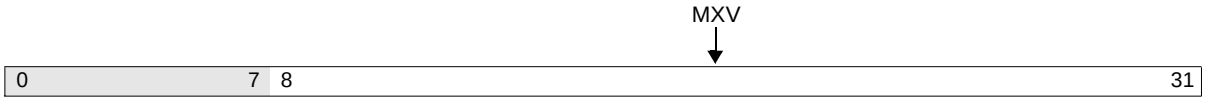


Figure 29-265. Duration Counter Maximum Registers 0:1 (PPM0_DCMXR0-PPM0_DCMXR1)

0:7		Reserved	
8:31	MXV	Highest Max value	This value can range from 0x0 to 0xFFFFFF. Its contents are the highest maximum duration value calculated via the selection in its corresponding Duration Selection register. This register can only be updated if its corresponding enable bit in the CR register is active

DCR Accessed using PPM0_CFGADDR; PPM0_CFGDATA; Offset 0x39, 0x3A R/W

See *Duration Counter Occurrence Total Register 0:1 (PPM0_DCOTR0-PPM0_DCOTR1)* on page 139.

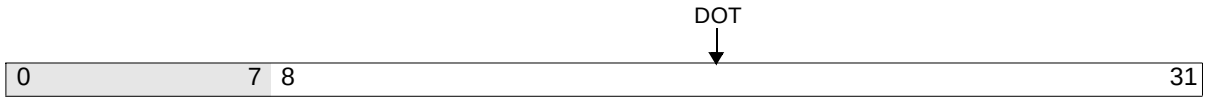


Figure 0-7. Duration Counter Occurrence Total Registers 0:1 (PPM0_DCOTR0-1)

0:7		Reserved	
8:31	DOT	Total number of event occurrences	This value can range from 0x0 to 0xFFFFFF. Its contents are incremented once for each occurrence of an event as selected in the corresponding Duration Selection register. This register can only be updated if its corresponding enable bit in the CR register is active

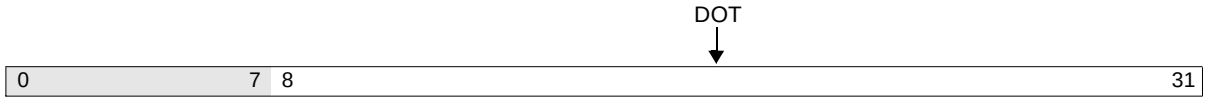


Figure 29-266. Duration Counter Occurrence Total Registers 0:1 (PPM0_DCOTR0-1)

0:7		Reserved	
8:31	DOT	Total number of event occurrences	This value can range from 0x0 to 0xFFFFFF. Its contents are incremented once for each occurrence of an event as selected in the corresponding Duration Selection register. This register can only be updated if its corresponding enable bit in the CR register is active

DCR Accessed using PPM0_CFGADDR; PPM0_CFGDATA; Offset 0x31 0x32 R/W

See *Duration Counter Selection Register 0:1 (PPM0_DCSR0-PPM0_DCSR1)* on page 135.

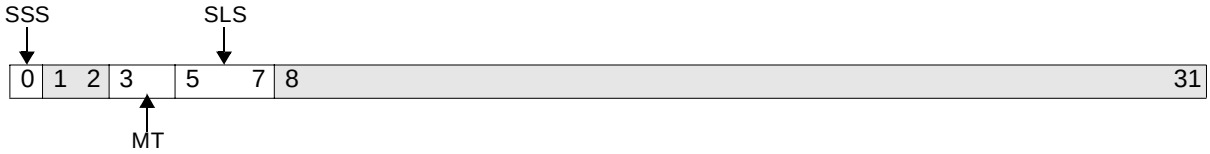


Figure 0-8. Duration Counter Selection Registers 0:1 (PPM0_DCSR0-PPM0_DCSR1)

0	SSS	Single Shot Selection 0 Disable single shot measurement 1 Enable single shot measurement	Selects which Slave's signal transitions to count with this counter set.
1:2		Reserved	
3:4	MT	Measurement Type 00 Write tenure 01 Read tenure 10 Wait tenure 11 Reserved	Selects the type of slave duration measurement to be performed (see Table 3-4).
5:7	SLS	Slave Selection 000 Selects Sl0 001 Selects Sl1 010 Selects Sl2 011 Selects Sl3 100 Selects Sl4 101 Selects Sl5 110 Selects Sl6 111 Selects Sl7	Selects the desired PLB Slave device in which the measurement is to be performed Slave 0 is DDR_SDRAM Slave 1 is PCIX Slave 2 is SRAM Slave 3 is reserved Slave 4 is reserved SLave 5 is PLB to OPB bridge Slave 6 is reserved Slave 7 is reserved
8:31		Reserved	

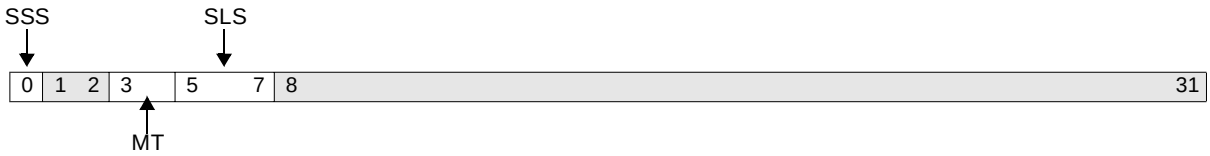


Figure 29-267. Duration Counter Selection Registers 0:1 (PPM0_DCSR0-PPM0_DCSR1)

0	SSS	Single Shot Selection 0 Disable single shot measurement 1 Enable single shot measurement	Selects which Slave's signal transitions to count with this counter set.
1:2		Reserved	
3:4	MT	Measurement Type 00 Write tenure 01 Read tenure 10 Wait tenure 11 Reserved	Selects the type of slave duration measurement to be performed (see Table 3-4).
5:7	SLS	Slave Selection 000 Selects Sl0 001 Selects Sl1 010 Selects Sl2 011 Selects Sl3 100 Selects Sl4 101 Selects Sl5 110 Selects Sl6 111 Selects Sl7	Selects the desired PLB Slave device in which the measurement is to be performed Slave 0 is DDR_SDRAM Slave 1 is PCIX Slave 2 is SRAM Slave 3 is reserved Slave 4 is reserved SLave 5 is PLB to OPB bridge Slave 6 is reserved Slave 7 is reserved
8:31		Reserved	

DCR Accessed using PPM0_CFGADDR; PPM0_CFGDATA; Offset 0x37 0x38 R/W

See *Duration Counter Total Value Register 0:1 (PPM0_DCTVR0-PPM0_DCTVR1)* on page 138.

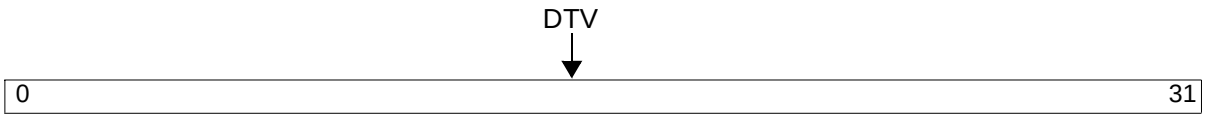


Figure 0-9. Duration Total Value Register 0:1 (PPM0_DCTVR0-PPM0_DCTVR1)

0:31	DTV	Duration Total Value	This value can range from 0x0 to 0xFFFFFFFF. The value is incremented in accordance with the duration values calculated via the selection in its corresponding Duration Selection register. This register can only be updated if its corresponding enable bit in the CR register is active.

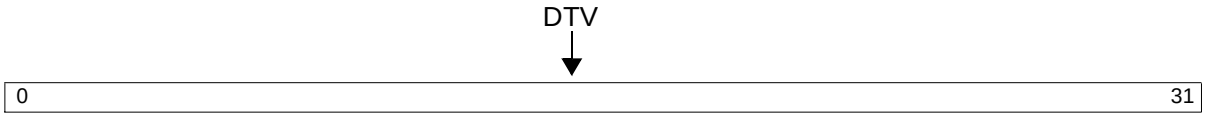


Figure 29-268. Duration Total Value Register 0:1 (PPM0_DCTVR0-PPM0_DCTVR1)

0:31	DTV	Duration Total Value	This value can range from 0x0 to 0xFFFFFFFF. The value is incremented in accordance with the duration values calculated via the selection in its corresponding Duration Selection register. This register can only be updated if its corresponding enable bit in the CR register is active.

DCR Accessed using PPM0_CFGADDR; PPM0_CFGDATA; Offset 0x32D, 0x2E, 0x2F, 0x30 R/W

See *Generic Pipeline Event Counter Register 0:3 (PPM0_GCR0-PPM0_GCR3)* on page 135.

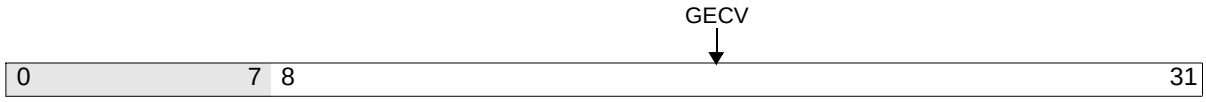


Figure 0-10. Generic Pipeline Event Counter Registers 0:3 (PPM0_GCR0-PPM0_GCR3)

0:7		Reserved	
8:31	GECV	Generic Event Counter Value	This value can range from 0x000000 to 0xFFFFFF. Its contents are incremented once for each event occurrence as selected in its corresponding GCounter Selection register. This register can only be updated if its corresponding GCn enable bit in the CR register is active

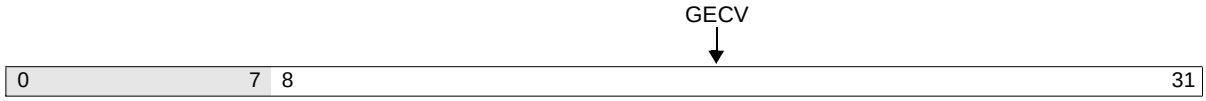


Figure 29-269. Generic Pipeline Event Counter Registers 0:3 (PPM0_GCR0-PPM0_GCR3)

0:7		Reserved	
8:31	GECV	Generic Event Counter Value	This value can range from 0x000000 to 0xFFFFFF. Its contents are incremented once for each event occurrence as selected in its corresponding GCounter Selection register. This register can only be updated if its corresponding GCn enable bit in the CR register is active

DCR Accessed using PPM0_CFGADDR; PPM0_CFGDATA; Offset 0x19, 0x1A, 0x1B, 0x1C R/W

See *Generic Event Counter Selection Register 0:3 (PPM0_GCSR0-PPM0_GCSR3)* on page 133.

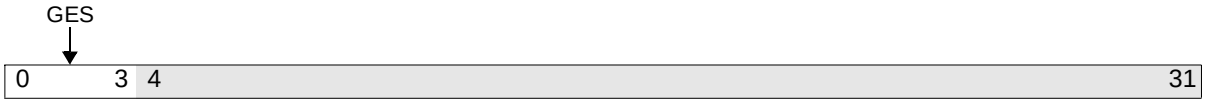


Figure 0-11. Generic Event Counter Selection Registers 0:3 (PPM0_GCSR0-PPM0_GCSR3)

0-3	GES	Generic Event Selection 0000 Selects G0_input 0001 Selects G1_input 0010 Selects G2_input 0011 Selects G3_input 0100 Selects write-pipeline-depth 1 0101 Selects write-pipeline-depth 2 0110 Selects read-pipeline-depth 1 0111 Selects read-pipeline-depth 2 1000 Selects read-pipeline-depth 3 1001 Selects read-pipeline-depth 4	Selects the external signal n, or the pipeline-depth n signal as input to this counter. **Note:** Using the pipeline depth selection(s) one can determine the depth of PLB pipelining that has been reached under specific applications.
4:31		Reserved	

Figure 29-270. Generic Event Counter Selection Registers 0:3 (PPM0_GCSR0-PPM0_GCSR3)

0-3	GES	Generic Event Selection 0000 Selects G0_input 0001 Selects G1_input 0010 Selects G2_input 0011 Selects G3_input 0100 Selects write-pipeline-depth 1 0101 Selects write-pipeline-depth 2 0110 Selects read-pipeline-depth 1 0111 Selects read-pipeline-depth 2 1000 Selects read-pipeline-depth 3 1001 Selects read-pipeline-depth 4	Selects the external signal n, or the pipeline-depth n signal as input to this counter. **Note:** Using the pipeline depth selection(s) one can determine the depth of PLB pipelining that has been reached under specific applications.
4:31		Reserved	

DCR Accessed using PPM0_CFGADDR; PPM0_CFGDATA; Offset 0x19, 0x1A, 0x1B, 0x1C R/W

See *Interrupt Status Register (PPM0_ISR)* on page 121.

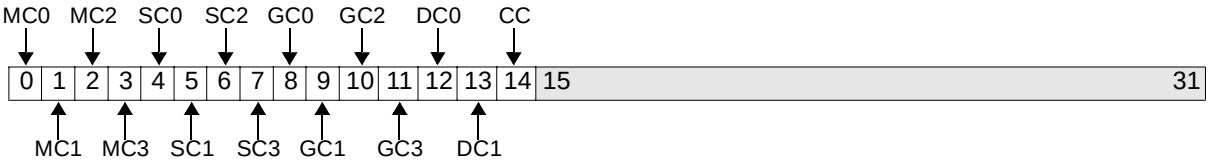


Figure 0-12. Interrupt Status Register (PPM0_ISR)

0	MC0	Master Counter 0 Interrupt Status 0 Master counter 0 interrupt did not occur 1 Master counter 0 interrupt occurred
1	MC1	Master Counter 1 Interrupt Status 0 Master counter 1 interrupt did not occur 1 Master counter 1 interrupt occurred
2	MC2	Master Counter 2 Interrupt Status 0 Master counter 2 interrupt did not occur 1 Master counter 2 interrupt occurred
3	MC3	Master Counter 3 Interrupt Status 0 Master counter 3 interrupt did not occur 1 Master counter 3 interrupt occurred
4	SC0	Slave Counter 0 Interrupt Status 0 Slave counter 0 interrupt did not occur 1 Slave counter 0 interrupt occurred
5	SC1	Slave Counter 1Interrupt Status 0 Slave counter 1 interrupt did not occur 1 Slave counter 1 interrupt occurred
6	SC2	Slave Counter 2 Interrupt Status 0 Slave counter 2 interrupt did not occur 1 Slave counter 2 interrupt occurred
7	SC3	Slave Counter 3 Interrupt Status 0 Slave counter 3 interrupt did not occur 1 Slave counter 3 interrupt occurred
8	GC0	Generic Counter 0 Interrupt Status 0 Generic counter 0 interrupt did not occur 1 Generic counter 0 interrupt occurred
9	GC1	Generic Counter 1 Interrupt Status 0 Generic counter 1 interrupt did not occur 1 Generic counter 1 interrupt occurred
10	GC2	Generic Counter 2 Interrupt Status 0 Generic counter 2 interrupt did not occur 1 Generic counter 2 interrupt occurred

11	GC3	Generic Counter 3 Interrupt Status 0 Generic counter 3 interrupt did not occur 1 Generic counter 3 interrupt occurred
12	DC0	Duration Counter 0 Interrupt Status 0 Duration counter 0 interrupt did not occur 1 Duration counter 0 interrupt occurred
13	DC1	Duration Counter 1 Interrupt Status 0 Duration counter 1 interrupt did not occur 1 Duration counter 1 interrupt occurred
14	CC	Cycle Counter Interrupt Status 0 Cycle counter interrupt did not occur 1 Cycle counter interrupt occurred
15:31		Reserved

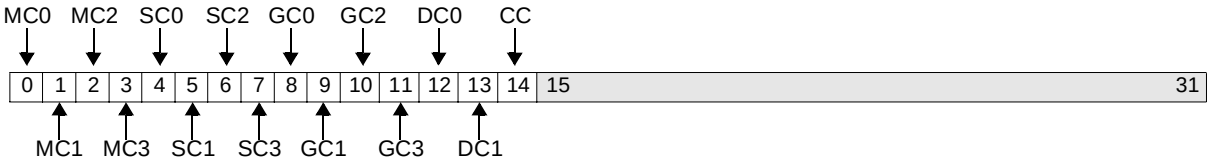


Figure 29-271. Interrupt Status Register (PPM0_ISR)

0	MC0	Master Counter 0 Interrupt Status 0 Master counter 0 interrupt did not occur 1 Master counter 0 interrupt occurred
1	MC1	Master Counter 1 Interrupt Status 0 Master counter 1 interrupt did not occur 1 Master counter 1 interrupt occurred
2	MC2	Master Counter 2 Interrupt Status 0 Master counter 2 interrupt did not occur 1 Master counter 2 interrupt occurred
3	MC3	Master Counter 3 Interrupt Status 0 Master counter 3 interrupt did not occur 1 Master counter 3 interrupt occurred
4	SC0	Slave Counter 0 Interrupt Status 0 Slave counter 0 interrupt did not occur 1 Slave counter 0 interrupt occurred
5	SC1	Slave Counter 1Interrupt Status 0 Slave counter 1 interrupt did not occur 1 Slave counter 1 interrupt occurred
6	SC2	Slave Counter 2 Interrupt Status 0 Slave counter 2 interrupt did not occur 1 Slave counter 2 interrupt occurred
7	SC3	Slave Counter 3 Interrupt Status 0 Slave counter 3 interrupt did not occur 1 Slave counter 3 interrupt occurred

8	GC0	Generic Counter 0 Interrupt Status 0 Generic counter 0 interrupt did not occur 1 Generic counter 0 interrupt occurred
9	GC1	Generic Counter 1 Interrupt Status 0 Generic counter 1 interrupt did not occur 1 Generic counter 1 interrupt occurred
10	GC2	Generic Counter 2 Interrupt Status 0 Generic counter 2 interrupt did not occur 1 Generic counter 2 interrupt occurred
11	GC3	Generic Counter 3 Interrupt Status 0 Generic counter 3 interrupt did not occur 1 Generic counter 3 interrupt occurred
12	DC0	Duration Counter 0 Interrupt Status 0 Duration counter 0 interrupt did not occur 1 Duration counter 0 interrupt occurred
13	DC1	Duration Counter 1 Interrupt Status 0 Duration counter 1 interrupt did not occur 1 Duration counter 1 interrupt occurred
14	CC	Cycle Counter Interrupt Status 0 Cycle counter interrupt did not occur 1 Cycle counter interrupt occurred
15:31		Reserved

DCR Accessed using PPM0_CFGADDR; PPM0_CFGDATA; Offset 0x7 R/W

See *Lower Address Mask Register (PPM0_LAMR)* on page 127.

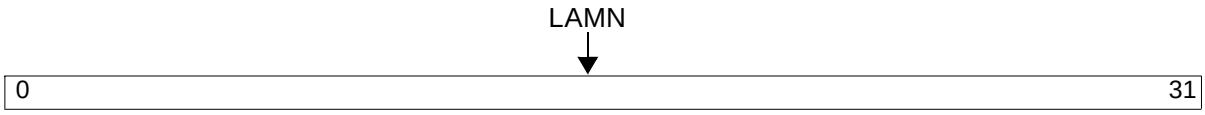


Figure 0-13. Lower Address Mask Register (PPM0_LAMR)

0:31	LAMn	Lower AddressBit-n Mask 0 Disable bit-n matching. 1 Enable bit-n matching.	If enabled, the corresponding bit in the LAR register is match against the same address bit-n of the PLB lower address bus.

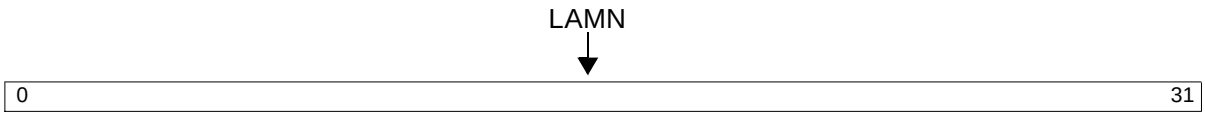


Figure 29-272. Lower Address Mask Register (PPM0_LAMR)

0:31	LAMn	Lower AddressBit-n Mask 0 Disable bit-n matching. 1 Enable bit-n matching.	If enabled, the corresponding bit in the LAR register is match against the same address bit-n of the PLB lower address bus.

DCR Accessed using PPM0_CFGADDR; PPM0_CFGDATA; Offset 0x5 R/W

See *Lower Address Register (PPM0_LAR)* on page 126.

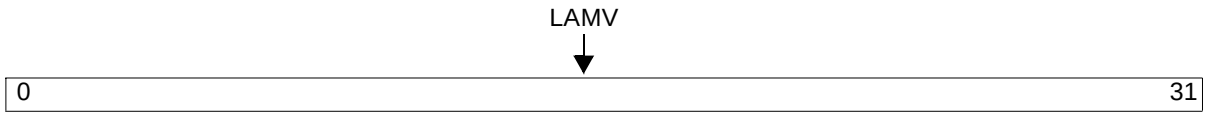


Figure 0-14. Lower Address Register (PPM0_LAR)

0:31	LAMV	Lower Address Match Value	If enabled, this value is match against the upper address bits of the PLB transaction event. Its contents can range from 0x00000000 to 0xFFFFFFF.

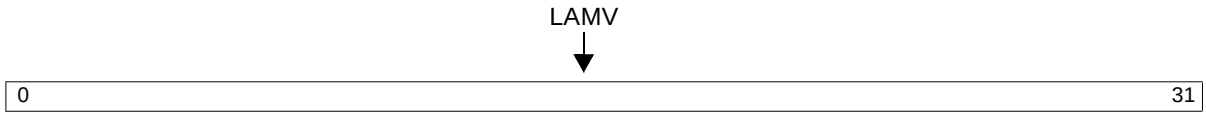


Figure 29-273. Lower Address Register (PPM0_LAR)

0:31	LAMV	Lower Address Match Value	If enabled, this value is match against the upper address bits of the PLB transaction event. Its contents can range from 0x00000000 to 0xFFFFFFF.

DCR Accessed using PPM0_CFGADDR; PPM0_CFGDATA; Offset 0x1D, 0x1E, 0x1F, 0x20 R/W

See *Master Event Counter Register 0:3 (PPM0_MCR0-PPM0_MCR3)* on page 133.

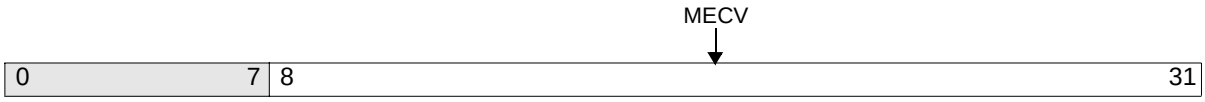


Figure 0-15. Master Event Counter Registers 0:3 (PPM0_MCR0-PPM0_MCR3)

0:7		Reserved	
8:31	MECV	Master Event Counter Value	This value can range from 0x000000 to 0xFFFFFF. Its contents are incremented once for each event occurrence as selected in its corresponding Mcounter Selection register. This register will only be updated if its corresponding MCn enable bit in the CR register is active.

Figure 29-274. Master Event Counter Registers 0:3 (PPM0_MCR0-PPM0_MCR3)

0:7		Reserved	
8:31	MECV	Master Event Counter Value	This value can range from 0x000000 to 0xFFFFFF. Its contents are incremented once for each event occurrence as selected in its corresponding Mcounter Selection register. This register will only be updated if its corresponding MCn enable bit in the CR register is active.

DCR Accessed using PPM0_CFGADDR; PPM0_CFGDATA; Offset 0x9, 0xA, 0xB, 0xC R/W

See *Master Event Counter Selection Register 0:3 (PPM0_MCSR0-PPM0_MCSR3)* on page 128.

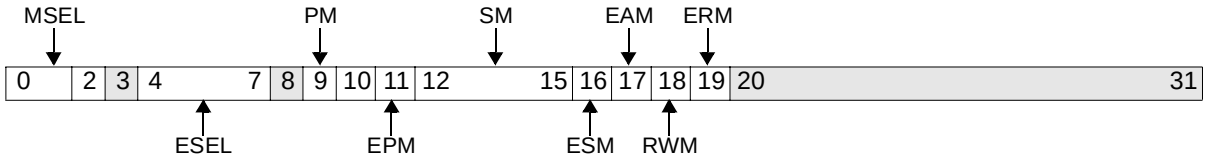


Figure 0-16. Master Event Counter Selection Register 0:7 (PPM0_MCSR0-PPM0_MCSR7)

0:2	MSEL	Master Selection 000 Selects Master 0 001 Selects Master 1 010 Selects Master 2 011 Selects Master 3 100 Selects Master 4 101 Selects Master 5 110 Selects Master 6 111 Selects Master 7	Selects which master's signal transitions to count with this counter set. Master 0 is ICU Read Master 1 is DCU Read Master 2 is DCU Write Master 3 is PCIX Master 4 is reserved Master 5 is MAL Master 6 is DMA Master 7 is OPB to PLB bridge
3		Reserved	
4:7	ESEL	Master Event Selection 0000 Selects PLB_MnAddrAck 0001 Selects PLB_MnRdDAck 0010 Selects PLB_MnWrDAck 0011 Selects PLB_MnRdErr 0100 Selects PLB_MnWrErr 0101 Selects PLB_MnRequest 0110 Selects PLB_MnAbort 0111 Selects PLB_MnBuslock 1000 Selects Mn_WrBurst 1001 Selects Mn_RdBurst 1010-1111 Reserved	Selects the master's PLB event to be tracked. (refer to Table 3-4, for signal event qualifications)
8		Reserved	
9:10	PM	Priority Matching 00 Lowest priority 01 Low priority 10 High priority 11 Highest priority	Sets the priorty decode value, which will be used if EPM bit is set.
11	EPM	Enable Priority Matching 0 Disable priority matching 1 Enable priority matching	Selects the priority bits to be used when tracking an event **Note:** This feature is functional when MES bits=0101 only. This bit is ignored for all other MES values.

12:15	SM	PLB transaction Size Matching 0000 One to 16 bytes 0001 4-word line 0010 8-word line 0011 16-word line 0100 Reserved 0101 Reserved 0110 Reserved 0111 Reserved 1000 Burst transfer 1001 Burst transfer 1010 Burst transfer 1011 Burst transfer 1100 Burst transfer 1101 Burst transfer 1110 Reserved 1111 Reserved	Sets the PLB transaction size decode value, which will be used during event tracking if the ESM bit is set.
16	ESM	Enable Size Matching 0 Disable transaction Size matching 1 Enable transaction Size matching	Selects the SM bits to be used when tracking an event **Note:** This feature is functional when MES bits=0000 only. This bit is ignored for all other values.
17	EAM	Enable Address Matching 0 Disable address matching feature 1 Enable address matching feature	Selects the contents of the UAR/LAR registers to be used when tracking an event **Note:** This feature is functional when MES bits=0000 only. This bit is ignored for all other values.
18	RWM	RnW Matching 0 Write tranastion matching 1 Read transaction matching	Selects the transaction matching criteria for Mn_Request transactions only.
19	ERM	Enable RnW Matching 0 Disable matching 1 Enable matching	Selects the priority bits to be used when tracking an event **Note:** This feature is functional when MES bits=0000 or 0110 only. This bit is ignored for all other values.
20:31		Reserved	

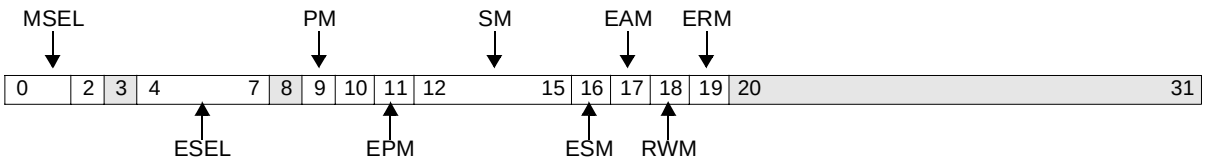


Figure 29-275. Master Event Counter Selection Register 0:7 (PPM0_MCSR0-PPM0_MCSR7)

0:2	MSEL	Master Selection 000 Selects Master 0 001 Selects Master 1 010 Selects Master 2 011 Selects Master 3 100 Selects Master 4 101 Selects Master 5 110 Selects Master 6 111 Selects Master 7	Selects which master's signal transitions to count with this counter set. Master 0 is ICU Read Master 1 is DCU Read Master 2 is DCU Write Master 3 is PCIX Master 4 is reserved Master 5 is MAL Master 6 is DMA Master 7 is OPB to PLB bridge
3		Reserved	
4:7	ESEL	Master Event Selection 0000 Selects PLB_MnAddrAck 0001 Selects PLB_MnRdDAck 0010 Selects PLB_MnWrDAck 0011 Selects PLB_MnRdErr 0100 Selects PLB_MnWrErr 0101 Selects PLB_MnRequest 0110 Selects PLB_MnAbort 0111 Selects PLB_MnBuslock 1000 Selects Mn_WrBurst 1001 Selects Mn_RdBurst 1010-1111 Reserved	Selects the master's PLB event to be tracked. (refer to Table 3-4, for signal event qualifications)
8		Reserved	
9:10	PM	Priority Matching 00 Lowest priority 01 Low priority 10 High priority 11 Highest priority	Sets the priorty decode value, which will be used if EPM bit is set.
11	EPM	Enable Priority Matching 0 Disable priority matching 1 Enable priority matching	Selects the priority bits to be used when tracking an event **Note:** This feature is functional when MES bits=0101 only. This bit is ignored for all other MES values.
12:15	SM	PLB transaction Size Matching 0000 One to 16 bytes 0001 4-word line 0010 8-word line 0011 16-word line 0100 Reserved 0101 Reserved 0110 Reserved 0111 Reserved 1000 Burst transfer 1001 Burst transfer 1010 Burst transfer 1011 Burst transfer 1100 Burst transfer 1101 Burst transfer 1110 Reserved 1111 Reserved	Sets the PLB transaction size decode value, which will be used during event tracking if the ESM bit is set.

16	ESM	Enable Size Matching 0 Disable transaction Size matching 1 Enable transaction Size matching	Selects the SM bits to be used when tracking an event **Note:** This feature is functional when MES bits=0000 only. This bit is ignored for all other values.
17	EAM	Enable Address Matching 0 Disable address matching feature 1 Enable address matching feature	Selects the contents of the UAR/LAR registers to be used when tracking an event **Note:** This feature is functional when MES bits=0000 only. This bit is ignored for all other values.
18	RWM	RnW Matching 0 Write tranastion matching 1 Read transaction matching	Selects the transaction matching criteria for Mn_Request transactions only.
19	ERM	Enable RnW Matching 0 Disable matching 1 Enable matching	Selects the priority bits to be used when tracking an event **Note:** This feature is functional when MES bits=0000 or 0110 only. This bit is ignored for all other values.
20:31		Reserved	

DCR Accessed using PPM0_CFGADDR; PPM0_CFGDATA; Offset 0x8 Read Only

See *Revision ID Register (PPM0_RIDR)* on page 128.

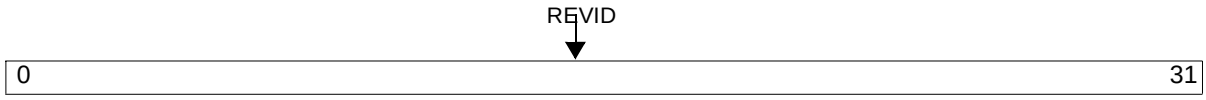


Figure 0-17. Revision ID Register (PPM0_RIDR)

0:31	REVID	RevID value: 0xC27E341x	Read Only field, where x is the Revision #. The default vaule for x in is 1.

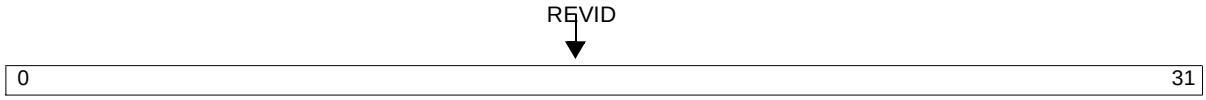


Figure 29-276. Revision ID Register (PPM0_RIDR)

0:31	REVID	RevID value: 0xC27E341x	Read Only field, where x is the Revision #. The default vaule for x in PPC440GP is 1.

DCR Accessed using PPM0_CFGADDR; PPM0_CFGDATA; Offset 0x25, 0x26,0x27,0x28 R/W

See *SLave Event Counter Register 0:3 (PPM0_SCR0-PPM0_SCR3)* on page 134.

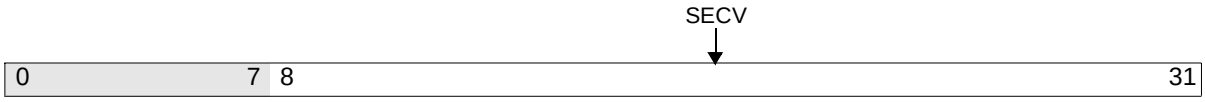


Figure 0-18. Slave Event Counter Registers 0:3 (PPM0_SCR0-PPM0_SCR3)

0:7		Reserved	
8:31	SECV	Slave Event Counter Value	This value can range from 0x000000 to 0xFFFFFF. Its contents are incremented once for each event occurrence as selected in its corresponding SCounter Selection register. This register can only be updated if its corresponding SCn enable bit in the CR register is active

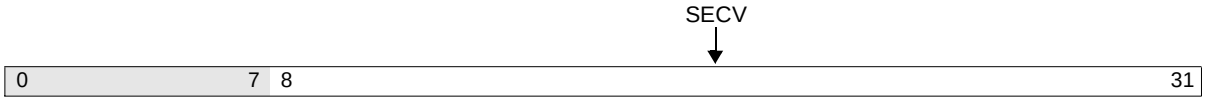


Figure 29-277. Slave Event Counter Registers 0:3 (PPM0_SCR0-PPM0_SCR3)

0:7		Reserved	
8:31	SECV	Slave Event Counter Value	This value can range from 0x000000 to 0xFFFFFF. Its contents are incremented once for each event occurrence as selected in its corresponding SCounter Selection register. This register can only be updated if its corresponding SCn enable bit in the CR register is active

DCR Accessed using PPM0_CFGADDR; PPM0_CFGDATA; Offset 0x11, 0x12,0x13,0x14 R/W

See *Slave Event Counter Selection Register 0:7 (PPM0_SCSR0-PPM0_SCSR7)* on page 130.

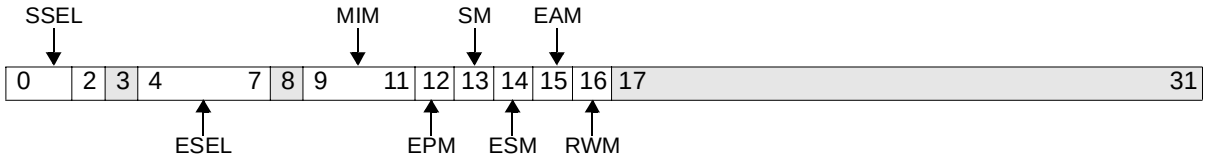


Figure 0-19. Slave Event Counter Selection Register 0:3 (PPM0_SCSR0-PPM0_SCSR3)

0:2	SSEL	Slave Selection 000 Selects Slave 0 001 Selects Slave 1 010 Selects Slave 2 011 Selects Slave 3 100 Selects Slave 4 101 Selects Slave 5 110 Selects Slave 6 111 Selects Slave 7	Selects which Slave's signal transitions to count with this counter set. Slave 0 is DDR_SDRAM Slave 1 is PCIX Slave 2 is SRAM Slave 3 is reserved Slave 4 is reserved SLave 5 is PLB to OPB bridge Slave 6 is reserved Slave 7 is reserved
3		Reserved	
4:7	ESEL	Slave Event Selection 0000 Selects PLB_SlnAddrAck 0001 Selects PLB_SlnRdDAck 0010 Selects PLB_SlnWrDAck 0011 Selects PLB_SlnRearbitrate 0100 Selects Sln_Wait 0101 Selects Sln_WrComp 0110 Selects Sln_RdComp 0111-1111 Reserved	Selects the PLB slave event to be tracked.

8		Reserved	
9:11	MIM	Master ID Matching 000 Selects Master 0 001 Selects Master 1 010 Selects Master 2 011 Selects Master 3 100 Selects Master 4 101 Selects Master 5 110 Selects Master 6 111 Selects Master 7	Sets the MasterID decode value, which will be used if EMIM bit is set. Master 0 is ICU Read Master 1 is DCU Read Master 2 is DCU Write Master 3 is PCIX Master 4 is Reserved Master 5 is MAL Master 6 is DMA Master 7 is OPB to PLB Bridge
12	EMIM	Enable Master ID Matching 0 Disable master ID matching 1 Enable master ID matching	Selects the MIM bits to be used when tracking an event **Note:** This feature is functional when SES bits=0000 only. This bit is ignored for all other SES values.
13	RWM	RnW Matching 0 Write tranastion matching 1 Read transaction matching	Selects the Read or Write value that will be used when ERM bit is set.
14	ERM	Enable RnW Matching 0 Disable matching 1 Enable matching	Selects the RWM bit to be used when tracking an event **Note:** This feature is functional when SES bits=0000 or 0011 only. This bit is ignored for all other SES values.
15	AVM	Address Valid Matching 0 PAValid matching 1 SAValid matching	Selects either the PAValid or SAValid signals to used when the EVM bit is set.
16	EVM	Enable Address Valid Matching 0 Disable matching 1 Enable matching	Enables the Primary of Secondary Address Valid matching feature to be used when tracking an event **Note:** This feature is functional when SES bits= 0011 only. This bit is ignored for all other SES values.
17:31		Reserved	

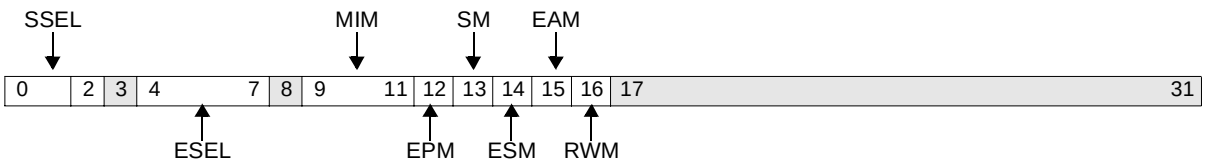


Figure 29-278. Slave Event Counter Selection Register 0:3 (PPM0_SCSR0-PPM0_SCSR3)

0:2	SSEL	Slave Selection 000 Selects Slave 0 001 Selects Slave 1 010 Selects Slave 2 011 Selects Slave 3 100 Selects Slave 4 101 Selects Slave 5 110 Selects Slave 6 111 Selects Slave 7	Selects which Slave's signal transitions to count with this counter set. Slave 0 is DDR_SDRAM Slave 1 is PCIX Slave 2 is SRAM Slave 3 is reserved Slave 4 is reserved SLave 5 is PLB to OPB bridge Slave 6 is reserved Slave 7 is reserved
3		Reserved	
4:7	ESEL	Slave Event Selection 0000 Selects PLB_SlnAddrAck 0001 Selects PLB_SlnRdDAck 0010 Selects PLB_SlnWrDAck 0011 Selects PLB_SlnRearbitrate 0100 Selects Sln_Wait 0101 Selects Sln_WrComp 0110 Selects Sln_RdComp 0111-1111 Reserved	Selects the PLB slave event to be tracked.

8		Reserved	
9:11	MIM	Master ID Matching 000 Selects Master 0 001 Selects Master 1 010 Selects Master 2 011 Selects Master 3 100 Selects Master 4 101 Selects Master 5 110 Selects Master 6 111 Selects Master 7	Sets the MasterID decode value, which will be used if EMIM bit is set. Master 0 is ICU Read Master 1 is DCU Read Master 2 is DCU Write Master 3 is PCIX Master 4 is Reserved Master 5 is MAL Master 6 is DMA Master 7 is OPB to PLB Bridge
12	EMIM	Enable Master ID Matching 0 Disable master ID matching 1 Enable master ID matching	Selects the MIM bits to be used when tracking an event **Note:** This feature is functional when SES bits=0000 only. This bit is ignored for all other SES values.
13	RWM	RnW Matching 0 Write tranastion matching 1 Read transaction matching	Selects the Read or Write value that will be used when ERM bit is set.
14	ERM	Enable RnW Matching 0 Disable matching 1 Enable matching	Selects the RWM bit to be used when tracking an event **Note:** This feature is functional when SES bits=0000 or 0011 only. This bit is ignored for all other SES values.
15	AVM	Address Valid Matching 0 PAValid matching 1 SAValid matching	Selects either the PAValid or SAValid signals to used when the EVM bit is set.
16	EVM	Enable Address Valid Matching 0 Disable matching 1 Enable matching	Enables the Primary of Secondary Address Valid matching feature to be used when tracking an event **Note:** This feature is functional when SES bits= 0011 only. This bit is ignored for all other SES values.
17:31		Reserved	

DCR Accessed using PPM0_CFGADDR; PPM0_CFGDATA; Offset 0x6 R/W

See *Upper Address Mask Register (PPM0_UAMR)* on page 127.

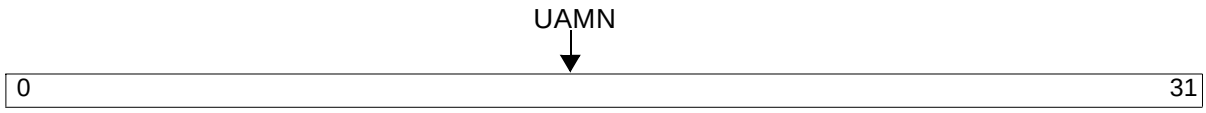


Figure 0-20. Upper Address Mask Register (PPM0_UAMR)

0:31	UAMn	Upper Address Bit-n Mask 0 Disable bit-n matching. 1 Enable bit-n matching.	If enabled, the corresponding bit in theUAR register is match against the same address bit-n of the PLB upper address bus.

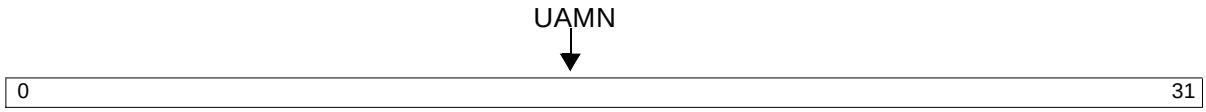


Figure 29-279. Upper Address Mask Register (PPM0_UAMR)

0:31	UAMn	Upper Address Bit-n Mask 0 Disable bit-n matching. 1 Enable bit-n matching.	If enabled, the corresponding bit in theUAR register is match against the same address bit-n of the PLB upper address bus.

DCR Accessed using PPM0_CFGADDR; PPM0_CFGDATA; Offset 0x4 R/W

See *Upper Address Register (PPM0_UAR)* on page 125.

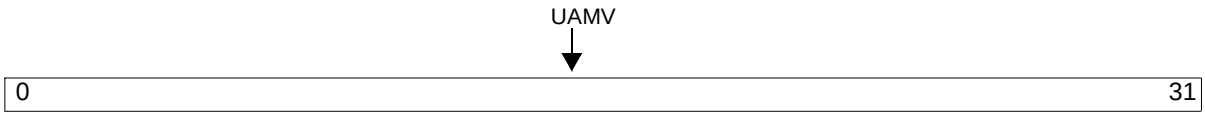


Figure 0-21. Upper Address Register (PPM0_UAR)

0:31	UAMV	Upper Address Match Value	If enabled, this value is matched against the upper PLB address bits UAbus(0-31) of the PLB transaction event.

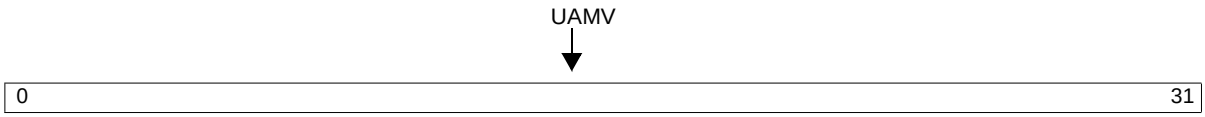


Figure 29-280. Upper Address Register (PPM0_UAR)

0:31	UAMV	Upper Address Match Value	If enabled, this value is matched against the upper PLB address bits UAbus(0-31) of the PLB transaction event.

DCR Accessed using SDRAM0_CFGADDR; SDRAM0_CFGDATA; Offset 0x40–0x4C R/W

See *Memory 0 - 3 Configuration (SDRAM0_B0CR-SDRAM0_B3CR)* on page 496.

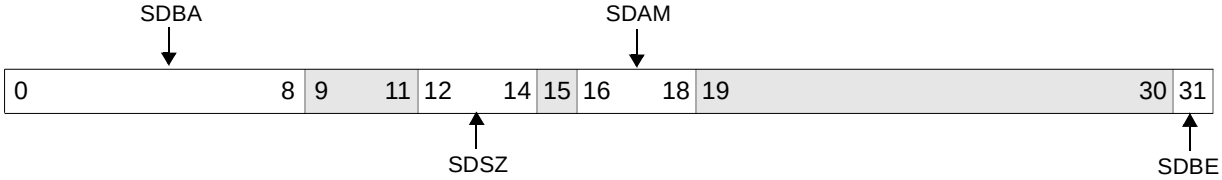


Figure 0-1. Memory 0-3 Configuration (SDRAM0_B0CR-SDRAM0_B3CR)

0:8	SDBA	Base Address	
9:11		Reserved	
12:14	SDSZ	Size 000 Reserved 001 8 MB 010 16 MB 011 32 MB 100 64 MB 101 128 MB 110 256 MB 111 512 MB	
15		Reserved	
16:18	SDAM	Addressing Mode 000 Mode 1 001 Mode 2 010 Mode 3 011 Mode 4 1xx Reserved	See the table on “DDR SDRAM Addressing Modes” on next page for details.
19:30		Reserved	
31	SDBE	Memory Bank Enable	

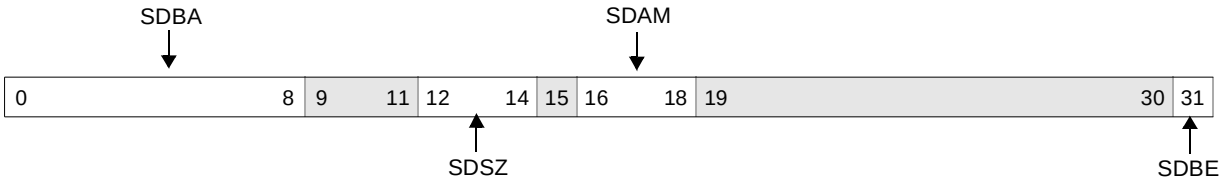


Figure 29-281. Memory 0-3 Configuration (SDRAM0_B0CR-SDRAM0_B3CR)

0:8	SDBA	Base Address	
9:11		Reserved	

12:14	SDSZ	Size 000 Reserved 001 8 MB 010 16 MB 011 32 MB 100 64 MB 101 128 MB 110 256 MB 111 512 MB	
15		Reserved	
16:18	SDAM	Addressing Mode 000 Mode 1 001 Mode 2 010 Mode 3 011 Mode 4 1xx Reserved	See the table on "DDR SDRAM Addressing Modes" on next page for details.
19:30		Reserved	
31	SDBE	Memory Bank Enable	

DCR Accessed using SDRAM0_CFGADDR; SDRAM0_CFGDATA; Offset 0x10 Read-Only

See *Master Bus Error Address Register (SDRAM0_BEAR)* on page 484.

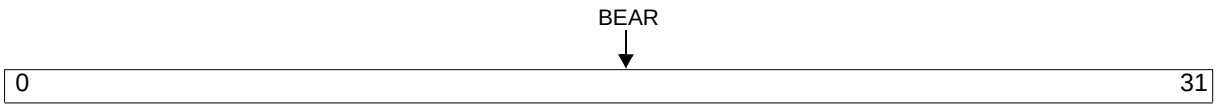


Figure 0-2. Bus Error Address Register (SDRAM0_BEAR)

0:31	BEAR	Address of Bus Error

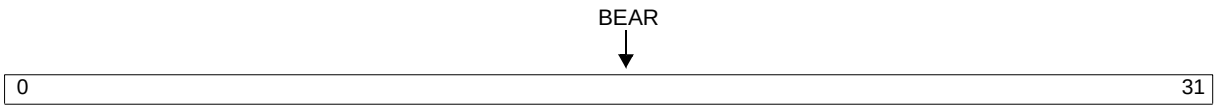


Figure 29-282. Bus Error Address Register (SDRAM0_BEAR)

0:31	BEAR	Address of Bus Error

DCR Accessed using SDRAM0_CFGADDR; SDRAM0_CFGDATA; Offset 0x00 R/W

See *Bus Error Syndrome Register 0 (SDRAM0_BESR0)* on page 480.

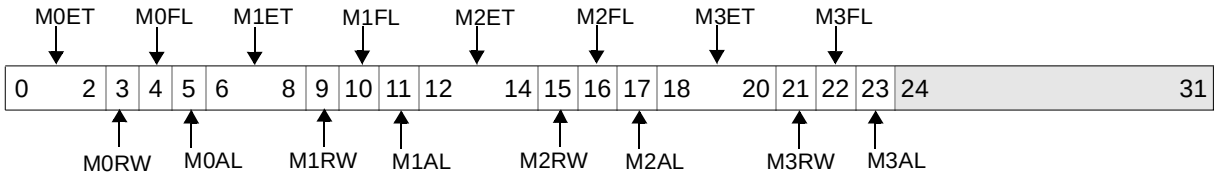


Figure 0-3. Bus Error Syndrome Register 0 (SDRAM0_BESR0)

0:2	M0ET	Error Type for 440 CPU Instruction Cache Controller 000 No Error 001 Reserved 01x ECC Uncorrectable Error 1xx Reserved
3	M0RW	Read/Write Status for 440 CPU Instruction Cache Controller (Read Access only) 0 Write Error 1 Read Error
4	M0FL	BESR Field Lock for 440 CPU Instruction Cache Controller 0 BESR0 Unlocked 1 BESR0 Locked
5	M0AL	BEAR Address Lock for 440 CPU Instruction Cache Controller 0 BEAR Not Locked by 440 CPU Instruction Cache Controller 1 BEAR Locked by 440 CPU Instruction Cache Controller
6:8	M1ET	Error Type for 440 CPU Data Cache Read Controller 000 No Error 001 Reserved 01x ECC Uncorrectable Error 1xx Reserved
9	M1RW	Read/Write Status for 440 CPU Data Cache Controller (Read Access only) 0 Reserved 1 Read Error
10	M1FL	BESR Field Lock for 440 CPU Data Cache Controller 0 BESR1 Unlocked 1 BESR1 Locked
11	M1AL	BEAR Address Lock for 440 CPU Data Cache Controller 0 BEAR Not Locked by 440 CPU Data Cache Controller 1 BEAR Locked by 440 CPU Data Cache Controller
12:14	M2ET	Error Type for 440 CPU Data Cache Write Controller 000 No Error 001 Reserved 01x ECC Uncorrectable Error 1xx Reserved

15	M2RW	Read/Write Status for 440 CPU Data Cache Controller (Write Access only) 0 Write Error 1 Reserved
16	M2FL	BESR Field Lock for 440 CPU Data Cache Controller 0 BESR2 Unlocked 1 BESR2 Locked
17	M2AL	BEAR Address Lock for 440 CPU Data Cache Controller 0 BEAR Not Locked 1 BEAR Locked
18:20	M3ET	Error Type for PLB-PCIX Bridge 000 001 Reserved 01x ECC Uncorrectable Error 1xx Reserved
21	M3RW	Read/Write Status for PLB-PCIX Bridge 0 Write Error 1 Read Error
22	M3FL	BESR Field Lock for Read/Write Status for PLB-PCIX Bridge 0 BESR3 Unlocked 1 BESR3 Locked
23	M3AL	BEAR Address Lock for Read/Write Status for PLB-PCIX Bridge 0 BEAR Not Locked by Read/Write Status for PLB-PCIX Bridge 1 BEAR Locked by Read/Write Status for PLB-PCIX Bridge
24:31		Reserved

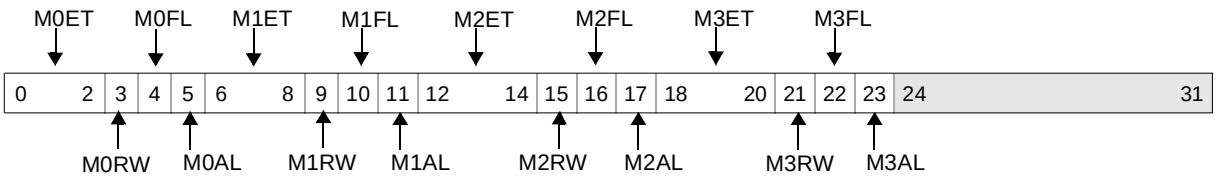


Figure 29-283. Bus Error Syndrome Register 0 (SDRAM0_BESR0)

0:2	M0ET	Error Type for 440 CPU Instruction Cache Controller 000 No Error 001 Reserved 01x ECC Uncorrectable Error 1xx Reserved
3	M0RW	Read/Write Status for 440 CPU Instruction Cache Controller (Read Access only) 0 Write Error 1 Read Error
4	M0FL	BESR Field Lock for 440 CPU Instruction Cache Controller 0 BESR0 Unlocked 1 BESR0 Locked

5	M0AL	BEAR Address Lock for 440 CPU Instruction Cache Controller 0 BEAR Not Locked by 440 CPU Instruction Cache Controller 1 BEAR Locked by 440 CPU Instruction Cache Controller
6:8	M1ET	Error Type for 440 CPU Data Cache Read Controller 000 No Error 001 Reserved 01x ECC Uncorrectable Error 1xx Reserved
9	M1RW	Read/Write Status for 440 CPU Data Cache Controller (Read Access only) 0 Reserved 1 Read Error
10	M1FL	BESR Field Lock for 440 CPU Data Cache Controller 0 BESR1 Unlocked 1 BESR1 Locked
11	M1AL	BEAR Address Lock for 440 CPU Data Cache Controller 0 BEAR Not Locked by 440 CPU Data Cache Controller 1 BEAR Locked by 440 CPU Data Cache Controller
12:14	M2ET	Error Type for 440 CPU Data Cache Write Controller 000 No Error 001 Reserved 01x ECC Uncorrectable Error 1xx Reserved
15	M2RW	Read/Write Status for 440 CPU Data Cache Controller (Write Access only) 0 Write Error 1 Reserved
16	M2FL	BESR Field Lock for 440 CPU Data Cache Controller 0 BESR2 Unlocked 1 BESR2 Locked
17	M2AL	BEAR Address Lock for 440 CPU Data Cache Controller 0 BEAR Not Locked 1 BEAR Locked
18:20	M3ET	Error Type for PLB-PCIX Bridge 000 001 Reserved 01x ECC Uncorrectable Error 1xx Reserved
21	M3RW	Read/Write Status for PLB-PCIX Bridge 0 Write Error 1 Read Error
22	M3FL	BESR Field Lock for Read/Write Status for PLB-PCIX Bridge 0 BESR3 Unlocked 1 BESR3 Locked
23	M3AL	BEAR Address Lock for Read/Write Status for PLB-PCIX Bridge 0 BEAR Not Locked by Read/Write Status for PLB-PCIX Bridge 1 BEAR Locked by Read/Write Status for PLB-PCIX Bridge
24:31		Reserved

DCR Accessed using SDRAM0_CFGADDR; SDRAM0_CFGDATA; Offset 0x08 R/W

See *Bus Error Syndrome Register 1 (SDRAM0_BESR1)* on page 482.

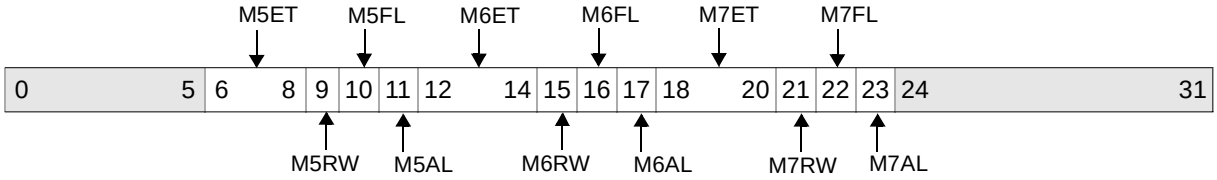


Figure 0-4. Bus Error Syndrome Register 1 (SDRAM0_BESR1)

0:5		Reserved
6:8	M5ET	Error Type for Memory Access Layer (MAL) 000 No Error 001 Reserved 01x ECC Uncorrectable Error 1xx Reserved
9	M5RW	Read/Write Status for MAL 0 Write Error 1 Read Error
10	M5FL	BESR Field Lock for MAL 0 BESR5 Unlocked 1 BESR5 Locked
11	M5AL	BEAR Address Lock for MAL 0 BEAR Not Locked by MAL 1 BEAR Locked by MAL
12:14	M6ET	Error Type for DMA Controller 000 No Error 001 Reserved 01x ECC Uncorrectable Error 1xx Reserved
15	M6RW	Read/Write Status for DMA Controller 0 Write Error 1 Read Error
16	M6FL	BESR Field Lock for DMA Controller 0 BESR6 Unlocked 1 BESR6 Locked
17	M6AL	BEAR Address Lock for DMA Controller 0 BEAR Not Locked by DMA Controller 1 BEAR Locked by DMA Controller
18:20	M7ET	Error Type for OPB to PLB Bridge 000 No Error 001 Reserved 01x ECC Uncorrectable Error 1xx Reserved

21	M7RW	Read/Write Status for OPB to PLB Bridge 0 Write Error 1 Read Error
22	M7FL	BESR Field Lock for OPB to PLB Bridge 0 BESR7 Unlocked 1 BESR7 Locked
23	M7AL	BEAR Address Lock for OPB to PLB Bridge 0 BEAR Not Locked by OPB to PLB Bridge 1 BEAR Locked by OPB to PLB Bridge
24:31		Reserved

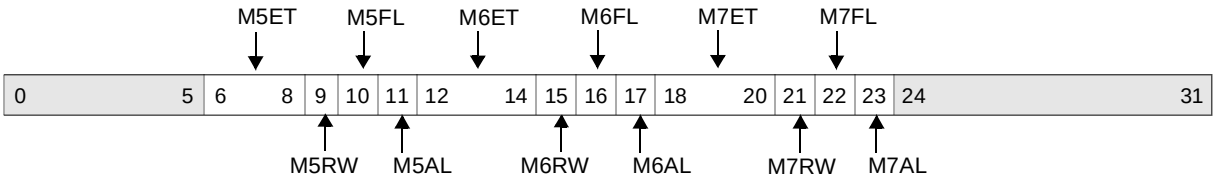


Figure 29-284. Bus Error Syndrome Register 1 (SDRAM0_BESR1)

0:5		Reserved
6:8	M5ET	Error Type for Memory Access Layer (MAL) 000 No Error 001 Reserved 01x ECC Uncorrectable Error 1xx Reserved
9	M5RW	Read/Write Status for Memory Access Layer (MAL) 0 Write Error 1 Read Error
10	M5FL	BESR Field Lock for Memory Access Layer (MAL) 0 BESR5 Unlocked 1 BESR5 Locked
11	M5AL	BEAR Address Lock for Memory Access Layer (MAL) 0 BEAR Not Locked by MAL 1 BEAR Locked by MAL
12:14	M6ET	Error Type for DMA Controller 000 No Error 001 Reserved 01x ECC Uncorrectable Error 1xx Reserved
15	M6RW	Read/Write Status for DMA Controller 0 Write Error 1 Read Error
16	M6FL	BESR Field Lock for DMA Controller 0 BESR6 Unlocked 1 BESR6 Locked

17	M6AL	BEAR Address Lock for DMA Controller 0 BEAR Not Locked by DMA Controller 1 BEAR Locked by DMA Controller
18:20	M7ET	Error Type for OPB to PLB Bridge 000 No Error 001 Reserved 01x ECC Uncorrectable Error 1xx Reserved
21	M7RW	Read/Write Status for OPB to PLB Bridge 0 Write Error 1 Read Error
22	M7FL	BESR Field Lock for OPB to PLB Bridge 0 BESR7 Unlocked 1 BESR7 Locked
23	M7AL	BEAR Address Lock for OPB to PLB Bridge 0 BEAR Not Locked by OPB to PLB Bridge 1 BEAR Locked by OPB to PLB Bridge
24:31		Reserved

DCR Accessed using SDRAM0_CFGADDR; SDRAM0_CFGDATA; Offset 0x20 R/W

See *Memory Controller Options 0 (SDRAM0_CFG0)* on page 487.

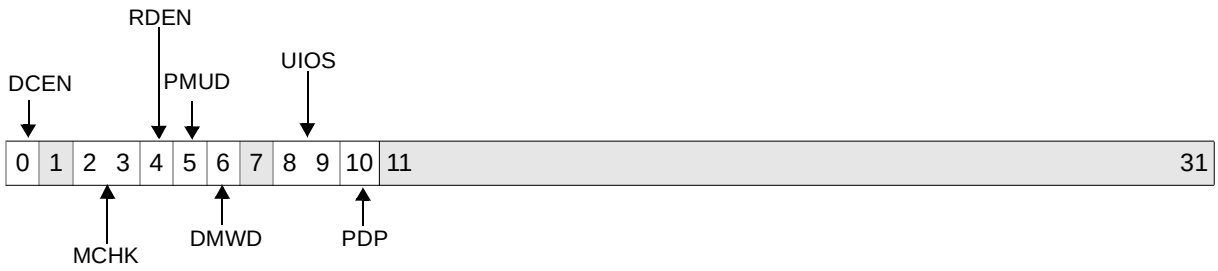


Figure 0-5. Memory Controller Options 0 (SDRAM0_CFG0)

0	DCEN	DDR SDRAM Controller Enable 0 Disable 1 Enable	Once enabled, DDR SDRAM will be initialized using the power-on sequence and subsequently available for access. All DDR SDRAM controller configuration registers should be initialized and valid when this is enabled.
1		Reserved	
2:3	MCHK	Memory Data Error Checking 00 None 01 Reserved 10 ECC generation only (no checking or correction) 11 ECC checking and correction	
4	RDEN	Registered DIMM Enable 0 Disable 1 Enable	
5	PMU	Page Management Unit 0 Enable 1 Disable	When PMU is enabled, up to 8 open pages will be maintained for subsequent accesses. When PMU is disabled, the accessed page will be opened for a given access, remain open for the duration of the access (allowing page hits within PLB sequential bursts) and be precharged (using read/write with autoprecharge command) upon completion of the access.
6	DMWD	DDR SDRAM Width 0 32-bit 1 64-bit	
7		Reserved	

8:9	UIOS[1]	Unused I/O State 00 Active (High-Z on Read, Driven on Write) -switching DQS 01 Active (High-Z on Read, Driven on Write) -static values 10 Static (Always Driven) - static values 11 High-Z	This field can be used to control the state of the unused I/O for various configurations through the corresponding I/O driver data and enable signals. Unused I/O are defined as the I/O associated with a function that is not enabled. Example: CB[0:7], DM[8], and DQS[8] would be unused I/O when ECC is not enabled. Likewise, DATA[32:63], DM[4:7], and DQS[4:7] would be unused I/O in 32-bit mode. All unused I/O receivers will be gated off.
10	PDP	Page Deallocation Policy 0 Pseudo Least Recently Allocated 1 Least Recently Used	This field selects the page deallocation policy when page mode is enabled. Page deallocation occurs when the PMU has 8 open pages and an access does not target a page or bank address associated with one of those open pages. A PMU entry must be deallocated or replaced (and the corresponding page closed). With PMU_DP=0, the PMU entry that has been open longest will be deallocated. With PMU_DP=1, the page that was least recently used (least recently accessed) will be deallocated. **Note:** This distinction is only relevant for memory subsystems employing more than two physical banks (chip selects) of memory.
11:31		Reserved	

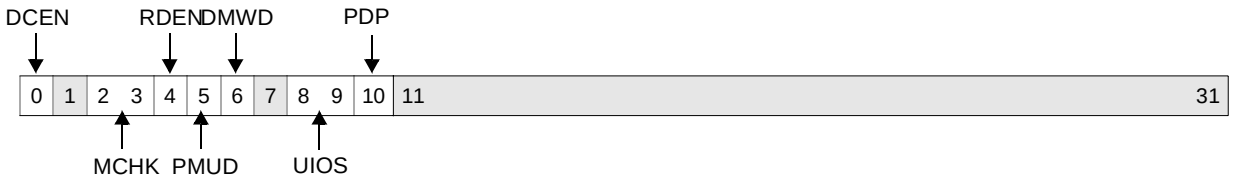


Figure 29-285. Memory Controller Options 0 (SDRAM0_CFG0)

0	DCEN	DDR SDRAM Controller Enable 0 Disable 1 Enable	Once enabled, DDR SDRAM will be initialized using the power-on sequence and subsequently available for access. All DDR SDRAM controller configuration registers should be initialized and valid when this is enabled.
1		Reserved	
2:3	MCHK	Memory Data Error Checking 00 None 01 Reserved 10 ECC generation only (no checking or correction) 11 ECC checking and correction	
4	RDEN	Registered DIMM Enable 0 Disable 1 Enable	
5	PMU	Page Management Unit 0 Enable 1 Disable	When PMU is enabled, up to 8 open pages will be maintained for subsequent accesses. When PMU is disabled, the accessed page will be opened for a given access, remain open for the duration of the access (allowing page hits within PLB sequential bursts) and be precharged (using read/write with autoprecharge command) upon completion of the access.
6	DMWD	DDR SDRAM Width 0 32-bit 1 64-bit	
7		Reserved	
8:9	UIOS[1]	Unused I/O State 00 Active (High-Z on Read, Driven on Write) - switching DQS 01 Active (High-Z on Read, Driven on Write) -static values 10 Static (Always Driven) - static values 11 High-Z	This field can be used to control the state of the unused I/O for various configurations through the corresponding I/O driver data and enable signals. Unused I/O are defined as the I/O associated with a function that is not enabled. Example: CB[0:7], DM[8], and DQS[8] would be unused I/O when ECC is not enabled. Likewise, DATA[32:63], DM[4:7], and DQS[4:7] would be unused I/O in 32-bit mode. All unused I/O receivers will be gated off.

10	PDP	Page Deallocation Policy 0 Pseudo Least Recently Allocated 1 Least Recently Used	This field selects the page deallocation policy when page mode is enabled. Page deallocation occurs when the PMU has 8 open pages and an access does not target a page or bank address associated with one of those open pages. A PMU entry must be deallocated or replaced (and the corresponding page closed). With PMU_DP=0, the PMU entry that has been open longest will be deallocated. With PMU_DP=1, the page that was least recently used (least recently accessed) will be deallocated. **Note: Note:** This distinction is only relevant for memory subsystems employing more than two physical banks (chip selects) of memory.
11:31		Reserved	

DCR Accessed using SDRAM0_CFGADDR; SDRAM0_CFGDATA; Offset 0x21 R/W

See *Memory Controller Options 1 (SDRAM0_CFG1)* on page 490.

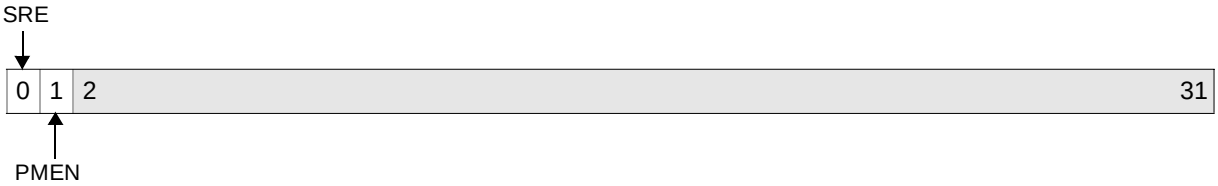


Figure 0-6. Memory Controller Options 1 (SDRAM0_CFG1)

0	SRE	Self-Refresh Entry 0 Exit 1 Enter	This bit may be set by software (using DCR write). Once set, the SDRAM is maintained in self-refresh mode until this bit is reset by software. The DDR_SDRAM controller responds to this bit independent of the state of SDRAM0_CFG0[DCEN].
1	PMEN	Power Management Enable 0 Disable 1 Enable	Refer to section on “Power Management” of this document for specific details.
2:31		Reserved	

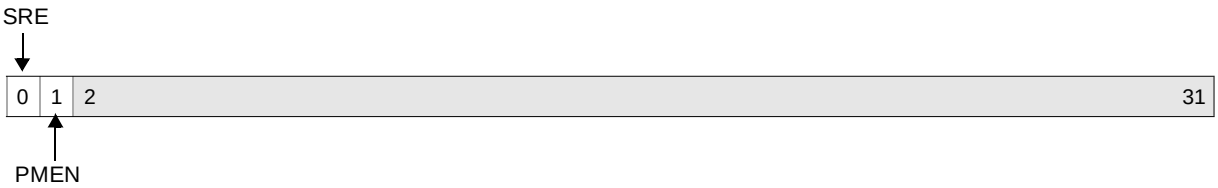


Figure 29-286. Memory Controller Options 1 (SDRAM0_CFG1)

0	SRE	Self-Refresh Entry 0 Exit 1 Enter	This bit may be set by software (using DCR write). Once set, the SDRAM is maintained in self-refresh mode until this bit is reset by software. The DDR_SDRAM controller responds to this bit independent of the state of SDRAM0_CFG0[DCEN].
1	PMEN	Power Management Enable 0 Disable 1 Enable	Refer to section on “Power Management” of this document for specific details.
2:31		Reserved	

DCR Accessed using SDRAM0_CFGADDR; SDRAM0_CFGDATA; Offset 0xA4 Read/Only

See *Controller ID Register (SDRAM0_CID)* on page 515.

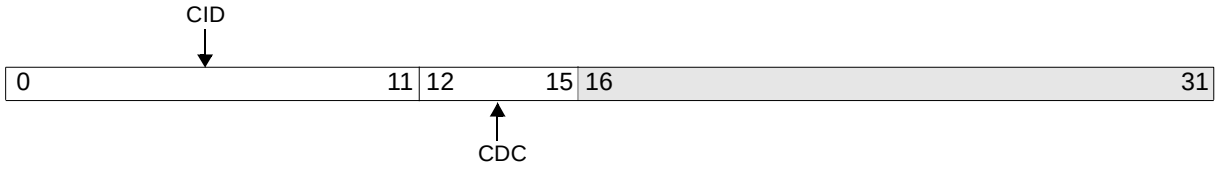


Figure 0-7. Controller ID Register (SDRAM0_CID)

0:11	CID	Core ID identifies the IBM specific core number associated with the DDR SDRAM.
12:15	CDC	Core Derivative Code B Base Library Core D Derivative of Base Library Core
16:31		Reserved

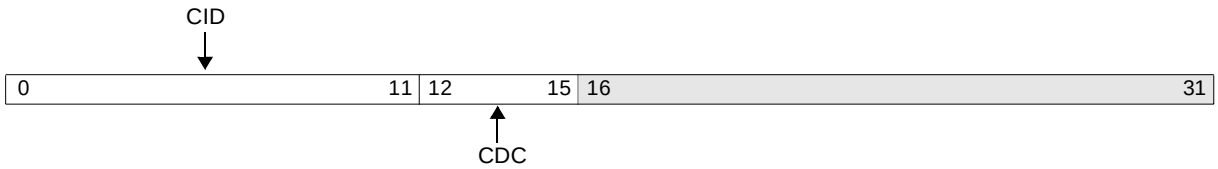


Figure 29-287. Controller ID Register (SDRAM0_CID)

0:11	CID	Core ID identifies the IBM specific core number associated with the DDR SDRAM.
12:15	CDC	Core Derivative Code B Base Library Core D Derivative of Base Library Core
16:31		Reserved

DCR Accessed using SDRAM0_CFGADDR; SDRAM0_CFGDATA; Offset 0x82 R/W

See *DDR SDRAM Clock Timing Register (SDRAM0_CLKTR)* on page 507.

Figure 0-8. DDR_SDRAM Clock Timing Register (SDRAM0_CLKTR)

0:1	CLKP	Write Clock Phase 00 Advance 0 degrees 01 Advance 90 degrees 10 Advance 180 degrees 11 Reserved	In most applications, CLKP should be set to 90 degrees phase advance.
2:22		Reserved	
23:31	DCDT	DDR Clock Delay Tuning Bit 23 is Most Significant Bit; bit 31 is Least Significant Bit. Bits 23:29 select increments of 1 full delay element. Bits 30:31 select increments of 1/4 delay element.	This field should never be programmed to exceed a delay equivalent to 1/4 the MemClkOut0 frequency.

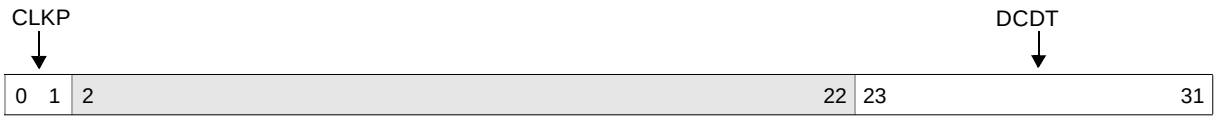


Figure 29-288. DDR_SDRAM Clock Timing Register (SDRAM0_CLKTR)

0:1	CLKP	Write Clock Phase 00 Advance 0 degrees 01 Advance 90 degrees 10 Advance 180 degrees 11 Reserved	In most applications, CLKP should be set to 90 degrees phase advance.
2:22		Reserved	
23:31	DCDT	DDR Clock Delay Tuning Bit 23 is Most Significant Bit; bit 31 is Least Significant Bit. Bits 23:29 select increments of 1 full delay element. Bits 30:31 select increments of 1/4 delay element.	This field should never be programmed to exceed a delay equivalent to 1/4 the MemClkOut0 frequency.

DCR Accessed using SDRAM0_CFGADDR; SDRAM0_CFGDATA; Offset 0x22 R/W

See *DDR SDRAM Device Options (SDRAM0_DEVOPT)* on page 491.

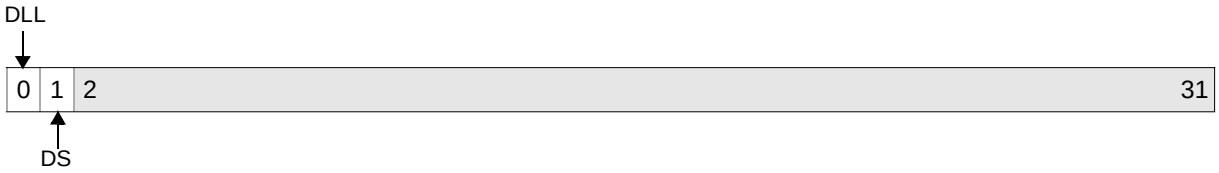


Figure 0-9. DDR_SDRAM Device Options (SDRAM0_DEVOPT)

0	DLL	DDR_SDRAM Device DLL Disable 0 Enable 1 Disable	For normal operation, enable DLL. The DLL setting controls the state of address bit during the initialization of the Extended Mode Register.
1	DS	DDR_SDRAM Device I/O Drive Strength 0 Normal 1 Weak	For normal operation, clear DS. The DS setting controls the state of address bit MemAddr1 during the initialization of the Extended Mode Register.
2:31		Reserved	

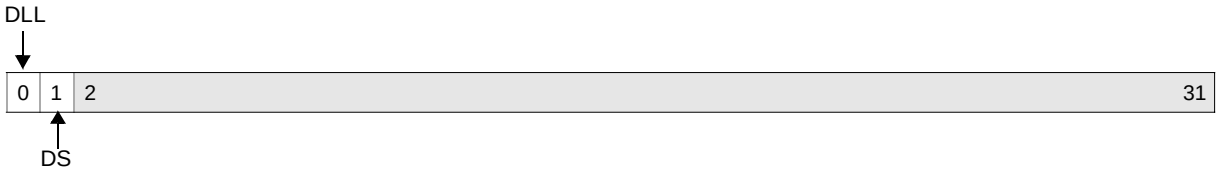


Figure 29-289. DDR_SDRAM Device Options (SDRAM0_DEVOPT)

0	DLL	DDR_SDRAM Device DLL Disable 0 Enable 1 Disable	For normal operation, enable DLL. The DLL setting controls the state of address bit during the initialization of the Extended Mode Register.
1	DS	DDR_SDRAM Device I/O Drive Strength 0 Normal 1 Weak	For normal operation, clear DS. The DS setting controls the state of address bit MemAddr1 during the initialization of the Extended Mode Register.
2:31		Reserved	

DCR Accessed using SDRAM0_CFGADDR; SDRAM0_CFGDATA; Offset 0x84 R/W

See *Delay Line Calibration Register (SDRAM0_DLYCAL)* on page 513.

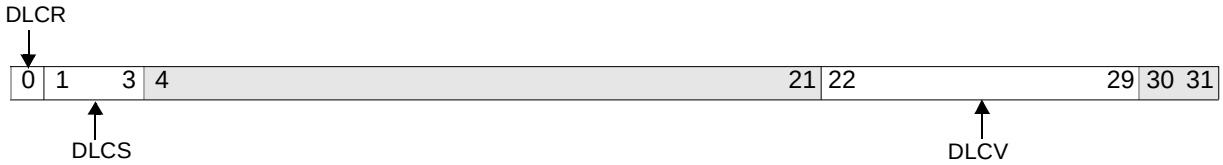


Figure 0-10. Delay Line Calibration Register (SDRAM0_DLYCAL)

0	DLCR	Delay Line Calibration Request 0 Delay Line Calibration not requested 1 Delay Line Calibration requested	Setting DLCR bit initiates the delay line calibration. During the re-initialization process, the DLCR bit is cleared. The delay line calibration occurs automatically following SysReset or upon request by setting the DLCR bit. The results of the calibration are provided to assist in the setting of SDRAM0_CLKTR, SDRAM0_WRDTR, and SDRAM0_SDTR2.
1:3	DLCS	Delay Line Calibration Status 000 Delay Line Calibration not run 001 Delay Line Calibration in progress 010 Delay Line Calibration complete 100 Delay Line Calibration error	This two bit field indicates the real-time status of the calibration process at the time of the register read. Once complete (DLCS = 010), the DLY_VAL field is valid. A delay line calibration status of “error” indicates that the calibration process completed without successfully determining the value of DLCV. Sufficient delay stages have been included in the calibration delay line such that this completion status should never occur within the supported operating frequency range.
4:21		Reserved	
22:29	DLCV	Delay Line Calibration Value	This 8-bit binary encoded value indicates the number of delay elements in a single 2x clock cycle or a single 1x half clock cycle. Bit 22 is Most Significant Bit; bit 29 is Least Significant Bit.
30:31		Reserved	

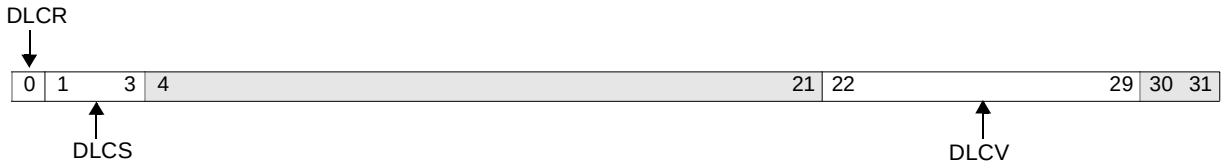


Figure 29-290. Delay Line Calibration Register (SDRAM0_DLYCAL)

0	DLCR	Delay Line Calibration Request 0 Delay Line Calibration not requested 1 Delay Line Calibration requested	Setting DLCR bit initiates the delay line calibration. During the re-initialization process, the DLCR bit is cleared. The delay line calibration occurs automatically following SysReset or upon request by setting the DLCR bit. The results of the calibration are provided to assist in the setting of SDRAM0_CLKTR, SDRAM0_WRDTR, and SDRAM0_SDTR2.
1:3	DLCS	Delay Line Calibration Status 000 Delay Line Calibration not run 001 Delay Line Calibration in progress 010 Delay Line Calibration complete 100 Delay Line Calibration error	This two bit field indicates the real-time status of the calibration process at the time of the register read. Once complete (DLCS = 010), the DLY_VAL field is valid. A delay line calibration status of "error" indicates that the calibration process completed without successfully determining the value of DLCV. Sufficient delay stages have been included in the calibration delay line such that this completion status should never occur within the supported operating frequency range.
4:21		Reserved	
22:29	DLCV	Delay Line Calibration Value	This 8-bit binary encoded value indicates the number of delay elements in a single 2x clock cycle or a single 1x half clock cycle. Bit 22 is Most Significant Bit; bit 29 is Least Significant Bit.
30:31		Reserved	

DCR Accessed using SDRAM0_CFGADDR; SDRAM0_CFGDATA; Offset 0x98 R/W

See *ECC Error Status Register (SDRAM0_ECCESR)* on page 514.

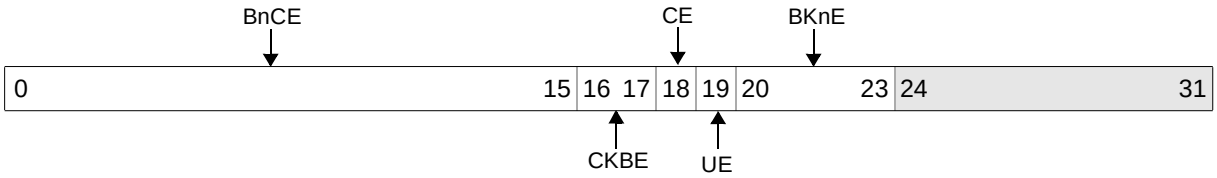


Figure 0-11. ECC Error Status Register (SDRAM0_ECCESR)

0:15	BnCE	Byte Lane n Corrected Error (where n=0-15) 0 No Error 1 Error Occurred in Byte Lane n
16:17	CKBE	Error Detected in Checkbits 64-bit Mode 00 No Error 01 Reserved 10 Error in Checkbits 11 Reserved 32-bit Mode 00 No Error 01 Error in Lower Checkbits 10 Error in Upper Checkbits 11 Error in Upper and Lower Checkbits
18	CE	Correctable Error
19	UE	Uncorrectable Error
20:23	BKnE	Bank n Error (where n=0-3) 0 No Error 1 Error Occurred in Bank n
24:31		Reserved

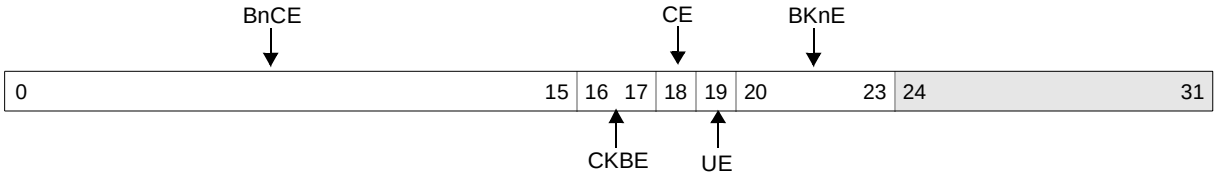


Figure 29-291. ECC Error Status Register (SDRAM0_ECCESR)

0:15	BnCE	Byte Lane n Corrected Error (where n=0-15) 0 No Error 1 Error Occurred in Byte Lane n

16:17	CKBE	Error Detected in Checkbits 64-bit Mode 00 No Error 01 Reserved 10 Error in Checkbits 11 Reserved 32-bit Mode 00 No Error 01 Error in Lower Checkbits 10 Error in Upper Checkbits 11 Error in Upper and Lower Checkbits
18	CE	Correctable Error
19	UE	Uncorrectable Error
20:23	BKnE	Bank n Error (where n=0-3) 0 No Error 1 Error Occurred in Bank n
24:31		Reserved

DCR Accessed using SDRAM0_CFGADDR; SDRAM0_CFGDATA; Offset 0x24 Read-only

See *Memory Controller Status (SDRAM0_MCSTS)* on page 492.

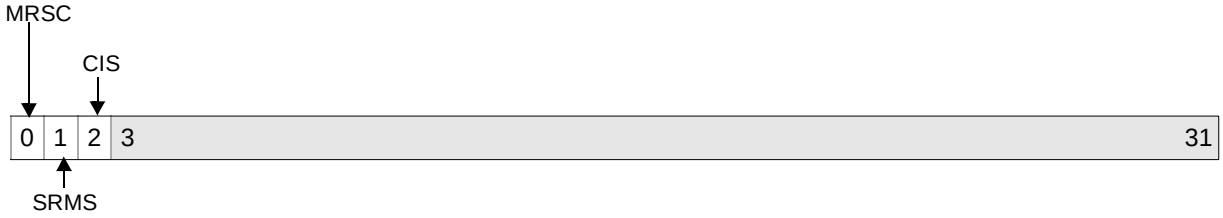


Figure 0-12. Memory Controller Status (SDRAM0_MCSTS)

0	MRSC	MRS Command Complete 0 MRS Not Complete 1 MRS Complete	This bit will be set to “1” once the DDR SDRAM controller has successfully completed the Mode Register Set Command, which results from setting DC_EN in SDRAM0_CFG0. Clearing DC_EN will clear this bit in the following cycle.
1	SRMS	Self-Refresh Mode Status 0 SDRAM is not in Self-Refresh Mode 1 SDRAM is in Self-Refresh Mode	This bit will be set upon the successful completion of Self-Refresh Mode entry, that is, the SDRAM device has been placed in Self-Refresh Mode. This bit will be cleared when Self-Refresh Mode has been exited. This bit applies to the software-initiated Self-Refresh Mode using SDRAM0_CFG0[SRE].
2	CIS	Core Idle Status 0 Busy 1 Idle	Indicates that there are no current or pending queued read/write accesses.
3:31		Reserved	

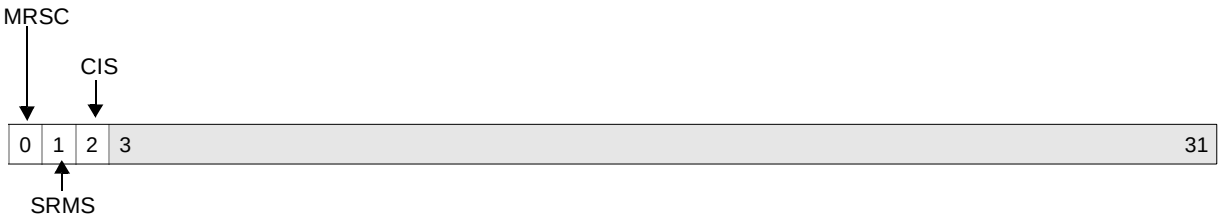


Figure 29-292. Memory Controller Status (SDRAM0_MCSTS)

0	MRSC	MRS Command Complete 0 MRS Not Complete 1 MRS Complete	This bit will be set to "1" once the DDR SDRAM controller has successfully completed the Mode Register Set Command, which results from setting DC_EN in SDRAM0_CFG0. Clearing DC_EN will clear this bit in the following cycle.
1	SRMS	Self-Refresh Mode Status 0 SDRAM is not in Self-Refresh Mode 1 SDRAM is in Self-Refresh Mode	This bit will be set upon the successful completion of Self-Refresh Mode entry, that is, the SDRAM device has been placed in Self-Refresh Mode. This bit will be cleared when Self-Refresh Mode has been exited. This bit applies to the software-initiated Self-Refresh Mode using SDRAM0_CFG0[SRE].
2	CIS	Core Idle Status 0 Busy 1 Idle	Indicates that there are no current or pending queued read/write accesses.
3:31		Reserved	

DCR Accessed using SDRAM0_CFGADDR; SDRAM0_CFGDATA; Offset 0x11 - 0x12 R/W

~~See "Master Write Interrupt (SDRAM0_MIRQ)" on page 17-14.~~See *Master Write Interrupt (SDRAM0_MIRQ)* on page 485.

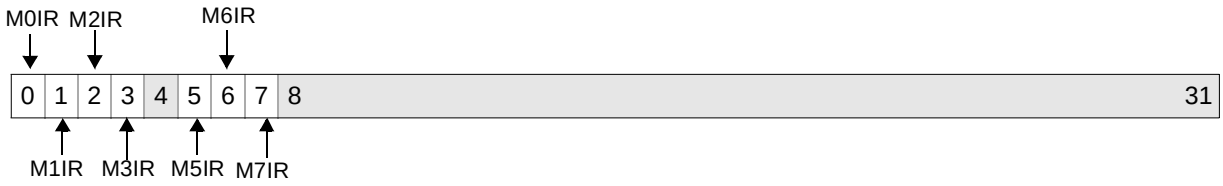


Figure 0-13. Master Write Interrupt (SDRAM0_MIRQ)

0	M0IR	440 CPU Instruction Cache Controller Write Interrupt 0 No Error 1 Write Error
1	M1IR	440 CPU Data Cache Controller Write Interrupt 0 No Error 1 Write Error
2	M2IR	440 CPU Data Cache Controller Write Interrupt 0 No Error 1 Write Error
3	M3IR	PLB_PCIX Bridge Write Interrupt 0 No Error 1 Write Error
4		Reserved
5	M5IR	MAL Write Interrupt 0 No Error 1 Write Error
6	M6IR	DMA Controller Write Interrupt 0 No Error 1 Write Error
7	M7IR	OPB to PLB Bridge Write Interrupt 0 No Error 1 Write Error
8:31		Reserved

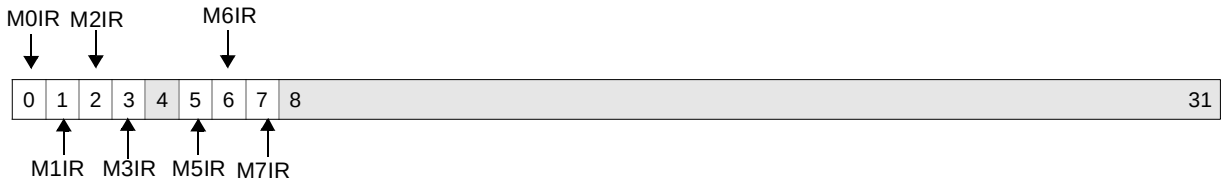


Figure 29-293. Master Write Interrupt (SDRAM0_MIRQ)

0:1		Reserved
0	M0IR	440 CPU Instruction Cache Controller Write Interrupt 0 No Error 1 Write Error
1	M1IR	440 CPU Data Cache Controller Write Interrupt 0 No Error 1 Write Error
2	M2IR	440 CPU Data Cache Controller Write Interrupt 0 No Error 1 Write Error
3	M3IR	PLB_PCIX Bridge Write Interrupt 0 No Error 1 Write Error
4		Reserved
5	M5IR	MAL Write Interrupt 0 No Error 1 Write Error
6	M6IR	DMA Controller Write Interrupt 0 No Error 1 Write Error
7	M7IR	OPB to PLB Bridge Write Interrupt 0 No Error 1 Write Error
8:31		Reserved

DCR Accessed using SDRAM0_CFGADDR; SDRAM0_CFGDATA; Offset 0x34 R/W

See *Power Management Idle Timer (SDRAM0_PMIT)* on page 494.

Figure 0-14. Power Management Idle Timer (SDRAM0_PMIT)

0:4	PM_C	Power Management Count	Programmable
5:9		Reserved	Hardcoded to ones
10:31		Reserved	Hardcoded to zero

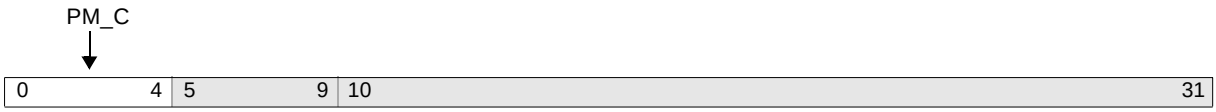


Figure 29-294. Power Management Idle Timer (SDRAM0_PMIT)

0:4	PM_C	Power Management Count	Programmable
5:9		Reserved	Hardcoded to ones
10:31		Reserved	Hardcoded to zero

DCR Accessed using SDRAM0_CFGADDR; SDRAM0_CFGDATA; Offset 0xA8 Read-only

See *Revision ID Register (SDRAM0_RID)* on page 516.

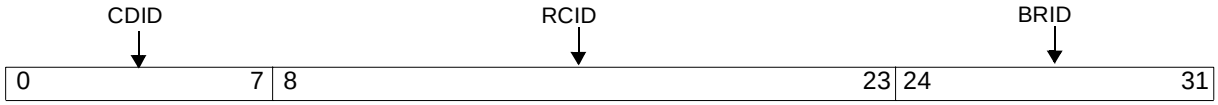


Figure 0-15. Revision ID Register (SDRAM0_RID)

0:7	CDID	Controller Derivative ID nn = controller derivative ID number nn = 00 indicates the Base Library Controller (CDC = B) nn = A non-zero values indicates the specific controller derivative ID number and is tracted by development (CDC = D).
8:23	RCID	Controller Revision Control ID cccc indicates the controller SCCS revision control ID associated with the specific version of the controller.
24:31	BRID	Controller Branch Revision Control ID bb indicates the core SCCS revision control branch ID associated with the specific version of the controller. This field is used to associate a specific Netlist with the corresponding RTL branch in the event of a modification at the Netlist level. bb = 00 indicates the an unmodified (post synthesis) netlist bb = non-zero value indicates the branch ID for the RTL and the corresponding netlist modification.

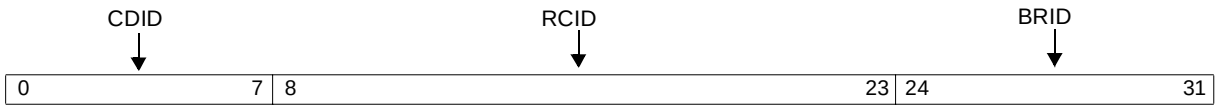


Figure 29-295. Revision ID Register (SDRAM0_RID)

0:7	CDID	Controller Derivative ID nn = controller derivative ID number nn = 00 indicates the Base Library Controller (CDC = B) nn = A non-zero values indicates the specific controller derivative ID number and is tracted by development (CDC = D).

8:23	RCID	Controller Revision Control ID cccc indicates the controller SCCS revision control ID associated with the specific version of the controller.	
24:31	BRID	Controller Branch Revision Control ID bb indicates the core SCCS revision control branch ID associated with the specific version of the controller. This field is used to associate a specific Netlist with the corresponding RTL branch in the event of a modification at the Netlist level. bb = 00 indicates the an unmodified (post synthesis) netlist bb = non-zero value indicates the branch ID for the RTL and the corresponding netlist modification.	

DCR Accessed using SDRAM0_CFGADDR; SDRAM0_CFGDATA; Offset 0x30 R/W

See *Refresh Timer Register (SDRAM0_RTR)* on page 493.

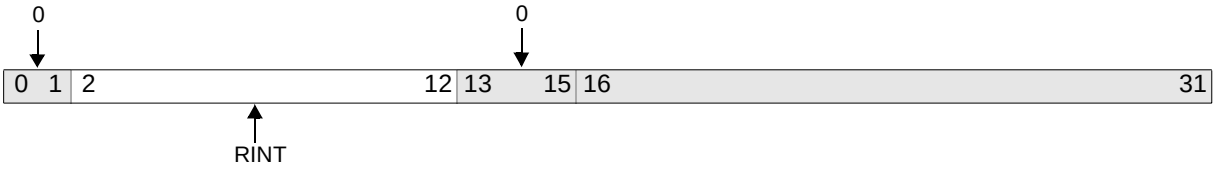


Figure 0-16. Refresh Timing Register (SDRAM0_RTR)

0:1		Reserved	Hardcoded to zero
2:12	RINT	Refresh Interval	Programmable
13:15		Reserved	Hardcoded to zero
16:31		Reserved	

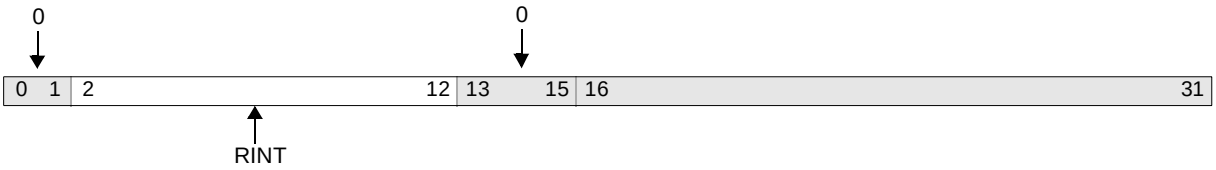


Figure 29-296. Refresh Timing Register (SDRAM0_RTR)

0:1		Reserved	Hardcoded to zero
2:12	RINT	Refresh Interval	Programmable
13:15		Reserved	Hardcoded to zero
16:31		Reserved	

DCR Accessed using SDRAM0_CFGADDR; SDRAM0_CFGDATA; Offset 0x18 R/W

See *PLB Slave Interface Options (SDRAM0_SLIO)* on page 486.

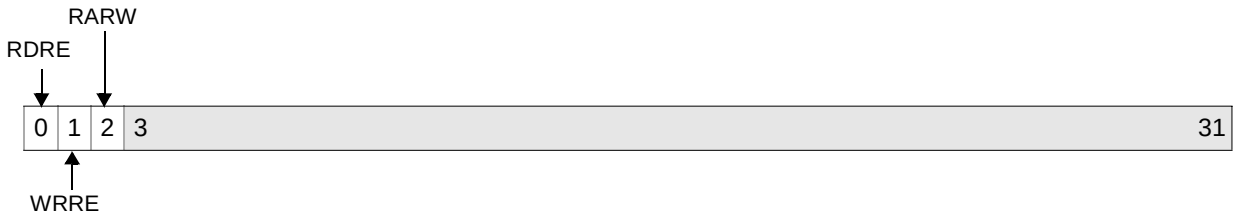


Figure 0-17. PLB Slave Interface Options (SDRAM0_SLIO)

0	RDRE	PLB Slave Read Rearbitrate Enable 0 Disable 1 Enable
1	WRRE	PLB Slave Write Rearbitrate Enable 0 Disable 1 Enable
2	RARW	PLB Read Around Write Enable 0 Disable 1 Enable
3:31		Reserved

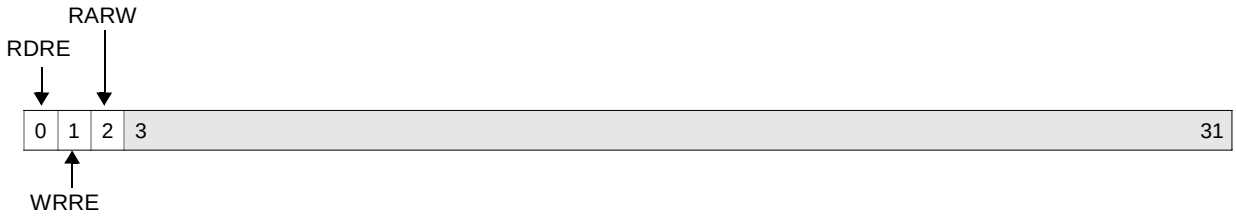


Figure 29-297. PLB Slave Interface Options (SDRAM0_SLIO)

0	RDRE	PLB Slave Read Rearbitrate Enable 0 Disable 1 Enable
1	WRRE	PLB Slave Write Rearbitrate Enable 0 Disable 1 Enable
2	RARW	PLB Read Around Write Enable 0 Disable 1 Enable
3:31		Reserved

DCR Accessed using SDRAM0_CFGADDR; SDRAM0_CFGDATA; Offset 0x80 R/W

See *SDRAM Timing Register 0 (SDRAM0_TR0)* on page 498.

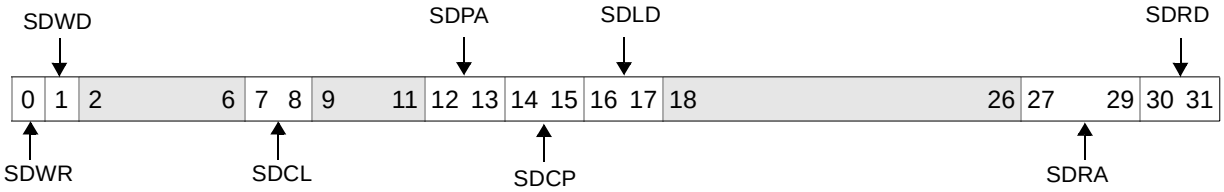


Figure 0-18. SDRAM Timing Register 0 (SDRAM0_TR0)

0	SDWR	DDR SDRAM Write Recovery 0 2 CLK 1 3 CLK	T_{wr}
1	SDWD	DDR SDRAM Write to Write delay when crossing a chip select boundary 0 0 CLK 1 1 CLK	This field configures the delay between back to back write cycles that cross chip select boundaries. A setting of 0 enables seamless writes when crossing chip select boundaries, while a setting of 1 will insert a single clock cycle delay between back-to-back writes when crossing a chip select boundary.
2:6		Reserved	
7:8	SDCL	DDR SDRAM CAS_ Latency 00 Reserved 01 2 CLK 10 2.5 CLK 11 3 CLK	T_{aa} This field indicates the DDR SDRAM device CAS latency (independent of SDRAM0_CFG0[R_DIMM_EN]) and is used directly during the SDRAM device mode set command to configure the DDR SDRAM. See the section on "Registered DIMM Support" for the supported configurations and CAS_ Latency settings. **Note:** Setting SD_CL to 2.5 CLK generally requires that SDRAM0_TR1[RDCD] be set to 1, SDRAM0_TR1[RDSS] be set to T2 Sample, and SDRAM0_TR1[RDSL] be set to Stage 3.
9:11		Reserved	

12:13	SDPA	DDR SDRAM CBR Precharge Command to next Activate Command minimum 00 Reserved 01 2 CLK 10 3 CLK 11 4 CLK	T_{rp}
14:15	SDCP	DDR SDRAM Read/Write Command to Precharge Command 00 2 CLK 01 3 CLK 10 4 CLK 11 5 CLK	T_{ras} - T_{rcd} The minimum value is calculated by first determining the values of T_{ras} and T_{rcd} in units of clock cycles (divide the respective device-specific AC timing specifications by the clock cycle time and round up to the nearest whole clock). Then apply the formula provided. SDCP settings that violate this rule (that is, settings less than the minimum calculated above for the device-specific timing parameters), independent of the programmed value of SDRD, are not supported.
	SDLD	DDR SDRAM Command Leadoff 00 1 CLK 01 2 CLK 10 Reserved 11 Reserved	
18:26		Reserved	
27:29	SDRA	DDR SDRAM CBR Refresh Command to next Activate Command minimum 000 6 CLK 001 7 CLK 010 8 CLK 011 9 CLK 100 10 CLK 101 11 CLK 110 12 CLK 111 13 CLK	T_{rfc}
30:31	SDRD	DDR SDRAM RAS to CAS Delay 00 Reserved 01 2 CLK 10 3 CLK 11 4 CLK	T_{rcd}

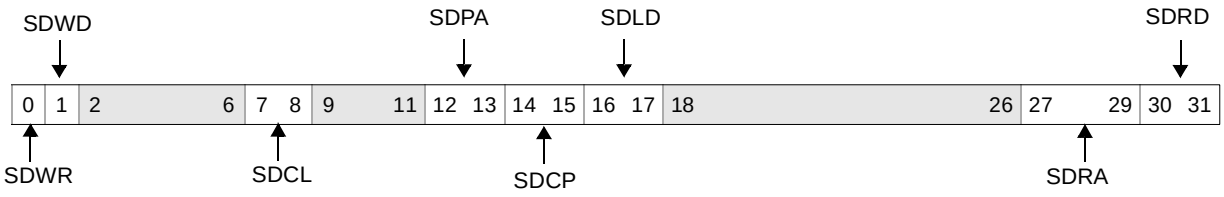


Figure 29-298. SDRAM Timing Register 0 (SDRAM0_TR0)

0	SDWR	DDR SDRAM Write Recovery 0 2 CLK 1 3 CLK	T_{wr}
1	SDWD	DDR SDRAM Write to Write delay when crossing a chip select boundary 0 0 CLK 1 1 CLK	This field configures the delay between back to back write cycles that cross chip select boundaries. A setting of 0 enables seamless writes when crossing chip select boundaries, while a setting of 1 will insert a single clock cycle delay between back-to-back writes when crossing a chip select boundary.
2:6		Reserved	
7:8	SDCL	DDR SDRAM CAS_ Latency 00 Reserved 01 2 CLK 10 2.5 CLK 11 3 CLK	T_{aa} This field indicates the DDR SDRAM device CAS latency (independent of SDRAM0_CFG0[R_DIMM_EN]) and is used directly during the SDRAM device mode set command to configure the DDR SDRAM. See the section on "Registered DIMM Support" for the supported configurations and CAS_ Latency settings. **Note:** Setting SD_CL to 2.5 CLK generally requires that SDRAM0_TR1[RDCD] be set to 1, SDRAM0_TR1[RDSS] be set to T2 Sample, and SDRAM0_TR1[RDSL] be set to Stage 3.
9:11		Reserved	
12:13	SDPA	DDR SDRAM CBR Precharge Command to next Activate Command minimum 00 Reserved 01 2 CLK 10 3 CLK 11 4 CLK	T_{rp}

14:15	SDCP	DDR SDRAM Read/Write Command to Precharge Command 00 2 CLK 01 3 CLK 10 4 CLK 11 Reserved	T_{ras} - T_{rcd} The minimum value is calculated by first determining the values of T_{ras} and T_{rcd} in units of clock cycles (divide the respective device-specific AC timing specifications by the clock cycle time and round up to the nearest whole clock). Then apply the formula provided. SDCP settings that violate this rule (that is, settings less than the minimum calculated above for the device-specific timing parameters), independent of the programmed value of SDRD, are not supported.
16:17	SDLD	DDR SDRAM Command Leadoff 00 1 CLK 01 2 CLK 10 Reserved 11 Reserved	
18:26		Reserved	
27:29	SDRA	DDR SDRAM CBR Refresh Command to next Activate Command minimum 000 6 CLK 001 7 CLK 010 8 CLK 011 9 CLK 100 10 CLK 101 11 CLK 110 12 CLK 111 13 CLK	T_{rfc}
30:31	SDRD	DDR SDRAM RAS to CAS Delay 00 Reserved 01 2 CLK 10 3 CLK 11 4 CLK	T_{rcd}

DCR Accessed using SDRAM0_CFGADDR; SDRAM0_CFGDATA; Offset 0x81R/W

See *SDRAM Timing Register 1 (SDRAM0_TR1)* on page 500.

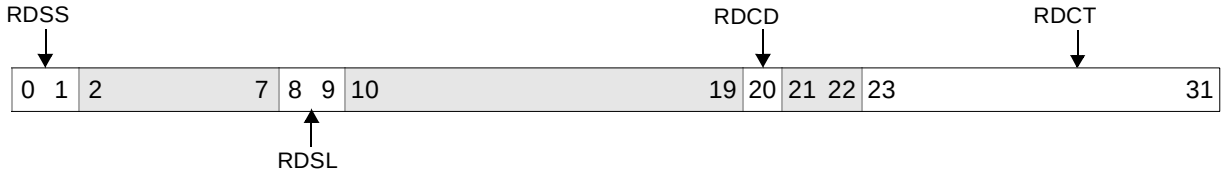


Figure 0-19. SDRAM Timing Register 1 (SDRAM0_TR1)

0:1	RDSS	Read Sample Cycle Select 00 T0 Sample 01 T1 Sample 10 T2 Sample 11 T3 Sample	**Note:** Set RDSS to T2 Sample for CAS latency 2.5 devices.
2:7		Reserved	
8:9	RDSL	Read Sample Stage Select 00 Stage 1 01 Stage 2 10 Stage 3 11 Reserved	**Note:** Set RDSL to Stage 3 for CAS latency 2.5 devices.
10:19		Reserved	
20	RDCD	Read Clock Delay - Stage 2 0 0 clock delay 1 1/2 clock delay	**Note:** Set RDCD to 1 for CAS latency 2.5 devices.
21:22		Reserved	
23:31	RDCT	Read Clock Delay Tuning Bits	Bit 23 is Most Significant Bit; bit 31 is Least Significant Bit. Bits 30:31, select increments of 1/4 delay element. Bits 23:29, select increments of 1 full delay element. This field should never be programmed to exceed a delay equivalent to 1/2 the Read Clock frequency.

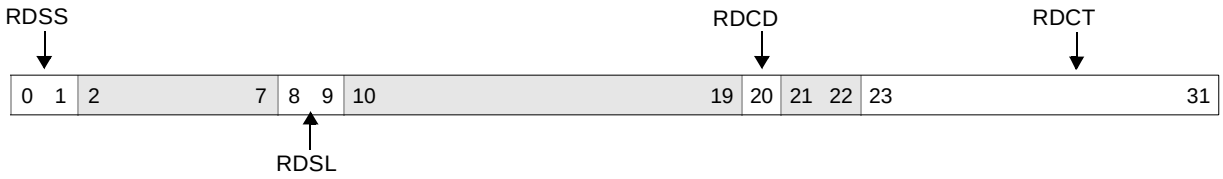


Figure 29-299. SDRAM Timing Register 1 (SDRAM0_TR1)

0:1	RDSS	Read Sample Cycle Select 00 T0 Sample 01 T1 Sample 10 T2 Sample 11 T3 Sample	**Note:** Set RDSS to T2 Sample for CAS latency 2.5 devices.
2:7		Reserved	
8:9	RDSL	Read Sample Stage Select 00 Stage 1 01 Stage 2 10 Stage 3 11 Reserved	**Note:** Set RDSL to Stage 3 for CAS latency 2.5 devices.
10:19		Reserved	
20	RDCD	Read Clock Delay - Stage 2 0 0 clock delay 1 1/2 clock delay	**Note:** Set RDCD to 1 for CAS latency 2.5 devices.
21:22		Reserved	
23:31	RDCT	Read Clock Delay Tuning Bits	Bit 23 is Most Significant Bit; bit 31 is Least Significant Bit. Bits 30:31, select increments of 1/4 delay element. Bits 23:29, select increments of 1 full delay element. This field should never be programmed to exceed a delay equivalent to 1/2 the Read Clock frequency.

DCR Accessed using SDRAM0_CFGADDR; SDRAM0_CFGDATA; Offset 0x38 R/W

See *PLB UABus Base Address (SDRAM0_UABBA)* on page 495.

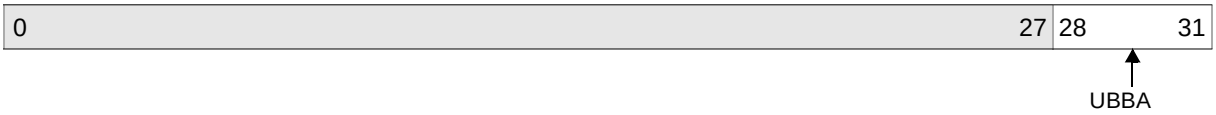


Figure 0-20. PLB UABus Base Address (SDRAM0_UABBA)

0:27		Reserved	
28:31	UBBA	PLB UABus Base Address	Defines the 4 GB aligned region in which the physical memory is located.

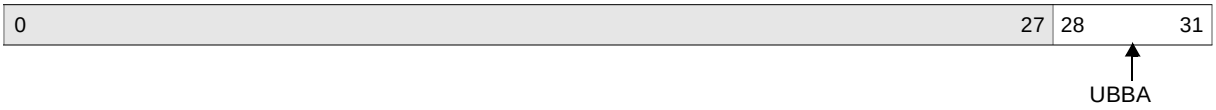


Figure 29-300. PLB UABus Base Address (SDRAM0_UABBA)

0:27		Reserved	
28:31	UBBA	PLB UABus Base Address	Defines the 4 GB aligned region in which the physical memory is located.

DCR Accessed using SDRAM0_CFGADDR; SDRAM0_CFGDATA; Offset 0x83 R/W

See *Write Data/DM/DQS Clock Timing Register (SDRAM0_WDDCTR)* on page 509.

Figure 0-21. Write Data/DM/DQS Clock Timing Register (SDRAM0_WDDCTR)

0:1	WRCP	Write Clock Phase 00 Advance 0 degrees 01 Advance 90 degrees 10 Advance 180 degrees 11 Reserved
2:22		Reserved
23:31	DCD	DDR Write Data/DM/DQS Clock Delay Tuning Bits Bit 23 is Most Significant Bit; bit 31 is Least Significant Bit. Bits 23:29 select increments of 1 full delay element. Bits 30:31 select increments of 1/4 delay element.

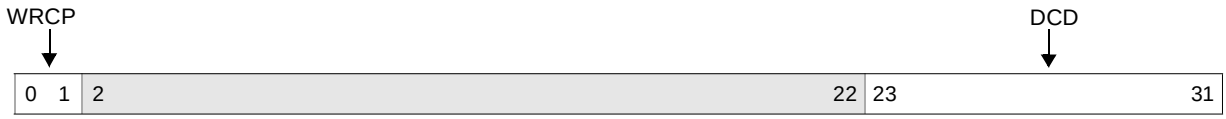


Figure 29-301. Write Data/DM/DQS Clock Timing Register (SDRAM0_WDDCTR)

0:1	WRCP	Write Clock Phase 00 Advance 0 degrees 01 Advance 90 degrees 10 Advance 180 degrees 11 Reserved
2:22		Reserved
23:31	DCD	DDR Write Data/DM/DQS Clock Delay Tuning Bits Bit 23 is Most Significant Bit; bit 31 is Least Significant Bit. Bits 23:29 select increments of 1 full delay element. Bits 30:31 select increments of 1/4 delay element.

DCR 0x024 Read/Write

See *Bus Error Address Register (SRAM0_BEAR)* on page 466.

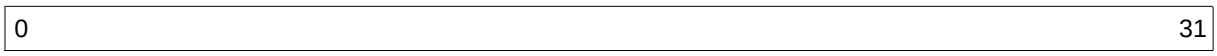


Figure 0-1. Bus Error Address Register (SRAM0_BEAR)

0:31		Address of Bus Error (asynchronous)

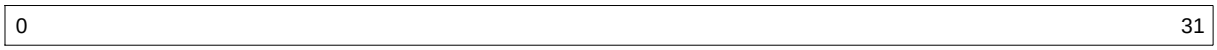


Figure 29-302. Bus Error Address Register (SRAM0_BEAR)

0:31		Address of Bus Error (asynchronous)

DCR 0x025 Read/Write

See *Bus Error Status Register 0 (SRAM0_BESR0)* on page 466.

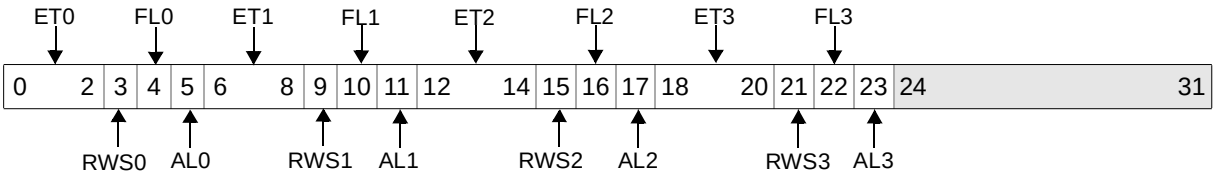


Figure 0-2. Bus Error Status Register 0 (SRAM0_BESR0)

0:2	ET0	Error type for master 0 (CPU ICU Read Operation) 000 No error 001 Reservd 010 Parity error 011 Reserved 100 Protection error 101 Reserved 110 Reserved 111 Reserved
3	RWS0	Read/write status for master 0 (CPU ICU Read Operation) 0 Error operation was a write operation 1 Error operation was a read operation
4	FL0	Field lock for master 0 (CPU ICU Read Operation) 0 Fields are unlocked 1 Fields are locked
5	AL0	BEAR address lock for master 0 (CPU ICU Read Operation) 0 BEAR address unlocked 1 BEAR address locked
6:8	ET1	Error type for master 1 (CPU DCU Read Operation) 000 No error 001 Reserved 010 Parity error 011 Reserved 100 Protection error 101 Reserved 110 Reserved 111 Reserved
9	RWS1	Read/write status for master 1 (CPU DCU Read Operation) 0 Error operation was a write operation 1 Error operation was a read operation
10	FL1	Field lock for master 1 (CPU DCU Read Operation) 0 Fields are unlocked 1 Fields are locked
11	AL1	BEAR address lock for master 1 (CPU DCU Read Operation) 0 BEAR address unlocked 1 BEAR address locked

12:14	ET2	Error type for master 2 000 No error 001 Reserved 010 Parity error 011 Reserved 100 Protection error 101 Reserved 110 Reserved 111 Reserved
15	RWS2	Read/write status for master 2 (CPU DCU Write Operation) 0 Error operation was a write operation 1 Error operation was a read operation
16	FL2	Field lock for master 2 (CPU DCU Write Operation) 0 Fields are unlocked 1 Fields are locked
17	AL2	BEAR address lock for master 2 (CPU DCU Write Operation) 0 BEAR address unlocked 1 BEAR address locked
18:20	ET3	Error type for master 3 (PCI-X) 000 No error 001 Reserved 010 Parity error 011 Reserved 100 Protection error 101 Reserved 110 Reserved 111 Reserved
21	RWS3	Read/write status for master 3 (PCI-X) 0 Error operation was a write operation 1 Error operation was a read operation
22	FL3	Field lock for master (PCI-X)3 0 Fields are unlocked 1 Fields are locked
23	AL3	BEAR address lock for master 3 (PCI-X) 0 BEAR address unlocked 1 BEAR address locked
24:31		Reserved

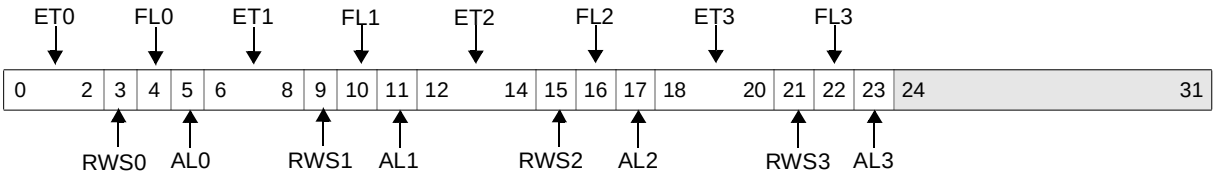


Figure 29-303. Bus Error Status Register 0 (SRAM0_BESR0)

0:2	ET0	Error type for master 0 (CPU ICU Read Operation) 000 No error 001 Reservd 010 Parity error 011 Reserved 100 Protection error 101 Reserved 110 Reserved 111 Reserved
3	RWS0	Read/write status for master 0 (CPU ICU Read Operation) 0 Error operation was a write operation 1 Error operation was a read operation
4	FL0	Field lock for master 0 (CPU ICU Read Operation) 0 Fields are unlocked 1 Fields are locked
5	AL0	BEAR address lock for master 0 (CPU ICU Read Operation) 0 BEAR address unlocked 1 BEAR address locked
6:8	ET1	Error type for master 1 (CPU DCU Read Operation) 000 No error 001 Reserved 010 Parity error 011 Reserved 100 Protection error 101 Reserved 110 Reserved 111 Reserved
9	RWS1	Read/write status for master 1 (CPU DCU Read Operation) 0 Error operation was a write operation 1 Error operation was a read operation
10	FL1	Field lock for master 1 (CPU DCU Read Operation) 0 Fields are unlocked 1 Fields are locked
11	AL1	BEAR address lock for master 1 (CPU DCU Read Operation) 0 BEAR address unlocked 1 BEAR address locked
12:14	ET2	Error type for master 2 000 No error 001 Reserved 010 Parity error 011 Reserved 100 Protection error 101 Reserved 110 Reserved 111 Reserved

15	RWS2	Read/write status for master 2 (CPU DCU Write Operation) 0 Error operation was a write operation 1 Error operation was a read operation
16	FL2	Field lock for master 2 (CPU DCU Write Operation) 0 Fields are unlocked 1 Fields are locked
17	AL2	BEAR address lock for master 2 (CPU DCU Write Operation) 0 BEAR address unlocked 1 BEAR address locked
18:20	ET3	Error type for master 3 (PCI-X) 000 No error 001 Reserved 010 Parity error 011 Reserved 100 Protection error 101 Reserved 110 Reserved 111 Reserved
21	RWS3	Read/write status for master 3 (PCI-X) 0 Error operation was a write operation 1 Error operation was a read operation
22	FL3	Field lock for master (PCI-X)3 0 Fields are unlocked 1 Fields are locked
23	AL3	BEAR address lock for master 3 (PCI-X) 0 BEAR address unlocked 1 BEAR address locked
24:31		Reserved

DCR 0x026 Read/Write

See *Bus Error Status Register 1 (SRAM0_BESR1)* on page 468.

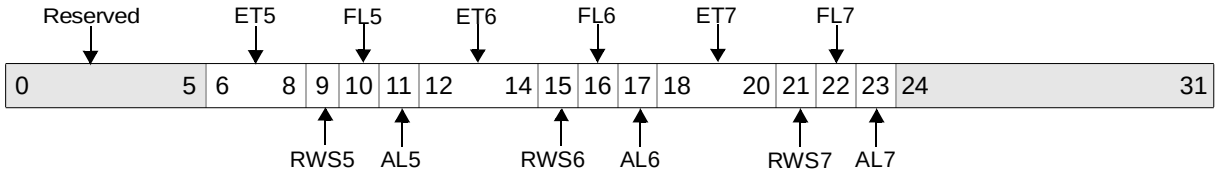


Figure 0-3. Bus Error Status Register 1 (SRAM0_BESR1)

0:5		Reserved
6:8	ET5	Error type for master 5 (MAL) 000 No error 001 Reserved 010 Parity error 011 Reserved 100 Protection error 101 Reserved 110 Reserved 111 Reserved
9	RWS5	Read/write status for master 5 (MAL) 0 Error operation was a write operation 1 Error operation was a read operation
10	FL5	Field lock for master 5 (MAL) 0 Fields are unlocked 1 Fields are locked
11	AL5	BEAR address lock for master 5 (MAL) 0 BEAR address unlocked 1 BEAR address locked
12:14	ET6	Error type for master 6 (DMA) 000 No error 001 Reserved 010 Parity error 011 Reserved 100 Protection error 101 Reserved 110 Reserved 111 Reserved
15	RWS6	Read/write status for master 6 (DMA) 0 Error operation was a write operation 1 Error operation was a read operation
16	FL6	Field lock for master 6 (DMA) 0 Fields are unlocked 1 Fields are locked

17	AL6	BEAR address lock for master 6 (DMA) 0 BEAR address unlocked 1 BEAR address locked
18:20	ET7	Error type for master 7 (BGI) 000 No error 001 Reserved 010 Parity error 011 Reserved 100 Protection error 101 Reserved 110 Reserved 111 Reserved
21	RWS7	Read/write status for master 7 (BGI) 0 Error operation was a write operation 1 Error operation was a read operation
22	FL7	Field lock for master 7 (BGI) 0 Fields are unlocked 1 Fields are locked
23	AL7	BEAR address lock for master 7 (BGI) 0 BEAR address unlocked 1 BEAR address locked
24:31		Reserved

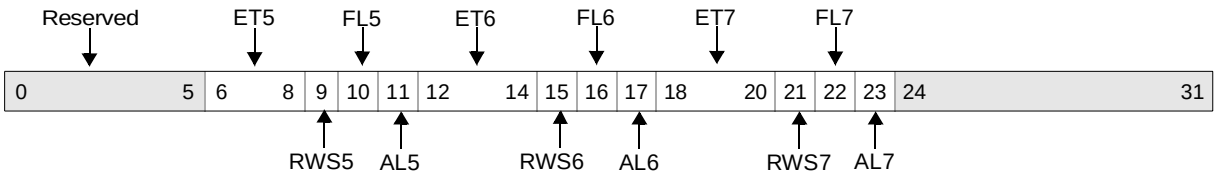


Figure 29-304. Bus Error Status Register 1 (SRAM0_BESR1)

0:5		Reserved
6:8	ET5	Error type for master 5 (MAL) 000 No error 001 Reserved 010 Parity error 011 Reserved 100 Protection error 101 Reserved 110 Reserved 111 Reserved
9	RWS5	Read/write status for master 5 (MAL) 0 Error operation was a write operation 1 Error operation was a read operation
10	FL5	Field lock for master 5 (MAL) 0 Fields are unlocked 1 Fields are locked

11	AL5	BEAR address lock for master 5 (MAL) 0 BEAR address unlocked 1 BEAR address locked
12:14	ET6	Error type for master 6 (DMA) 000 No error 001 Reserved 010 Parity error 011 Reserved 100 Protection error 101 Reserved 110 Reserved 111 Reserved
15	RWS6	Read/write status for master 6 (DMA) 0 Error operation was a write operation 1 Error operation was a read operation
16	FL6	Field lock for master 6 (DMA) 0 Fields are unlocked 1 Fields are locked
17	AL6	BEAR address lock for master 6 (DMA) 0 BEAR address unlocked 1 BEAR address locked
18:20	ET7	Error type for master 7 (BGI) 000 No error 001 Reserved 010 Parity error 011 Reserved 100 Protection error 101 Reserved 110 Reserved 111 Reserved
21	RWS7	Read/write status for master 7 (BGI) 0 Error operation was a write operation 1 Error operation was a read operation
22	FL7	Field lock for master 7 (BGI) 0 Fields are unlocked 1 Fields are locked
23	AL7	BEAR address lock for master 7 (BGI) 0 BEAR address unlocked 1 BEAR address locked
24:31		Reserved

DCR 0x28 Read only

See *Core ID Register (SRAM0_CID)* on page 470.

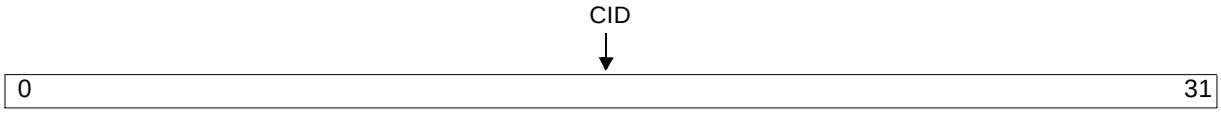


Figure 0-4. SRAM Internal Core Device ID Register (SRAM0_CID)

0:31	CID	Internal Core Device ID	H322B0000 (Read only).

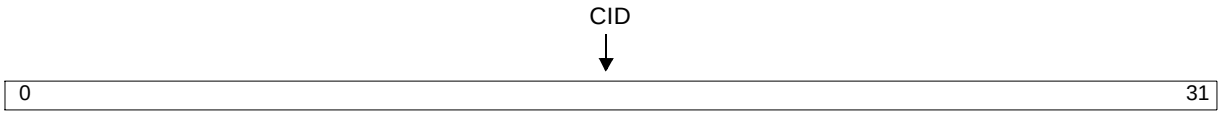


Figure 29-305. SRAM Internal Core Device ID Register (SRAM0_CID)

0:31	CID	Internal Core Device ID	H322B0000 (Read only).

DCR 0x2A Read/Write

~~See "Data Parity Checking Register (SRAM0_DPC)" on page 16-11.~~See *Data Parity Checking Register (SRAM0_DPC)* on page 471.

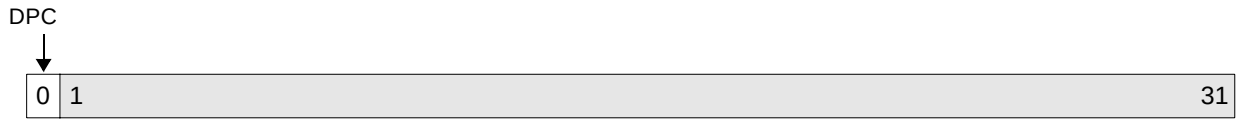


Figure 0-5. Data Parity Checking Register (SRAM0_DPC)

0	DPC	Data Parity Check: 0 Disabled 1 Enabled	When set to 1, this bit enables data parity generation during write transactions and read transactions. When set to 0, this bit disables both functions.
1:31		Reserved	

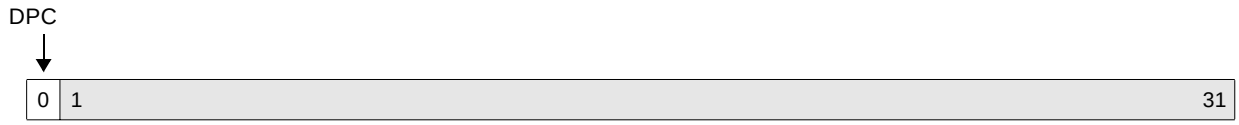


Figure 29-306. Data Parity Checking Register (SRAM0_DPC)

0	DPC	Data Parity Check: 0 Disabled 1 Enabled	When set to 1, this bit enables data parity generation during write transactions and read transactions. When set to 0, this bit disables both functions.
1:31		Reserved	

DCR 0x029 Read only

See *Revision ID Register (SRAM0_REVID)* on page 471.

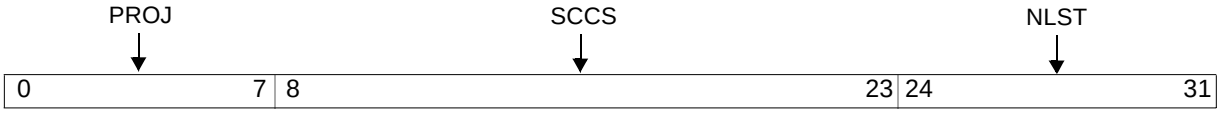


Figure 0-6. SRAM Internal Core Revision ID Register (SRAM0_REVID)

0:7	PROJ	Project Number	Read only.
8:23	SCCS	SCCS Version	Read only.
24:31	NLST	Netlist Number	Read only

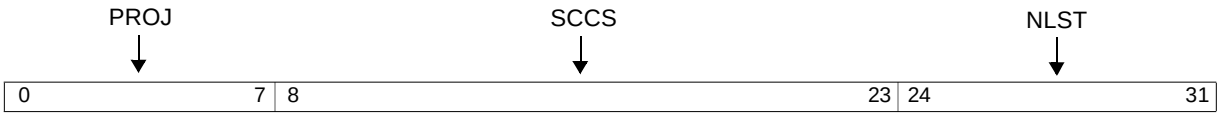


Figure 29-307. SRAM Internal Core Revision ID Register (SRAM0_REVID)

0:7	PROJ	Project Number	Read only.
8:23	SCCS	SCCS Version	Read only.
24:31	NLST	Netlist Number	Read only

DCR 0x027 Read/Write

See *Power Management Register (SRAM0_PMEG)* on page 470.

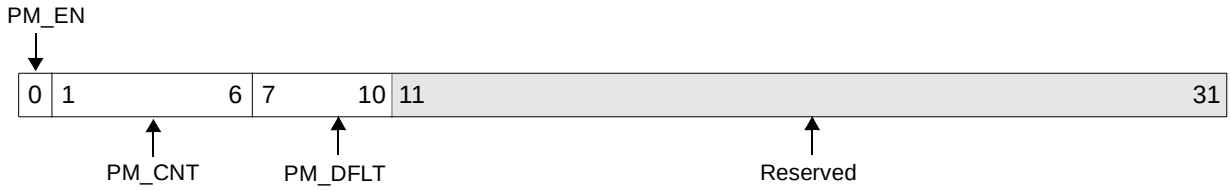


Figure 0-7. Power Management Register (SRAM0_PMEG)

0	PM_EN	Power Management enable: 0 Sleep mode disabled 1 Sleep mode enabled	
1:6	PM_CNT	Power Management Counter	The value (n) programmed into these register bits are the multiples of 16 clock cycles (n x 16). If PM_EN is set to 1, SRAM controller will go to sleep mode after being idle for (n x 16) clock cycles.
7:10	PM_DFLT	Power Management Default Wait Interval Hardwired to 1111.	The default value of 16 clock cycles. If PM_EN is set to 1, SRAM controller will go to sleep mode after being idle for 16 clock cycles.
11:31		Reserved	

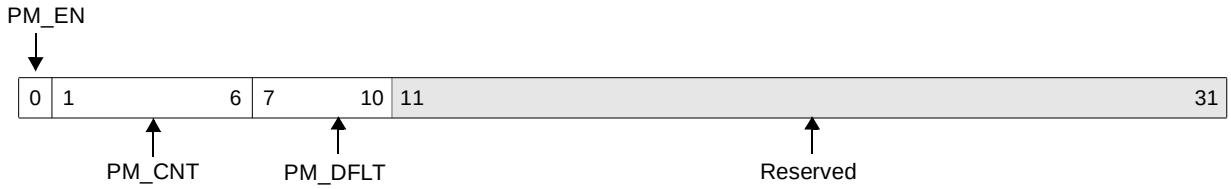


Figure 29-308. Power Management Register (SRAM0_PMEG)

0	PM_EN	Power Management enable: 0 Sleep mode disabled 1 Sleep mode enabled	
1:6	PM_CNT	Power Management Counter	The value (n) programmed into these register bits are the multiples of 16 clock cycles (n x 16). If PM_EN is set to 1, SRAM controller will go to sleep mode after being idle for (n x 16) clock cycles.
7:10	PM_DFLT	Power Management Default Wait Interval Hardwired to 1111.	The default value of 16 clock cycles. If PM_EN is set to 1, SRAM controller will go to sleep mode after being idle for 16 clock cycles.
11:31		Reserved	

DCR 0x020–0x023 Read/Write

See *SRAM Bank Configuration Registers (SRAM0_SB0CR - SRAM0_SB3CR)* on page 465.

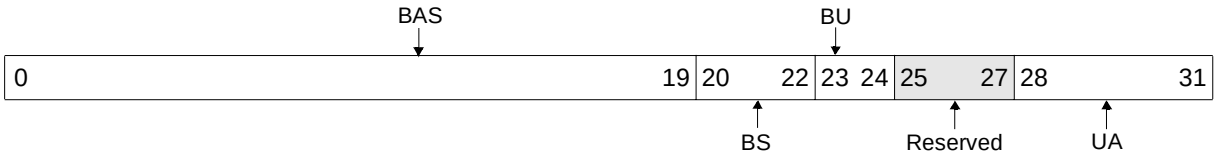


Figure 0-8. Memory Configuration (SRAM0_SB0CR - SRAM0_SB3CR)

0: 19	BAS	Base Address Select	Specifies the starting address of the SRAM bank.
20:22	BS	Bank Size 000 4 KB 001 8 KB 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved	Specifies the size of the logical bank address range.
23:24	BU	Bank Usage 00 Disabled 01 Bank is valid for read only (RO) 10 Reserved 11 Bank is valid for read/write (R/W)	
25:27		Reserved	
28:31	UA	Four least significant bits of the upper 32-bit address.	

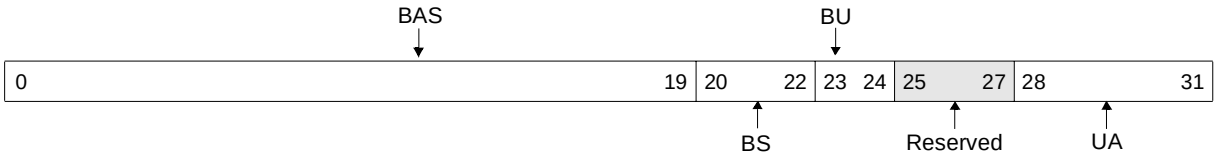


Figure 29-309. Memory Configuration (SRAM0_SB0CR - SRAM0_SB3CR)

0: 19	BAS	Base Address Select	Specifies the starting address of the SRAM bank.
20:22	BS	Bank Size 000 4 KB 001 8 KB 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved	Specifies the size of the logical bank address range.

23:24	BU	Bank Usage 00 Disabled 01 Bank is valid for read only (RO) 10 Reserved 11 Bank is valid for read/write (R/W)
25:27		Reserved
28:31	UA	Four least significant bits of the upper 32-bit address.

MMIO 0x140000200 (UART0), 0x140000300 (UART1) Read/Write

See *UART Baud-Rate Divisor Latch (LSB) Registers (UARTx_DLL)* on page 792.

Figure 0-1. UART Baud-Rate Divisor Latch (LSB) Registers (UARTx_DLL)

8:15		Data bits
Note: UARTx_DLL is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.		

Figure 29-310. UART Baud-Rate Divisor Latch (LSB) Registers (UARTx_DLL)

8:15		Data bits
Note: UARTx_DLL is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.		

MMIO 0x140000201 (UART0), 0x140000301(UART 1) R/eadWrite

See *UART Baud-Rate Divisor Latch (MSB) Registers (UARTx_DLM)* on page 792.

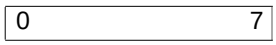


Figure 0-2. UART Baud-Rate Divisor Latch (MSB) Registers (UARTx_DLM)

0:7		Data bits
Note: UARTx_DLM is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.		

Figure 29-311. UART Baud-Rate Divisor Latch (MSB) Registers (UARTx_DLM)

0:7		Data bits
Note: UARTx_DLM is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.		

MMIO 0x140000202 (UART0), 0x140000302 Write-Only

See *FIFO Control Registers (UARTx_FCR)* on page 786.

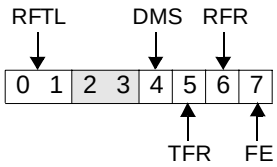


Figure 0-3. UART FIFO Control Registers (UARTx_FCR)

0:1	RFTL	Receiver FIFO Trigger Level 00 01 byte 01 16 bytes 10 32 bytes 11 56 bytes	
2:3		Reserved	
4	DMS	DMA Mode Select 0 Mode 0 = single transfer 1 Mode 1 = multiple transfers	Select single or multiple transfer mode if UARTx_FCR[7] = 1.
5	TFR	Transmitter FIFO Reset 0 Operation complete 1 Reset the transmitter FIFO	A 1 written to this bit clears all bytes in the transmitter FIFO and resets all of its counter logic to 0. The transmitter shift register is not cleared. This bit is self-clearing.
6	RFR	Receiver FIFO Reset 0 Operation complete 1 Reset the receiver FIFO	A 1 written to this bit clears all bytes in the receiver FIFO and resets all of its counter logic to 0. The receiver shift register is not cleared. This bit is self-clearing.
7	FE	FIFO Enable 0 Disable FIFOs 1 Enable FIFOs	When set to 1, both the receiver and transmitter FIFOs are enabled. When set to 0, both receiver and transmitter FIFOs are reset. Data is automatically cleared from both FIFOs when changing to and from FIFO and 16450 modes. Programming other bits will be ignored if this bit is not a 1.
Note: UARTx_FCR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.			

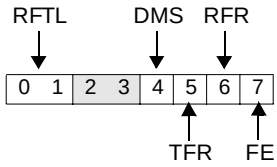


Figure 29-312. UART FIFO Control Registers (UARTx_FCR)

0:1	RFTL	Receiver FIFO Trigger Level 00 01 byte 01 16 bytes 10 32 bytes 11 56 bytes	
2:3		Reserved	
4	DMS	DMA Mode Select 0 Mode 0 = single transfer 1 Mode 1 = multiple transfers	Select single or multiple transfer mode if UARTx_FCR[7] = 1.
5	TFR	Transmitter FIFO Reset 0 Operation complete 1 Reset the transmitter FIFO	A 1 written to this bit clears all bytes in the transmitter FIFO and resets all of its counter logic to 0. The transmitter shift register is not cleared. This bit is self-clearing.
6	RFR	Receiver FIFO Reset 0 Operation complete 1 Reset the receiver FIFO	A 1 written to this bit clears all bytes in the receiver FIFO and resets all of its counter logic to 0. The receiver shift register is not cleared. This bit is self-clearing.
7	FE	FIFO Enable 0 Disable FIFOs 1 Enable FIFOs	When set to 1, both the receiver and transmitter FIFOs are enabled. When set to 0, both receiver and transmitter FIFOs are reset. Data is automatically cleared from both FIFOs when changing to and from FIFO and 16450 modes. Programming other bits will be ignored if this bit is not a 1.
Note: UARTx_FCR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.			

MMIO 0x140000201 (UART0), 0x140000301 (UART1) Read/Write

See *Interrupt Enable Registers (UARTx_IER)* on page 783.

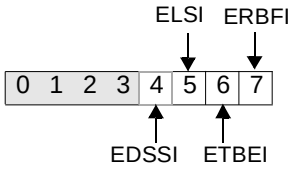


Figure 0-4. UART Interrupt Enable Registers (UARTx_IER)

0:3		Reserved	Always 0.
4	EDSSI	Modem Status Interrupt 0 Disable modem status interrupt 1 Enable modem status interrupt	
5	ELSI	Receiver Line Status Interrupt enable 0 Disable receiver line status interrupt 1 Enable receiver line status interrupt	
6	ETBEI	Transmitter Holding Register Empty Interrupt enable 0 Disable transmitter holding register empty interrupt 1 Enable transmitter holding register empty interrupt	
7	ERBFI	Received Data Available Interrupt enable 0 Disable received data available interrupt 1 Enable received data available interrupt	In FIFO mode, timeout interrupts follow the enable/disable state of ERBFI.
Note: UARTx_IER is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.			

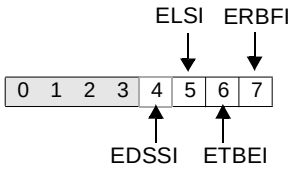


Figure 29-313. UART Interrupt Enable Registers (UARTx_IER)

0:3		Reserved	Always 0.
4	EDSSI	Modem Status Interrupt 0 Disable modem status interrupt 1 Enable modem status interrupt	
5	ELSI	Receiver Line Status Interrupt enable 0 Disable receiver line status interrupt 1 Enable receiver line status interrupt	

6	ETBEI	Transmitter Holding Register Empty Interrupt enable 0 Disable transmitter holding register empty interrupt 1 Enable transmitter holding register empty interrupt	
7	ERBFI	Received Data Available Interrupt enable 0 Disable received data available interrupt 1 Enable received data available interrupt	In FIFO mode, timeout interrupts follow the enable/disable state of ERBFI.
Note: UARTx_IER is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.			

MMIO 0x140000202 (UART0), 0x140000302 (UART1) Read-Only

See *Interrupt Identification Registers (UARTx_IIR)* on page 784.

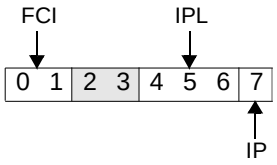


Figure 0-5. UART Interrupt Identification Registers (UARTx_IIR)

0:1	FCI	FIFO Control Indicator 00 FIFOs disabled (UARTx_FCR[FC] = 0) 01 Reserved 10 Reserved 11 FIFOs enabled (UARTx_FCR[FC] = 1)	
2:3		Reserved	
4:6	IPL	Interrupt Priority Level 000 Priority level 4 001 Priority level 3 010 Priority level 2 011 Priority level 1 100 Reserved 101 Reserved 110 Priority level 2 111 Reserved	Note: Priority 1 is highest priority.
7	IP	Interrupt Pending 0 Interrupt is pending 1 No interrupt pending	When set to 0, IIR contents can be used as a pointer to the appropriate interrupt service routine.
Note: UARTx_IIR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.			

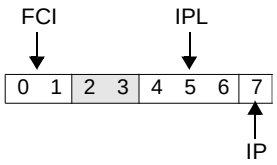


Figure 29-314. UART Interrupt Identification Registers (UARTx_IIR)

0:1	FCI	FIFO Control Indicator 00 FIFOs disabled (UARTx_FCR[FC] = 0) 01 Reserved 10 Reserved 11 FIFOs enabled (UARTx_FCR[FC] = 1)
2:3		Reserved

4:6	IPL	Interrupt Priority Level 000 Priority level 4 001 Priority level 3 010 Priority level 2 011 Priority level 1 100 Reserved 101 Reserved 110 Priority level 2 111 Reserved	Note: Priority 1 is highest priority.
7	IP	Interrupt Pending 0 Interrupt is pending 1 No interrupt pending	When set to 0, IIR contents can be used as a pointer to the appropriate interrupt service routine.
Note: UARTx_IIR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.			

MMIO 0x140000203 (UART0), 0x140000303 (UART1) Read/Write

See *UART Line Control Registers (UARTx_LCR)* on page 787.

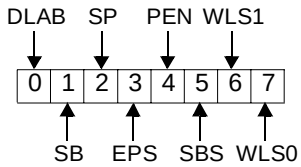


Figure 0-6. UART Line Control Registers (UARTx_LCR)

0	DLAB	Divisor Latch Access Bit 0 Address RBR, THR and IER with LTADR2-0 for read or write operation 1 Address Divisor Latches with LTADR2-0 for read or write operation	
1	SB	Set Break 0 Disable Break 1 Enable Break	Causes a break condition to be transmitted to the UART when the core is receiving. SOUT is forced to the spacing state (0). This bit acts only on SOUT and has no effect on the transmitter logic.
2	SP	Sticky Parity 0 Disable sticky parity 1 Enable sticky parity	If UARTx_LCR[EPS] = 1 and UARTx_LCR[PEN] = 1, the parity bit is transmitted and checked as 0. If UARTx_LCR [EPS] = 0 and UARTx_LCR[PEN] = 1,the parity bit is transmitted and checked as 1.
3	EPS	Even Parity Select 0 Generate odd parity 1 Generate even parity	This bit is significant only if UARTx_LCR[PEN] = 1.
4	PEN	Parity Enable 0 Disable parity checking 1 Enable parity checking	
5	SBS	Stop Bit Select 0 Characters have 1 stop bit 1 Characters have 1.5 or 2 stop bits	If UARTx_LCR[WPS1,WPS0] = 00, characters have 1.5 stop bits. For any other value of UARTx_LCR[WPS1,WPS0], characters have 2 stop bits. The receiver checks the first stop bit only, regardless of how many stop bits are selected.
6	WLS1	Word Length Select Bits 1 00 Use 5-bit characters 01 Use 6-bit characters 10 Use 7-bit characters 11 Use 8-bit characters	

7	WLS0	Word Length Select Bits 0 00 Use 5-bit characters 01 Use 6-bit characters 10 Use 7-bit characters 11 Use 8-bit characters
Note: UARTx_LCR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.		

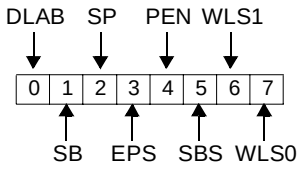


Figure 29-315. UART Line Control Registers (UARTx_LCR)

0	DLAB	Divisor Latch Access Bit 0 Address RBR, THR and IER with LTADR2-0 for read or write operation 1 Address Divisor Latches with LTADR2-0 for read or write operation	
1	SB	Set Break 0 Disable Break 1 Enable Break	Causes a break condition to be transmitted to the UART when the core is receiving. SOUT is forced to the spacing state (0). This bit acts only on SOUT and has no effect on the transmitter logic.
2	SP	Sticky Parity 0 Disable sticky parity 1 Enable sticky parity	If UARTx_LCR[EPS] = 1 and UARTx_LCR[PEN] = 1, the parity bit is transmitted and checked as 0. If UARTx_LCR [EPS] = 0 and UARTx_LCR[PEN] = 1,the parity bit is transmitted and checked as 1.
3	EPS	Even Parity Select 0 Generate odd parity 1 Generate even parity	This bit is significant only if UARTx_LCR[PEN] = 1.
4	PEN	Parity Enable 0 Disable parity checking 1 Enable parity checking	
5	SBS	Stop Bit Select 0 Characters have 1 stop bit 1 Characters have 1.5 or 2 stop bits	If UARTx_LCR[WPS1,WPS0] = 00, characters have 1.5 stop bits. For any other value of UARTx_LCR[WPS1,WPS0], characters have 2 stop bits. The receiver checks the first stop bit only, regardless of how many stop bits are selected.
6	WLS1	Word Length Select Bits 1 00 Use 5-bit characters 01 Use 6-bit characters 10 Use 7-bit characters 11 Use 8-bit characters	

7	WLS0	Word Length Select Bits 0 00 Use 5-bit characters 01 Use 6-bit characters 10 Use 7-bit characters 11 Use 8-bit characters
Note: UARTx_LCR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.		

MMIO 0x140000205 (UART0), 0x140000305 (UART1) R/W

See *Line Status Registers (UARTx_LSR)* on page 789.

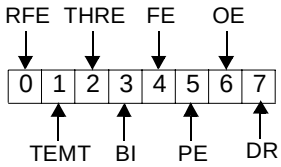


Figure 0-7. UART Line Status Registers (UARTx_LSR)

0	RFE	Receiver FIFO Error Indicator 0 In FIFO mode, reset to 0 when the processor reads the UARTx_LSR, provided there are no subsequent errors in the FIFO. 1 There are one or more instances of parity error, framing error or break indication in the FIFO.	Always 0 in 16450 mode.
1	TEMT	Transmitter Empty Indicator 0 Reset to 0 whenever the THR or the transmitter shift register contain a character. In FIFO mode, it is reset to 0 whenever the transmitter FIFO or the transmitter shift register contain a character. 1 Set to 1 when the THR and the Transmitter shift register are both empty. In FIFO mode, it is set to 1 when the transmitter FIFO and the transmitter shift register are both empty.	
2	THRE	Transmitter Holding Register Empty Indicator 0 Concurrent reset to 0 with the loading of the THR by the processor. In FIFO mode it is reset to 0 when at least one byte is written to the transmitter FIFO. 1 Set to 1 when the UART is ready to accept a new character for transmission. In FIFO mode, this bit is set when the transmitter FIFO is empty.	When UARTx_IER[THRE] = 1, the UART issues an interrupt to the interrupt controller. This bit is set to 1 when a character is transferred from the THR to the transmitter shift register.

3	BI	Break Interrupt Indicator. 0 Reset to 0 whenever processor reads Line Status Register (LSR). 1 Set to 1 whenever the received data input is held at the spacing level (0) for longer than a full word transmission time.	The full word transmission time is the time required for the start bit, data bits (can be 5–8 bits), parity and stop bits. In FIFO mode, this error is revealed to the processor when the character this error is associated with is at the top of the FIFO. Only one 0 character is loaded into the receiver FIFO when a break occurs. After the next valid start bit is received and has gone into the marking state, the next character transfer is enabled. Error causes a Receiver Line Status Interrupt.
4	FE	Framing Error Indicator. 0 Reset to 0 whenever processor reads LSR. 1 Set to 1 whenever stop bit following the last data bit or parity bit is detected as 0 (spacing level).Indicates that a valid stop bit was not found in the received character.	Error causes a Receiver Line Status Interrupt.
5	PE	Parity Error Indicator. 0 Reset to 0 whenever processor reads UARTx_LSR. 1 Indicates that the received data character does not have the correct parity as determined by the even parity select bit (UARTx_LCR.[EPS]). Set to 1 upon detection of a parity error.	In FIFO mode, this error is revealed to the processor when the character this error is associated with is at the top of the FIFO. Error causes a Receiver Line Status Interrupt.
6	OE	Overrun Error Indicator. 0 Reset to 0 whenever processor reads UARTx_LSR. 1 Data in the RBR was read by the processor before the next character was transferred into the UARTx_RBR, hence the original data was lost.	In FIFO mode, if the incoming data continues to fill the FIFO beyond the trigger level, an OE occurs only after the FIFO is completely full and the entire next character has been received in the receiver shift register. The processor is informed of the OE immediately upon occurrence. The character in the shift register will be overwritten and will not be transferred to the FIFO. Error causes a Receiver Line Status Interrupt.
7	DR	Receiver Data Ready Indicator. 0 Reset to 0 when all data has been read from the receiver FIFO or the UARTx_RBR. 1 An entire incoming character has been received into the UARTx_RBR or receiver FIFO.	
Note: UARTx_LSR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.			

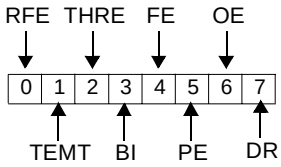


Figure 29-316. UART Line Status Registers (UARTx_LSR)

0	RFE	Receiver FIFO Error Indicator 0 In FIFO mode, reset to 0 when the processor reads the UARTx_LSR, provided there are no subsequent errors in the FIFO. 1 There are one or more instances of parity error, framing error or break indication in the FIFO.	Always 0 in 16450 mode.
1	TEMT	Transmitter Empty Indicator 0 Reset to 0 whenever the THR or the transmitter shift register contain a character. In FIFO mode, it is reset to 0 whenever the transmitter FIFO or the transmitter shift register contain a character. 1 Set to 1 when the THR and the Transmitter shift register are both empty. In FIFO mode, it is set to 1 when the transmitter FIFO and the transmitter shift register are both empty.	
2	THRE	Transmitter Holding Register Empty Indicator 0 Concurrent reset to 0 with the loading of the THR by the processor. In FIFO mode it is reset to 0 when at least one byte is written to the transmitter FIFO. 1 Set to 1 when the UART is ready to accept a new character for transmission. In FIFO mode, this bit is set when the transmitter FIFO is empty.	When UARTx_IER[THRE] = 1, the UART issues an interrupt to the PPC440GP interrupt controller. This bit is set to 1 when a character is transferred from the THR to the transmitter shift register.
3	BI	Break Interrupt Indicator. 0 Reset to 0 whenever processor reads Line Status Register (LSR). 1 Set to 1 whenever the received data input is held at the spacing level (0) for longer than a full word transmission time.	The full word transmission time is the time required for the start bit, data bits (can be 5–8 bits), parity and stop bits. In FIFO mode, this error is revealed to the processor when the character this error is associated with is at the top of the FIFO. Only one 0 character is loaded into the receiver FIFO when a break occurs. After the next valid start bit is received and has gone into the marking state, the next character transfer is enabled. Error causes a Receiver Line Status Interrupt.
4	FE	Framing Error Indicator. 0 Reset to 0 whenever processor reads LSR. 1 Set to 1 whenever stop bit following the last data bit or parity bit is detected as 0 (spacing level).Indicates that a valid stop bit was not found in the received character.	Error causes a Receiver Line Status Interrupt.
5	PE	Parity Error Indicator. 0 Reset to 0 whenever processor reads UARTx_LSR. 1 Indicates that the received data character does not have the correct parity as determined by the even parity select bit (UARTx_LCR.[EPS]). Set to 1 upon detection of a parity error.	In FIFO mode, this error is revealed to the processor when the character this error is associated with is at the top of the FIFO. Error causes a Receiver Line Status Interrupt.

6	OE	Overrun Error Indicator. 0 Reset to 0 whenever processor reads UARTx_LSR. 1 Data in the RBR was read by the processor before the next character was transferred into the UARTx_RBR, hence the original data was lost.	In FIFO mode, if the incoming data continues to fill the FIFO beyond the trigger level, an OE occurs only after the FIFO is completely full and the entire next character has been received in the receiver shift register. The processor is informed of the OE immediately upon occurrence. The character in the shift register will be overwritten and will not be transferred to the FIFO. Error causes a Receiver Line Status Interrupt.
7	DR	Receiver Data Ready Indicator. 0 Reset to 0 when all data has been read from the receiver FIFO or the UARTx_RBR. 1 An entire incoming character has been received into the UARTx_RBR or receiver FIFO.	
Note: UARTx_LSR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.			

MMIO 0x140000204 (UART0), 0x140000304 (UART1) Read/Write

See *Modem Control Registers (UARTx_MCR)* on page 788.

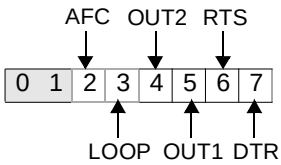


Figure 0-8. UART Modem Control Registers (UARTx_MCR)

0:1		Reserved	Always 0.
2	AFC	Auto Flow Control 0 Hardware flow control disabled 1 Hardware flow control enabled	
3	LOOP	Loopback Mode 0 Disabled 1 Enabled	Provides a local loopback feature for diagnostic testing of the UART. The following occurs: 1. SOUT is set to the marking state (logic 1) SIN is disconnected. 2. The output of the transmitter shift register feeds the input of the receiver shift register. 3. The four modem control inputs $\overline{DSR}$, $\overline{CTS}$, $\overline{RI}$, and $\overline{DCD}$ are disconnected. 4. The four modem control outputs $\overline{DTR}$, $\overline{RTS}$, $\overline{OUT1}$, and $\overline{OUT2}$ are set to a logic 1 (their inactive state). 5. The four modem control outputs are connected internally to the four modem control inputs. Transmitted data is immediately received to verify the UART transmit and receive data paths. Receiver and transmitter interrupts are operational. Their sources are external to the UART. Also operational are the modem control interrupts, but their source is the low-order 4 bits of UARTx_MCR instead of the modem control inputs to the UART. UARTx_IER still controls the interrupts.
4	OUT2	User Output 2 0 $\overline{OUT2}$ inactive (1) 1 $\overline{OUT2}$ active (0)	The $\overline{OUT2}$ bit may be written and read but it provides no function
5	OUT1	User Output 1 0 $\overline{OUT1}$ inactive (1) 1 $\overline{OUT1}$ active (0)	The $\overline{OUT1}$ bit may be written and read but it provides no function
6	RTS	Request To Send 0 $\overline{RTS}$ inactive (1) 1 $\overline{RTS}$ active (0)	

7	DTR	Data Terminal Ready 0 $\overline{\text{DTR}}$ inactive (1) 1 $\overline{\text{DTR}}$ active (0)
Note: UARTx_MCR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.		

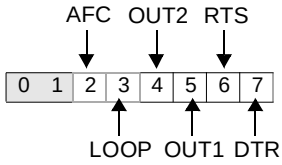


Figure 29-317. UART Modem Control Registers (UARTx_MCR)

0:1		Reserved	Always 0.
2	AFC	Auto Flow Control 0 Hardware flow control disabled 1 Hardware flow control enabled	
3	LOOP	Loopback Mode 0 Disabled 1 Enabled	Provides a local loopback feature for diagnostic testing of the UART. The following occurs: 1. SOUT is set to the marking state (logic 1) SIN is disconnected. 2. The output of the transmitter shift register feeds the input of the receiver shift register. 3. The four modem control inputs $\overline{\text{DSR}}$, $\overline{\text{CTS}}$, $\overline{\text{RI}}$, and $\overline{\text{DCD}}$ are disconnected. 4. The four modem control outputs $\overline{\text{DTR}}$, $\overline{\text{RTS}}$, $\overline{\text{OUT1}}$, and $\overline{\text{OUT2}}$ are set to a logic 1 (their inactive state). 5. The four modem control outputs are connected internally to the four modem control inputs. Transmitted data is immediately received to verify the UART transmit and receive data paths. Receiver and transmitter interrupts are operational. Their sources are external to the UART. Also operational are the modem control interrupts, but their source is the low-order 4 bits of UARTx_MCR instead of the modem control inputs to the UART. UARTx_IER still controls the interrupts.
4	OUT2	User Output 2 0 $\overline{\text{OUT2}}$ inactive (1) 1 $\overline{\text{OUT2}}$ active (0)	The $\overline{\text{OUT2}}$ bit may be written and read but it provides no function
5	OUT1	User Output 1 0 $\overline{\text{OUT1}}$ inactive (1) 1 $\overline{\text{OUT1}}$ active (0)	The $\overline{\text{OUT1}}$ bit may be written and read but it provides no function
6	RTS	Request To Send 0 $\overline{\text{RTS}}$ inactive (1) 1 $\overline{\text{RTS}}$ active (0)	
7	DTR	Data Terminal Ready 0 $\overline{\text{DTR}}$ inactive (1) 1 $\overline{\text{DTR}}$ active (0)	
Note: UARTx_MCR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.			

MMIO 0x140000206 (UART0), 0x140000306 (UART1) Read/Write

See *Modem Control Registers (UARTx_MCR)* on page 788.

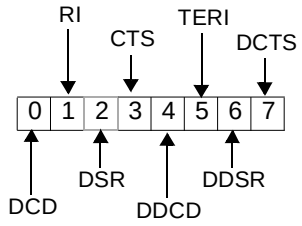


Figure 0-9. UART Modem Status Registers (UARTx_MSR)

0	DCD	Data Carrier Detect	In loopback mode (UARTx_MCR[LB] is 1), it is equivalent to UARTx_MCR[OUT2].
1	RI	Ring Indicator	In loopback mode (UARTx_MCR[LB] is 1), it is equivalent to UARTx_MCR[OUT1].
2	DSR	Data Set Ready	In loopback mode (UARTx_MCR[LB] is 1), it is equivalent to UARTx_MCR[DTR].
3	CTS	Clear To Send	In loopback mode (UARTx_MCR[LB] is 1), it is equivalent to UARTx_MCR[RTS].
4	DDCD	Delta Data Carrier Detect 0 Set when processor reads the Modem Status Register 1 $\overline{\text{DCD}}$ input changed state	Indicates that the $\overline{\text{DCD}}$ input to the UART has changed state since the processor last read the Modem Status Register. A modem status interrupt is generated.
5	TERI	Trailing Edge of Ring Indicator 0 Set when processor reads the Modem Status Register 1 $\overline{\text{RI}}$ input changed from 0 to 1	Indicates that the $\overline{\text{RI}}$ input to the UART changed from 0 to 1 since the processor last read the Modem Status Register. A modem status interrupt is generated.
6	DDSR	Delta Data Set Ready 0 Set when processor reads the Modem Status Register 1 $\overline{\text{DSR}}$ input changed state	Indicates that the $\overline{\text{DSR}}$ input to the UART has changed state since the processor last read the Modem Status Register. A modem status interrupt is generated.
7	DCTS	Delta Clear To Send 0 Set when processor reads the Modem Status Register 1 $\overline{\text{CTS}}$ input changed state	Indicates that the $\overline{\text{CTS}}$ input to the UART has changed state since the processor last read the Modem Status Register. A modem status interrupt is generated.
Note: UARTx_MSR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.			

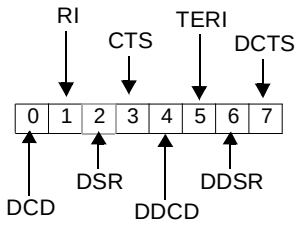


Figure 29-318. UART Modem Status Registers (UARTx_MSR)

0	DCD	Data Carrier Detect	In loopback mode (UARTx_MCR[LB] is 1), it is equivalent to UARTx_MCR[OUT2].
1	RI	Ring Indicator	In loopback mode (UARTx_MCR[LB] is 1), it is equivalent to UARTx_MCR[OUT1].
2	DSR	Data Set Ready	In loopback mode (UARTx_MCR[LB] is 1), it is equivalent to UARTx_MCR[DTR].
3	CTS	Clear To Send	In loopback mode (UARTx_MCR[LB] is 1), it is equivalent to UARTx_MCR[RTS].
4	DDCD	Delta Data Carrier Detect 0 Set when processor reads the Modem Status Register 1 $\overline{DCD}$ input changed state	Indicates that the $\overline{DCD}$ input to the UART has changed state since the processor last read the Modem Status Register. A modem status interrupt is generated.
5	TERI	Trailing Edge of Ring Indicator 0 Set when processor reads the Modem Status Register 1 $\overline{RI}$ input changed from 0 to 1	Indicates that the $\overline{RI}$ input to the UART changed from 0 to 1 since the processor last read the Modem Status Register. A modem status interrupt is generated.
6	DDSR	Delta Data Set Ready 0 Set when processor reads the Modem Status Register 1 $\overline{DSR}$ input changed state	Indicates that the $\overline{DSR}$ input to the UART has changed state since the processor last read the Modem Status Register. A modem status interrupt is generated.
7	DCTS	Delta Clear To Send 0 Set when processor reads the Modem Status Register 1 $\overline{CTS}$ input changed state	Indicates that the $\overline{CTS}$ input to the UART has changed state since the processor last read the Modem Status Register. A modem status interrupt is generated.
Note: UARTx_MSR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.			

MMIO 0x140000200, 0x140000300 Read-Only

See *Receiver Buffer Registers (UARTx_RBR)* on page 783.

Figure 0-10. UART Receiver Buffer Registers (UARTx_RBR)

0:7		Data bit
Note: UARTx_RBR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.		

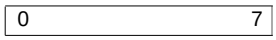


Figure 29-319. UART Receiver Buffer Registers (UARTx_RBR)

0:7		Data bit
Note: UARTx_RBR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.		

MMIO 0x140000207 (UART0), 0x140000307 (UART1) Read/Write

See *Scratchpad Registers (UARTx_SCR)* on page 791.

Figure 0-11. Scratchpad Registers (UARTx_SCR)

0:7		Data bits
Note: UARTx_SCR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.		

Figure 29-320. Scratchpad Registers (UARTx_SCR)

0:7		Data bits
Note: UARTx_SCR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.		

MMIO 0x140000200 (UART0), 0x140000300 (UART1) Write-Only

See *Transmitter Holding Registers (UARTx_THR)* on page 783.

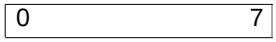


Figure 0-12. UART Transmitter Holding Registers (UARTx_THR)

0:7		Data bit
Note: UARTx_THR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.		

Figure 29-321. UART Transmitter Holding Registers (UARTx_THR)

0:7		Data bit
Note: UARTx_THR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.		

DCR 0x0C3 Read/Write

See *UIC0 Critical Register (UIC0_CR)* on page 313.

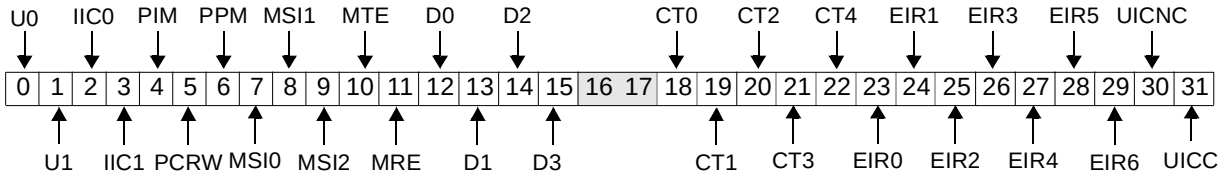


Figure 0-1. UIC0 Critical Register (UIC0_CR)

0	U0	UART0 Interrupt Class 0 UART0 interrupt non-critical. 1 UART0 interrupt is critical.
1	U1	UART1 Interrupt Class 0 UART1 interrupt is non-critical. 1 UART1 interrupt is critical.
2	IIC0	IIC0 Interrupt Class 0 IIC0 interrupt is non-critical. 1 IIC0 interrupt is critical.
3	IIC1	IIC1 Interrupt Class 0 IIC1 interrupt is non-critical. 1 IIC1 interrupt is critical.
4	PIM	PCI Inbound Message Interrupt Class 0 PCI inbound message interrupt is non-critical. 1 PCI inbound message interrupt is critical.
5	PCRW	PCI Command Register Write Interrupt Class 0 PCI command register write interrupt is non-critical. 1 PCI command register write interrupt is critical.
6	PPM	PCI Power Management Interrupt Class 0 PCI power management interrupt is non-critical. 1 PCI power management interrupt is critical.
7	MSI0	PCI MSI Level 0 Interrupt Class 0 PCI MSI level 0 interrupt is non-critical. 1 PCI MSI level 0 interrupt is critical.
8	MSI1	PCI MSI Level 1 Interrupt Class 0 PCI MSI level 1 interrupt is non-critical. 1 PCI MSI level 1 interrupt is critical.

9	MSI2	PCI MSI Level 2 Interrupt Class 0 PCI MSI level 2 interrupt is non-critical. 1 PCI MSI level 2 interrupt is critical.
10	MTE	MAL TX EOB Interrupt Class 0 MAL TX EOB interrupt is non-critical. 1 MAL TX EOB interrupt has is critical.
11	MRE	MAL RX EOB Interrupt Class 0 MAL RX EOB interrupt is non-critical. 1 MAL RX EOB interrupt has is critical.
12	D0	DMA Channel 0 Interrupt Class 0 DMA channel 0 interrupt is non-critical. 1 DMA channel 0 interrupt is critical.
13	D1	DMA Channel 1 Interrupt Class 0 DMA channel 1 interrupt is non-critical. 1 DMA channel 1 interrupt is critical.
14	D2	DMA Channel 2 Interrupt Class 0 DMA channel 2 interrupt is non-critical. 1 DMA channel 2 interrupt is critical.
15	D3	DMA Channel 3 Interrupt Class 0 DMA channel 3 interrupt is non-critical. 1 DMA channel 3 interrupt is critical.
16:17		Reserved
18	CT0	GPT Compare Timer 0 Interrupt Class 0 Compare timer 0 interrupt is non-critical. 1 Compare timer 0 interrupt is critical.
19	CT1	GPT Compare Timer 1 Interrupt Class 0 Compare timer1 interrupt is non-critical. 1 Compare timer 1 interrupt is critical.
20	CT2	GPT Compare Timer 2 Interrupt Class 0 Compare timer 2 interrupt is non-critical. 1 Compare timer 2 interrupt is critical.
21	CT3	GPT Compare Timer 3 Interrupt Class 0 Compare timer 3 interrupt is non-critical. 1 Compare timer 3 interrupt is critical.
22	CT4	GPT Compare Timer 4 Interrupt Class 0 Compare timer 4 interrupt is non-critical. 1 Compare timer 4 interrupt is critical.
23	EIR0	External IRQ 0 Interrupt Class 0 External IRQ 0 interrupt is non-critical. 1 External IRQ 0 interrupt is critical.
24	EIR1	External IRQ 1 Interrupt Class 0 External IRQ 1 interrupt is non-critical. 1 External IRQ 1 interrupt is critical.

25	EIR2	External IRQ 2 Interrupt Class 0 External IRQ 2 interrupt is non-critical. 1 External IRQ 2 interrupt is critical.
26	EIR3	External IRQ 3 Interrupt Class 0 External IRQ 3 interrupt is non-critical. 1 External IRQ 3 interrupt is critical.
27	EIR4	External IRQ 4 Interrupt Class 0 External IRQ 4 interrupt is non-critical. 1 External IRQ 4 interrupt is critical.
28	EIR5	External IRQ 5 Interrupt Class 0 External IRQ 5 interrupt is non-critical. 1 An external IRQ 5 interrupt is critical.
29	EIR6	External IRQ 6 Interrupt Class 0 External IRQ 6 interrupt is non-critical. 1 External IRQ 6 interrupt is critical.
30	UIC1NC	UIC1 Non-Critical Interrupt Class 0 UICNC interrupt is non-critical. 1 UICNC interrupt is critical.
31	UIC1C	UIC1 Critical Interrupt Class 0 UICC interrupt is non-critical. 1 UICC interrupt is critical.

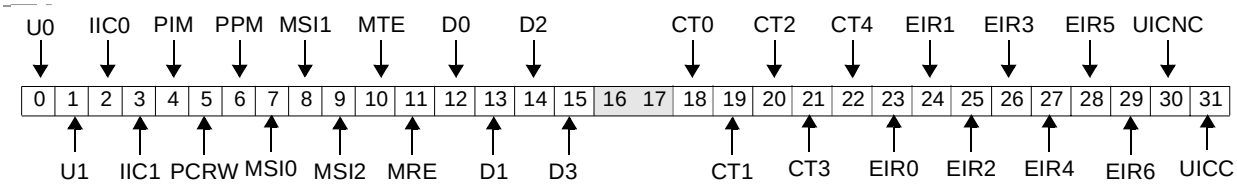


Figure 29-322. UIC0 Critical Register (UIC0_CR)

0	U0	UART0 Interrupt Class 0 UART0 interrupt non-critical. 1 UART0 interrupt is critical.
1	U1	UART1 Interrupt Class 0 UART1 interrupt is non-critical. 1 UART1 interrupt is critical.
2	IIC0	IIC0 Interrupt Class 0 IIC0 interrupt is non-critical. 1 IIC0 interrupt is critical.
3	IIC1	IIC1 Interrupt Class 0 IIC1 interrupt is non-critical. 1 IIC1 interrupt is critical.
4	PIM	PCI Inbound Message Interrupt Class 0 PCI inbound message interrupt is non-critical. 1 PCI inbound message interrupt is critical.

5	PCRW	PCI Command Register Write Interrupt Class 0 PCI command register write interrupt is non-critical. 1 PCI command register write interrupt is critical.
6	PPM	PCI Power Management Interrupt Class 0 PCI power management interrupt is non-critical. 1 PCI power management interrupt is critical.
7	MSI0	PCI MSI Level 0 Interrupt Class 0 PCI MSI level 0 interrupt is non-critical. 1 PCI MSI level 0 interrupt is critical.
8	MSI1	PCI MSI Level 1 Interrupt Class 0 PCI MSI level 1 interrupt is non-critical. 1 PCI MSI level 1 interrupt is critical.
9	MSI2	PCI MSI Level 2 Interrupt Class 0 PCI MSI level 2 interrupt is non-critical. 1 PCI MSI level 2 interrupt is critical.
10	MTE	MAL TX EOB Interrupt Class 0 MAL TX EOB interrupt is non-critical. 1 MAL TX EOB interrupt has is critical.
11	MRE	MAL RX EOB Interrupt Class 0 MAL RX EOB interrupt is non-critical. 1 MAL RX EOB interrupt has is critical.
12	D0	DMA Channel 0 Interrupt Class 0 DMA channel 0 interrupt is non-critical. 1 DMA channel 0 interrupt is critical.
13	D1	DMA Channel 1 Interrupt Class 0 DMA channel 1 interrupt is non-critical. 1 DMA channel 1 interrupt is critical.
14	D2	DMA Channel 2 Interrupt Class 0 DMA channel 2 interrupt is non-critical. 1 DMA channel 2 interrupt is critical.
15	D3	DMA Channel 3 Interrupt Class 0 DMA channel 3 interrupt is non-critical. 1 DMA channel 3 interrupt is critical.
16:17		Reserved
18	CT0	GPT Compare Timer 0 Interrupt Class 0 Compare timer 0 interrupt is non-critical. 1 Compare timer 0 interrupt is critical.
19	CT1	GPT Compare Timer 1 Interrupt Class 0 Compare timer 1 interrupt is non-critical. 1 Compare timer 1 interrupt is critical.
20	CT2	GPT Compare Timer 2 Interrupt Class 0 Compare timer 2 interrupt is non-critical. 1 Compare timer 2 interrupt is critical.
21	CT3	GPT Compare Timer 3 Interrupt Class 0 Compare timer 3 interrupt is non-critical. 1 Compare timer 3 interrupt is critical.
22	CT4	GPT Compare Timer 4 Interrupt Class 0 Compare timer 4 interrupt is non-critical. 1 Compare timer 4 interrupt is critical.

23	EIR0	External IRQ 0 Interrupt Class 0 External IRQ 0 interrupt is non-critical. 1 External IRQ 0 interrupt is critical.
24	EIR1	External IRQ 1 Interrupt Class 0 External IRQ 1 interrupt is non-critical. 1 External IRQ 1 interrupt is critical.
25	EIR2	External IRQ 2 Interrupt Class 0 External IRQ 2 interrupt is non-critical. 1 External IRQ 2 interrupt is critical.
26	EIR3	External IRQ 3 Interrupt Class 0 External IRQ 3 interrupt is non-critical. 1 External IRQ 3 interrupt is critical.
27	EIR4	External IRQ 4 Interrupt Class 0 External IRQ 4 interrupt is non-critical. 1 External IRQ 4 interrupt is critical.
28	EIR5	External IRQ 5 Interrupt Class 0 External IRQ 5 interrupt is non-critical. 1 An external IRQ 5 interrupt is critical.
29	EIR6	External IRQ 6 Interrupt Class 0 External IRQ 6 interrupt is non-critical. 1 External IRQ 6 interrupt is critical.
30	UIC1NC	UIC1 Non-Critical Interrupt Class 0 UICNC interrupt is non-critical. 1 UICNC interrupt is critical.
31	UIC1C	UIC1 Critical Interrupt Class 0 UICC interrupt is non-critical. 1 UICC interrupt is critical.

DCR 0x0C2 Read/Write

See *UIC0 Enable Register (UIC0_ER)* on page 308.

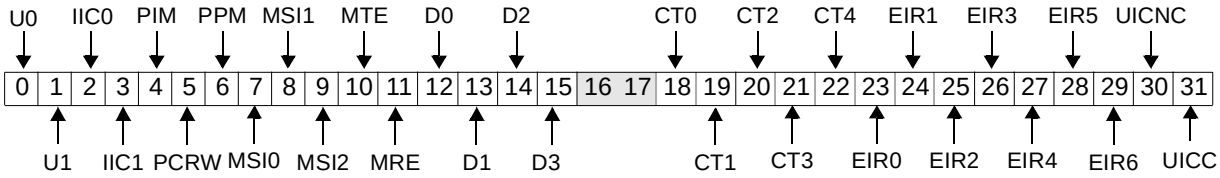


Figure 0-2. UIC0 Enable Register (UIC0_ER)

0	U0	UART0 Interrupt Enable 0 UART0 interrupt is disabled. 1 UART0 interrupt is enabled.
1	U1	UART1 Interrupt Enable 0 UART1 interrupt is disabled. 1 UART1 interrupt is enabled.
2	IIC0	IIC0 Interrupt Enable 0 IIC0 interrupt is disabled. 1 IIC0 interrupt is enabled.
3	IIC1	IIC1 Interrupt Enable 0 IIC1 interrupt is disabled. 1 IIC1 interrupt is enabled.
4	PIM	PCI Inbound Message Interrupt Enable 0 PCI inbound message interrupt is disabled. 1 PCI inbound message interrupt is enabled.
5	PCRW	PCI Command Register Write Interrupt Enable 0 PCI command register write interrupt is disabled. 1 PCI command register write interrupt is enabled.
6	PPM	PCI Power Management Interrupt Enable 0 PCI power management interrupt is disabled. 1 PCI power management interrupt is enabled.
7	MSI0	PCI MSI Level 0 Interrupt Enable 0 PCI MSI Level 0 interrupt is disabled. 1 PCI MSI Level 0 interrupt is enabled.
8	MSI1	PCI MSI Level 1 Interrupt Enable 0 PCI MSI Level 1 Interrupt is disabled. 1 PCI MSI Level 1 interrupt is enabled.

9	MSI2	PCI MSI Level 2 Interrupt Enable 0 PCI MSI Level 2 interrupt is disabled. 1 PCI MSI Level 2 interrupt is enabled.
10	MTE	MAL TX EOB Interrupt Enable 0 MAL TX EOB interrupt is disabled. 1 MAL TX EOB interrupt has is enabled.
11	MRE	MAL RX EOB Interrupt Enable 0 MAL RX EOB interrupt is disabled. 1 MAL RX EOB interrupt has is enabled.
12	D0	DMA Channel 0 Interrupt Enable 0 DMA channel 0 interrupt is disabled. 1 DMA channel 0 interrupt is enabled.
13	D1	DMA Channel 1 Interrupt Enable 0 DMA channel 1 interrupt is disabled. 1 DMA channel 1 interrupt is enabled.
14	D2	DMA Channel 2 Interrupt Enable 0 DMA channel 2 interrupt is disabled. 1 DMA channel 2 interrupt is enabled.
15	D3	DMA Channel 3 Interrupt Enable 0 DMA channel 3 interrupt is disabled. 1 DMA channel 3 interrupt is enabled.
16:17		Reserved
18	CT0	GPT Compare Timer 0 Interrupt Enable 0 Compare timer 0 interrupt is disabled. 1 Compare timer 0 interrupt is enabled.
19	CT1	GPT Compare Timer 1 Interrupt Enable 0 Compare timer1 interrupt is disabled. 1 Compare timer 1 interrupt is enabled.
20	CT2	GPT Compare Timer 2 Interrupt Enable 0 Compare timer 2 interrupt is disabled. 1 Compare timer 2 interrupt is enabled.
21	CT3	GPT Compare Timer 3 Interrupt Enable 0 Compare timer 3 interrupt is disabled. 1 Compare timer 3 interrupt is enabled.
22	CT4	GPT Compare Timer 4 Interrupt Enable 0 Compare timer 4 interrupt is disabled. 1 Compare timer 4 interrupt is enabled.
23	EIR0	External IRQ 0 Interrupt Enable 0 External IRQ 0 interrupt is disabled. 1 External IRQ 0 interrupt is enabled.
24	EIR1	External IRQ 1 Interrupt Enable 0 External IRQ 1 interrupt is disabled. 1 External IRQ 1 interrupt is enabled.

25	EIR2	External IRQ 2 Interrupt Enable 0 External IRQ 2 interrupt is disabled. 1 External IRQ 2 interrupt is enabled.
26	EIR3	External IRQ 3 Interrupt Enable 0 External IRQ 3 interrupt is disabled. 1 External IRQ 3 interrupt is enabled.
27	EIR4	External IRQ 4 Interrupt Enable 0 External IRQ 4 interrupt is disabled. 1 External IRQ 4 interrupt is enabled.
28	EIR5	External IRQ 5 Interrupt Enable 0 External IRQ 5 interrupt is disabled. 1 An external IRQ 5 interrupt is enabled.
29	EIR6	External IRQ 6 Interrupt Enable 0 External IRQ 6 interrupt is disabled. 1 External IRQ 6 interrupt is enabled.
30	UIC1NC	UIC1 Non-Critical Interrupt Enable 0 UICNC interrupt is disabled. 1 UICNC interrupt is enabled.
31	UIC1C	UIC1 Critical Interrupt Enable 0 UICC interrupt is disabled. 1 UICC interrupt is enabled.

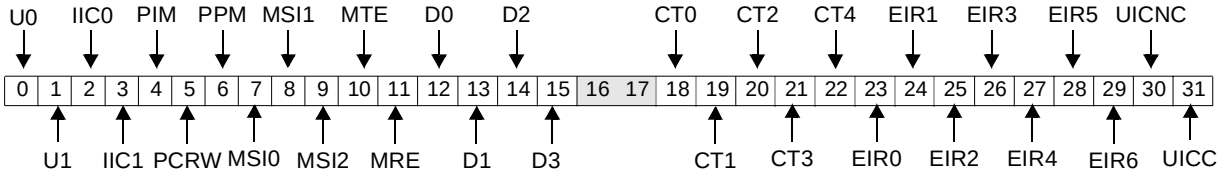


Figure 29-323. UIC0 Enable Register (UIC0_ER)

0	U0	UART0 Interrupt Enable 0 UART0 interrupt is disabled. 1 UART0 interrupt is enabled.
1	U1	UART1 Interrupt Enable 0 UART1 interrupt is disabled. 1 UART1 interrupt is enabled.
2	IIC0	IIC0 Interrupt Enable 0 IIC0 interrupt is disabled. 1 IIC0 interrupt is enabled.
3	IIC1	IIC1 Interrupt Enable 0 IIC1 interrupt is disabled. 1 IIC1 interrupt is enabled.
4	PIM	PCI Inbound Message Interrupt Enable 0 PCI inbound message interrupt is disabled. 1 PCI inbound message interrupt is enabled.

5	PCRW	PCI Command Register Write Interrupt Enable 0 PCI command register write interrupt is disabled. 1 PCI command register write interrupt is enabled.
6	PPM	PCI Power Management Interrupt Enable 0 PCI power management interrupt is disabled. 1 PCI power management interrupt is enabled.
7	MSI0	PCI MSI Level 0 Interrupt Enable 0 PCI MSI Level 0 interrupt is disabled. 1 PCI MSI Level 0 interrupt is enabled.
8	MSI1	PCI MSI Level 1 Interrupt Enable 0 PCI MSI Level 1 Interrupt is disabled. 1 PCI MSI Level 1 interrupt is enabled.
9	MSI2	PCI MSI Level 2 Interrupt Enable 0 PCI MSI Level 2 interrupt is disabled. 1 PCI MSI Level 2 interrupt is enabled.
10	MTE	MAL TX EOB Interrupt Enable 0 MAL TX EOB interrupt is disabled. 1 MAL TX EOB interrupt has is enabled.
11	MRE	MAL RX EOB Interrupt Enable 0 MAL RX EOB interrupt is disabled. 1 MAL RX EOB interrupt has is enabled.
12	D0	DMA Channel 0 Interrupt Enable 0 DMA channel 0 interrupt is disabled. 1 DMA channel 0 interrupt is enabled.
13	D1	DMA Channel 1 Interrupt Enable 0 DMA channel 1 interrupt is disabled. 1 DMA channel 1 interrupt is enabled.
14	D2	DMA Channel 2 Interrupt Enable 0 DMA channel 2 interrupt is disabled. 1 DMA channel 2 interrupt is enabled.
15	D3	DMA Channel 3 Interrupt Enable 0 DMA channel 3 interrupt is disabled. 1 DMA channel 3 interrupt is enabled.
16:17		Reserved
18	CT0	GPT Compare Timer 0 Interrupt Enable 0 Compare timer 0 interrupt is disabled. 1 Compare timer 0 interrupt is enabled.
19	CT1	GPT Compare Timer 1 Interrupt Enable 0 Compare timer 1 interrupt is disabled. 1 Compare timer 1 interrupt is enabled.
20	CT2	GPT Compare Timer 2 Interrupt Enable 0 Compare timer 2 interrupt is disabled. 1 Compare timer 2 interrupt is enabled.
21	CT3	GPT Compare Timer 3 Interrupt Enable 0 Compare timer 3 interrupt is disabled. 1 Compare timer 3 interrupt is enabled.
22	CT4	GPT Compare Timer 4 Interrupt Enable 0 Compare timer 4 interrupt is disabled. 1 Compare timer 4 interrupt is enabled.

23	EIR0	External IRQ 0 Interrupt Enable 0 External IRQ 0 interrupt is disabled. 1 External IRQ 0 interrupt is enabled.
24	EIR1	External IRQ 1 Interrupt Enable 0 External IRQ 1 interrupt is disabled. 1 External IRQ 1 interrupt is enabled.
25	EIR2	External IRQ 2 Interrupt Enable 0 External IRQ 2 interrupt is disabled. 1 External IRQ 2 interrupt is enabled.
26	EIR3	External IRQ 3 Interrupt Enable 0 External IRQ 3 interrupt is disabled. 1 External IRQ 3 interrupt is enabled.
27	EIR4	External IRQ 4 Interrupt Enable 0 External IRQ 4 interrupt is disabled. 1 External IRQ 4 interrupt is enabled.
28	EIR5	External IRQ 5 Interrupt Enable 0 External IRQ 5 interrupt is disabled. 1 An external IRQ 5 interrupt is enabled.
29	EIR6	External IRQ 6 Interrupt Enable 0 External IRQ 6 interrupt is disabled. 1 External IRQ 6 interrupt is enabled.
30	UIC1NC	UIC1 Non-Critical Interrupt Enable 0 UICNC interrupt is disabled. 1 UICNC interrupt is enabled.
31	UIC1C	UIC1 Critical Interrupt Enable 0 UICC interrupt is disabled. 1 UICC interrupt is enabled.

DCR 0x0C6 Read-Only

See *UIC0 Masked Status Register (UIC0_MSR)* on page 327.

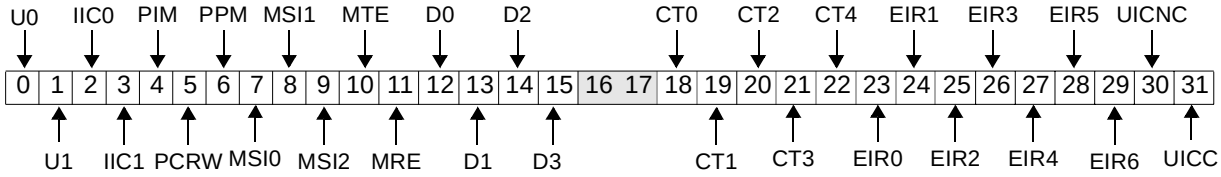


Figure 0-3. UIC0 Masked Status Register (UIC0_MSR)

0	U0	UART0 Masked Interrupt Status 0 UART0masked interrupt has not occurred. 1 UART0 masked interrupt occurred.
1	U1	UART1 Masked Interrupt Status 0 UART1 masked interrupt has not occurred. 1 UART1 masked interrupt occurred.
2	IIC0	IIC0 Masked Interrupt Status 0 IIC0 masked interrupt has not occurred. 1 IIC0 masked interrupt occurred.
3	IIC1	IIC1 Masked Interrupt Status 0 IIC1 masked interrupt has not occurred. 1 IIC1 masked interrupt occurred.
4	PIM	PCI Inbound Message Masked Interrupt Status 0 PCI inbound message masked interrupt has not occurred. 1 PCI inbound message masked interrupt occurred.
5	PCRW	PCI Command Register Write Masked Interrupt Status 0 PCI command register write masked interrupt has not occurred. 1 PCI command register write masked interrupt occurred.
6	PPM	PCI Power Management Masked Interrupt Status 0 PCI power management masked interrupt has not occurred. 1 PCI power management masked interrupt occurred.

7	MSI0	PCI MSI Level 0 Masked Interrupt Status 0 PCI MSI level 0 masked interrupt has not occurred. 1 PCI MSI level 0 masked interrupt occurred.
8	MSI1	PCI MSI Level 1 Masked Interrupt Status 0 PCI MSI level 1 masked interrupt has not occurred. 1 PCI MSI level 1 masked interrupt occurred.
9	MSI2	PCI MSI Level 2 Masked Interrupt Status 0 PCI MSI level 2 masked interrupt has not occurred. 1 PCI MSI level 2 masked interrupt occurred.
10	MTE	MAL TX EOB Masked Interrupt Status 0 MAL TX EOB interrupt has not occurred. 1 MAL TX EOB interrupt has occurred.
11	MRE	MAL RX EOB Masked Interrupt Status 0 MAL RX EOB interrupt has not occurred. 1 MAL RX EOB interrupt has occurred.
12	D0	DMA Channel 0 Masked Interrupt Status 0 DMA channel 0 interrupt has not occurred. 1 DMA channel 0 interrupt occurred.
13	D1	DMA Channel 1 Masked Interrupt Status 0 DMA channel 1 interrupt has not occurred. 1 DMA channel 1 interrupt occurred.
14	D2	DMA Channel 2 Masked Interrupt Status 0 DMA channel 2 interrupt has not occurred. 1 DMA channel 2 interrupt occurred.
15	D3	DMA Channel 3 Masked Interrupt Status 0 DMA channel 3 interrupt has not occurred. 1 DMA channel 3 interrupt occurred.
16:17		Reserved
18	CT0	GPT Compare Timer 0 Masked Interrupt Status 0 Compare timer 0 interrupt has not occurred. 1 Compare timer 0 interrupt occurred.

19	CT1	GPT Compare Timer 1 Masked Interrupt Status 0 Compare timer1 interrupt has not occurred. 1 Compare timer 1 interrupt occurred.
20	CT2	GPT Compare Timer 2 Masked Interrupt Status 0 Compare timer 2 interrupt has not occurred. 1 Compare timer 2 interrupt occurred.
21	CT3	GPT Compare Timer 3 Masked Interrupt Status 0 Compare timer 3 interrupt has not occurred. 1 Compare timer 3 interrupt occurred.
22	CT4	GPT Compare Timer 4 Masked Interrupt Status 0 Compare timer 4 interrupt has not occurred. 1 Compare timer 4 interrupt occurred.
23	EIR0	External IRQ 0 Masked Interrupt Status 0 External IRQ 0 interrupt has not occurred. 1 External IRQ 0 interrupt occurred.
24	EIR1	External IRQ 1 Masked Interrupt Status 0 External IRQ 1 interrupt has not occurred. 1 External IRQ 1 interrupt occurred.
25	EIR2	External IRQ 2 Masked Interrupt Status 0 External IRQ 2 interrupt has not occurred. 1 External IRQ 2 interrupt occurred.
26	EIR3	External IRQ 3 Masked Interrupt Status 0 External IRQ 3 interrupt has not occurred. 1 External IRQ 3 interrupt occurred.
27	EIR4	External IRQ 4 Masked Interrupt Status 0 External IRQ 4 interrupt has not occurred. 1 External IRQ 4 interrupt occurred.
28	EIR5	External IRQ 5 Masked Interrupt Status 0 External IRQ 5 interrupt has not occurred. 1 An external IRQ 5 interrupt occurred.

29	EIR6	External IRQ 6 Masked Interrupt Status 0 External IRQ 6 interrupt has not occurred. 1 External IRQ 6 interrupt occurred.
30	UIC1NC	UIC1 Non-Critical Masked Interrupt Status 0 UICNC interrupt has not occurred. 1 UICNC interrupt occurred.
31	UIC1C	UIC1 Critical Masked Interrupt Status 0 UICC interrupt has not occurred. 1 UICC interrupt occurred.

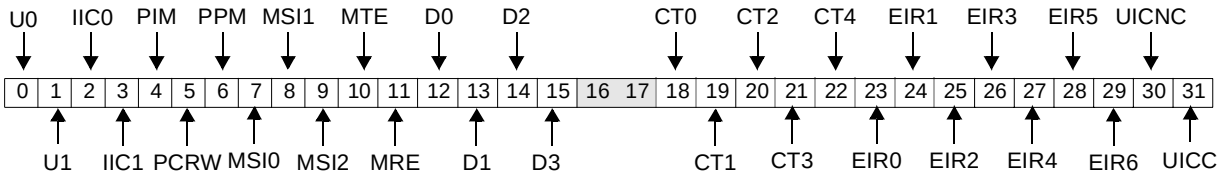


Figure 29-324. UIC0 Masked Status Register (UIC0_MSR)

0	U0	UART0 Masked Interrupt Status 0 UART0masked interrupt has not occurred. 1 UART0 masked interrupt occurred.
1	U1	UART1 Masked Interrupt Status 0 UART1 masked interrupt has not occurred. 1 UART1 masked interrupt occurred.
2	IIC0	IIC0 Masked Interrupt Status 0 IIC0 masked interrupt has not occurred. 1 IIC0 masked interrupt occurred.
3	IIC1	IIC1 Masked Interrupt Status 0 IIC1 masked interrupt has not occurred. 1 IIC1 masked interrupt occurred.
4	PIM	PCI Inbound Message Masked Interrupt Status 0 PCI inbound message masked interrupt has not occurred. 1 PCI inbound message masked interrupt occurred.
5	PCRW	PCI Command Register Write Masked Interrupt Status 0 PCI command register write masked interrupt has not occurred. 1 PCI command register write masked interrupt occurred.
6	PPM	PCI Power Management Masked Interrupt Status 0 PCI power management masked interrupt has not occurred. 1 PCI power management masked interrupt occurred.
7	MSI0	PCI MSI Level 0 Masked Interrupt Status 0 PCI MSI level 0 masked interrupt has not occurred. 1 PCI MSI level 0 masked interrupt occurred.

8	MSI1	PCI MSI Level 1 Masked Interrupt Status 0 PCI MSI level 1 masked interrupt has not occurred. 1 PCI MSI level 1 masked interrupt occurred.
9	MSI2	PCI MSI Level 2 Masked Interrupt Status 0 PCI MSI level 2 masked interrupt has not occurred. 1 PCI MSI level 2 masked interrupt occurred.
10	MTE	MAL TX EOB Masked Interrupt Status 0 MAL TX EOB interrupt has not occurred. 1 MAL TX EOB interrupt has occurred.
11	MRE	MAL RX EOB Masked Interrupt Status 0 MAL RX EOB interrupt has not occurred. 1 MAL RX EOB interrupt has occurred.
12	D0	DMA Channel 0 Masked Interrupt Status 0 DMA channel 0 interrupt has not occurred. 1 DMA channel 0 interrupt occurred.
13	D1	DMA Channel 1 Masked Interrupt Status 0 DMA channel 1 interrupt has not occurred. 1 DMA channel 1 interrupt occurred.
14	D2	DMA Channel 2 Masked Interrupt Status 0 DMA channel 2 interrupt has not occurred. 1 DMA channel 2 interrupt occurred.
15	D3	DMA Channel 3 Masked Interrupt Status 0 DMA channel 3 interrupt has not occurred. 1 DMA channel 3 interrupt occurred.
16:17		Reserved
18	CT0	GPT Compare Timer 0 Masked Interrupt Status 0 Compare timer 0 interrupt has not occurred. 1 Compare timer 0 interrupt occurred.
19	CT1	GPT Compare Timer 1 Masked Interrupt Status 0 Compare timer 1 interrupt has not occurred. 1 Compare timer 1 interrupt occurred.
20	CT2	GPT Compare Timer 2 Masked Interrupt Status 0 Compare timer 2 interrupt has not occurred. 1 Compare timer 2 interrupt occurred.
21	CT3	GPT Compare Timer 3 Masked Interrupt Status 0 Compare timer 3 interrupt has not occurred. 1 Compare timer 3 interrupt occurred.
22	CT4	GPT Compare Timer 4 Masked Interrupt Status 0 Compare timer 4 interrupt has not occurred. 1 Compare timer 4 interrupt occurred.
23	EIR0	External IRQ 0 Masked Interrupt Status 0 External IRQ 0 interrupt has not occurred. 1 External IRQ 0 interrupt occurred.
24	EIR1	External IRQ 1 Masked Interrupt Status 0 External IRQ 1 interrupt has not occurred. 1 External IRQ 1 interrupt occurred.
25	EIR2	External IRQ 2 Masked Interrupt Status 0 External IRQ 2 interrupt has not occurred. 1 External IRQ 2 interrupt occurred.

26	EIR3	External IRQ 3 Masked Interrupt Status 0 External IRQ 3 interrupt has not occurred. 1 External IRQ 3 interrupt occurred.
27	EIR4	External IRQ 4 Masked Interrupt Status 0 External IRQ 4 interrupt has not occurred. 1 External IRQ 4 interrupt occurred.
28	EIR5	External IRQ 5 Masked Interrupt Status 0 External IRQ 5 interrupt has not occurred. 1 An external IRQ 5 interrupt occurred.
29	EIR6	External IRQ 6 Masked Interrupt Status 0 External IRQ 6 interrupt has not occurred. 1 External IRQ 6 interrupt occurred.
30	UIC1NC	UIC1 Non-Critical Masked Interrupt Status 0 UICNC interrupt has not occurred. 1 UICNC interrupt occurred.
31	UIC1C	UIC1 Critical Masked Interrupt Status 0 UICC interrupt has not occurred. 1 UICC interrupt occurred.

DCR 0x0C4 Read/Write

See *UIC0 Polarity Register (UIC0_PR)* on page 317.

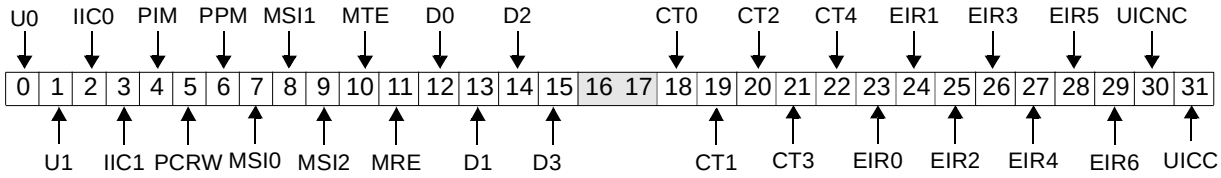


Figure 0-4. UIC0 Polarity Register (UIC0_PR)

0	U0	UART0 Interrupt Polarity 0 UART0 interrupt has negative polarity. 1 UART0 interrupt has positive polarity.
1	U1	UART1 Interrupt Polarity 0 UART1 interrupt has negative polarity. 1 UART1 interrupt has positive polarity.
2	IIC0	IIC0 Interrupt Polarity 0 IIC0 interrupt has negative polarity. 1 IIC0 interrupt has positive polarity.
3	IIC1	IIC1 Interrupt Polarity 0 IIC1 interrupt has negative polarity. 1 IIC1 interrupt has positive polarity.
4	PIM	PCI Inbound Message Interrupt Polarity 0 PCI inbound message interrupt has negative polarity. 1 PCI inbound message interrupt has positive polarity.
5	PCRW	PCI Command Register Write Interrupt Polarity 0 PCI command register write interrupt has negative polarity. 1 PCI command register write interrupt has positive polarity.
6	PPM	PCI Power Management Interrupt Polarity 0 PCI power management interrupt has negative polarity. 1 PCI power management interrupt has positive polarity.
7	MSI0	PCI MSI Level 0 Interrupt Polarity 0 PCI MSI level 0 interrupt has negative polarity. 1 PCI MSI level 0 interrupt has positive polarity.
8	MSI1	PCI MSI Level 1 Interrupt Polarity 0 PCI MSI level 1 interrupt has negative polarity. 1 PCI MSI level 1interrupt has positive polarity.

9	MSI2	PCI MSI Level 2 Interrupt Polarity 0 PCI MSI level 2 interrupt has negative polarity. 1 PCI MSI level 2 interrupt has positive polarity.
10	MTE	MAL TX EOB Interrupt Polarity 0 MAL TX EOB interrupt has negative polarity. 1 MAL TX EOB interrupt has has positive polarity.
11	MRE	MAL RX EOB Interrupt Polarity 0 MAL RX EOB interrupt has negative polarity. 1 MAL RX EOB interrupt has has positive polarity.
12	D0	DMA Channel 0 Interrupt Polarity 0 DMA channel 0 interrupt has negative polarity. 1 DMA channel 0 interrupt has positive polarity.
13	D1	DMA Channel 1 Interrupt Polarity 0 DMA channel 1 interrupt has negative polarity. 1 DMA channel 1 interrupt has positive polarity.
14	D2	DMA Channel 2 Interrupt Polarity 0 DMA channel 2 interrupt has negative polarity. 1 DMA channel 2 interrupt has positive polarity.
15	D3	DMA Channel 3 Interrupt Polarity 0 DMA channel 3 interrupt has negative polarity. 1 DMA channel 3 interrupt has positive polarity.
16:17		Reserved
18	CT0	GPT Compare Timer 0 Interrupt Polarity 0 Compare timer 0 interrupt has negative polarity. 1 Compare timer 0 interrupt has positive polarity.
19	CT1	GPT Compare Timer 1 Interrupt Polarity 0 Compare timer1 interrupt has negative polarity. 1 Compare timer 1 interrupt has positive polarity.
20	CT2	GPT Compare Timer 2 Interrupt Polarity 0 Compare timer 2 interrupt has negative polarity. 1 Compare timer 2 interrupt has positive polarity.
21	CT3	GPT Compare Timer 3 Interrupt Polarity 0 Compare timer 3 interrupt has negative polarity. 1 Compare timer 3 interrupt has positive polarity.

22	CT4	GPT Compare Timer 4 Interrupt Polarity 0 Compare timer 4 interrupt has negative polarity. 1 Compare timer 4 interrupt has positive polarity.
23	EIR0	External IRQ 0 Interrupt Polarity 0 External IRQ 0 interrupt has negative polarity. 1 External IRQ 0 interrupt has positive polarity.
24	EIR1	External IRQ 1 Interrupt Polarity 0 External IRQ 1 interrupt has negative polarity. 1 External IRQ 1 interrupt has positive polarity.
25	EIR2	External IRQ 2 Interrupt Polarity 0 External IRQ 2 interrupt has negative polarity. 1 External IRQ 2 interrupt has positive polarity.
26	EIR3	External IRQ 3 Interrupt Polarity 0 External IRQ 3 interrupt has negative polarity. 1 External IRQ 3 interrupt has positive polarity.
27	EIR4	External IRQ 4 Interrupt Polarity 0 External IRQ 4 interrupt has negative polarity. 1 External IRQ 4 interrupt has positive polarity.
28	EIR5	External IRQ 5 Interrupt Polarity 0 External IRQ 5 interrupt has negative polarity. 1 An external IRQ 5 interrupt has positive polarity.
29	EIR6	External IRQ 6 Interrupt Polarity 0 External IRQ 6 interrupt has negative polarity. 1 External IRQ 6 interrupt has positive polarity.
30	UIC1NC	UIC1 Non-Critical Interrupt Polarity 0 UICNC interrupt has negative polarity. 1 UICNC interrupt has positive polarity.
31	UIC1C	UIC1 Critical Interrupt Polarity 0 UICC interrupt has negative polarity. 1 UICC interrupt has positive polarity.

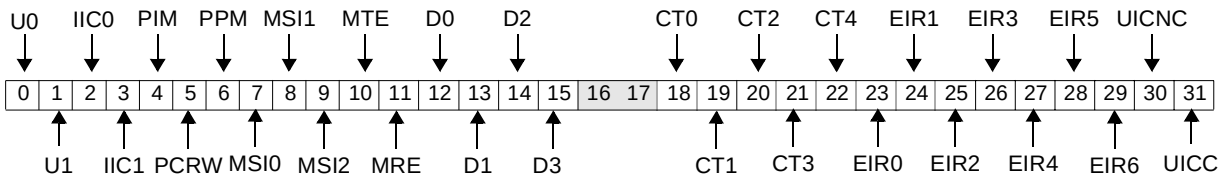


Figure 29-325. UIC0 Polarity Register (UIC0_PR)

0	U0	UART0 Interrupt Polarity 0 UART0 interrupt has negative polarity. 1 UART0 interrupt has positive polarity.

1	U1	UART1 Interrupt Polarity 0 UART1 interrupt has negative polarity. 1 UART1 interrupt has positive polarity.
2	IIC0	IIC0 Interrupt Polarity 0 IIC0 interrupt has negative polarity. 1 IIC0 interrupt has positive polarity.
3	IIC1	IIC1 Interrupt Polarity 0 IIC1 interrupt has negative polarity. 1 IIC1 interrupt has positive polarity.
4	PIM	PCI Inbound Message Interrupt Polarity 0 PCI inbound message interrupt has negative polarity. 1 PCI inbound message interrupt has positive polarity.
5	PCRW	PCI Command Register Write Interrupt Polarity 0 PCI command register write interrupt has negative polarity. 1 PCI command register write interrupt has positive polarity.
6	PPM	PCI Power Management Interrupt Polarity 0 PCI power management interrupt has negative polarity. 1 PCI power management interrupt has positive polarity.
7	MSI0	PCI MSI Level 0 Interrupt Polarity 0 PCI MSI level 0 interrupt has negative polarity. 1 PCI MSI level 0 interrupt has positive polarity.
8	MSI1	PCI MSI Level 1 Interrupt Polarity 0 PCI MSI level 1 interrupt has negative polarity. 1 PCI MSI level 1interrupt has positive polarity.
9	MSI2	PCI MSI Level 2 Interrupt Polarity 0 PCI MSI level 2 interrupt has negative polarity. 1 PCI MSI level 2 interrupt has positive polarity.
10	MTE	MAL TX EOB Interrupt Polarity 0 MAL TX EOB interrupt has negative polarity. 1 MAL TX EOB interrupt has has positive polarity.
11	MRE	MAL RX EOB Interrupt Polarity 0 MAL RX EOB interrupt has negative polarity. 1 MAL RX EOB interrupt has has positive polarity.
12	D0	DMA Channel 0 Interrupt Polarity 0 DMA channel 0 interrupt has negative polarity. 1 DMA channel 0 interrupt has positive polarity.
13	D1	DMA Channel 1 Interrupt Polarity 0 DMA channel 1 interrupt has negative polarity. 1 DMA channel 1 interrupt has positive polarity.
14	D2	DMA Channel 2 Interrupt Polarity 0 DMA channel 2 interrupt has negative polarity. 1 DMA channel 2 interrupt has positive polarity.
15	D3	DMA Channel 3 Interrupt Polarity 0 DMA channel 3 interrupt has negative polarity. 1 DMA channel 3 interrupt has positive polarity.
16:17		Reserved
18	CT0	GPT Compare Timer 0 Interrupt Polarity 0 Compare timer 0 interrupt has negative polarity. 1 Compare timer 0 interrupt has positive polarity.

19	CT1	GPT Compare Timer 1 Interrupt Polarity 0 Compare timer 1 interrupt has negative polarity. 1 Compare timer 1 interrupt has positive polarity.
20	CT2	GPT Compare Timer 2 Interrupt Polarity 0 Compare timer 2 interrupt has negative polarity. 1 Compare timer 2 interrupt has positive polarity.
21	CT3	GPT Compare Timer 3 Interrupt Polarity 0 Compare timer 3 interrupt has negative polarity. 1 Compare timer 3 interrupt has positive polarity.
22	CT4	GPT Compare Timer 4 Interrupt Polarity 0 Compare timer 4 interrupt has negative polarity. 1 Compare timer 4 interrupt has positive polarity.
23	EIR0	External IRQ 0 Interrupt Polarity 0 External IRQ 0 interrupt has negative polarity. 1 External IRQ 0 interrupt has positive polarity.
24	EIR1	External IRQ 1 Interrupt Polarity 0 External IRQ 1 interrupt has negative polarity. 1 External IRQ 1 interrupt has positive polarity.
25	EIR2	External IRQ 2 Interrupt Polarity 0 External IRQ 2 interrupt has negative polarity. 1 External IRQ 2 interrupt has positive polarity.
26	EIR3	External IRQ 3 Interrupt Polarity 0 External IRQ 3 interrupt has negative polarity. 1 External IRQ 3 interrupt has positive polarity.
27	EIR4	External IRQ 4 Interrupt Polarity 0 External IRQ 4 interrupt has negative polarity. 1 External IRQ 4 interrupt has positive polarity.
28	EIR5	External IRQ 5 Interrupt Polarity 0 External IRQ 5 interrupt has negative polarity. 1 An external IRQ 5 interrupt has positive polarity.
29	EIR6	External IRQ 6 Interrupt Polarity 0 External IRQ 6 interrupt has negative polarity. 1 External IRQ 6 interrupt has positive polarity.
30	UIC1NC	UIC1 Non-Critical Interrupt Polarity 0 UICNC interrupt has negative polarity. 1 UICNC interrupt has positive polarity.
31	UIC1C	UIC1 Critical Interrupt Polarity 0 UICC interrupt has negative polarity. 1 UICC interrupt has positive polarity.

DCR 0x0C0 Read/Write

See *UIC0 Status Register (UIC0_SR)* on page 303.

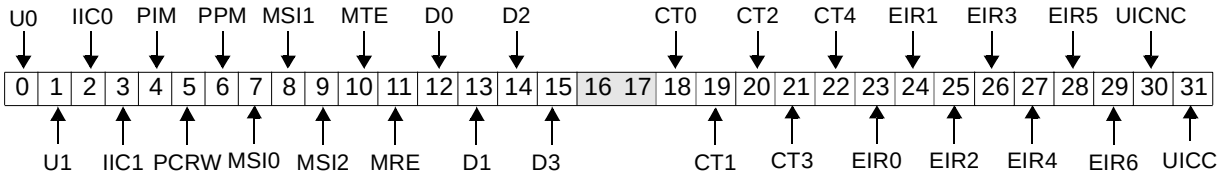


Figure 0-5. UIC0 Status Register (UIC0_SR)

0	U0	UART0 Interrupt Status 0 UART0 interrupt has not occurred. 1 UART0 interrupt occurred.
1	U1	UART1 Interrupt Status 0 UART1 interrupt has not occurred. 1 UART1 interrupt occurred.
2	IIC0	IIC0 Interrupt Status 0 IIC0 interrupt has not occurred. 1 IIC0 interrupt occurred.
3	IIC1	IIC1 Interrupt Status 0 IIC1 interrupt has not occurred. 1 IIC1 interrupt occurred.
4	PIM	PCI Inbound Message Interrupt Status 0 PCI inbound message interrupt has not occurred. 1 PCI inbound message interrupt occurred.
5	PCRW	PCI Command Register Write Interrupt Status 0 PCI command register write interrupt has not occurred. 1 PCI command register write interrupt occurred.
6	PPM	PCI Power Management Interrupt Status 0 PCI power management interrupt has not occurred. 1 PCI power management interrupt occurred.
7	MSI0	PCI MSI Level 0 Interrupt Status 0 PCI MSI Level 0 interrupt has not occurred. 1 PCI MSI Level 0 interrupt occurred.
8	MSI1	PCI MSI Level 1 Interrupt Status 0 PCI MSI Level 1 interrupt has not occurred. 1 PCI MSI Level 1 interrupt occurred.

9	MSI2	PCI MSI Level 2 Interrupt Status 0 PCI MSI Level 2 interrupt has not occurred. 1 PCI MSI Level 2 interrupt occurred.
10	MTE	MAL TX EOB Interrupt Status 0 MAL TX EOB interrupt has not occurred. 1 MAL TX EOB interrupt has occurred.
11	MRE	MAL RX EOB Interrupt Status 0 MAL RX EOB interrupt has not occurred. 1 MAL RX EOB interrupt has occurred.
12	D0	DMA Channel 0 Interrupt Status 0 DMA channel 0 interrupt has not occurred. 1 DMA channel 0 interrupt occurred.
13	D1	DMA Channel 1 Interrupt Status 0 DMA channel 1 interrupt has not occurred. 1 DMA channel 1 interrupt occurred.
14	D2	DMA Channel 2 Interrupt Status 0 DMA channel 2 interrupt has not occurred. 1 DMA channel 2 interrupt occurred.
15	D3	DMA Channel 3 Interrupt Status 0 DMA channel 3 interrupt has not occurred. 1 DMA channel 3 interrupt occurred.
16:17		Reserved
18	CT0	GPT Compare Timer 0 Interrupt Status 0 Compare timer 0 interrupt has not occurred. 1 Compare timer 0 interrupt occurred.
19	CT1	GPT Compare Timer 1 Interrupt Status 0 Compare timer1 interrupt has not occurred. 1 Compare timer 1 interrupt occurred.
20	CT2	GPT Compare Timer 2 Interrupt Status 0 Compare timer 2 interrupt has not occurred. 1 Compare timer 2 interrupt occurred.
21	CT3	GPT Compare Timer 3 Interrupt Status 0 Compare timer 3 interrupt has not occurred. 1 Compare timer 3 interrupt occurred.

22	CT4	GPT Compare Timer 4 Interrupt Status 0 Compare timer 4 interrupt has not occurred. 1 Compare timer 4 interrupt occurred.
23	EIR0	External IRQ 0 Interrupt Status 0 External IRQ 0 interrupt has not occurred. 1 External IRQ 0 interrupt occurred.
24	EIR1	External IRQ 1 Interrupt Status 0 External IRQ 1 interrupt has not occurred. 1 External IRQ 1 interrupt occurred.
25	EIR2	External IRQ 2 Interrupt Status 0 External IRQ 2 interrupt has not occurred. 1 External IRQ 2 interrupt occurred.
26	EIR3	External IRQ 3 Interrupt Status 0 External IRQ 3 interrupt has not occurred. 1 External IRQ 3 interrupt occurred.
27	EIR4	External IRQ 4 Interrupt Status 0 External IRQ 4 interrupt has not occurred. 1 External IRQ 4 interrupt occurred.
28	EIR5	External IRQ 5 Interrupt Status 0 External IRQ 5 interrupt has not occurred. 1 An external IRQ 5 interrupt occurred.
29	EIR6	External IRQ 6 Interrupt Status 0 External IRQ 6 interrupt has not occurred. 1 External IRQ 6 interrupt occurred.
30	UIC1NC	UIC1 Non-Critical Interrupt Status 0 UICNC interrupt has not occurred. 1 UICNC interrupt occurred.
31	UIC1C	UIC1 Critical Interrupt Status 0 UICC interrupt has not occurred. 1 UICC interrupt occurred.

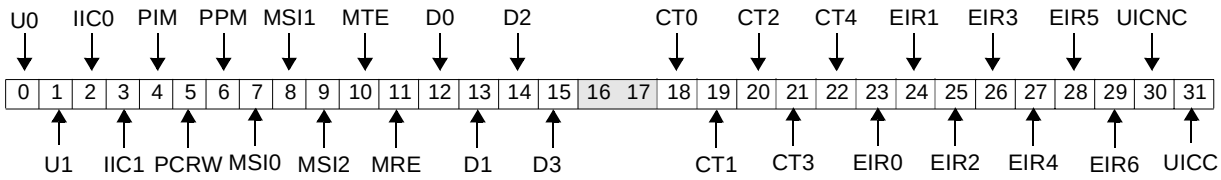


Figure 29-326. UIC0 Status Register (UIC0_SR)

0	U0	UART0 Interrupt Status 0 UART0 interrupt has not occurred. 1 UART0 interrupt occurred.
1	U1	UART1 Interrupt Status 0 UART1 interrupt has not occurred. 1 UART1 interrupt occurred.
2	IIC0	IIC0 Interrupt Status 0 IIC0 interrupt has not occurred. 1 IIC0 interrupt occurred.
3	IIC1	IIC1 Interrupt Status 0 IIC1 interrupt has not occurred. 1 IIC1 interrupt occurred.
4	PIM	PCI Inbound Message Interrupt Status 0 PCI inbound message interrupt has not occurred. 1 PCI inbound message interrupt occurred.
5	PCRW	PCI Command Register Write Interrupt Status 0 PCI command register write interrupt has not occurred. 1 PCI command register write interrupt occurred.
6	PPM	PCI Power Management Interrupt Status 0 PCI power management interrupt has not occurred. 1 PCI power management interrupt occurred.
7	MSI0	PCI MSI Level 0 Interrupt Status 0 PCI MSI Level 0 interrupt has not occurred. 1 PCI MSI Level 0 interrupt occurred.
8	MSI1	PCI MSI Level 1 Interrupt Status 0 PCI MSI Level 1 interrupt has not occurred. 1 PCI MSI Level 1 interrupt occurred.
9	MSI2	PCI MSI Level 2 Interrupt Status 0 PCI MSI Level 2 interrupt has not occurred. 1 PCI MSI Level 2 interrupt occurred.
10	MTE	MAL TX EOB Interrupt Status 0 MAL TX EOB interrupt has not occurred. 1 MAL TX EOB interrupt has occurred.
11	MRE	MAL RX EOB Interrupt Status 0 MAL RX EOB interrupt has not occurred. 1 MAL RX EOB interrupt has occurred.
12	D0	DMA Channel 0 Interrupt Status 0 DMA channel 0 interrupt has not occurred. 1 DMA channel 0 interrupt occurred.

13	D1	DMA Channel 1 Interrupt Status 0 DMA channel 1 interrupt has not occurred. 1 DMA channel 1 interrupt occurred.
14	D2	DMA Channel 2 Interrupt Status 0 DMA channel 2 interrupt has not occurred. 1 DMA channel 2 interrupt occurred.
15	D3	DMA Channel 3 Interrupt Status 0 DMA channel 3 interrupt has not occurred. 1 DMA channel 3 interrupt occurred.
16:17		Reserved
18	CT0	GPT Compare Timer 0 Interrupt Status 0 Compare timer 0 interrupt has not occurred. 1 Compare timer 0 interrupt occurred.
19	CT1	GPT Compare Timer 1 Interrupt Status 0 Compare timer 1 interrupt has not occurred. 1 Compare timer 1 interrupt occurred.
20	CT2	GPT Compare Timer 2 Interrupt Status 0 Compare timer 2 interrupt has not occurred. 1 Compare timer 2 interrupt occurred.
21	CT3	GPT Compare Timer 3 Interrupt Status 0 Compare timer 3 interrupt has not occurred. 1 Compare timer 3 interrupt occurred.
22	CT4	GPT Compare Timer 4 Interrupt Status 0 Compare timer 4 interrupt has not occurred. 1 Compare timer 4 interrupt occurred.
23	EIR0	External IRQ 0 Interrupt Status 0 External IRQ 0 interrupt has not occurred. 1 External IRQ 0 interrupt occurred.
24	EIR1	External IRQ 1 Interrupt Status 0 External IRQ 1 interrupt has not occurred. 1 External IRQ 1 interrupt occurred.
25	EIR2	External IRQ 2 Interrupt Status 0 External IRQ 2 interrupt has not occurred. 1 External IRQ 2 interrupt occurred.
26	EIR3	External IRQ 3 Interrupt Status 0 External IRQ 3 interrupt has not occurred. 1 External IRQ 3 interrupt occurred.
27	EIR4	External IRQ 4 Interrupt Status 0 External IRQ 4 interrupt has not occurred. 1 External IRQ 4 interrupt occurred.
28	EIR5	External IRQ 5 Interrupt Status 0 External IRQ 5 interrupt has not occurred. 1 An external IRQ 5 interrupt occurred.
29	EIR6	External IRQ 6 Interrupt Status 0 External IRQ 6 interrupt has not occurred. 1 External IRQ 6 interrupt occurred.
30	UIC1NC	UIC1 Non-Critical Interrupt Status 0 UICNC interrupt has not occurred. 1 UICNC interrupt occurred.
31	UIC1C	UIC1 Critical Interrupt Status 0 UICC interrupt has not occurred. 1 UICC interrupt occurred.

DCR 0x0C5 Read/Write

See *UIC0 Trigger Register (UIC0_TR)* on page 322.

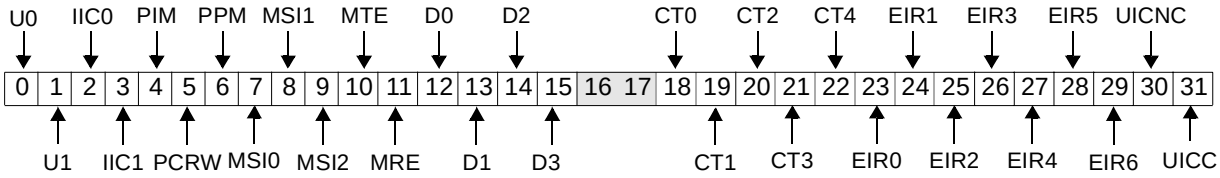


Figure 0-6. UIC0 Trigger Register (UIC0_TR)

0	U0	UART0 Interrupt Trigger 0 UART0 interrupt is level-sensitive. 1 UART0 interrupt is edge-sensitive.
1	U1	UART1 Interrupt Trigger 0 UART1 interrupt is level-sensitive. 1 UART1 interrupt is edge-sensitive.
2	IIC0	IIC0 Interrupt Trigger 0 IIC0 interrupt is level-sensitive. 1 IIC0 interrupt is edge-sensitive.
3	IIC1	IIC1 Interrupt Trigger 0 IIC1 interrupt is level-sensitive. 1 IIC1 interrupt is edge-sensitive.
4	PIM	PCI Inbound Message Interrupt Trigger 0 PCI inbound message interrupt is level-sensitive. 1 PCI inbound message interrupt is edge-sensitive.
5	PCRW	PCI Command Register Write Interrupt Trigger 0 PCI command register write interrupt is level-sensitive. 1 PCI command register write interrupt is edge-sensitive.
6	PPM	PCI Power Management Interrupt Trigger 0 PCI power management interrupt is level-sensitive. 1 PCI power management interrupt is edge-sensitive.
7	MSI0	PCI MSI Level 0 Interrupt Trigger 0 PCI MSI level 0 interrupt is level-sensitive. 1 PCI MSI level 0 interrupt is edge-sensitive.

8	MSI1	PCI MSI Level 1 Interrupt Trigger 0 PCI MSI level 1 interrupt is level-sensitive. 1 PCI MSI level 1 interrupt is edge-sensitive.
9	MSI2	PCI MSI Level 2 Interrupt Trigger 0 PCI MSI level 2 interrupt is level-sensitive. 1 PCI MSI level 2 interrupt is edge-sensitive.
10	MTE	MAL TX EOB Interrupt Trigger 0 MAL TX EOB interrupt is level-sensitive. 1 MAL TX EOB interrupt has is edge-sensitive.
11	MRE	MAL RX EOB Interrupt Trigger 0 MAL RX EOB interrupt is level-sensitive. 1 MAL RX EOB interrupt has is edge-sensitive.
12	D0	DMA Channel 0 Interrupt Trigger 0 DMA channel 0 interrupt is level-sensitive. 1 DMA channel 0 interrupt is edge-sensitive.
13	D1	DMA Channel 1 Interrupt Trigger 0 DMA channel 1 interrupt is level-sensitive. 1 DMA channel 1 interrupt is edge-sensitive.
14	D2	DMA Channel 2 Interrupt Trigger 0 DMA channel 2 interrupt is level-sensitive. 1 DMA channel 2 interrupt is edge-sensitive.
15	D3	DMA Channel 3 Interrupt Trigger 0 DMA channel 3 interrupt is level-sensitive. 1 DMA channel 3 interrupt is edge-sensitive.
16:17		Reserved
18	CT0	GPT Compare Timer 0 Interrupt Trigger 0 Compare timer 0 interrupt is level-sensitive. 1 Compare timer 0 interrupt is edge-sensitive.

19	CT1	GPT Compare Timer 1 Interrupt Trigger 0 Compare timer1 interrupt is level-sensitive. 1 Compare timer 1 interrupt is edge-sensitive.
20	CT2	GPT Compare Timer 2 Interrupt Trigger 0 Compare timer 2 interrupt is level-sensitive. 1 Compare timer 2 interrupt is edge-sensitive.
21	CT3	GPT Compare Timer 3 Interrupt Trigger 0 Compare timer 3 interrupt is level-sensitive. 1 Compare timer 3 interrupt is edge-sensitive.
22	CT4	GPT Compare Timer 4 Interrupt Trigger 0 Compare timer 4 interrupt is level-sensitive. 1 Compare timer 4 interrupt is edge-sensitive.
23	EIR0	External IRQ 0 Interrupt Trigger 0 External IRQ 0 interrupt is level-sensitive. 1 External IRQ 0 interrupt is edge-sensitive.
24	EIR1	External IRQ 1 Interrupt Trigger 0 External IRQ 1 interrupt is level-sensitive. 1 External IRQ 1 interrupt is edge-sensitive.
25	EIR2	External IRQ 2 Interrupt Trigger 0 External IRQ 2 interrupt is level-sensitive. 1 External IRQ 2 interrupt is edge-sensitive.
26	EIR3	External IRQ 3 Interrupt Trigger 0 External IRQ 3 interrupt is level-sensitive. 1 External IRQ 3 interrupt is edge-sensitive.
27	EIR4	External IRQ 4 Interrupt Trigger 0 External IRQ 4 interrupt is level-sensitive. 1 External IRQ 4 interrupt is edge-sensitive.

28	EIR5	External IRQ 5 Interrupt Trigger 0 External IRQ 5 interrupt is level-sensitive. 1 An external IRQ 5 interrupt is edge-sensitive.
29	EIR6	External IRQ 6 Interrupt Trigger 0 External IRQ 6 interrupt is level-sensitive. 1 External IRQ 6 interrupt is edge-sensitive.
30	UIC1NC	UIC1 Non-Critical Interrupt Trigger 0 UICNC interrupt is level-sensitive. 1 UICNC interrupt is edge-sensitive.
31	UIC1C	UIC1 Critical Interrupt Trigger 0 UICC interrupt is level-sensitive. 1 UICC interrupt is edge-sensitive.

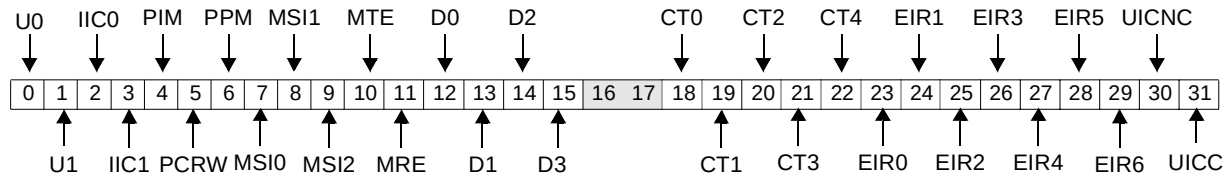


Figure 29-327. UIC0 Trigger Register (UIC0_TR)

0	U0	UART0 Interrupt Trigger 0 UART0 interrupt is level-sensitive. 1 UART0 interrupt is edge-sensitive.
1	U1	UART1 Interrupt Trigger 0 UART1 interrupt is level-sensitive. 1 UART1 interrupt is edge-sensitive.
2	IIC0	IIC0 Interrupt Trigger 0 IIC0 interrupt is level-sensitive. 1 IIC0 interrupt is edge-sensitive.
3	IIC1	IIC1 Interrupt Trigger 0 IIC1 interrupt is level-sensitive. 1 IIC1 interrupt is edge-sensitive.
4	PIM	PCI Inbound Message Interrupt Trigger 0 PCI inbound message interrupt is level-sensitive. 1 PCI inbound message interrupt is edge-sensitive.
5	PCRW	PCI Command Register Write Interrupt Trigger 0 PCI command register write interrupt is level-sensitive. 1 PCI command register write interrupt is edge-sensitive.

6	PPM	PCI Power Management Interrupt Trigger 0 PCI power management interrupt is level-sensitive. 1 PCI power management interrupt is edge-sensitive.
7	MSI0	PCI MSI Level 0 Interrupt Trigger 0 PCI MSI level 0 interrupt is level-sensitive. 1 PCI MSI level 0 interrupt is edge-sensitive.
8	MSI1	PCI MSI Level 1 Interrupt Trigger 0 PCI MSI level 1 interrupt is level-sensitive. 1 PCI MSI level 1 interrupt is edge-sensitive.
9	MSI2	PCI MSI Level 2 Interrupt Trigger 0 PCI MSI level 2 interrupt is level-sensitive. 1 PCI MSI level 2 interrupt is edge-sensitive.
10	MTE	MAL TX EOB Interrupt Trigger 0 MAL TX EOB interrupt is level-sensitive. 1 MAL TX EOB interrupt has is edge-sensitive.
11	MRE	MAL RX EOB Interrupt Trigger 0 MAL RX EOB interrupt is level-sensitive. 1 MAL RX EOB interrupt has is edge-sensitive.
12	D0	DMA Channel 0 Interrupt Trigger 0 DMA channel 0 interrupt is level-sensitive. 1 DMA channel 0 interrupt is edge-sensitive.
13	D1	DMA Channel 1 Interrupt Trigger 0 DMA channel 1 interrupt is level-sensitive. 1 DMA channel 1 interrupt is edge-sensitive.
14	D2	DMA Channel 2 Interrupt Trigger 0 DMA channel 2 interrupt is level-sensitive. 1 DMA channel 2 interrupt is edge-sensitive.
15	D3	DMA Channel 3 Interrupt Trigger 0 DMA channel 3 interrupt is level-sensitive. 1 DMA channel 3 interrupt is edge-sensitive.
16:17		Reserved
18	CT0	GPT Compare Timer 0 Interrupt Trigger 0 Compare timer 0 interrupt is level-sensitive. 1 Compare timer 0 interrupt is edge-sensitive.
19	CT1	GPT Compare Timer 1 Interrupt Trigger 0 Compare timer 1 interrupt is level-sensitive. 1 Compare timer 1 interrupt is edge-sensitive.
20	CT2	GPT Compare Timer 2 Interrupt Trigger 0 Compare timer 2 interrupt is level-sensitive. 1 Compare timer 2 interrupt is edge-sensitive.
21	CT3	GPT Compare Timer 3 Interrupt Trigger 0 Compare timer 3 interrupt is level-sensitive. 1 Compare timer 3 interrupt is edge-sensitive.
22	CT4	GPT Compare Timer 4 Interrupt Trigger 0 Compare timer 4 interrupt is level-sensitive. 1 Compare timer 4 interrupt is edge-sensitive.
23	EIR0	External IRQ 0 Interrupt Trigger 0 External IRQ 0 interrupt is level-sensitive. 1 External IRQ 0 interrupt is edge-sensitive.

24	EIR1	External IRQ 1 Interrupt Trigger 0 External IRQ 1 interrupt is level-sensitive. 1 External IRQ 1 interrupt is edge-sensitive.
25	EIR2	External IRQ 2 Interrupt Trigger 0 External IRQ 2 interrupt is level-sensitive. 1 External IRQ 2 interrupt is edge-sensitive.
26	EIR3	External IRQ 3 Interrupt Trigger 0 External IRQ 3 interrupt is level-sensitive. 1 External IRQ 3 interrupt is edge-sensitive.
27	EIR4	External IRQ 4 Interrupt Trigger 0 External IRQ 4 interrupt is level-sensitive. 1 External IRQ 4 interrupt is edge-sensitive.
28	EIR5	External IRQ 5 Interrupt Trigger 0 External IRQ 5 interrupt is level-sensitive. 1 An external IRQ 5 interrupt is edge-sensitive.
29	EIR6	External IRQ 6 Interrupt Trigger 0 External IRQ 6 interrupt is level-sensitive. 1 External IRQ 6 interrupt is edge-sensitive.
30	UIC1NC	UIC1 Non-Critical Interrupt Trigger 0 UICNC interrupt is level-sensitive. 1 UICNC interrupt is edge-sensitive.
31	UIC1C	UIC1 Critical Interrupt Trigger 0 UICC interrupt is level-sensitive. 1 UICC interrupt is edge-sensitive.

DCR 0x0C8 Write-Only

See *UIC0 Vector Configuration Register (UIC0_VCR)* on page 332.

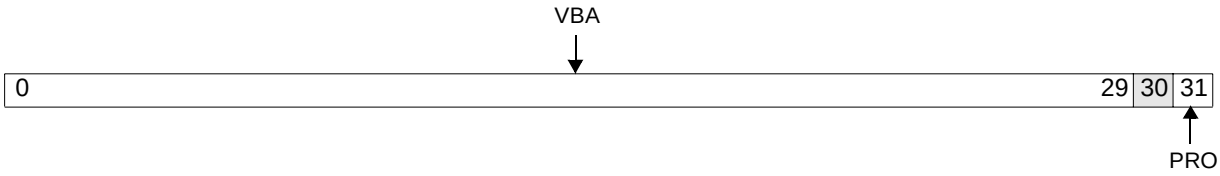


Figure 0-7. UIC0 Vector Configuration Register (UIC0_VCR)

0:29	VBA	Vector Base Address
30		Reserved
31	PRO	Priority Ordering 0 UIC0_SR[31] is the highest priority interrupt. 1 UIC0_SR[0] is the highest priority interrupt. Note: Vector generation is not performed for non-critical interrupts.

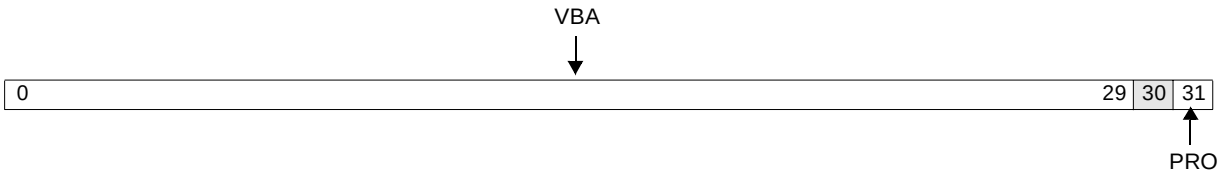


Figure 29-328. UIC0 Vector Configuration Register (UIC0_VCR)

0:29	VBA	Vector Base Address
30		Reserved
31	PRO	Priority Ordering 0 UIC0_SR[31] is the highest priority interrupt. 1 UIC0_SR[0] is the highest priority interrupt. **Note:** Vector generation is not performed for non-critical interrupts.

DCR 0x0C7 Read-Only

See *UIC0 Vector Register (UIC0_VR)* on page 333.

Figure 0-8. UIC Vector Register (UIC*x*_VR)

0:31		Interrupt Vector

Figure 29-329. UIC Vector Register (UIC0_VR)

0:31		Interrupt Vector

DCR 0x0D3 Read/Write

See *UIC1 Critical Register (UIC1_CR)* on page 315.

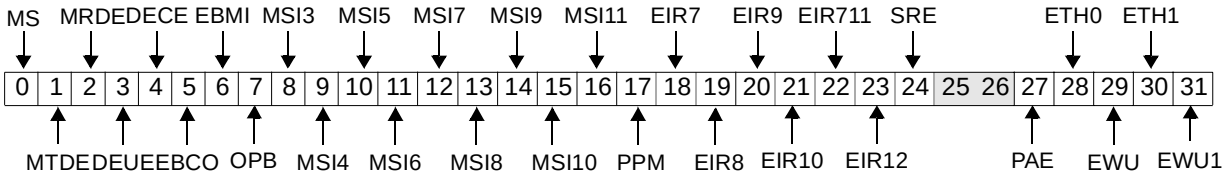


Figure 0-9. UIC1 Critical Register (UIC1_CR)

0	MS	MAL SERR Interrupt Class 0 MAL SERR interrupt is non-critical. 1 MAL SERR interrupt is critical.
1	MTDE	MAL TXDE Interrupt Class 0 MAL TXDE interrupt is non-critical. 1 MAL TXDE interrupt is critical.
2	MRDE	MAL RXDE Interrupt Class 0 MAL RXDE interrupt is non-critical. 1 MAL RXDE interrupt is critical.
3	DEUE	DDRSDRAM ECC Uncorrectable Error Interrupt Class 0 DDRSDRAM ECC uncorrectable error interrupt is non-critical. 1 DDRSDRAM ECC uncorrectable error interrupt is critical.
4	DECE	DDRSDRAM ECC Correctable Error Interrupt Class 0 DDRSDRAM ECC correctable error interrupt is non-critical. 1 DDRSDRAM ECC correctable error interrupt is critical.
5	EBCO	EBCO Interrupt Class 0 EBCO interrupt is non-critical. 1 EBCO interrupt is critical.
6	EBMI	EBMI Interrupt Class 0 EBMI interrupt is non-critical. 1 EBMI interrupt is critical.
7	OPB	OPB to PLB Bridge Interrupt Class 0 OPB to PLB bridge interrupt is non-critical. 1 OPB to PLB bridge interrupt is critical.
8	MSI3	PCI MSI Level 3 Interrupt Class 0 PCI MSI level 3 interrupt is non-critical. 1 PCI MSI level 3 interrupt is critical.

9	MSI4	PCI MSI Level 4 Interrupt Class 0 PCI MSI level 4 interrupt is non-critical. 1 PCI MSI level 4 interrupt is critical.
10	MSI5	PCI MSI Level 5 Interrupt Class 0 PCI MSI level 5 interrupt is non-critical. 1 PCI MSI level 5 interrupt is critical.
11	MSI6	PCI MSI Level 6 Interrupt Class 0 PCI MSI level 6 interrupt is non-critical. 1 PCI MSI level 6 interrupt is critical.
12	MSI7	PCI MSI Level 7 Interrupt Class 0 PCI MSI level 7 interrupt is non-critical. 1 PCI MSI level 7 interrupt is critical.
13	MSI8	PCI MSI Level 8 Interrupt Class 0 PCI MSI level 8 interrupt is non-critical. 1 PCI MSI level 8 interrupt is critical.
14	MSI9	PCI MSI Level 9 Interrupt Class 0 PCI MSI level 9 interrupt is non-critical. 1 PCI MSI level 9 interrupt is critical.
15	MSI10	PCI MSI Level 10 Interrupt Class 0 PCI MSI level 10 interrupt is non-critical. 1 PCI MSI level 10 interrupt is critical.
16	MS11	PCI MSI Level 11 Interrupt Class 0 PCI MSI level 11 interrupt is non-critical. 1 PCI MSI level 11 interrupt is critical.
17	PPM	PPM Interrupt Class 0 PPM interrupt is non-critical. 1 PPM interrupt is critical.
18	EIR7	External IRQ 7 Interrupt Class 0 External IRQ 7 interrupt is non-critical. 1 External IRQ 7 interrupt is critical.
19	EIR8	External IRQ 8 Interrupt Class 0 External IRQ 8 interrupt is non-critical. 1 External IRQ 8 interrupt is critical.
20	EIR9	External IRQ 9 Interrupt Class 0 External IRQ 9 interrupt is non-critical. 1 External IRQ 9 interrupt is critical.
21	EIR10	External IRQ 10 Interrupt Class 0 External IRQ 10 interrupt is non-critical. 1 External IRQ 10 interrupt is critical.
22	EIR11	External IRQ 11 Interrupt Class 0 External IRQ 11 interrupt is non-critical. 1 External IRQ 11 interrupt is critical.
23	EIR12	External IRQ 12 Interrupt Class 0 External IRQ 12 interrupt is non-critical. 1 External IRQ 12 interrupt is critical.

24	SRE	Serial ROM Error Interrupt Class 0 Serial ROM error interrupt is non-critical. 1 Serial ROM error interrupt is critical.
25:26		Reserved
27	PAE	PCI Asynchronous Error Interrupt Class 0 PCI asynchronous error interrupt is non-critical. 1 PCI asynchronous error interrupt is critical.
28	ETH0	Ethernet 0 Interrupt Class 0 Ethernet 0 interrupt is non-critical. 1 Ethernet 0 interrupt is critical.
29	EWU0	Ethernet 0 Wake-up Interrupt Class 0 Ethernet 0 wake-upinterrupt is non-critical. 1 Ethernet 0 wake-up interrupt is critical.
30	ETH1	Ethernet 1 Interrupt Class 0 Ethernet 1 interrupt is non-critical. 1 Ethernet 1 interrupt is critical.
31	EWU1	Ethernet 1 Wake-up Interrupt Class 0 Ethernet 1 interrupt is non-critical. 1 Ethernet 1 interrupt is critical.

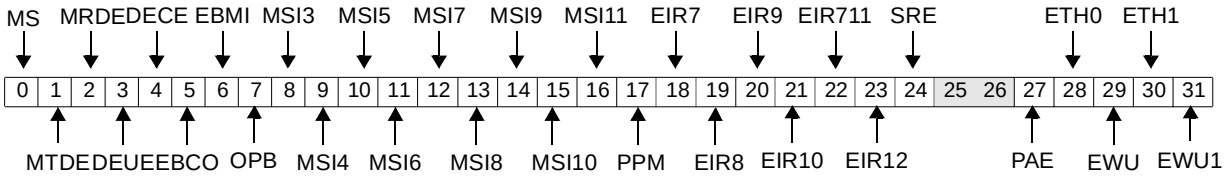


Figure 29-330. UIC1 Critical Register (UIC1_CR)

0	MS	MAL SERR Interrupt Class 0 MAL SERR interrupt is non-critical. 1 MAL SERR interrupt is critical.
1	MTDE	MAL TXDE Interrupt Class 0 MAL TXDE interrupt is non-critical. 1 MAL TXDE interrupt is critical.
2	MRDE	MAL RXDE Interrupt Class 0 MAL RXDE interrupt is non-critical. 1 MAL RXDE interrupt is critical.
3	DEUE	DDRSDRAM ECC Uncorrectable Error Interrupt Class 0 DDRSDRAM ECC uncorrectable error interrupt is non-critical. 1 DDRSDRAM ECC uncorrectable error interrupt is critical.

4	DECE	DDRSDRAM ECC Correctable Error Interrupt Class 0 DDRSDRAM ECC correctable error interrupt is non-critical. 1 DDRSDRAM ECC correctable error interrupt is critical.
5	EBCO	EBCO Interrupt Class 0 EBCO interrupt is non-critical. 1 EBCO interrupt is critical.
6	EBMI	EBMI Interrupt Class 0 EBMI interrupt is non-critical. 1 EBMI interrupt is critical.
7	OPB	OPB to PLB Bridge Interrupt Class 0 OPB to PLB bridge interrupt is non-critical. 1 OPB to PLB bridge interrupt is critical.
8	MSI3	PCI MSI Level 3 Interrupt Class 0 PCI MSI level 3 interrupt is non-critical. 1 PCI MSI level 3 interrupt is critical.
9	MSI4	PCI MSI Level 4 Interrupt Class 0 PCI MSI level 4 interrupt is non-critical. 1 PCI MSI level 4 interrupt is critical.
10	MSI5	PCI MSI Level 5 Interrupt Class 0 PCI MSI level 5 interrupt is non-critical. 1 PCI MSI level 5 interrupt is critical.
11	MSI6	PCI MSI Level 6 Interrupt Class 0 PCI MSI level 6 interrupt is non-critical. 1 PCI MSI level 6 interrupt is critical.
12	MSI7	PCI MSI Level 7 Interrupt Class 0 PCI MSI level 7 interrupt is non-critical. 1 PCI MSI level 7 interrupt is critical.
13	MSI8	PCI MSI Level 8 Interrupt Class 0 PCI MSI level 8 interrupt is non-critical. 1 PCI MSI level 8 interrupt is critical.
14	MSI9	PCI MSI Level 9 Interrupt Class 0 PCI MSI level 9 interrupt is non-critical. 1 PCI MSI level 9 interrupt is critical.
15	MSI10	PCI MSI Level 10 Interrupt Class 0 PCI MSI level 10 interrupt is non-critical. 1 PCI MSI level 10 interrupt is critical.
16	MS11	PCI MSI Level 11 Interrupt Class 0 PCI MSI level 11 interrupt is non-critical. 1 PCI MSI level 11 interrupt is critical.
17	PPM	PPM Interrupt Class 0 PPM interrupt is non-critical. 1 PPM interrupt is critical.
18	EIR7	External IRQ 7 Interrupt Class 0 External IRQ 7 interrupt is non-critical. 1 External IRQ 7 interrupt is critical.
19	EIR8	External IRQ 8 Interrupt Class 0 External IRQ 8 interrupt is non-critical. 1 External IRQ 8 interrupt is critical.
20	EIR9	External IRQ 9 Interrupt Class 0 External IRQ 9 interrupt is non-critical. 1 External IRQ 9 interrupt is critical.

21	EIR10	External IRQ 10 Interrupt Class 0 External IRQ 10 interrupt is non-critical. 1 External IRQ 10 interrupt is critical.
22	EIR11	External IRQ 11 Interrupt Class 0 External IRQ 11 interrupt is non-critical. 1 External IRQ 11 interrupt is critical.
23	EIR12	External IRQ 12 Interrupt Class 0 External IRQ 12 interrupt is non-critical. 1 External IRQ 12 interrupt is critical.
24	SRE	Serial ROM Error Interrupt Class 0 Serial ROM error interrupt is non-critical. 1 Serial ROM error interrupt is critical.
25:26		Reserved
27	PAE	PCI Asynchronous Error Interrupt Class 0 PCI asynchronous error interrupt is non-critical. 1 PCI asynchronous error interrupt is critical.
28	ETH0	Ethernet 0 Interrupt Class 0 Ethernet 0 interrupt is non-critical. 1 Ethernet 0 interrupt is critical.
29	EWU0	Ethernet 0 Wake-up Interrupt Class 0 Ethernet 0 wake-upinterrupt is non-critical. 1 Ethernet 0 wake-up interrupt is critical.
30	ETH1	Ethernet 1 Interrupt Class 0 Ethernet 1 interrupt is non-critical. 1 Ethernet 1 interrupt is critical.
31	EWU1	Ethernet 1 Wake-up Interrupt Class 0 Ethernet 1 interrupt is non-critical. 1 Ethernet 1 interrupt is critical.

DCR 0x0D2 Read/Write

See *UIC1 Enable Register (UIC1_ER)* on page 310.

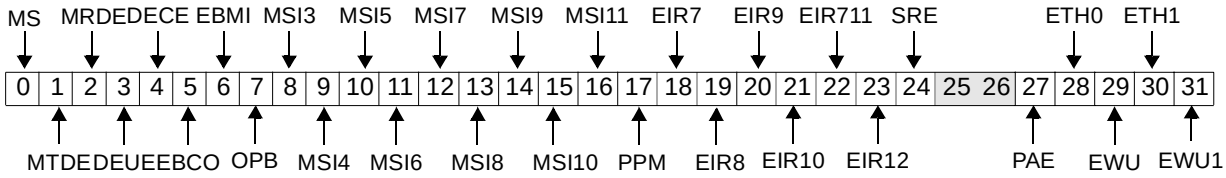


Figure 0-10. UIC1 Enable Register (UIC1_ER)

0	MS	MAL SERR Interrupt Enable 0 MAL SERR interrupt is disabled. 1 MAL SERR interrupt is enabled.
1	MTDE	MAL TXDE Interrupt Enable 0 MAL TXDE interrupt is disabled. 1 MAL TXDE interrupt is enabled.
2	MRDE	MAL RXDE Interrupt Enable 0 MAL RXDE interrupt is disabled. 1 MAL RXDE interrupt is enabled.
3	DEUE	DDRSDRAM ECC Uncorrectable Error Interrupt Enable 0 DDRSDRAM ECC uncorrectable error interrupt is disabled. 1 DDRSDRAM ECC uncorrectable error interrupt is enabled.
4	DECE	DDRSDRAM ECC Correctable Error Interrupt Enable 0 DDRSDRAM ECC correctable error interrupt is disabled. 1 DDRSDRAM ECC correctable error interrupt is enabled.
5	EBCO	EBCO Interrupt Enable 0 EBCO interrupt is disabled. 1 EBCO interrupt is enabled.
6	EBMI	EBMI Interrupt Enable 0 EBMI interrupt is disabled. 1 EBMI interrupt is enabled.
7	OPB	OPB to PLB Bridge Interrupt Enable 0 OPB to PLB bridge interrupt is disabled. 1 OPB to PLB bridge interrupt is enabled.
8	MSI3	PCI MSI Level 3 Interrupt Enable 0 PCI MSI level 3 interrupt is disabled. 1 PCI MSI level 3 interrupt is enabled.

9	MSI4	PCI MSI Level 4 Interrupt Enable 0 PCI MSI level 4 interrupt is disabled. 1 PCI MSI level 4 interrupt is enabled.
10	MSI5	PCI MSI Level 5 Interrupt Enable 0 PCI MSI level 5 interrupt is disabled. 1 PCI MSI level 5 interrupt is enabled.
11	MSI6	PCI MSI Level 6 Interrupt Enable 0 PCI MSI level 6 interrupt is disabled. 1 PCI MSI level 6 interrupt is enabled.
12	MSI7	PCI MSI Level 7 Interrupt Enable 0 PCI MSI level 7 interrupt is disabled. 1 PCI MSI level 7 interrupt is enabled.
13	MSI8	PCI MSI Level 8 Interrupt Enable 0 PCI MSI level 8 interrupt is disabled. 1 PCI MSI level 8 interrupt is enabled.
14	MSI9	PCI MSI Level 9 Interrupt Enable 0 PCI MSI level 9 interrupt is disabled. 1 PCI MSI level 9 interrupt is enabled.
15	MSI10	PCI MSI Level 10 Interrupt Enable 0 PCI MSI level 10 interrupt is disabled. 1 PCI MSI level 10 interrupt is enabled.
16	MSI11	PCI MSI Level 11 Interrupt Enable 0 PCI MSI level 11 interrupt is disabled. 1 PCI MSI level 11 interrupt is enabled.
17	PPM	PPM Interrupt Enable 0 PPM interrupt is disabled. 1 PPM interrupt is enabled.
18	EIR7	External IRQ 7 Interrupt Enable 0 External IRQ 7 interrupt is disabled. 1 External IRQ 7 interrupt is enabled.
19	EIR8	External IRQ 8 Interrupt Enable 0 External IRQ 8 interrupt is disabled. 1 External IRQ 8 interrupt is enabled.
20	EIR9	External IRQ 9 Interrupt Enable 0 External IRQ 9 interrupt is disabled. 1 External IRQ 9 interrupt is enabled.
21	EIR10	External IRQ 10 Interrupt Enable 0 External IRQ 10 interrupt is disabled. 1 External IRQ 10 interrupt is enabled.
22	EIR11	External IRQ 11 Interrupt Enable 0 External IRQ 11 interrupt is disabled. 1 External IRQ 11 interrupt is enabled.
23	EIR12	External IRQ 12 Interrupt Enable 0 External IRQ 12 interrupt is disabled. 1 External IRQ 12 interrupt is enabled.

24	SRE	Serial ROM Error Interrupt Enable 0 Serial ROM error interrupt is disabled. 1 Serial ROM error interrupt is enabled.
25:26		Reserved
27	PAE	PCI Asynchronous Error Interrupt Enable 0 PCI asynchronous error interrupt is disabled. 1 PCI asynchronous error interrupt is enabled.
28	ETH0	Ethernet 0 Interrupt Enable 0 Ethernet 0 interrupt is disabled. 1 Ethernet 0 interrupt is enabled.
29	EWU0	Ethernet 0 Wake-up Interrupt Enable 0 Ethernet 0 wake-up interrupt is disabled. 1 Ethernet 0 wake-up interrupt is enabled.
30	ETH1	Ethernet 1 Interrupt Enable 0 Ethernet 1 interrupt is disabled. 1 Ethernet 1 interrupt is enabled.
31	EWU1	Ethernet 1 Wake-up Interrupt Enable 0 Ethernet 1 wake-up interrupt is disabled. 1 Ethernet 1 wake-up interrupt is enabled.

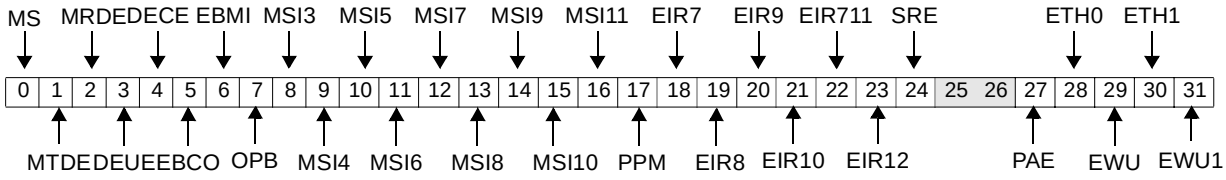


Figure 29-331. UIC1 Enable Register (UIC1_ER)

0	MS	MAL SERR Interrupt Enable 0 MAL SERR interrupt is disabled. 1 MAL SERR interrupt is enabled.
1	MTDE	MAL TXDE Interrupt Enable 0 MAL TXDE interrupt is disabled. 1 MAL TXDE interrupt is enabled.
2	MRDE	MAL RXDE Interrupt Enable 0 MAL RXDE interrupt is disabled. 1 MAL RXDE interrupt is enabled.
3	DEUE	DDRSDRAM ECC Uncorrectable Error Interrupt Enable 0 DDRSDRAM ECC uncorrectable error interrupt is disabled. 1 DDRSDRAM ECC uncorrectable error interrupt is enabled.

4	DECE	DDRSDRAM ECC Correctable Error Interrupt Enable 0 DDRSDRAM ECC correctable error interrupt is disabled. 1 DDRSDRAM ECC correctable error interrupt is enabled.
5	EBCO	EBCO Interrupt Enable 0 EBCO interrupt is disabled. 1 EBCO interrupt is enabled.
6	EBMI	EBMI Interrupt Enable 0 EBMI interrupt is disabled. 1 EBMI interrupt is enabled.
7	OPB	OPB to PLB Bridge Interrupt Enable 0 OPB to PLB bridge interrupt is disabled. 1 OPB to PLB bridge interrupt is enabled.
8	MSI3	PCI MSI Level 3 Interrupt Enable 0 PCI MSI level 3 interrupt is disabled. 1 PCI MSI level 3 interrupt is enabled.
9	MSI4	PCI MSI Level 4 Interrupt Enable 0 PCI MSI level 4 interrupt is disabled. 1 PCI MSI level 4 interrupt is enabled.
10	MSI5	PCI MSI Level 5 Interrupt Enable 0 PCI MSI level 5 interrupt is disabled. 1 PCI MSI level 5 interrupt is enabled.
11	MSI6	PCI MSI Level 6 Interrupt Enable 0 PCI MSI level 6 interrupt is disabled. 1 PCI MSI level 6 interrupt is enabled.
12	MSI7	PCI MSI Level 7 Interrupt Enable 0 PCI MSI level 7 interrupt is disabled. 1 PCI MSI level 7 interrupt is enabled.
13	MSI8	PCI MSI Level 8 Interrupt Enable 0 PCI MSI level 8 interrupt is disabled. 1 PCI MSI level 8 interrupt is enabled.
14	MSI9	PCI MSI Level 9 Interrupt Enable 0 PCI MSI level 9 interrupt is disabled. 1 PCI MSI level 9 interrupt is enabled.
15	MSI10	PCI MSI Level 10 Interrupt Enable 0 PCI MSI level 10 interrupt is disabled. 1 PCI MSI level 10 interrupt is enabled.
16	MSI11	PCI MSI Level 11 Interrupt Enable 0 PCI MSI level 11 interrupt is disabled. 1 PCI MSI level 11 interrupt is enabled.
17	PPM	PPM Interrupt Enable 0 PPM interrupt is disabled. 1 PPM interrupt is enabled.
18	EIR7	External IRQ 7 Interrupt Enable 0 External IRQ 7 interrupt is disabled. 1 External IRQ 7 interrupt is enabled.
19	EIR8	External IRQ 8 Interrupt Enable 0 External IRQ 8 interrupt is disabled. 1 External IRQ 8 interrupt is enabled.
20	EIR9	External IRQ 9 Interrupt Enable 0 External IRQ 9 interrupt is disabled. 1 External IRQ 9 interrupt is enabled.

21	EIR10	External IRQ 10 Interrupt Enable 0 External IRQ 10 interrupt is disabled. 1 External IRQ 10 interrupt is enabled.
22	EIR11	External IRQ 11 Interrupt Enable 0 External IRQ 11 interrupt is disabled. 1 External IRQ 11 interrupt is enabled.
23	EIR12	External IRQ 12 Interrupt Enable 0 External IRQ 12 interrupt is disabled. 1 External IRQ 12 interrupt is enabled.
24	SRE	Serial ROM Error Interrupt Enable 0 Serial ROM error interrupt is disabled. 1 Serial ROM error interrupt is enabled.
25:26		Reserved
27	PAE	PCI Asynchronous Error Interrupt Enable 0 PCI asynchronous error interrupt is disabled. 1 PCI asynchronous error interrupt is enabled.
28	ETH0	Ethernet 0 Interrupt Enable 0 Ethernet 0 interrupt is disabled. 1 Ethernet 0 interrupt is enabled.
29	EWU0	Ethernet 0 Wake-up Interrupt Enable 0 Ethernet 0 wake-up interrupt is disabled. 1 Ethernet 0 wake-up interrupt is enabled.
30	ETH1	Ethernet 1 Interrupt Enable 0 Ethernet 1 interrupt is disabled. 1 Ethernet 1 interrupt is enabled.
31	EWU1	Ethernet 1 Wake-up Interrupt Enable 0 Ethernet 1 wake-up interrupt is disabled. 1 Ethernet 1 wake-up interrupt is enabled.

DCR 0x0D6 Read-Only

See *UIC0 Masked Status Register (UIC0_MSR)* on page 327.

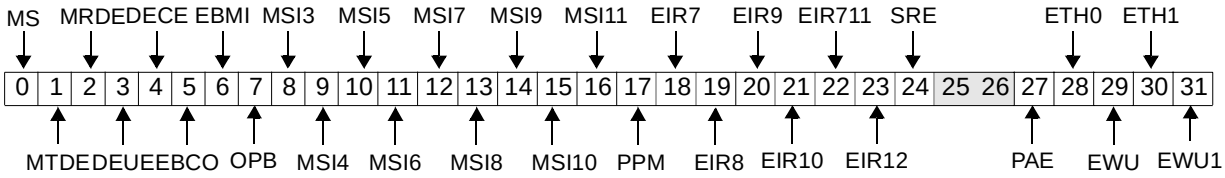


Figure 0-11. UIC1 Masked Status Register (UIC1_MSR)

0	MS	MAL SERR Masked Interrupt Status 0 MAL SERR masked interrupt has not occurred. 1 MAL SERR masked interrupt occurred.
1	MTDE	MAL TXDE Masked Interrupt Status 0 MAL TXDE masked interrupt has not occurred. 1 MAL TXDE masked interrupt occurred.
2	MRDE	MAL RXDE Masked Interrupt Status 0 MAL RXDE masked interrupt has not occurred. 1 MAL RXDE masked interrupt occurred.
3	DEUE	DDRSDRAM ECC Uncorrectable Error Masked Interrupt Status 0 DDRSDRAM ECC uncorrectable error masked interrupt has not occurred. 1 DDRSDRAM ECC uncorrectable error masked interrupt occurred.
4	DECE	DDRSDRAM ECC Correctable Error Masked Interrupt Status 0 DDRSDRAM ECC correctable error masked interrupt has not occurred. 1 DDRSDRAM ECC correctable error masked interrupt occurred.
5	EBCO	EBCO Masked Interrupt Status 0 EBCO interrupt has not occurred. 1 EBCO interrupt occurred.
6	EBMI	EBMI Masked Interrupt Status 0 EBMI masked interrupt has not occurred. 1 EBMI masked interrupt occurred.

7	OPB	OPB to PLB Bridge Masked Interrupt Status 0 OPB to PLB bridge masked interrupt has not occurred. 1 OPB to PLB bridge masked interrupt occurred.
8	MSI3	PCI MSI Level 3 Masked Interrupt Status 0 PCI MSI level 3 masked interrupt has not occurred. 1 PCI MSI level 3 masked interrupt occurred.
9	MSI4	PCI MSI Level 4 Masked Interrupt Status 0 PCI MSI level 4 masked interrupt has not occurred. 1 PCI MSI level 4 masked interrupt occurred.
10	MSI5	PCI MSI Level 5 Masked Interrupt Status 0 PCI MSI level 5 masked interrupt has not occurred. 1 PCI MSI level 5 masked interupt occurred.
11	MSI6	PCI MSI Level 6 Masked Interrupt Status 0 PCI MSI level 6 masked interrupt has not occurred. 1 PCI MSI level 6 masked interrupt occurred.
12	MSI7	PCI MSI Level 7 Masked Interrupt Status 0 PCI MSI level 7 masked interrupt has not occurred. 1 PCI MSI level 7 masked interrupt input occurred.
13	MSI8	PCI MSI Level 8 Masked Interrupt Status 0 PCI MSI level 8 masked interrupt has not occurred. 1 PCI MSI level 8 masked interrupt occurred.
14	MSI9	PCI MSI Level 9 Masked Interrupt Status 0 PCI MSI level 9 masked interrupt has not occurred. 1 PCI MSI level 9 masked interrupt occurred.
15	MSI10	PCI MSI Level 10 Masked Interrupt Status 0 PCI MSI level 10 masked interrupt has not occurred. 1 PCI MSI level 10 masked interrupt occurred.

16	MSI11	PCI MSI Level 11 Masked Interrupt Status 0 PCI MSI level 11 masked interrupt has not occurred. 1 PCI MSI level 11 masked interrupt occurred.
17	PPM	PPM Masked Interrupt Status 0 PPM masked interrupt has not occurred. 1 PPM masked interrupt occurred.
18	EIR7	External IRQ 7 Masked Interrupt Status 0 External IRQ 7 interrupt has not occurred. 1 External IRQ 7 interrupt occurred.
19	EIR8	External IRQ 8 Masked Interrupt Status 0 External IRQ 8 interrupt has not occurred. 1 External IRQ 8 interrupt occurred.
20	EIR9	External IRQ 9 Masked Interrupt Status 0 External IRQ 9 interrupt has not occurred. 1 External IRQ 9 interrupt occurred.
21	EIR10	External IRQ 10 Masked Interrupt Status 0 External IRQ 10 interrupt has not occurred. 1 External IRQ 10 interrupt occurred.
22	EIR11	External IRQ 11 Masked Interrupt Status 0 External IRQ 11 interrupt has not occurred. 1 External IRQ 11 interrupt occurred.
23	EIR12	External IRQ 12 Masked Interrupt Status 0 External IRQ 12 interrupt has not occurred. 1 External IRQ 12 interrupt occurred.
24	SRE	Serial ROM Error Interrupt Status 0 Serial ROM error interrupt has not occurred. 1 Serial ROM error interrupt occurred.
25:26		Reserved
27	PAE	PCI Asynchronous Error Masked Interrupt Status 0 PCI asynchronous error interrupt has not occurred. 1 PCI asynchronous error interrupt occurred.
28	ETH0	Ethernet 0 Masked Interrupt Status 0 Ethernet 0 interrupt has not occurred. 1 Ethernet 0 interrupt occurred.

29	EWU0	Ethernet 0 Wake-up Masked Interrupt Status 0 Ethernet 0 wake-up interrupt has not occurred. 1 Ethernet 0 wake-up interrupt occurred.
30	ETH1	Ethernet 1 Masked Interrupt Status 0 Ethernet 1 interrupt has not occurred. 1 Ethernet 1 interrupt occurred.
31	EWU1	Ethernet 1 Wake-up Masked Interrupt Status 0 Ethernet 1 wake-up interrupt has not occurred. 1 Ethernet 1wake-up interrupt occurred.

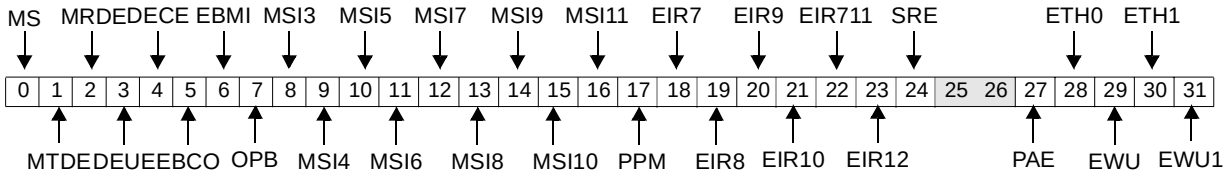


Figure 29-332. UIC1 Masked Status Register (UIC1_MSR)

0	MS	MAL SERR Masked Interrupt Status 0 MAL SERR masked interrupt has not occurred. 1 MAL SERR masked interrupt occurred.
1	MTDE	MAL TXDE Masked Interrupt Status 0 MAL TXDE masked interrupt has not occurred. 1 MAL TXDE masked interrupt occurred.
2	MRDE	MAL RXDE Masked Interrupt Status 0 MAL RXDE masked interrupt has not occurred. 1 MAL RXDE masked interrupt occurred.
3	DEUE	DDRSDRAM ECC Uncorrectable Error Masked Interrupt Status 0 DDRSDRAM ECC uncorrectable error masked interrupt has not occurred. 1 DDRSDRAM ECC uncorrectable error masked interrupt occurred.
4	DECE	DDRSDRAM ECC Correctable Error Masked Interrupt Status 0 DDRSDRAM ECC correctable error masked interrupt has not occurred. 1 DDRSDRAM ECC correctable error masked interrupt occurred.
5	EBCO	EBCO Masked Interrupt Status 0 EBCO interrupt has not occurred. 1 EBCO interrupt occurred.
6	EBMI	EBMI Masked Interrupt Status 0 EBMI masked interrupt has not occurred. 1 EBMI masked interrupt occurred.

7	OPB	OPB to PLB Bridge Masked Interrupt Status 0 OPB to PLB bridge masked interrupt has not occurred. 1 OPB to PLB bridge masked interrupt occurred.
8	MSI3	PCI MSI Level 3 Masked Interrupt Status 0 PCI MSI level 3 masked interrupt has not occurred. 1 PCI MSI level 3 masked interrupt occurred.
9	MSI4	PCI MSI Level 4 Masked Interrupt Status 0 PCI MSI level 4 masked interrupt has not occurred. 1 PCI MSI level 4 masked interrupt occurred.
10	MSI5	PCI MSI Level 5 Masked Interrupt Status 0 PCI MSI level 5 masked interrupt has not occurred. 1 PCI MSI level 5 masked interupt occurred.
11	MSI6	PCI MSI Level 6 Masked Interrupt Status 0 PCI MSI level 6 masked interrupt has not occurred. 1 PCI MSI level 6 masked interrupt occurred.
12	MSI7	PCI MSI Level 7 Masked Interrupt Status 0 PCI MSI level 7 masked interrupt has not occurred. 1 PCI MSI level 7 masked interrupt input occurred.
13	MSI8	PCI MSI Level 8 Masked Interrupt Status 0 PCI MSI level 8 masked interrupt has not occurred. 1 PCI MSI level 8 masked interrupt occurred.
14	MSI9	PCI MSI Level 9 Masked Interrupt Status 0 PCI MSI level 9 masked interrupt has not occurred. 1 PCI MSI level 9 masked interrupt occurred.
15	MSI10	PCI MSI Level 10 Masked Interrupt Status 0 PCI MSI level 10 masked interrupt has not occurred. 1 PCI MSI level 10 masked interrupt occurred.
16	MSI11	PCI MSI Level 11 Masked Interrupt Status 0 PCI MSI level 11 masked interrupt has not occurred. 1 PCI MSI level 11 masked interrupt occurred.
17	PPM	PPM Masked Interrupt Status 0 PPM masked interrupt has not occurred. 1 PPM masked interrupt occurred.
18	EIR7	External IRQ 7 Masked Interrupt Status 0 External IRQ 7 interrupt has not occurred. 1 External IRQ 7 interrupt occurred.
19	EIR8	External IRQ 8 Masked Interrupt Status 0 External IRQ 8 interrupt has not occurred. 1 External IRQ 8 interrupt occurred.
20	EIR9	External IRQ 9 Masked Interrupt Status 0 External IRQ 9 interrupt has not occurred. 1 External IRQ 9 interrupt occurred.

21	EIR10	External IRQ 10 Masked Interrupt Status 0 External IRQ 10 interrupt has not occurred. 1 External IRQ 10 interrupt occurred.
22	EIR11	External IRQ 11 Masked Interrupt Status 0 External IRQ 11 interrupt has not occurred. 1 External IRQ 11 interrupt occurred.
23	EIR12	External IRQ 12 Masked Interrupt Status 0 External IRQ 12 interrupt has not occurred. 1 External IRQ 12 interrupt occurred.
24	SRE	Serial ROM Error Interrupt Status 0 Serial ROM error interrupt has not occurred. 1 Serial ROM error interrupt occurred.
25:26		Reserved
27	PAE	PCI Asynchronous Error Masked Interrupt Status 0 PCI asynchronous error interrupt has not occurred. 1 PCI asynchronous error interrupt occurred.
28	ETH0	Ethernet 0 Masked Interrupt Status 0 Ethernet 0 interrupt has not occurred. 1 Ethernet 0 interrupt occurred.
29	EWU0	Ethernet 0 Wake-up Masked Interrupt Status 0 Ethernet 0 wake-up interrupt has not occurred. 1 Ethernet 0 wake-up interrupt occurred.
30	ETH1	Ethernet 1 Masked Interrupt Status 0 Ethernet 1 interrupt has not occurred. 1 Ethernet 1 interrupt occurred.
31	EWU1	Ethernet 1 Wake-up Masked Interrupt Status 0 Ethernet 1 wake-up interrupt has not occurred. 1 Ethernet 1wake-up interrupt occurred.

DCR 0x0D4 Read/Write

See *UIC1 Polarity Register (UIC1_PR)* on page 320.

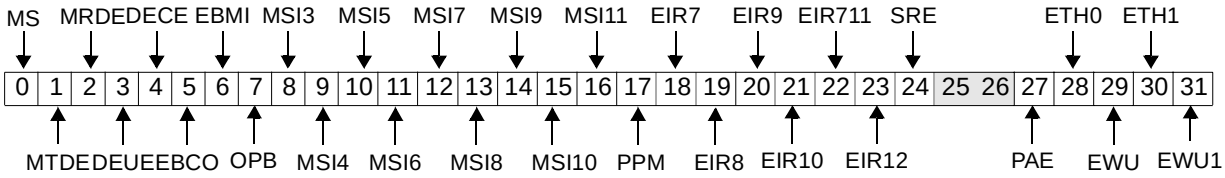


Figure 0-12. UIC1 Polarity Register (UIC1_PR)

0	MS	MAL SERR Interrupt Polarity 0 MAL SERR interrupt has negative polarity. 1 MAL SERR interrupt has positive polarity.
1	MTDE	MAL TXDE Interrupt Polarity 0 MAL TXDE interrupt has negative polarity. 1 MAL TXDE interrupt has positive polarity.
2	MRDE	MAL RXDE Interrupt Polarity 0 MAL RXDE interrupt has negative polarity. 1 MAL RXDE interrupt has positive polarity.
3	DEUE	DDRSDRAM ECC Uncorrectable Error Interrupt Polarity 0 DDRSDRAM ECC uncorrectable error interrupt has negative polarity. 1 DDRSDRAM ECC uncorrectable error interrupt has positive polarity.
4	DECE	DDRSDRAM ECC Correctable Error Interrupt Polarity 0 DDRSDRAM ECC correctable error interrupt has negative polarity. 1 DDRSDRAM ECC correctable error interrupt has positive polarity.
5	EBCO	EBCO Interrupt Polarity 0 EBCO interrupt has negative polarity. 1 EBCO interrupt has positive polarity.
6	EBMI	EBMI Interrupt Polarity 0 EBMI interrupt has negative polarity. 1 EBMI interrupt has positive polarity.

7	OPB	OPB to PLB Bridge Interrupt Polarity 0 OPB to PLB bridge interrupt has negative polarity. 1 OPB to PLB bridge interrupt has positive polarity.
8	MSI3	PCI MSI Level 3 Interrupt Polarity 0 PCI MSI level 3 interrupt has negative polarity. 1 PCI MSI level 3 interrupt has positive polarity.
9	MSI4	PCI MSI Level 4 Interrupt Polarity 0 PCI MSI level 4 interrupt has negative polarity. 1 PCI MSI level 4 interrupt has positive polarity.
10	MSI5	PCI MSI Level 5 Interrupt Polarity 0 PCI MSI level 5 interrupt has negative polarity. 1 PCI MSI level 5 interrupt has positive polarity.
11	MSI6	PCI MSI Level 6 Interrupt Polarity 0 PCI MSI level 6 interrupt has negative polarity. 1 PCI MSI level 6 interrupt has positive polarity.
12	MSI7	PCI MSI Level 7 Interrupt Polarity 0 PCI MSI level 7 interrupt has negative polarity. 1 PCI MSI level 7 interrupt has positive polarity.
13	MSI8	PCI MSI Level 8 Interrupt Polarity 0 PCI MSI level 8 interrupt has negative polarity. 1 PCI MSI level 8 interrupt has positive polarity.
14	MSI9	PCI MSI Level 9 Interrupt Polarity 0 PCI MSI level 9 interrupt has negative polarity. 1 PCI MSI level 9 interrupt has positive polarity.
15	MSI10	PCI MSI Level 10 Interrupt Polarity 0 PCI MSI level 10 interrupt has negative polarity. 1 PCI MSI level 10 interrupt has positive polarity.

16	MSI11	PCI MSI Level 11 Interrupt Polarity 0 PCI MSI level 11 interrupt has negative polarity. 1 PCI MSI level 11 interrupt has positive polarity.
17	PPM	PPM Interrupt Polarity 0 PPM interrupt has negative polarity. 1 PPM interrupt has positive polarity.
18	EIR7	External IRQ 7 Interrupt Polarity 0 External IRQ 7 interrupt has negative polarity. 1 External IRQ 7 interrupt has positive polarity.
19	EIR8	External IRQ 8 Interrupt Polarity 0 External IRQ 8 interrupt has negative polarity. 1 External IRQ 8 interrupt has positive polarity.
20	EIR9	External IRQ 9 Interrupt Polarity 0 External IRQ 9 interrupt has negative polarity. 1 External IRQ 9 interrupt has positive polarity.
21	EIR10	External IRQ 10 Interrupt Polarity 0 External IRQ 10 interrupt has negative polarity. 1 External IRQ 10 interrupt has positive polarity.
22	EIR11	External IRQ 11 Interrupt Polarity 0 External IRQ 11 interrupt has negative polarity. 1 External IRQ 11 interrupt has positive polarity.
23	EIR12	External IRQ 12 Interrupt Polarity 0 External IRQ 12 interrupt has negative polarity. 1 External IRQ 12 interrupt has positive polarity.
24	SRE	Serial ROM Error Interrupt Polarity 0 Serial ROM error interrupt has negative polarity. 1 Serial ROM error interrupt has positive polarity.
25:26		Reserved

27	PAE	PCI Asynchronous Error Interrupt Polarity 0 PCI asynchronous error interrupt has negative polarity. 1 PCI asynchronous error interrupt has positive polarity.
28	ETH0	Ethernet 0 Interrupt Polarity 0 Ethernet 0 interrupt has negative polarity. 1 Ethernet 0 interrupt has positive polarity.
29	EWU0	Ethernet 0 Wake-up Interrupt Polarity 0 Ethernet 0 wake-up interrupt has negative polarity. 1 Ethernet 0 wake-up interrupt has positive polarity.
30	ETH1	Ethernet 1 Interrupt Polarity 0 Ethernet 1 interrupt has negative polarity. 1 Ethernet 1 interrupt has positive polarity.
31	EWU1	Ethernet 1 Wake-up Interrupt Polarity 0 Ethernet 1 interrupt has negative polarity. 1 Ethernet 1 interrupt has positive polarity.

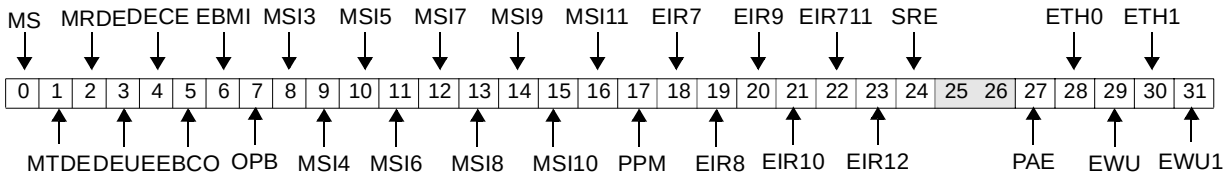


Figure 29-333. UIC1 Polarity Register (UIC1_PR)

0	MS	MAL SERR Interrupt Polarity 0 MAL SERR interrupt has negative polarity. 1 MAL SERR interrupt has positive polarity.
1	MTDE	MAL TXDE Interrupt Polarity 0 MAL TXDE interrupt has negative polarity. 1 MAL TXDE interrupt has positive polarity.
2	MRDE	MAL RXDE Interrupt Polarity 0 MAL RXDE interrupt has negative polarity. 1 MAL RXDE interrupt has positive polarity.
3	DEUE	DDRSDRAM ECC Uncorrectable Error Interrupt Polarity 0 DDRSDRAM ECC uncorrectable error interrupt has negative polarity. 1 DDRSDRAM ECC uncorrectable error interrupt has positive polarity.

4	DECE	DDRSDRAM ECC Correctable Error Interrupt Polarity 0 DDRSDRAM ECC correctable error interrupt has negative polarity. 1 DDRSDRAM ECC correctable error interrupt has positive polarity.
5	EBCO	EBCO Interrupt Polarity 0 EBCO interrupt has negative polarity. 1 EBCO interrupt has positive polarity.
6	EBMI	EBMI Interrupt Polarity 0 EBMI interrupt has negative polarity. 1 EBMI interrupt has positive polarity.
7	OPB	OPB to PLB Bridge Interrupt Polarity 0 OPB to PLB bridge interrupt has negative polarity. 1 OPB to PLB bridge interrupt has positive polarity.
8	MSI3	PCI MSI Level 3 Interrupt Polarity 0 PCI MSI level 3 interrupt has negative polarity. 1 PCI MSI level 3 interrupt has positive polarity.
9	MSI4	PCI MSI Level 4 Interrupt Polarity 0 PCI MSI level 4 interrupt has negative polarity. 1 PCI MSI level 4 interrupt has positive polarity.
10	MSI5	PCI MSI Level 5 Interrupt Polarity 0 PCI MSI level 5 interrupt has negative polarity. 1 PCI MSI level 5 interrupt has positive polarity.
11	MSI6	PCI MSI Level 6 Interrupt Polarity 0 PCI MSI level 6 interrupt has negative polarity. 1 PCI MSI level 6 interrupt has positive polarity.
12	MSI7	PCI MSI Level 7 Interrupt Polarity 0 PCI MSI level 7 interrupt has negative polarity. 1 PCI MSI level 7 interrupt has positive polarity.
13	MSI8	PCI MSI Level 8 Interrupt Polarity 0 PCI MSI level 8 interrupt has negative polarity. 1 PCI MSI level 8 interrupt has positive polarity.
14	MSI9	PCI MSI Level 9 Interrupt Polarity 0 PCI MSI level 9 interrupt has negative polarity. 1 PCI MSI level 9 interrupt has positive polarity.
15	MSI10	PCI MSI Level 10 Interrupt Polarity 0 PCI MSI level 10 interrupt has negative polarity. 1 PCI MSI level 10 interrupt has positive polarity.
16	MSI11	PCI MSI Level 11 Interrupt Polarity 0 PCI MSI level 11 interrupt has negative polarity. 1 PCI MSI level 11 interrupt has positive polarity.
17	PPM	PPM Interrupt Polarity 0 PPM interrupt has negative polarity. 1 PPM interrupt has positive polarity.
18	EIR7	External IRQ 7 Interrupt Polarity 0 External IRQ 7 interrupt has negative polarity. 1 External IRQ 7 interrupt has positive polarity.
19	EIR8	External IRQ 8 Interrupt Polarity 0 External IRQ 8 interrupt has negative polarity. 1 External IRQ 8 interrupt has positive polarity.

20	EIR9	External IRQ 9 Interrupt Polarity 0 External IRQ 9 interrupt has negative polarity. 1 External IRQ 9 interrupt has positive polarity.
21	EIR10	External IRQ 10 Interrupt Polarity 0 External IRQ 10 interrupt has negative polarity. 1 External IRQ 10 interrupt has positive polarity.
22	EIR11	External IRQ 11 Interrupt Polarity 0 External IRQ 11 interrupt has negative polarity. 1 External IRQ 11 interrupt has positive polarity.
23	EIR12	External IRQ 12 Interrupt Polarity 0 External IRQ 12 interrupt has negative polarity. 1 External IRQ 12 interrupt has positive polarity.
24	SRE	Serial ROM Error Interrupt Polarity 0 Serial ROM error interrupt has negative polarity. 1 Serial ROM error interrupt has positive polarity.
25:26		Reserved
27	PAE	PCI Asynchronous Error Interrupt Polarity 0 PCI asynchronous error interrupt has negative polarity. 1 PCI asynchronous error interrupt has positive polarity.
28	ETH0	Ethernet 0 Interrupt Polarity 0 Ethernet 0 interrupt has negative polarity. 1 Ethernet 0 interrupt has positive polarity.
29	EWU0	Ethernet 0 Wake-up Interrupt Polarity 0 Ethernet 0 wake-up interrupt has negative polarity. 1 Ethernet 0 wake-up interrupt has positive polarity.
30	ETH1	Ethernet 1 Interrupt Polarity 0 Ethernet 1 interrupt has negative polarity. 1 Ethernet 1 interrupt has positive polarity.
31	EWU1	Ethernet 1 Wake-up Interrupt Polarity 0 Ethernet 1 interrupt has negative polarity. 1 Ethernet 1 interrupt has positive polarity.

DCR 0x0D0 Read/Clear

See *UIC1 Status Register (UIC1_SR)* on page 306.

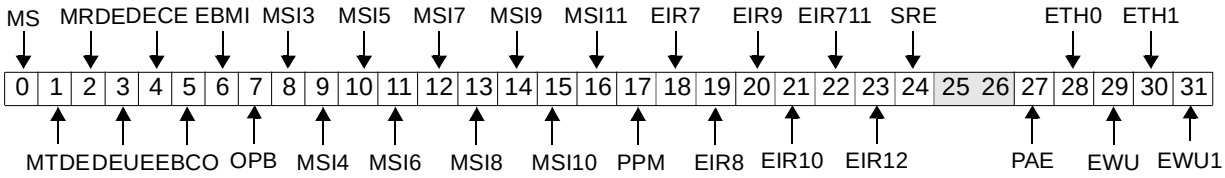


Figure 0-13. UIC1 Status Register (UIC1_SR)

0	MS	MAL SERR Interrupt Status 0 MAL SERR interrupt has not occurred. 1 MAL SERR interrupt occurred.
1	MTDE	MAL TXDE Interrupt Status 0 MAL TXDE interrupt has not occurred. 1 MAL TXDE interrupt occurred.
2	MRDE	MAL RXDE Interrupt Status 0 MAL RXDE interrupt has not occurred. 1 MAL RXDE interrupt occurred.
3	DEUE	DDRSDRAM ECC Uncorrectable Error Interrupt Status 0 DDRSDRAM ECC uncorrectable error interrupt has not occurred. 1 DDRSDRAM ECC uncorrectable error interrupt occurred.
4	DECE	DDRSDRAM ECC Correctable Error Interrupt Status 0 DDRSDRAM ECC correctable error interrupt has not occurred. 1 DDRSDRAM ECC correctable error interrupt occurred.
5	EBCO	EBCO Interrupt Status 0 EBCO interrupt has not occurred. 1 EBCO interrupt occurred.
6	EBMI	EBMI Interrupt Status 0 EBMI interrupt has not occurred. 1 EBMI interrupt occurred.
7	OPB	OPB to PLB Bridge Interrupt Status 0 OPB to PLB bridge interrupt has not occurred. 1 OPB to PLB bridge interrupt occurred.
8	MSI3	PCI MSI Level 3 Interrupt Status 0 PCI MSI level 3 interrupt has not occurred. 1 PCI MSI level 3 interrupt occurred.

9	MSI4	PCI MSI Level 4 Interrupt Status 0 PCI MSI level 4 interrupt has not occurred. 1 PCI MSI level 4 interrupt occurred.
10	MSI5	PCI MSI Level 5 Interrupt Status 0 PCI MSI level 5 interrupt has not occurred. 1 PCI MSI level 5 interrupt occurred.
11	MSI6	PCI MSI Level 6 Interrupt Status 0 PCI MSI level 6 interrupt has not occurred. 1 PCI MSI level 6 interrupt occurred.
12	MSI7	PCI MSI Level 7 Interrupt Status 0 PCI MSI level 7 interrupt has not occurred. 1 PCI MSI level 7 interrupt occurred.
13	MSI8	PCI MSI Level 8 Interrupt Status 0 PCI MSI level 8 interrupt has not occurred. 1 PCI MSI level 8 interrupt occurred.
14	MSI9	PCI MSI Level 9 Interrupt Status 0 PCI MSI level 9 interrupt has not occurred. 1 PCI MSI level 9 interrupt occurred.
15	MSI10	PCI MSI Level 10 Interrupt Status 0 PCI MSI level 10 interrupt has not occurred. 1 PCI MSI level 10 interrupt occurred.
16	MSI11	PCI MSI Level 11 Interrupt Status 0 PCI MSI level 11 interrupt has not occurred. 1 PCI MSI level 11 interrupt occurred.
17	PPM	PPM Interrupt Status 0 PPM interrupt has not occurred. 1 PPM interrupt occurred.
18	EIR7	External IRQ 7 Interrupt Status 0 External IRQ 7 interrupt has not occurred. 1 External IRQ 7 interrupt occurred.
19	EIR8	External IRQ 8 Interrupt Status 0 External IRQ 8 interrupt has not occurred. 1 External IRQ 8 interrupt occurred.
20	EIR9	External IRQ 9 Interrupt Status 0 External IRQ 9 interrupt has not occurred. 1 External IRQ 9 interrupt occurred.
21	EIR10	External IRQ 10 Interrupt Status 0 External IRQ 10 interrupt has not occurred. 1 External IRQ 10 interrupt occurred.
22	EIR11	External IRQ 11 Interrupt Status 0 External IRQ 11 interrupt has not occurred. 1 External IRQ 11 interrupt occurred.
23	EIR12	External IRQ 12 Interrupt Status 0 External IRQ 12 interrupt has not occurred. 1 External IRQ 12 interrupt occurred.

24	SRE	Serial ROM Error Interrupt Status 0 Serial ROM error interrupt has not occurred. 1 Serial ROM error interrupt occurred.
25:26		Reserved
27	PAE	PCI Asynchronous Error Interrupt Status 0 PCI ashynchronous error interrupt has not occurred. 1 PCI asynchronous error interrupt occurred.
28	ETH0	Ethernet 0 Interrupt Status 0 Ethernet 0 interrupt has not occurred. 1 Ethernet 0 interrupt occurred.
29	EWU0	Ethernet 0 Wake-up Interrupt Status 0 Ethernet 0 wake-up interrupt has not occurred. 1 Ethernet 0 wake-up interrupt occurred.
30	ETH1	Ethernet 1 Interrupt Status 0 Ethernet 1 interrupt has not occurred. 1 Ethernet 1 interrupt occurred.
31	EWU1	Ethernet 1 Wake-up Interrupt Status 0 Ethernet 1 wake-up interrupt has not occurred. 1 Ethernet 1 wake-up nterrupt occurred.

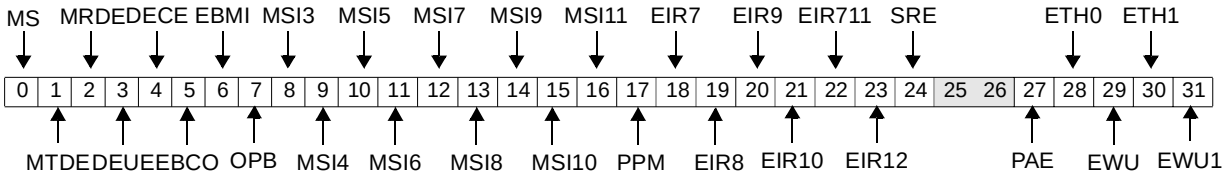


Figure 29-334. UIC1 Status Register (UIC1_SR)

0	MS	MAL SERR Interrupt Status 0 MAL SERR interrupt has not occurred. 1 MAL SERR interrupt occurred.
1	MTDE	MAL TXDE Interrupt Status 0 MAL TXDE interrupt has not occurred. 1 MAL TXDE interrupt occurred.
2	MRDE	MAL RXDE Interrupt Status 0 MAL RXDE interrupt has not occurred. 1 MAL RXDE interrupt occurred.
3	DEUE	DDRSDRAM ECC Uncorrectable Error Interrupt Status 0 DDRSDRAM ECC uncorrectable error interrupt has not occurred. 1 DDRSDRAM ECC uncorrectable error interrupt occurred.

4	DECE	DDRSDRAM ECC Correctable Error Interrupt Status 0 DDRSDRAM ECC correctable error interrupt has not occurred. 1 DDRSDRAM ECC correctable error interrupt occurred.
5	EBCO	EBCO Interrupt Status 0 EBCO interrupt has not occurred. 1 EBCO interrupt occurred.
6	EBMI	EBMI Interrupt Status 0 EBMI interrupt has not occurred. 1 EBMI interrupt occurred.
7	OPB	OPB to PLB Bridge Interrupt Status 0 OPB to PLB bridge interrupt has not occurred. 1 OPB to PLB bridge interrupt occurred.
8	MSI3	PCI MSI Level 3 Interrupt Status 0 PCI MSI level 3 interrupt has not occurred. 1 PCI MSI level 3 interrupt occurred.
9	MSI4	PCI MSI Level 4 Interrupt Status 0 PCI MSI level 4 interrupt has not occurred. 1 PCI MSI level 4 interrupt occurred.
10	MSI5	PCI MSI Level 5 Interrupt Status 0 PCI MSI level 5 interrupt has not occurred. 1 PCI MSI level 5 interrupt occurred.
11	MSI6	PCI MSI Level 6 Interrupt Status 0 PCI MSI level 6 interrupt has not occurred. 1 PCI MSI level 6 interrupt occurred.
12	MSI7	PCI MSI Level 7 Interrupt Status 0 PCI MSI level 7 interrupt has not occurred. 1 PCI MSI level 7 interrupt occurred.
13	MSI8	PCI MSI Level 8 Interrupt Status 0 PCI MSI level 8 interrupt has not occurred. 1 PCI MSI level 8 interrupt occurred.
14	MSI9	PCI MSI Level 9 Interrupt Status 0 PCI MSI level 9 interrupt has not occurred. 1 PCI MSI level 9 interrupt occurred.
15	MSI10	PCI MSI Level 10 Interrupt Status 0 PCI MSI level 10 interrupt has not occurred. 1 PCI MSI level 10 interrupt occurred.
16	MSI11	PCI MSI Level 11 Interrupt Status 0 PCI MSI level 11 interrupt has not occurred. 1 PCI MSI level 11 interrupt occurred.
17	PPM	PPM Interrupt Status 0 PPM interrupt has not occurred. 1 PPM interrupt occurred.
18	EIR7	External IRQ 7 Interrupt Status 0 External IRQ 7 interrupt has not occurred. 1 External IRQ 7 interrupt occurred.
19	EIR8	External IRQ 8 Interrupt Status 0 External IRQ 8 interrupt has not occurred. 1 External IRQ 8 interrupt occurred.
20	EIR9	External IRQ 9 Interrupt Status 0 External IRQ 9 interrupt has not occurred. 1 External IRQ 9 interrupt occurred.

21	EIR10	External IRQ 10 Interrupt Status 0 External IRQ 10 interrupt has not occurred. 1 External IRQ 10 interrupt occurred.
22	EIR11	External IRQ 11 Interrupt Status 0 External IRQ 11 interrupt has not occurred. 1 External IRQ 11 interrupt occurred.
23	EIR12	External IRQ 12 Interrupt Status 0 External IRQ 12 interrupt has not occurred. 1 External IRQ 12 interrupt occurred.
24	SRE	Serial ROM Error Interrupt Status 0 Serial ROM error interrupt has not occurred. 1 Serial ROM error interrupt occurred.
25:26		Reserved
27	PAE	PCI Asynchronous Error Interrupt Status 0 PCI ashynchronous error interrupt has not occurred. 1 PCI asynchronous error interrupt occurred.
28	ETH0	Ethernet 0 Interrupt Status 0 Ethernet 0 interrupt has not occurred. 1 Ethernet 0 interrupt occurred.
29	EWU0	Ethernet 0 Wake-up Interrupt Status 0 Ethernet 0 wake-up interrupt has not occurred. 1 Ethernet 0 wake-up interrupt occurred.
30	ETH1	Ethernet 1 Interrupt Status 0 Ethernet 1 interrupt has not occurred. 1 Ethernet 1 interrupt occurred.
31	EWU1	Ethernet 1 Wake-up Interrupt Status 0 Ethernet 1 wake-up interrupt has not occurred. 1 Ethernet 1 wake-up nterrupt occurred.

DCR 0x0D5 Read/Write

See *UIC1 Trigger Register (UIC1_TR)* on page 324.

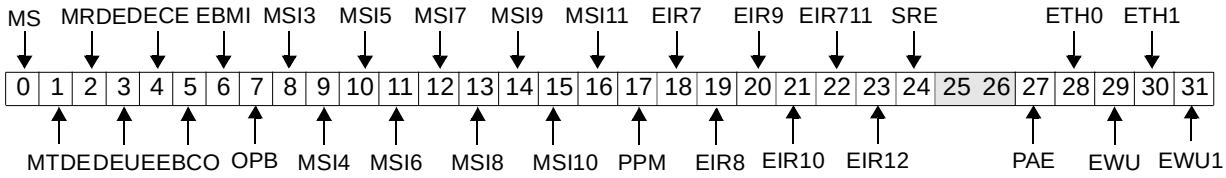


Figure 0-14. UIC1 Trigger Register (UIC1_TR)

0	MS	MAL SERR Interrupt Trigger 0 MAL SERR interrupt is level-sensitive. 1 MAL SERR interrupt is edge-sensitive.
1	MTDE	MAL TXDE Interrupt Trigger 0 MAL TXDE interrupt is level-sensitive. 1 MAL TXDE interrupt is edge-sensitive.
2	MRDE	MAL RXDE Interrupt Trigger 0 MAL RXDE interrupt is level-sensitive. 1 MAL RXDE interrupt is edge-sensitive.
3	DEUE	DDRSDRAM ECC Uncorrectable Error Interrupt Trigger 0 DDRSDRAM ECC uncorrectable error interrupt is level-sensitive. 1 DDRSDRAM ECC uncorrectable error interrupt is edge-sensitive.
4	DECE	DDRSDRAM ECC Correctable Error Interrupt Trigger 0 DDRSDRAM ECC correctable error interrupt is level-sensitive. 1 DDRSDRAM ECC correctable error interrupt is edge-sensitive.
5	EBCO	EBCO Interrupt Trigger 0 EBCO interrupt is level-sensitive. 1 EBCO interrupt is edge-sensitive.
6	EBMI	EBMI Interrupt Trigger 0 EBMI interrupt is level-sensitive. 1 EBMI interrupt is edge-sensitive.
7	OPB	OPB to PLB Bridge Interrupt Trigger 0 OPB to PLB bridge interrupt is level-sensitive. 1 OPB to PLB bridge interrupt is edge-sensitive.
8	MSI3	PCI MSI Level 3 Interrupt Trigger 0 PCI MSI level 3 interrupt is level-sensitive. 1 PCI MSI level 3 interrupt is edge-sensitive.

9	MSI4	PCI MSI Level 4 Interrupt Trigger 0 PCI MSI level 4 interrupt is level-sensitive. 1 PCI MSI level 4 interrupt is edge-sensitive.
10	MSI5	PCI MSI Level 5 Interrupt Trigger 0 PCI MSI level 5 interrupt is level-sensitive. 1 PCI MSI level 5 interrupt is edge-sensitive.
11	MSI6	PCI MSI Level 6 Interrupt Trigger 0 PCI MSI level 6 interrupt is level-sensitive. 1 PCI MSI level 6 interrupt is edge-sensitive.
12	MSI7	PCI MSI Level 7 Interrupt Trigger 0 PCI MSI level 7 interrupt is level-sensitive. 1 PCI MSI level 7 interrupt is edge-sensitive.
13	MSI8	PCI MSI Level 8 Interrupt Trigger 0 PCI MSI level 8 interrupt is level-sensitive. 1 PCI MSI level 8 interrupt is edge-sensitive.
14	MSI9	PCI MSI Level 9 Interrupt Trigger 0 PCI MSI level 9 interrupt is level-sensitive. 1 PCI MSI level 9 interrupt is edge-sensitive.
15	MSI10	PCI MSI Level 10 Interrupt Trigger 0 PCI MSI level 10 interrupt is level-sensitive. 1 PCI MSI level 10 interrupt is edge-sensitive.
16	MSI11	PCI MSI Level 11 Interrupt Trigger 0 PCI MSI level 11 interrupt is level-sensitive. 1 PCI MSI level 11 interrupt is edge-sensitive.
17	PPM	PPM Interrupt Trigger 0 PPM interrupt is level-sensitive. 1 PPM interrupt is edge-sensitive.
18	EIR7	External IRQ 7 Interrupt Trigger 0 External IRQ 7 interrupt is level-sensitive. 1 External IRQ 7 interrupt is edge-sensitive.
19	EIR8	External IRQ 8 Interrupt Trigger 0 External IRQ 8 interrupt is level-sensitive. 1 External IRQ 8 interrupt is edge-sensitive.
20	EIR9	External IRQ 9 Interrupt Trigger 0 External IRQ 9 interrupt is level-sensitive. 1 External IRQ 9 interrupt is edge-sensitive.
21	EIR10	External IRQ 10 Interrupt Trigger 0 External IRQ 10 interrupt is level-sensitive. 1 External IRQ 10 interrupt is edge-sensitive.
22	EIR11	External IRQ 11 Interrupt Trigger 0 External IRQ 11 interrupt is level-sensitive. 1 External IRQ 11 interrupt is edge-sensitive.
23	EIR12	External IRQ 12 Interrupt Trigger 0 External IRQ 12 interrupt is level-sensitive. 1 External IRQ 12 interrupt is edge-sensitive.

24	SRE	Serial ROM Error Interrupt Trigger 0 Serial ROM error interrupt is level-sensitive. 1 Serial ROM error interrupt is edge-sensitive.
25:26		Reserved
27	PAE	PCI Asynchronous Error Interrupt Trigger 0 PCI asynchronous error interrupt is level-sensitive. 1 PCI asynchronous error interrupt is edge-sensitive.
28	ETH0	Ethernet 0 Interrupt Trigger 0 Ethernet 0 interrupt is level-sensitive. 1 Ethernet 0 interrupt is edge-sensitive.
29	EWU0	Ethernet 0 Wake-up Interrupt Trigger 0 Ethernet 0 wake-up interrupt is level-sensitive. 1 Ethernet 0 wake-up interrupt is edge-sensitive.
30	ETH1	Ethernet 1 Interrupt Trigger 0 Ethernet 1 interrupt is level-sensitive. 1 Ethernet 1 interrupt is edge-sensitive.
31	EWU1	Ethernet 1 Wake-up Interrupt Trigger 0 Ethernet 1 interrupt is level-sensitive. 1 Ethernet 1 interrupt is edge-sensitive.

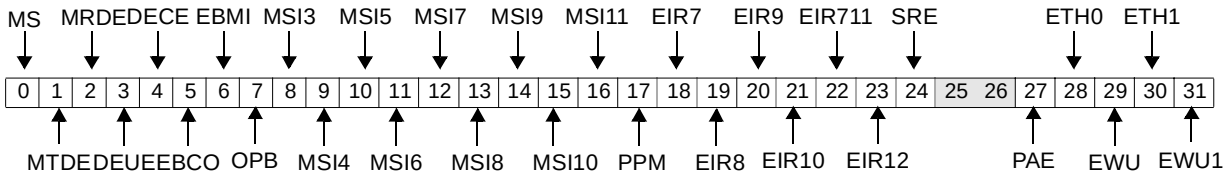


Figure 29-335. UIC1 Trigger Register (UIC1_TR)

0	MS	MAL SERR Interrupt Trigger 0 MAL SERR interrupt is level-sensitive. 1 MAL SERR interrupt is edge-sensitive.
1	MTDE	MAL TXDE Interrupt Trigger 0 MAL TXDE interrupt is level-sensitive. 1 MAL TXDE interrupt is edge-sensitive.
2	MRDE	MAL RXDE Interrupt Trigger 0 MAL RXDE interrupt is level-sensitive. 1 MAL RXDE interrupt is edge-sensitive.
3	DEUE	DDRSDRAM ECC Uncorrectable Error Interrupt Trigger 0 DDRSDRAM ECC uncorrectable error interrupt is level-sensitive. 1 DDRSDRAM ECC uncorrectable error interrupt is edge-sensitive.

4	DECE	DDRSDRAM ECC Correctable Error Interrupt Trigger 0 DDRSDRAM ECC correctable error interrupt is level-sensitive. 1 DDRSDRAM ECC correctable error interrupt is edge-sensitive.
5	EBCO	EBCO Interrupt Trigger 0 EBCO interrupt is level-sensitive. 1 EBCO interrupt is edge-sensitive.
6	EBMI	EBMI Interrupt Trigger 0 EBMI interrupt is level-sensitive. 1 EBMI interrupt is edge-sensitive.
7	OPB	OPB to PLB Bridge Interrupt Trigger 0 OPB to PLB bridge interrupt is level-sensitive. 1 OPB to PLB bridge interrupt is edge-sensitive.
8	MSI3	PCI MSI Level 3 Interrupt Trigger 0 PCI MSI level 3 interrupt is level-sensitive. 1 PCI MSI level 3 interrupt is edge-sensitive.
9	MSI4	PCI MSI Level 4 Interrupt Trigger 0 PCI MSI level 4 interrupt is level-sensitive. 1 PCI MSI level 4 interrupt is edge-sensitive.
10	MSI5	PCI MSI Level 5 Interrupt Trigger 0 PCI MSI level 5 interrupt is level-sensitive. 1 PCI MSI level 5 interrupt is edge-sensitive.
11	MSI6	PCI MSI Level 6 Interrupt Trigger 0 PCI MSI level 6 interrupt is level-sensitive. 1 PCI MSI level 6 interrupt is edge-sensitive.
12	MSI7	PCI MSI Level 7 Interrupt Trigger 0 PCI MSI level 7 interrupt is level-sensitive. 1 PCI MSI level 7 interrupt is edge-sensitive.
13	MSI8	PCI MSI Level 8 Interrupt Trigger 0 PCI MSI level 8 interrupt is level-sensitive. 1 PCI MSI level 8 interrupt is edge-sensitive.
14	MSI9	PCI MSI Level 9 Interrupt Trigger 0 PCI MSI level 9 interrupt is level-sensitive. 1 PCI MSI level 9 interrupt is edge-sensitive.
15	MSI10	PCI MSI Level 10 Interrupt Trigger 0 PCI MSI level 10 interrupt is level-sensitive. 1 PCI MSI level 10 interrupt is edge-sensitive.
16	MSI11	PCI MSI Level 11 Interrupt Trigger 0 PCI MSI level 11 interrupt is level-sensitive. 1 PCI MSI level 11 interrupt is edge-sensitive.
17	PPM	PPM Interrupt Trigger 0 PPM interrupt is level-sensitive. 1 PPM interrupt is edge-sensitive.
18	EIR7	External IRQ 7 Interrupt Trigger 0 External IRQ 7 interrupt is level-sensitive. 1 External IRQ 7 interrupt is edge-sensitive.
19	EIR8	External IRQ 8 Interrupt Trigger 0 External IRQ 8 interrupt is level-sensitive. 1 External IRQ 8 interrupt is edge-sensitive.
20	EIR9	External IRQ 9 Interrupt Trigger 0 External IRQ 9 interrupt is level-sensitive. 1 External IRQ 9 interrupt is edge-sensitive.

21	EIR10	External IRQ 10 Interrupt Trigger 0 External IRQ 10 interrupt is level-sensitive. 1 External IRQ 10 interrupt is edge-sensitive.
22	EIR11	External IRQ 11 Interrupt Trigger 0 External IRQ 11 interrupt is level-sensitive. 1 External IRQ 11 interrupt is edge-sensitive.
23	EIR12	External IRQ 12 Interrupt Trigger 0 External IRQ 12 interrupt is level-sensitive. 1 External IRQ 12 interrupt is edge-sensitive.
24	SRE	Serial ROM Error Interrupt Trigger 0 Serial ROM error interrupt is level-sensitive. 1 Serial ROM error interrupt is edge-sensitive.
25:26		Reserved
27	PAE	PCI Asynchronous Error Interrupt Trigger 0 PCI asynchronous error interrupt is level-sensitive. 1 PCI asynchronous error interrupt is edge-sensitive.
28	ETH0	Ethernet 0 Interrupt Trigger 0 Ethernet 0 interrupt is level-sensitive. 1 Ethernet 0 interrupt is edge-sensitive.
29	EWU0	Ethernet 0 Wake-up Interrupt Trigger 0 Ethernet 0 wake-up interrupt is level-sensitive. 1 Ethernet 0 wake-up interrupt is edge-sensitive.
30	ETH1	Ethernet 1 Interrupt Trigger 0 Ethernet 1 interrupt is level-sensitive. 1 Ethernet 1 interrupt is edge-sensitive.
31	EWU1	Ethernet 1 Wake-up Interrupt Trigger 0 Ethernet 1 interrupt is level-sensitive. 1 Ethernet 1 interrupt is edge-sensitive.

DCR 0x0D8 Write-Only

See *UIC0 Vector Configuration Register (UIC0_VCR)* on page 332.

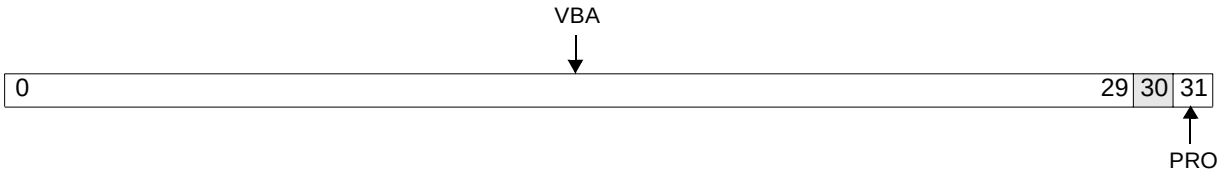


Figure 0-15. UIC1 Vector Configuration Register (UIC1_VCR)

0:29	VBA	Vector Base Address
30		Reserved
31	PRO	Priority Ordering 0 UIC1_SR[31] is the highest priority interrupt. 1 UIC1_SR[0] is the highest priority interrupt. Note: Vector generation is not performed for non-critical interrupts.

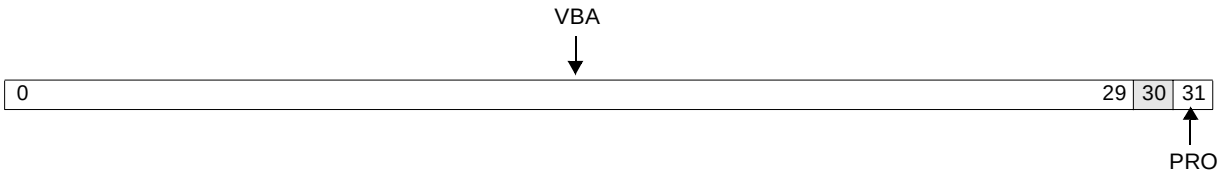


Figure 29-336. UIC1 Vector Configuration Register (UIC1_VCR)

0:29	VBA	Vector Base Address
30		Reserved
31	PRO	Priority Ordering 0 UIC1_SR[31] is the highest priority interrupt. 1 UIC1_SR[0] is the highest priority interrupt. **Note:** Vector generation is not performed for non-critical interrupts.

DCR 0x0D7 Read-Only

See *UIC0 Vector Register (UIC0_VR)* on page 333.

Figure 0-16. UIC1 Vector Register (UIC1_VR)

0:31		Interrupt Vector

Figure 29-337. UIC1 Vector Register (UIC1_VR)

0:31		Interrupt Vector

MMIO 0x014000780 Read/Write

See *Function Enable Register (ZMII0_FER)* on page 723.

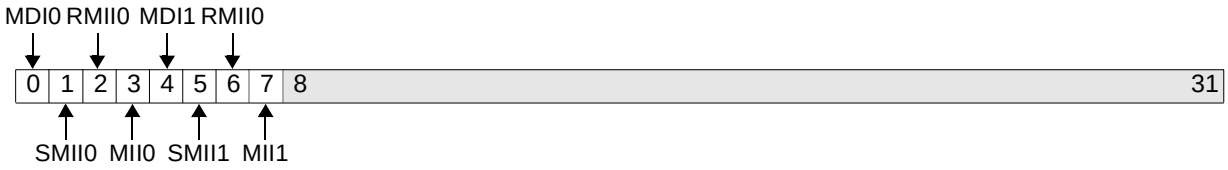


Figure 0-1. Function Enable Register (ZMII0_FER)

0	MDI0	EMAC0 MDI Enable 0 EMAC0 management data and clock are ignored. 1 EMAC0 management data and clock are driven to the PHY.	Must be 0 if MDI1 = 1.
1	SMII0	EMAC0 SMII Enable 0 EMAC0 SMII PHY interface is disabled. 1 EMAC0 SMII PHY interface is enabled.	Must be 0 if RMII0, MII0, RMII1, or MII1 is 1.
2	RMII0	EMAC0 RMII Enable 0 EMAC0 RMII PHY interface is disabled. 1 EMAC0 RMII PHY interface is enabled.	Must be 0 if SMII0, MII0, SMII1, or MII1 is 1.
3	MII0	EMAC0 MII Enable 0 EMAC0 MII PHY interface is disabled. 1 EMAC0 MII PHY interface is enabled.	Must be 0 if SMII0, RMII0, SMII1, RMII1, or MII1 is 1.
4	MDI1	EMAC1 MDI Enable 0 EMAC1 management data and clock are ignored. 1 EMAC1 management data and clock are driven to the PHY.	Must be 0 if MDI0 = 1.
5	SMII1	EMAC1 SMII Enable 0 EMAC1 SMII PHY interface is disabled. 1 EMAC1 SMII PHY interface is enabled.	Must be 0 if RMII0, MII0, RMII1, or MII1 is 1.
6	RMII1	EMAC0 RMII Enable 0 EMAC1 RMII PHY interface is disabled. 1 EMAC1 RMII PHY interface is enabled.	Must be 0 if SMII0, MII0, SMII1, or MII1 is 1.
7	MII1	EMAC0 MII Enable 0 EMAC1 MII PHY interface is disabled. 1 EMAC1 MII PHY interface is enabled.	Must be 0 if SMII0, RMII0, MII0, SMII1, or RMII1 is 1.
8:31		Reserved	

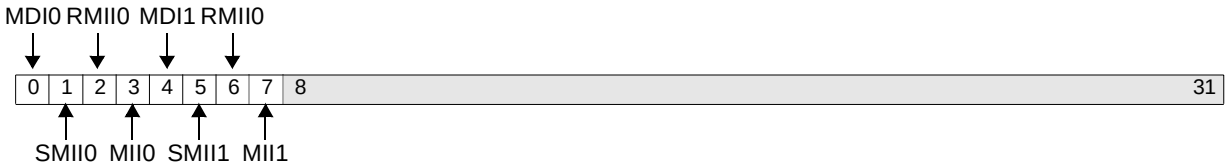


Figure 29-338. Function Enable Register (ZMII0_FER)

0	MDI0	EMAC0 MDI Enable 0 EMAC0 management data and clock are ignored. 1 EMAC0 management data and clock are driven to the PHY.	Must be 0 if MDI1 = 1.
1	SMII0	EMAC0 SMII Enable 0 EMAC0 SMII PHY interface is disabled. 1 EMAC0 SMII PHY interface is enabled.	Must be 0 if RMII0, MII0, RMII1, or MII1 is 1.
2	RMII0	EMAC0 RMII Enable 0 EMAC0 RMII PHY interface is disabled. 1 EMAC0 RMII PHY interface is enabled.	Must be 0 if SMII0, MII0, SMII1, or MII1 is 1.
3	MII0	EMAC0 MII Enable 0 EMAC0 MII PHY interface is disabled. 1 EMAC0 MII PHY interface is enabled.	Must be 0 if SMII0, RMII0, SMII1, RMII1, or MII1 is 1.
4	MDI1	EMAC1 MDI Enable 0 EMAC1 management data and clock are ignored. 1 EMAC1 management data and clock are driven to the PHY.	Must be 0 if MDI0 = 1.
5	SMII1	EMAC1 SMII Enable 0 EMAC1 SMII PHY interface is disabled. 1 EMAC1 SMII PHY interface is enabled.	Must be 0 if RMII0, MII0, RMII1, or MII1 is 1.
6	RMII1	EMAC0 RMII Enable 0 EMAC1 RMII PHY interface is disabled. 1 EMAC1 RMII PHY interface is enabled.	Must be 0 if SMII0, MII0, SMII1, or MII1 is 1.
7	MII1	EMAC0 MII Enable 0 EMAC1 MII PHY interface is disabled. 1 EMAC1 MII PHY interface is enabled.	Must be 0 if SMII0, RMII0, MII0, SMII1, or RMII1 is 1.
8:31		Reserved	

MMIO 0x014000788 Read/Write

See *SMII Status Register (ZMII0_SMIISR)* on page 726.

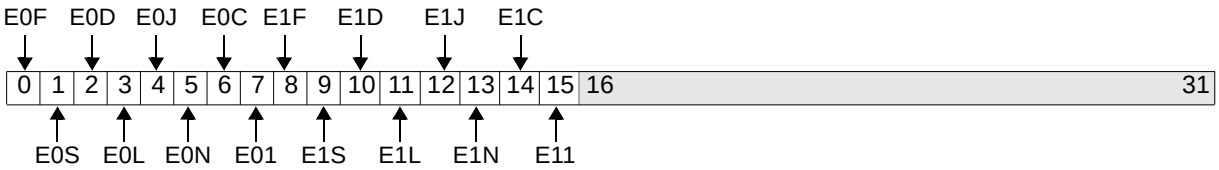


Figure 0-2. SMII Status Register (ZMII0_SMIISR)

0	E0F	EMAC0RxD from Previous Frame	Indicates whether or not the PHY detected an error somewhere in the previous frame. Should be valid in the segment immediately following a frame, and should stay valid until the first data segment of the next frame begin.
1	E0S	EMAC0RxD Speed 0 10MBit 1 100MBit	
2	E0D	EMAC0RxD Duplex 0 Half 1 Full	
3	E0L	EMAC0RxD Link 0 Down 1 Up	
4	E0J	EMAC0RxD Jabber 0 OK 1 Error	
5	E0N	EMAC0RxD Nibble 0 Invalid 1 Valid	Conveys the validity of the upper nibble of the last byte of the previous frame. Should be valid in the segment immediately following a frame, and should stay valid until the first data segment of the next frame begin.
6	E0C	EMAC0RxD False Carrier Detected	When asserted, this bit indicates that the PHY has detected a false carrier event.
7	E01	EMAC0RxD Set to 1	
8	E1F	EMAC1RxD from Previous Frame	Indicates whether or not the PHY detected an error somewhere in the previous frame. Should be valid in the segment immediately following a frame, and should stay valid until the first data segment of the next frame begin.
9	E1S	EMAC1RxD Speed 0 10MBit 1 100MBit	

10	E1D	EMAC1RxD Duplex 0 Half 1 Full	
11	E1L	EMAC1RxD Link 0 Down 1 Up	
12	E1J	EMAC1RxD Jabber 0 OK 1 Error	
13	E1N	EMAC1RxD Nibble 0 Invalid 1 Valid	Conveys the validity of the upper nibble of the last byte of the previous frame. Should be valid in the segment immediately following a frame, and should stay valid until the first data segment of the next frame begin.
14	E1C	EMAC1RxD False Carrier Detected	When asserted, this bit indicates that the PHY has detected a false carrier event.
15	E11	EMAC1RxD Set to 1	
16:31		Reserved.	

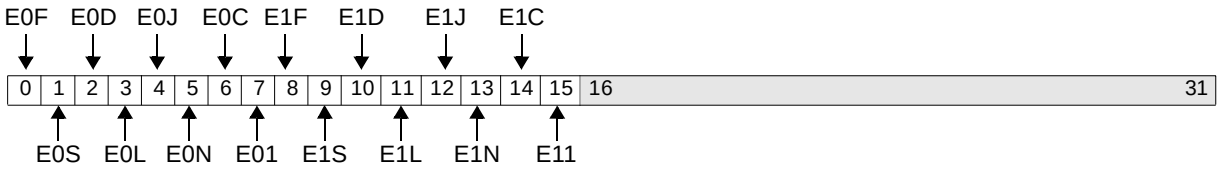


Figure 29-339. SMII Status Register (ZMII0_SMIISR)

0	E0F	EMAC0RxD from Previous Frame	Indicates whether or not the PHY detected an error somewhere in the previous frame. Should be valid in the segment immediately following a frame, and should stay valid until the first data segment of the next frame begin.
1	E0S	EMAC0RxD Speed 0 10MBit 1 100MBit	
2	E0D	EMAC0RxD Duplex 0 Half 1 Full	
3	E0L	EMAC0RxD Link 0 Down 1 Up	
4	E0J	EMAC0RxD Jabber 0 OK 1 Error	

5	E0N	EMAC0RxD Nibble 0 Invalid 1 Valid	Conveys the validity of the upper nibble of the last byte of the previous frame. Should be valid in the segment immediately following a frame, and should stay valid until the first data segment of the next frame begin.
6	E0C	EMAC0RxD False Carrier Detected	When asserted, this bit indicates that the PHY has detected a false carrier event.
7	E01	EMAC0RxD Set to 1	
8	E1F	EMAC1RxD from Previous Frame	Indicates whether or not the PHY detected an error somewhere in the previous frame. Should be valid in the segment immediately following a frame, and should stay valid until the first data segment of the next frame begin.
9	E1S	EMAC1RxD Speed 0 10MBit 1 100MBit	
10	E1D	EMAC1RxD Duplex 0 Half 1 Full	
11	E1L	EMAC1RxD Link 0 Down 1 Up	
12	E1J	EMAC1RxD Jabber 0 OK 1 Error	
13	E1N	EMAC1RxD Nibble 0 Invalid 1 Valid	Conveys the validity of the upper nibble of the last byte of the previous frame. Should be valid in the segment immediately following a frame, and should stay valid until the first data segment of the next frame begin.
14	E1C	EMAC1RxD False Carrier Detected	When asserted, this bit indicates that the PHY has detected a false carrier event.
15	E11	EMAC1RxD Set to 1	
16:31		Reserved.	

MMIO 0x014000784 Read/Write

See *Speed Select Register (ZMII0_SSR)* on page 724.

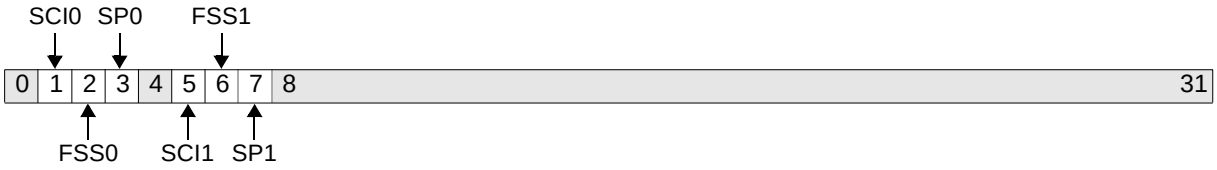


Figure 0-3. Speed Selection Register (ZMII0_SSR)

0		Reserved	
1	SCI0	EMAC0 Suppress Collision Indication 0 EMAC0 collision indication signal is asserted. 1 EMAC0 collision indication signal is deasserted.	
2	FSS0	EMAC0 Force Speed Selection 0 ZMII0_SP0 does not override IPG status field. 1 ZMII0_SP0 overrides IPG status field.	SMII only.
3	SP0	EMAC0 Speed Selection 0 10 Mbps selected. 1 100 Mbps selected.	Must be programmed for RMII; can be programmed for SMII; ignored for MII.
4		Reserved	
5	SCI1	EMAC1 Suppress Collision Indication 0 EMAC1 collision indication signal is asserted. 1 EMAC1 collision indication signal is deasserted.	
6	FSS1	EMAC1 Force Speed Selection 0 ZMII0_SP1 does not override IPG status field. 1 ZMII0_SP1 overrides IPG status field.	SMII only.
7	SP1	EMAC1 Speed Selection 0 10 Mbps selected. 1 100 Mbps selected.	Must be programmed for RMII; can be programmed for SMII; ignored for MII.
8:31		Reserved	

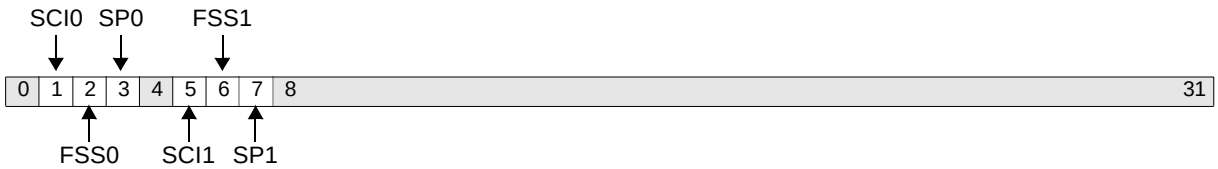


Figure 29-340. Speed Selection Register (ZMII0_SSR)

0		Reserved	
1	SCI0	EMAC0 Suppress Collision Indication 0 EMAC0 collision indication signal is asserted. 1 EMAC0 collision indication signal is deasserted.	
2	FSS0	EMAC0 Force Speed Selection 0 ZMII0_SP0 does not override IPG status field. 1 ZMII0_SP0 overrides IPG status field.	SMII only.
3	SP0	EMAC0 Speed Selection 0 10 Mbps selected. 1 100 Mbps selected.	Must be programmed for RMII; can be programmed for SMII; ignored for MII.
4		Reserved	
5	SCI1	EMAC1 Suppress Collision Indication 0 EMAC1 collision indication signal is asserted. 1 EMAC1 collision indication signal is deasserted.	
6	FSS1	EMAC1 Force Speed Selection 0 ZMII0_SP1 does not override IPG status field. 1 ZMII0_SP1 overrides IPG status field.	SMII only.
7	SP1	EMAC1 Speed Selection 0 10 Mbps selected. 1 100 Mbps selected.	Must be programmed for RMII; can be programmed for SMII; ignored for MII.
8:31		Reserved	

30. Signal Summary

This chapter provides detailed information on the PPC440GP I/O signals.

30.1 Signals Listed Alphabetically

~~Table 30-1~~Table 30-1 lists the PPC440GP signals in alphabetical order with an indication of the interface group to which it belongs to. Table 30-2 lists the PPC440GP signals by interface category with a brief description.

Multiplexed signals are shown in brackets following the first signal name assigned to each multiplexed ball (for example, IRQ0:6[GPIO17:23]). Active-low signals are shown with an overbar on the signal name (for example, $\overline{\text{ExtAck}}$).

Table 0-1. Signals Listed Alphabetically

Signal Name	Interface Group	Page
AGND	Power	30-20
AMV_{DD}	Power MemClkOut PLL analog voltage	30-20
APV_{DD}	Power PCI PLL analog voltage	30-20
ASV_{DD}	Power SysClk PLL analog voltage	30-20
BA0 BA1	DDR SDRAM	30-16
$\overline{\text{BankSel0}}$ $\overline{\text{BankSel1}}$ $\overline{\text{BankSel2}}$ $\overline{\text{BankSel3}}$	DDR SDRAM	30-16
[$\overline{\text{BE0}}$]PCIXC0 [$\overline{\text{BE1}}$]PCIXC1 [$\overline{\text{BE2}}$]PCIXC2 [$\overline{\text{BE3}}$]PCIXC3 [$\overline{\text{BE4}}$]PCIXC4 [$\overline{\text{BE5}}$]PCIXC5 [$\overline{\text{BE6}}$]PCIXC6 [$\overline{\text{BE7}}$]PCIXC7	PCI-X	30-15
BusReq	External Master Peripheral	30-18
$\overline{\text{CAS}}$	DDR SDRAM	30-16
ClkEn0 ClkEn1 ClkEn2 ClkEn3	DDR SDRAM	30-16

Table 0-1. Signals Listed Alphabetically

Signal Name	Interface Group	Page
DM0 DM1 DM2 DM3 DM4 DM5 DM6 DM7 DM8	DDR SDRAM	30-16
DMAAck0 DMAAck1 DMAAck2 DMAAck3	External Slave Peripheral	30-17
DMAReq0 DMAReq1 DMAReq2 DMAReq3	External Slave Peripheral	30-17
DQS0 DQS1 DQS2 DQS3 DQS4 DQS5 DQS6 DQS7 DQS8	DDR SDRAM	30-16
DrvrInh1 DrvrInh2	System	30-19
ECC0 ECC1 ECC2 ECC3 ECC4 ECC5 ECC6 ECC7	DDR SDRAM	30-16
EMCCD[EMC1RxErr]	Ethernet	30-16
EMCCRS[EMC0CRSDV]	Ethernet	30-16
EMCMDClk	Ethernet	30-16
EMCMDIO	Ethernet	30-16
EMCRxClk	Ethernet	30-16
EMCRxD0[EMC0RxD0][EMC0RxD] EMCRxD1[EMC0RxD1][EMC1RxD] EMCRxD2[EMC1RxD0] EMCRxD3[EMC1RxD1]	Ethernet	30-16

Table 0-1. Signals Listed Alphabetically

Signal Name	Interface Group	Page
EMCRxDV[EMC1CRSDV]	Ethernet	30-16
EMCRxErr[EMC0RxErr]	Ethernet	30-16
EMCTxClk[EMCRefClk]	Ethernet	30-16
EMCTxD0[EMC0TxD0][EMC0TxD] EMCTxD1[EMC0TxD1][EMC1TxD] EMCTxD2[EMC1TxD0] EMCTxD3[EMC1TxD1]	Ethernet	30-16
EMCTxEn[EMC0TxEn][EMCSync]	Ethernet	30-16
EMCTxErr[EMC1TxEn]	Ethernet	30-16
EOT0/TC0 EOT1/TC1 EOT2/TC2 EOT3/TC3	External Slave Peripheral	30-17
$\overline{\text{ExtAck}}$	External Master Peripheral	30-18
$\overline{\text{ExtReq}}$	External Master Peripheral	30-18
$\overline{\text{ExtReset}}$	External Master Peripheral	30-18
GND	Power	30-20
GND	Power	30-20

Table 0-1. Signals Listed Alphabetically

Signal Name	Interface Group	Page
[GPIO0]IRQ0 [GPIO1]IRQ1 [GPIO2]IRQ2 [GPIO3]IRQ3 [GPIO4]IRQ4 [GPIO5]IRQ5 [GPIO6]IRQ6 [GPIO7]IRQ7 [GPIO8]IRQ8 [GPIO9]IRQ9 [GPIO10]IRQ10 GPIO11 [GPIO12]UART1_Rx [GPIO13]UART1_Tx [GPIO14]$\overline{\text{UART1\_DSR/CTS}}$ [GPIO15]$\overline{\text{UART1\_RTS/DTR}}$ [GPIO16]IIC1SClk [GPIO17]IIC1SDA [GPIO18]TrcBS0 [GPIO19]TrcBS1 [GPIO20]TrcBS2 [GPIO21]TrcES0 [GPIO22]TrcES1 [GPIO23]TrcES2 [GPIO24]TrcES3 [GPIO25]TrcES4 [GPIO26]TrcTS0 [GPIO27]TrcTS1 [GPIO28]TrcTS2 [GPIO29]TrcTS3 [GPIO30]TrcTS4 [GPIO31]TrcTS5	System	30-19
$\overline{\text{Halt}}$	System	30-19
HoldAck	External Master Peripheral	30-18
HoldReq	External Master Peripheral	30-18
IIC0SClk	IIC Peripheral	30-19
IIC0SDA	IIC Peripheral	30-19
IIC1SClk[GPIO16]	IIC Peripheral	30-19
IIC1SDA[GPIO17]	IIC Peripheral	30-19

Table 0-1. Signals Listed Alphabetically

Signal Name	Interface Group	Page
IRQ0[GPIO0] IRQ1[GPIO1] IRQ2[GPIO2] IRQ3[GPIO3 IRQ4[GPIO4] IRQ5[GPIO5] IRQ6[GPIO6] IRQ7[GPIO7] IRQ8[GPIO8] IRQ9[GPIO9] IRQ10[GPIO10] [IRQ11]$\overline{\text{PCIReq1}}$ [IRQ12]$\overline{\text{PCIGnt1}}$	Interrupts	30-19
MemAddr0 MemAddr1 MemAddr2 MemAddr3 MemAddr4 MemAddr5 MemAddr6 MemAddr7 MemAddr8 MemAddr9 MemAddr10 MemAddr11 MemAddr12	DDR SDRAM	30-16
$\overline{\text{MemClkOut0}}$ $\overline{\text{MemClkOut0}}$	DDR SDRAM	30-16

Table 0-1. Signals Listed Alphabetically

Signal Name	Interface Group	Page
MemData0 MemData1 MemData2 MemData3 MemData4 MemData5 MemData6 MemData7 MemData8 MemData9 MemData10 MemData11 MemData12 MemData13 MemData14 MemData15 MemData16 MemData17 MemData18 MemData19 MemData20 MemData21 MemData22 MemData23 MemData24 MemData25 MemData26 MemData27 MemData28 MemData29 MemData30 MemData31	DDR SDRAM	30-16

Table 0-1. Signals Listed Alphabetically

Signal Name	Interface Group	Page
MemData32 MemData33 MemData34 MemData35 MemData36 MemData37 MemData38 MemData39 MemData40 MemData41 MemData42 MemData43 MemData44 MemData45 MemData46 MemData47 MemData48 MemData49 MemData50 MemData51 MemData52 MemData53 MemData54 MemData55 MemData56 MemData57 MemData58 MemData59 MemData60 MemData61 MemData62 MemData63	DDR SDRAM	30-16
MemVRef1 MemVRef2	DDR SDRAM	30-16
No ball	A physical ball does not exist at these ball coordinates.	
OV_{DD}	Power	30-20
$\overline{\text{PCIXAck64}}$	PCI-X	30-15

Table 0-1. Signals Listed Alphabetically

Signal Name	Interface Group	Page
PCIXAD0 PCIXAD1 PCIXAD2 PCIXAD3 PCIXAD4 PCIXAD5 PCIXAD6 PCIXAD7 PCIXAD8 PCIXAD9 PCIXAD10 PCIXAD11 PCIXAD12 PCIXAD13 PCIXAD14 PCIXAD15 PCIXAD16 PCIXAD17 PCIXAD18 PCIXAD19 PCIXAD20 PCIXAD21 PCIXAD22 PCIXAD23 PCIXAD24 PCIXAD25 PCIXAD26 PCIXAD27 PCIXAD28 PCIXAD29 PCIXAD30 PCIXAD31	PCI-X	30-15

Table 0-1. Signals Listed Alphabetically

Signal Name	Interface Group	Page
PCIXAD32 PCIXAD33 PCIXAD34 PCIXAD35 PCIXAD36 PCIXAD37 PCIXAD38 PCIXAD39 PCIXAD40 PCIXAD41 PCIXAD42 PCIXAD43 PCIXAD44 PCIXAD45 PCIXAD46 PCIXAD47 PCIXAD48 PCIXAD49 PCIXAD50 PCIXAD51 PCIXAD52 PCIXAD53 PCIXAD54 PCIXAD55 PCIXAD56 PCIXAD57 PCIXAD58 PCIXAD59 PCIXAD60 PCIXAD61 PCIXAD62 PCIXAD63	PCI-X	30-15
PCIXC0[$\overline{\text{BE0}}$] PCIXC1[$\overline{\text{BE1}}$] PCIXC2[$\overline{\text{BE2}}$] PCIXC3[$\overline{\text{BE3}}$] PCIXC4[$\overline{\text{BE4}}$] PCIXC5[$\overline{\text{BE5}}$] PCIXC6[$\overline{\text{BE6}}$] PCIXC7[$\overline{\text{BE7}}$]	PCI-X	30-15
PCIXClk	PCI-X	30-15
$\overline{\text{PCIXDevSel}}$	PCI-X	30-15
$\overline{\text{PCIXFrame}}$	PCI-X	30-15
$\overline{\text{PCIXGnt0}}$ $\overline{\text{PCIXGnt1}}$[IRQ12] $\overline{\text{PCIXGnt2}}$ $\overline{\text{PCIXGnt3}}$ $\overline{\text{PCIXGnt4}}$ $\overline{\text{PCIXGnt5}}$	PCI-X	30-15

Table 0-1. Signals Listed Alphabetically

Signal Name	Interface Group	Page
PCIXIDSel	PCI-X	30-15
$\overline{\text{PCIXINT}}$	PCI-X	30-15
$\overline{\text{PCIXIRDY}}$	PCI-X	30-15
PCIXM66En	PCI-X	30-15
PCIXParHigh	PCI-X	30-15
PCIXParLow	PCI-X	30-15
$\overline{\text{PCIXPErr}}$	PCI-X	30-15
$\overline{\text{PCIXReq0}}$ $\overline{\text{PCIXReq1}}$[IRQ11] $\overline{\text{PCIXReq2}}$ $\overline{\text{PCIXReq3}}$ $\overline{\text{PCIXReq4}}$ $\overline{\text{PCIXReq5}}$	PCI-X	30-15
$\overline{\text{PCIXReq64}}$	PCI-X	30-15
$\overline{\text{PCIXReset}}$	PCI-X	30-15
$\overline{\text{PCIXSErr}}$	PCI-X	30-15
$\overline{\text{PCIXStop}}$	PCI-X	30-15
$\overline{\text{PCIXTRDY}}$	PCI-X	30-15
PCIX133Cap	PCI-X	30-15
PCIXCap	PCI-X	30-15

Table 0-1. Signals Listed Alphabetically

Signal Name	Interface Group	Page
PerAddr0 PerAddr1 PerAddr2 PerAddr3 PerAddr4 PerAddr5 PerAddr6 PerAddr7 PerAddr8 PerAddr9 PerAddr10 PerAddr11 PerAddr12 PerAddr13 PerAddr14 PerAddr15 PerAddr16 PerAddr17 PerAddr18 PerAddr19 PerAddr20 PerAddr21 PerAddr22 PerAddr23 PerAddr24 PerAddr25 PerAddr26 PerAddr27 PerAddr28 PerAddr29 PerAddr30 PerAddr31	External Slave Peripheral **Note:** PerAddr0 is the most significant bit (msb) on this bus.	30-17
$\overline{\text{PerBE0}}$ $\overline{\text{PerBE1}}$ $\overline{\text{PerBE2}}$ $\overline{\text{PerBE3}}$	External Slave Peripheral	30-17
$\overline{\text{PerBLast}}$	External Slave Peripheral	30-17
PerClk	External Master Peripheral	30-18
$\overline{\text{PerCS0}}$ $\overline{\text{PerCS1}}$ $\overline{\text{PerCS2}}$ $\overline{\text{PerCS3}}$ $\overline{\text{PerCS4}}$ $\overline{\text{PerCS5}}$ $\overline{\text{PerCS6}}$ $\overline{\text{PerCS7}}$	External Slave Peripheral	30-17

Table 0-1. Signals Listed Alphabetically

Signal Name	Interface Group	Page
PerData0 PerData1 PerData2 PerData3 PerData4 PerData5 PerData6 PerData7 PerData8 PerData9 PerData10 PerData11 PerData12 PerData13 PerData14 PerData15 PerData16 PerData17 PerData18 PerData19 PerData20 PerData21 PerData22 PerData23 PerData24 PerData25 PerData26 PerData27 PerData28 PerData29 PerData30 PerData31	External Slave Peripheral **Note:** PerData0 is the most significant bit (msb) on this bus.	30-17
PerErr	External Master Peripheral	30-18
$\overline{\text{PerOE}}$	External Slave Peripheral	30-17
PerPar0 PerPar1 PerPar2 PerPar3	External Slave Peripheral	30-17
PerR/$\overline{\text{W}}$	External Slave Peripheral	30-17
PerReady[RcvrInh]	External Slave Peripheral	30-17
$\overline{\text{PerWE}}$	External Slave Peripheral	30-17
$\overline{\text{RAS}}$	DDR SDRAM	30-16
[RcvrInh]PerReady	System	30-19
RefVEn	System	30-19
Reserved	Reserved	30-20

Table 0-1. Signals Listed Alphabetically

Signal Name	Interface Group	Page
SV_{DD}	Power	30-20
SysClk	System	30-19
SysErr	System	30-19
$\overline{\text{SysReset}}$	System	30-19
TCK	JTAG	30-19
TDI	JTAG	30-19
TDO	JTAG	30-19
TestEn	System	30-19
TmrClk	System	30-19
TMS	JTAG	30-19
TrcBS0[GPIO18] TrcBS1[GPIO19] TrcBS2[GPIO20]	Trace	30-20
TrcClk	Trace	30-20
TrcES0[GPIO21] TrcES1[GPIO22] TrcES2[GPIO23] TrcES3[GPIO24] TrcES4[GPIO25]	Trace	30-20
TrcTS0[GPIO26] TrcTS1[GPIO27] TrcTS2[GPIO28] TrcTS3[GPIO29] TrcTS4[GPIO30] TrcTS5[GPIO31] TrcTS6	Trace	30-20
$\overline{\text{TRST}}$	JTAG	30-19
$\overline{\text{UART0\_CTS}}$	UART Peripheral	30-18
$\overline{\text{UART0\_DCD}}$	UART Peripheral **Note:** Used as initialization strapping input.	30-18
$\overline{\text{UART0\_DSR}}$	UART Peripheral **Note:** Used as initialization strapping input.	30-18
$\overline{\text{UART0\_DTR}}$	UART Peripheral	30-18
$\overline{\text{UART0\_RI}}$	UART Peripheral	30-18
$\overline{\text{UART0\_RTS}}$	UART Peripheral	30-18
UART0_Rx	UART Peripheral	30-18
UART0_Tx	UART Peripheral	30-18
$\overline{\text{UART1\_DSR/CTS}}$[GPIO14]	UART Peripheral	30-18

Table 0-1. Signals Listed Alphabetically

Signal Name	Interface Group	Page
$\overline{\text{UART1\_RTS/DTR}}$[GPIO15]	UART Peripheral	30-18
UART1_Rx[GPIO12]	UART Peripheral	30-18
UART1_Tx[GPIO13]	UART Peripheral	30-18
UARTSerClk	UART Peripheral	30-18
V_{DD}	Power	30-20
$\overline{\text{WE}}$	DDR SDRAM	30-16

Table 30-1. Signals Listed Alphabetically

Signal Name	Interface	I/O
AGND	Power Interface	1538
AxV_{DD}	Power Interface	1538
BA0:1	DDR SDRAM Interface	1534
$\overline{\text{BankSel0:3}}$	DDR SDRAM Interface	1534
BusReq	External Master Peripheral Interface	1536
$\overline{\text{CAS}}$	DDR SDRAM Interface	1534
ClkEn0:3	DDR SDRAM Interface	1534
DM0:8	DDR SDRAM Interface	1534
DMAAck0:3	External Slave Peripheral Interface	1535
DMAReq0:3	External Slave Peripheral Interface	1535
DrvrInh1:2	System Interface	1537
DQS0:8	DDR SDRAM Interface	1534
ECC0:7	DDR SDRAM Interface	1534
EMCCD, EMC1RxErr	Ethernet Interface	1534
EMCCrS, EMC0CrSDV	Ethernet Interface	1534
EMCMDClk	Ethernet Interface	1534
EMCMDIO	Ethernet Interface	1534
EMCRxD0:3, EMC0RxD0:1, EMC1RxD0:1, EMC0RxD, EMC1RxD	Ethernet Interface	1534
EMCRxDV, EMC1CrSDV	Ethernet Interface	1534
EMCRxClk	Ethernet Interface	1534
EMCRxErr, EMC0RxErr	Ethernet Interface	1534
EMCTxClk, EMCRefClk	Ethernet Interface	1534

Table 30-1. Signals Listed Alphabetically (continued)

Signal Name	Interface	I/O
EMCTxD0:3, EMC0TxD0:1, EMC1TxD0:1, EMC0TxD, EMC1TxD	Ethernet Interface	1534
EMCTxEn, EMC0TxEn, EMCSync	Ethernet Interface	1534
EMCTxErr, EMC1TxEn	Ethernet Interface	1534
EOT0:3/TC0:3	External Slave Peripheral Interface	1535
$\overline{\text{ExtAck}}$	External Master Peripheral Interface	1536
$\overline{\text{ExtReq}}$	External Master Peripheral Interface	1536
$\overline{\text{ExtReset}}$	System Interface	1537
$\overline{\text{ExtReset}}$	External Master Peripheral Interface	1536
GND	Power Interface	1538
GPIO0:31	System Interface	1537
$\overline{\text{Halt}}$	System Interface	1537
HoldAck	External Master Peripheral Interface	1536
HoldReq	External Master Peripheral Interface	1536
IIC0SClk	IIC Interface	1536
IIC0SDA	IIC Interface	1536
IIC1SClk	IIC Interface	1536
IIC1SDA	IIC Interface	1536
IRQ0:10	Interrupt Interface	1537
IRQ11:12	Interrupt Interface	1537
MemAddr0:12	DDR SDRAM Interface	1534
MemClkOut0 $\overline{\text{MemClkOut0}}$	DDR SDRAM Interface	1534
MemData0:63	DDR SDRAM Interface	1534
MemVRef1	DDR SDRAM Interface	1534
MemVRef2	DDR SDRAM Interface	1534
OV_{DD}	Power Interface	1538
PCIXAD0:63	PCIX Interface	1533
PCIXC0:7[$\overline{\text{BE0:7}}$]	PCIX Interface	1533
PCIXCap	PCIX Interface	1533
PCIX133Cap	PCIX Interface	1533
PCIXClk	PCIX Interface	1533
$\overline{\text{PCIXDevSel}}$	PCIX Interface	1533

Table 30-1. Signals Listed Alphabetically (continued)

Signal Name	Interface	I/O
$\overline{\text{PCIXFrame}}$	PCIX Interface	1533
$\overline{\text{PCIXGnt0}}$	PCIX Interface	1533
$\overline{\text{PCIXGnt1}}$	PCIX Interface	1533
$\overline{\text{PCIXGnt2:5}}$	PCIX Interface	1533
PCIXIDSel	PCIX Interface	1533
$\overline{\text{PCIXINT}}$	PCIX Interface	1533
$\overline{\text{PCIXIRDY}}$	PCIX Interface	1533
PCIXM66En	PCIX Interface	1533
PCIXParHigh	PCIX Interface	1533
PCIXParLow	PCIX Interface	1533
$\overline{\text{PCIXPErr}}$	PCIX Interface	1533
$\overline{\text{PCIXReq0}}$	PCIX Interface	1533
$\overline{\text{PCIXReq1:5}}$	PCIX Interface	1533
$\overline{\text{PCIXReq64}}$	PCIX Interface	1533
$\overline{\text{PCIXAck64}}$	PCIX Interface	1533
$\overline{\text{PCIXReset}}$	PCIX Interface	1533
$\overline{\text{PCIXSErr}}$	PCIX Interface	1533
$\overline{\text{PCIXStop}}$	PCIX Interface	1533
$\overline{\text{PCIXTRDY}}$	PCIX Interface	1533
PerAddr0:31	External Slave Peripheral Interface	1535
$\overline{\text{PerBE0:3}}$	External Slave Peripheral Interface	1535
$\overline{\text{PerBLast}}$	External Slave Peripheral Interface	1535
PerClk	External Master Peripheral Interface	1536
$\overline{\text{PerCS0:7}}$	External Slave Peripheral Interface	1535
PerData0:31	External Slave Peripheral Interface	1535
PerErr	External Master Peripheral Interface	1536
$\overline{\text{PerOE}}$	External Slave Peripheral Interface	1535
PerPar0:3	External Slave Peripheral Interface	1535
PerReady	External Slave Peripheral Interface	1535
PerR/$\overline{\text{W}}$	External Slave Peripheral Interface	1535
$\overline{\text{PerWE}}$	External Slave Peripheral Interface	1535
$\overline{\text{RAS}}$	DDR SDRAM Interface	1534
UARTSerClk	UART Interface	1536
UART0_Rx	UART Interface	1536
UART0_Tx	UART Interface	1536

Table 30-1. Signals Listed Alphabetically (continued)

Signal Name	Interface	I/O
$\overline{\text{UART0\_DCD}}$	UART Interface	1536
$\overline{\text{UART0\_DSR}}$	UART Interface	1536
$\overline{\text{UART0\_CTS}}$	UART Interface	1536
$\overline{\text{UART0\_DTR}}$	UART Interface	1536
$\overline{\text{UART0\_RTS}}$	UART Interface	1536
$\overline{\text{UART0\_RI}}$	UART Interface	1536
UART1_Rx	UART Interface	1536
UART1_Tx	UART Interface	1536
$\overline{\text{UART1\_DSR}}/\overline{\text{CTS}}$	UART Interface	1536
$\overline{\text{UART1\_RTS/DTR}}$	UART Interface	1536
RcvrInh	System Interface	1537
RefVEn	System Interface	1537
SV_{DD}	Power Interface	1538
SysClk	System Interface	1537
SysErr	System Interface	1537
$\overline{\text{SysReset}}$	System Interface	1537
TCK	JTAG Interface	1537
TDI	JTAG Interface	1537
TDO	JTAG Interface	1537
TestEn	System Interface	1537
TmrClk	System Interface	1537
TMS	JTAG Interface	1537
TrcBS0:2	Trace Interface	1537
TrcClk	Trace Interface	1537
TrcES0:4	Trace Interface	1537
TrcTS0:6	Trace Interface	1537
$\overline{\text{TRST}}$	JTAG Interface	1537
V_{DD}	Power Interface	1538
$\overline{\text{WE}}$	DDR SDRAM Interface	1534

30.2 Signal Descriptions

Table 30-2 lists the PPC440GP signals by interface category with a brief description. Multiplexed signals are shown in brackets following the first signal name assigned to each multiplexed ball (for example, IRQ0:6[GPIO17:23]). Active-low signals are shown with an overbar on the signal name (for example, $\overline{\text{ExtAck}}$).

Table 0-2. Signal Descriptions by Interface Category

Signal Name	Description	I/O
PCI-X Interface		
PCIXAD0:63	Address/Data bus (bidirectional).	I/O
PCIXC0:7[$\overline{\text{BE0:7}}$]	PCI-X Command[Byte Enables].	I/O
PCIXClk	Provides timing to the PCI interface for PCI transactions.	I
$\overline{\text{PCIXDevSel}}$	Indicates the driving device has decoded its address as the target of the current access.	I/O
$\overline{\text{PCIXFrame}}$	Driven by the current master to indicate beginning and duration of an access.	I/O
$\overline{\text{PCIXGnt0}}$	Indicates that the specified agent is granted access to the bus.	O
$\overline{\text{PCIXGnt1}}$	Indicates that the specified agent is granted access to the bus.	I/O
$\overline{\text{PCIXGnt2:5}}$	Indicates that the specified agent is granted access to the bus.	O
PCIXIDSel	Used as a chip select during configuration read and write transactions.	I
$\overline{\text{PCIXINT}}$	Level sensitive PCI interrupt.	O
$\overline{\text{PCIXIRDY}}$	Indicates initiating agent's ability to complete the current data phase of the transaction.	I/O
PCIXM66En	Capable of 66MHz operation.	I
PCIXParHigh	Even parity across PCIAD32:63 and PCIXC0:3[BE4:7].	I/O
PCIXParLow	Even parity across PCIAD0:31 and PCIXC0:3[BE0:3].	I/O
$\overline{\text{PCIXPErr}}$	Reports data parity errors during all PCI transactions except a Special Cycle.	I/O
$\overline{\text{PCIXReq0:5}}$	An indication to the PCI-X arbiter that the specified agent wishes to use the bus.	I
$\overline{\text{PCIXReq64}}$	Asserted by the current bus master, indicating a 64-bit transfer.	I/O
$\overline{\text{PCIXAck64}}$	Indicates the target can transfer data using 64 bits.	I/O
$\overline{\text{PCIXReset}}$	Brings PCI device registers and logic to a consistent state.	O
$\overline{\text{PCIXSErr}}$	Reports address parity errors, data parity errors on the Special Cycle command, or other catastrophic system errors.	I/O

Table 0-2. Signal Descriptions by Interface Category (continued)

Signal Name	Description	I/O
$\overline{\text{PCIXStop}}$	Indicates the current target is requesting the master to stop the current transaction.	I/O
$\overline{\text{PCIXTRDY}}$	Indicates the target agent's ability to complete the current data phase of the transaction.	I/O
PCIXCap	Capable of PCI-X operation.	I
PCIX133Cap	PCI-X devices are 133 MHz capable.	I
DDR SDRAM Interface		
BA0:1	Bank Address supporting up to four internal banks.	O
$\overline{\text{BankSel0:3}}$	Selects up to four external DDR SDRAM banks.	O
$\overline{\text{CAS}}$	Column Address Strobe.	O
ClkEn0:3	Clock Enable. One for each bank.	O
DM0:8	Memory write data byte lane masks. MEMDM8 is the byte lane mask for the ECC byte lane.	O
DQS0:8	Byte lane data strobe. DQS8 is the data strobe for the ECC byte lane.	I/O
ECC0:7	ECC check bits 0:7.	I/O
MemAddr0:12	Memory address bus.	O
MemClkOut0 $\overline{\text{MemClkOut0}}$	Subsystem clock.	O
MemData0:63	Memory data bus.	I/O
MemVRef1 MemVRef2	Memory reference voltages (SV_{REF}).	I
$\overline{\text{RAS}}$	Row Address Strobe.	O
$\overline{\text{WE}}$	Write Enable.	O
Ethernet Interface		
EMCCD[EMC1RxErr]	Collision Detection[Receive error] (MII[RMII 1]).	I/O
EMCCRS[EMC0CRSDV]	Carrier Sense[Receive data valid] (MII[RMII 0]).	I/O
EMCMDClk	Management Data Clock (MII and RMI 0 and 1).	O
EMCMDIO	Transfer command and status information between MII and PHY (MII and RMII 0 and 1).	I/O
EMCRxClk	Receive Clock (MII).	I

Table 0-2. Signal Descriptions by Interface Category (continued)

Signal Name	Description	I/O
EMCRxD0[EMC0RxD0][EMC0RxD] EMCRxD1[EMC0RxD1][EMC1RxD] EMCRxD2[EMC1RxD0] EMCRxD3[EMC1RxD1]	Received Data (MII[RMII 0 and 1][SMII 0 and 1]).	I/O
EMCRxDV[EMC1CRSDV]	Receive Data Valid (MII[RMII 1]).	I
EMCRxErr[EMC0RxErr]	Receive Error (MII[RMII 0]).	I
EMCTxClk[EMCRefClk]	Transmit Clock (MII[RMI 0 and 1, and SMII]).	I
EMCTxD0[EMC0TxD0][EMC0TxD] EMCTxD1[EMC0TxD1][EMC1TxD] EMCTxD2[EMC1TxD0] EMCTxD3[EMC1TxD1]	Transmitted Data (MII[RMII 0 and 1][SMII 0 and 1]).	O
EMCTxEn[EMC0TxEn][EMCSync]	Transmit data Enabled (MII[RMII 0][SMII]).	O
EMCTxErr[EMC1TxEn]	Transmit Error (MII) or Transmit data Enabled (RMII1).	O
External Slave Peripheral Interface		
DMAAck0:3	Used by the PPC440GP to indicate that data transfers have occurred.	O
DMAReq0:3	Used by slave peripherals to indicate they are prepared to transfer data.	I
EOT0:3/TC0:3	End Of Transfer/Terminal Count.	I/O
PerAddr0:31	Peripheral address bus used by PPC440GP when not in external master mode, otherwise used by external master. **Note:** PerAddr0 is the most significant bit (msb) on this bus.	I/O
$\overline{\text{PerBE0:3}}$	External peripheral data bus byte enables.	I/O
$\overline{\text{PerBLast}}$	Used by either the peripheral controller, DMA controller, or external master to indicates the last transfer of a memory access.	I/O
$\overline{\text{PerCS0:7}}$	External peripheral device select.	O
PerData0:31	Peripheral data bus used by PPC440GP when not in external master mode, otherwise used by external master. **Note:** PerData0 is the most significant bit (msb) on this bus.	I/O

Table 0-2. Signal Descriptions by Interface Category (continued)

Signal Name	Description	I/O
$\overline{\text{PerOE}}$	Used by either peripheral controller or DMA controller depending upon the type of transfer involved. When the PPC440GP is the bus master, it enables the selected DDR SDRAMs to drive the bus.	O
PerPar0:3	External peripheral data bus byte parity.	I/O
PerReady[RcvrInh]	Used by a peripheral slave to indicate it is ready to transfer data.	I
PerR/$\overline{\text{W}}$	Used by the PPC440GP when not in external master mode, as output by either the peripheral controller or DMA controller depending upon the type of transfer involved. High indicates a read from memory, low indicates a write to memory. Otherwise, it used by the external master as an input to indicate the direction of transfer.	I/O
$\overline{\text{PerWE}}$	Write Enable. Low when any of the four $\overline{\text{PerBE0:3}}$ signals are low.	O
External Master Peripheral Interface		
BusReq	Bus Request. Used when the PPC440GP needs to regain control of peripheral interface from an external master.	O
$\overline{\text{ExtAck}}$	External Acknowledgement. Used by the PPC440GP to indicate that a data transfer occurred.	O
$\overline{\text{ExtReq}}$	External Request. Used by an external master to indicate it is prepared to transfer data.	I
$\overline{\text{ExtReset}}$	Peripheral Reset. Used by an external master and by synchronous peripheral slaves.	O
HoldAck	Hold Acknowledge. Used by the PPC440GP to transfer ownership of peripheral bus to an external master.	O
HoldReq	Hold Request. Used by an external master to request ownership of the peripheral bus.	I
PerClk	Peripheral Clock. Used by an external master and by synchronous peripheral slaves.	O
PerErr	External Error. Used as an input used to record external master errors and external slave peripheral errors.	I/O
UART Peripheral Interface		
UARTSerClk	Serial Clock used to provide an alternative clock to the internally generated serial clock. Used in cases where the allowable internally generated baud rates are not satisfactory. This input can be individually connected to either or both UART0 and UART1.	I
UART0_Rx	UART0 Receive data.	I
UART0_Tx	UART0 Transmit data.	O
$\overline{\text{UART0\_DCD}}$	UART0 Data Carrier Detect.	I

Table 0-2. Signal Descriptions by Interface Category (continued)

Signal Name	Description	I/O
$\overline{\text{UART0\_DSR}}$	UART0 Data Set Ready.	I
$\overline{\text{UART0\_CTS}}$	UART0 Clear To Send.	I
$\overline{\text{UART0\_DTR}}$	UART0 Data Terminal Ready.	O
$\overline{\text{UART0\_RTS}}$	UART0 Request To Send.	O
$\overline{\text{UART0\_RI}}$	UART0 Ring Indicator.	I
UART1_Rx	UART1 Receive data.	I/O
UART1_Tx	UART1 Transmit data.	I/O
$\overline{\text{UART1\_DSR}}/\overline{\text{CTS}}$	UART1 Data Set Ready or Clear To Send. The choice is determined by a DCR register bit setting.	I/O
$\overline{\text{UART1\_RTS/DTR}}$	UART1 Request To Send or Data Terminal Ready. The choice is determined by a DCR register bit setting.	I/O
IIC Peripheral Interface		
IIC0SClk	IIC0 Serial Clock.	I/O
IIC0SDA	IIC0 Serial Data.	I/O
IIC1SClk	IIC1 Serial Clock.	I/O
IIC1SDA	IIC1 Serial Data.	I/O
Interrupts Interface		
IRQ0:10	Interrupt Requests 0 through 10.	I
[IRQ11]	Interrupt Request 11.	I
[IRQ12]	Interrupt Request 12.	I
JTAG Interface		
TCK	Test Clock.	I
TDI	Test Data In.	I
TDO	Test Data Out.	O
TMS	Test Mode Select.	I
$\overline{\text{TRST}}$	Test Reset.	I
System Interface		
$\overline{\text{ExtReset}}$	External reset	O
SysClk	Main system clock input.	Clk
SysErr	Set to 1 when a machine check is generated.	O

Table 0-2. Signal Descriptions by Interface Category (continued)

Signal Name	Description	I/O
$\overline{\text{SysReset}}$	Main system reset.	I/O
TmrClk	Processor timer external input clock.	I
$\overline{\text{Halt}}$	Halt from external debugger.	I
[GPIO0:10]	General purpose I/O 0 through 10. To access these functions, software must toggle DCR register bits.	I/O
GPIO11	General purpose I/O 11. To access this function, software must toggle a DCR register bit.	I/O
[GPIO12:31]	General purpose I/O 12 through 31. To access these functions, software must toggle a DCR register bit.	I/O
TestEn	Test Enable.	I
[RcvrInh]	Receiver Inhibit. Active only when TestEn is active.	I
RefVEn	Reference Voltage Enable. Used for wafer testing. Do not connect for normal operation.	I
DrvrInh1:2	Driver Inhibit. Used for test purposes only. Tie up for normal operation	I
Trace Interface		
TrcBS0:2	Branch execution status.	I/O
TrcClk	Trace data capture clock, runs at 1/4 the frequency of the processor.	O
TrcES0:4	Trace Execution Status is presented every fourth processor clock cycle.	I/O
TrcTS0:6	Additional information on trace execution and branch status.	I/O
Power Pins		
AGND	PLL (analog) voltage ground.	n/a
GND	Ground.	n/a
AxV_{DD}	Filtered voltages input for PLLs (analog circuits) **Note:** A separate filter for each of the three voltages is recommended.	n/a
OV_{DD}	I/O (except DDR SDRAM) voltage—3.3V.	n/a
SV_{DD}	DDR SDRAM voltage—2.5V.	n/a
V_{DD}	Logic voltage—1.8V.	n/a
Reserved Pins		
Reserved	Do not connect signals, voltage, or ground to these balls.	n/a

Table 30-2. Signal Descriptions by Interface Category

Signal Name	Description	I/O
PCI-X Interface		
PCIXAD0:63	Address/Data bus (bidirectional).	I/O
PCIXC0:7[$\overline{\text{BE0:7}}$]	PCI-X Command[Byte Enables].	I/O
PCIXCap	Capable of PCI-X operation.	I
PCIX133Cap	PCI-X devices are 133 MHz capable.	I
PCIXClk	Provides timing to the PCI interface for PCI transactions.	I
$\overline{\text{PCIXDevSel}}$	Indicates the driving device has decoded its address as the target of the current access.	I/O
$\overline{\text{PCIXFrame}}$	Driven by the current master to indicate beginning and duration of an access.	I/O
$\overline{\text{PCIXGnt0}}$	Indicates that the specified agent is granted access to the bus.	I/O
$\overline{\text{PCIXGnt1}}$	Indicates that the specified agent is granted access to the bus.	I/O
$\overline{\text{PCIXGnt2:5}}$	Indicates that the specified agent is granted access to the bus.	O
PCIXIDSel	Used as a chip select during configuration read and write transactions.	I
$\overline{\text{PCIXINT}}$	Level sensitive PCI interrupt.	O
$\overline{\text{PCIXIRDY}}$	Indicates initiating agent's ability to complete the current data phase of the transaction.	I/O
PCIXM66En	Capable of 66 MHz operation.	I
PCIXParHigh	Even parity across PCIAD32:63 and PCIXC0:3[BE4:7].	I/O
PCIXParLow	Even parity across PCIAD0:31 and PCIXC0:3[BE0:3].	I/O
$\overline{\text{PCIXPErr}}$	Reports data parity errors during all PCI transactions except a Special Cycle.	I/O
$\overline{\text{PCIXReq0}}$	An indication to the PCI-X arbiter that the specified agent wishes to use the bus.	I/O
$\overline{\text{PCIXReq1:5}}$	An indication to the PCI-X arbiter that the specified agent wishes to use the bus.	I
$\overline{\text{PCIXReq64}}$	Asserted by the current bus master, indicating a 64-bit transfer.	I/O
$\overline{\text{PCIXAck64}}$	Indicates the target can transfer data using 64 bits.	I/O
$\overline{\text{PCIXReset}}$	Brings PCI device registers and logic to a consistent state.	O
$\overline{\text{PCIXSErr}}$	Reports address parity errors, data parity errors on the Special Cycle command, or other catastrophic system errors.	I/O
$\overline{\text{PCIXStop}}$	Indicates the current target is requesting the master to stop the current transaction.	I/O
$\overline{\text{PCIXTRDY}}$	Indicates the target agent's ability to complete the current data phase of the transaction.	I/O
DDR SDRAM Interface		
BA0:1	Bank Address supporting up to four internal banks.	O
$\overline{\text{BankSel0:3}}$	Selects up to four external DDR SDRAM banks.	O
$\overline{\text{CAS}}$	Column Address Strobe.	O
ClkEn0:3	Clock Enable. One for each bank.	O
DM0:8	Memory write data byte lane masks. MEMDM8 is the byte lane mask for the ECC byte lane.	O

Table 30-2. Signal Descriptions by Interface Category (continued)

Signal Name	Description	I/O
DQS0:8	Byte lane data strobe. DQS8 is the data strobe for the ECC byte lane.	I/O
ECC0:7	ECC check bits 0:7.	I/O
MemAddr0:12	Memory address bus.	O
MemClkOut0 $\overline{\text{MemClkOut0}}$	Subsystem clock.	O
MemData0:63	Memory data bus.	I/O
MemVRef1	Memory reference voltage (SV_{REF}) input.	I
MemVRef2	Memory reference voltage (SV_{REF}) output.	O
$\overline{\text{RAS}}$	Row Address Strobe.	O
$\overline{\text{WE}}$	Write Enable.	O
Ethernet Interface		
EMCCD, EMC1RxErr	MII: Collision detection RMII 1: Receive error	I/O
EMCCrS, EMC0CrSDV	MII: Carrier sense RMII 0: Carrier sense data valid	I/O
EMCMDClk	MII and RMII: Management data clock	O
EMCMDIO	MII and RMII: Transfer command and status information between MII and PHY	I/O
EMCRxD0:3, EMC0RxD0:1, EMC1RxD0:1, EMC0RxD, EMC1RxD	MII: Receive data RMII 0: Receive data RMII 1: Receive data SMII 0: Receive data SMII 1: Receive data	I/O
EMCRxDV, EMC1CrSDV	MII: Receive data valid RMII 1: Carrier sense data valid	I
EMCRxClk	MII: Receive clock:	I
EMCRxErr, EMC0RxErr	MII: Receive error RMII 0: Receive error	I
EMCTxClk, EMCRefClk	MII: Transmit clock RMII and SMII: Transmit clock	I
EMCTxD0:3, EMC0TxD0:1, EMC1TxD0:1, EMC0TxD, EMC1TxD	MII: Transmit data RMII 0: Transmit data RMII 1: Transmit data SMII 0: Transmit data SMII 1: Transmit data	O
EMCTxEn, EMC0TxEn, EMCSync	MII: Transmit data enabled RMII 0: Transmit data enabled SMII: Sync signal	O
EMCTxErr, EMC1TxEn	MII: Transmit error: RMII : Transmit data enabled	O
External Slave Peripheral Interface		
DMAAck0:3	Used by the PPC440GP to indicate that data transfers have occurred.	O
DMAReq0:3	Used by slave peripherals to indicate they are prepared to transfer data.	I
EOT0:3/TC0:3	End Of Transfer/Terminal Count.	I/O

Table 30-2. Signal Descriptions by Interface Category (continued)

Signal Name	Description	I/O
PerAddr0:31	Peripheral address bus used by PPC440GP when not in external master mode, otherwise used by external master. **Note:** PerAddr0 is the most significant bit (msb) on this bus.	I/O
$\overline{\text{PerBE0:3}}$	External peripheral data bus byte enables.	I/O
$\overline{\text{PerBLast}}$	Used by either the peripheral controller, DMA controller, or external master to indicates the last transfer of a memory access.	I/O
$\overline{\text{PerCS0:7}}$	External peripheral device select.	O
PerData0:31	Peripheral data bus used by PPC440GP when not in external master mode, otherwise used by external master. **Note:** PerData0 is the most significant bit (msb) on this bus.	I/O
$\overline{\text{PerOE}}$	Used by either peripheral controller or DMA controller depending upon the type of transfer involved. When the PPC440GP is the bus master, it enables the selected DDR SDRAMs to drive the bus.	O
PerPar0:3	External peripheral data bus byte parity.	I/O
PerReady	Used by a peripheral slave to indicate it is ready to transfer data.	I
PerR/$\overline{\text{W}}$	Used by the PPC440GP when not in external master mode, as output by either the peripheral controller or DMA controller depending upon the type of transfer involved. High indicates a read from memory, low indicates a write to memory. Otherwise, it used by the external master as an input to indicate the direction of transfer.	I/O
$\overline{\text{PerWE}}$	Write Enable. Low when any of the four $\overline{\text{PerBE0:3}}$ signals are low.	O
External Master Peripheral Interface		
BusReq	Bus Request. Used when the PPC440GP needs to regain control of peripheral interface from an external master.	O
$\overline{\text{ExtAck}}$	External Acknowledgement. Used by the PPC440GP to indicate that a data transfer occurred.	O
$\overline{\text{ExtReq}}$	External Request. Used by an external master to indicate it is prepared to transfer data.	I
$\overline{\text{ExtReset}}$	Peripheral Reset. Used by an external master and by synchronous peripheral slaves.	O
HoldAck	Hold Acknowledge. Used by the PPC440GP to transfer ownership of peripheral bus to an external master.	O
HoldReq	Hold Request. Used by an external master to request ownership of the peripheral bus.	I
PerClk	Peripheral Clock. Used by an external master and by synchronous peripheral slaves.	O
PerErr	External Error. Used as an input used to record external master errors and external slave peripheral errors.	I/O
UART Peripheral Interface		
UARTSerClk	Serial clock input that provides an alternative to the internally generated serial clock. Used in cases where the allowable internally generated clock rates are not satisfactory. This input can be individually connected to either or both UART0 and UART1.	I
UART0_Rx	UART0 Receive data.	I
UART0_Tx	UART0 Transmit data.	O
$\overline{\text{UART0\_DCD}}$	UART0 Data Carrier Detect.	I

Table 30-2. Signal Descriptions by Interface Category (continued)

Signal Name	Description	I/O
$\overline{\text{UART0\_DSR}}$	UART0 Data Set Ready.	I
$\overline{\text{UART0\_CTS}}$	UART0 Clear To Send.	I
$\overline{\text{UART0\_DTR}}$	UART0 Data Terminal Ready.	O
$\overline{\text{UART0\_RTS}}$	UART0 Request To Send.	O
$\overline{\text{UART0\_RI}}$	UART0 Ring Indicator.	I
UART1_Rx	UART1 Receive data.	I/O
UART1_Tx	UART1 Transmit data.	I/O
$\overline{\text{UART1\_DSR}}$/$\overline{\text{CTS}}$	UART1 Data Set Ready or Clear To Send. The choice is determined by a DCR register bit setting.	I/O
$\overline{\text{UART1\_RTS/DTR}}$	UART1 Request To Send or Data Terminal Ready. The choice is determined by a DCR register bit setting.	I/O
IIC Peripheral Interface		
IIC0SClk	IIC0 Serial Clock.	I/O
IIC0SDA	IIC0 Serial Data.	I/O
IIC1SClk	IIC1 Serial Clock.	I/O
IIC1SDA	IIC1 Serial Data.	I/O
Interrupts Interface		
IRQ0:10	Interrupt Requests 0 through 10.	I
IRQ11:12	Interrupt Requests 11 through 12.	I
JTAG Interface		
TCK	Test Clock.	I
TDI	Test Data In.	I
TDO	Test Data Out.	O
TMS	Test Mode Select.	I
$\overline{\text{TRST}}$	Test Reset.	I
System Interface		
$\overline{\text{ExtReset}}$	External reset	O
SysClk	Main system reset. External logic can drive this bidirectional pin low (minimum of 16 cycles) to initiate a system reset. A system reset can also be initiated by software. Implemented as an open-drain output (two states; 0 or open circuit).	Clock
SysErr	Set to 1 when a machine check is generated.	O
$\overline{\text{SysReset}}$	Main system reset.	I/O
TmrClk	Processor timer external input clock.	I
$\overline{\text{Halt}}$	Halt from external debugger.	I
GPIO0:31	General purpose I/O 0 through 10. To access these functions, software must set DCR register bits.	I/O
TestEn	Test Enable.	I
RcvrInh	Receiver Inhibit. Active only when TestEn is active.	I

Table 30-2. Signal Descriptions by Interface Category (continued)

Signal Name	Description	I/O
RefVEn	Reference Voltage Enable. Used for wafer testing. Do not connect for normal operation.	I
DrvrInh1:2	Driver Inhibit. Used for test purposes only. Tie up for normal operation	I
Trace Interface		
TrcBS0:2	Trace branch execution status.	I/O
TrcClk	Trace data capture clock, runs at 1/4 the frequency of the processor.	O
TrcES0:4	Trace Execution Status is presented every fourth processor clock cycle.	I/O
TrcTS0:6	Additional information on trace execution and branch status.	I/O
Power Pins		
AGND	PLL (analog) voltage ground.	n/a
GND	Ground.	n/a
AxV_{DD}	Filtered voltages input for PLLs (analog circuits) **Note:** A separate filter for each of the three voltages is recommended.	n/a
OV_{DD}	I/O (except DDR SDRAM) voltage—3.3V.	n/a
SV_{DD}	DDR SDRAM voltage—2.5V.	n/a
V_{DD}	Logic voltage—1.8V.	n/a
Reserved Pins		
Reserved	Do not connect signals, voltage, or ground to these balls.	n/a

.

Appendix A. Instruction Summary

This appendix describes the various instruction formats, and lists all of the PPC440GP instructions summarized alphabetically and by opcode.

"Instruction Formats", illustrates the PPC440GP instruction forms (allowed arrangements of fields within instructions).

"Alphabetical Summary of Implemented Instructions," on page A-7*Appendix A.2* on page 1544 lists all PPC440GP mnemonics, including extended mnemonics. A short functional description is included for each mnemonic.

"Allocated Instruction Opcodes," on page A-40*Appendix A.3* on page 1575 identifies those opcodes which are allocated by PowerPC Book-E for implementation-dependent usage, including auxiliary processors.

"Preserved Instruction Opcodes," on page A-41*Appendix A.4* on page 1575 identifies those opcodes which are identified by PowerPC Book-E as "preserved" for compatibility with previous versions of the architecture.

"Reserved Instruction Opcodes," on page A-41*Appendix A.5* on page 1576 indentifies those opcodes which are "reserved" for use by future versions of the architecture.

"Implemented Instructions Sorted by Opcode," on page A-42*Appendix A.6* on page 1577, lists all instructions implemented within the PPC440GP, sorted by primary and secondary opcodes. Extended mnemonics are not included in the opcode list, but allocated, preserved, and reserved-nop opcodes are included.

A.1 Instruction Formats

Instructions are four bytes long. Instruction addresses are always word-aligned.

Instruction bits 0 through 5 always contain the primary opcode. Many instructions have an extended opcode in another field. Remaining instruction bits contain additional fields. All instruction fields belong to one of the following categories:

- Defined

  These instructions contain values, such as opcodes, that cannot be altered. The instruction format diagrams specify the values of defined fields.

- Variable

  These fields contain operands, such as GPR selectors and immediate values, that can vary from execution to execution. The instruction format diagrams specify the operands in the variable fields.

- Reserved

  Bits in reserved fields should be set to 0. In the instruction format diagrams, /, //, or /// indicate reserved fields.

If any bit in a defined field does not contain the expected value, the instruction is illegal and an illegal instruction exception occurs. If any bit in a reserved field does not contain 0, the instruction form is invalid; its result is architecturally undefined. The PPC440GP executes all invalid instruction forms without causing an illegal instruction exception.

A.1.1 Instruction Fields

PPC440GP instructions contain various combinations of the following fields, as indicated in the instruction format diagrams that follow the field definitions. Numbers, enclosed in parentheses, that follow the field names indicate bit positions; bit fields are indicated by starting and stopping bit positions separated by colons.

AA (30) Absolute address bit.

- 0 The immediate field represents an address relative to the current instruction address (CIA). The effective address (EA) of the branch is either the sum of the LI field sign-extended to 32 bits and the branch instruction address, or the sum of the BD field sign-extended to 32 bits and the branch instruction address.
- 1 The immediate field represents an absolute address. The EA of the branch is either the LI field or the BD field, sign-extended to 32 bits.

BA (11:15) Specifies a bit in the CR used as a source of a CR-logical instruction.

BB (16:20) Specifies a bit in the CR used as a source of a CR-logical instruction.

BD (16:29) An immediate field specifying a 14-bit signed twos complement branch displacement. This field is concatenated on the right with 0b00 and sign-extended to 32 bits.

BF (6:8) Specifies a field in the CR used as a target in a compare or **mcrf** instruction.

BFA (11:13) Specifies a field in the CR used as a source in a **mcrf** instruction.

BI (11:15) Specifies a bit in the CR used as a source for the condition of a conditional branch instruction.

BO (6:10) Specifies options for conditional branch instructions. See ~~"Branch Instruction BO Field" on page 4-46~~*Branch Instruction BO Field* on page 214.

BT (6:10) Specifies a bit in the CR used as a target as the result of a CR-Logical instruction.

D (16:31) Specifies a 16-bit signed two's-complement integer displacement for load/store instructions.

DCRF (11:20) Specifies a device control register (DCR). This field represents the DCR Number (DCRN) with the upper and lower five bits reversed (that is, DCRF = DCRN[5:9] || DCRN[0:4]).

FXM (12:19) Field mask used to identify CR fields to be updated by the **mtcrf** instruction.

IM (16:31) An immediate field used to specify a 16-bit value (either signed integer or unsigned).

LI (6:29) An immediate field specifying a 24-bit signed twos complement branch displacement; this field is concatenated on the right with b'00' and sign-extended to 32 bits.

LK (31) Link bit.

- 0 Do not update the link register (LR).
- 1 Update the LR with the address of the next instruction.

MB (21:25) Mask begin.

Used in rotate-and-mask instructions to specify the beginning bit of a mask.

ME (26:30) Mask end.

Used in rotate-and-mask instructions to specify the ending bit of a mask.

NB (16:20) Specifies the number of bytes to move in an immediate string load or store.

OPCD (0:5) Primary opcode. Primary opcodes, in decimal, appear in the instruction format diagrams presented with individual instructions. The OPCD field name does not appear in instruction descriptions.

OE (21) Enables setting the OV and SO fields in the fixed-point exception register (XER) for extended arithmetic.

RA (11:15) A GPR used as a source or target.

RB (16:20) A GPR used as a source.

Rc (31) Record bit.

0 Do not set the CR.

1 Set the CR to reflect the result of an operation.

See ~~"Condition Register (CR)" on page 4-50~~*Condition Register (CR)* on page 217 for a further discussion of how the CR bits are set.

RS (6:10) A GPR used as a source.

RT (6:10) A GPR used as a target.

SH (16:20) Specifies a shift amount.

SPRF (11:20) Specifies a special purpose register (SPR). This field represents the SPR Number (SPRN) with the upper and lower five bits reversed (that is, SPRF = SPRN[5:9] || SPRN[0:4]).

TO (6:10) Specifies the conditions on which to trap, as described under **tw** and **twi** instructions.

WS (16:20) Specifies the portion of a TLB entry to be read/written by **tlbre**/**tlbwe**.

XO (21:30) Extended opcode for instructions without an OE field. Extended opcodes, in decimal, appear in the instruction format diagrams presented with individual instructions. The XO field name does not appear in instruction descriptions.

XO (22:30) Extended opcode for instructions with an OE field. Extended opcodes, in decimal, appear in the instruction format diagrams presented with individual instructions. The XO field name does not appear in instruction descriptions.

A.1.2 Instruction Format Diagrams

The instruction formats (allso called "forms") illustrated in Figure A-1 through Figure A-9 are valid combinations of instruction fields. ~~Table A-5 on page A-43~~*Table A-5* on page 1577 indicates which "form" is utilized by each PPC440GP opcode. Fields indicated by slashes (/, //, or ///) are reserved. The figures are adapted from the PowerPC User Instruction Set Architecture.

A.1.2.1 I-Form

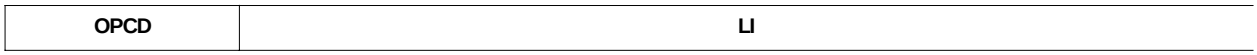


Figure A-1. I Instruction Format

A.1.2.2 B-Form

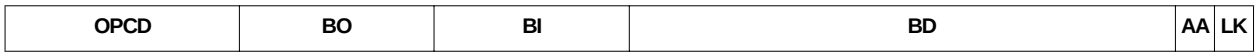


Figure A-2. B Instruction Format

A.1.2.3 SC-Form

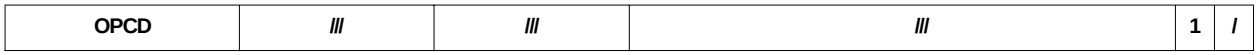


Figure A-3. SC Instruction Format

A.1.2.4 D-Form

OPCD	**RT**		**RA**	**D**
OPCD	**RS**		**RA**	**SI**
OPCD	**RS**		**RA**	**D**
OPCD	**RS**		**RA**	**UI**
OPCD	**BF** / **L**		**RA**	**SI**
OPCD	**BF** / **L**		**RA**	**UI**
OPCD	**TO**		**RA**	**SI**

0 6 11 16 31

Figure A-4. D Instruction Format

A.1.2.5 X-Form

OPCD	**RT**	**RA**	**RB**	**XO**	**Rc**
OPCD	**RT**	**RA**	**RB**	**XO**	**/**
OPCD	**RT**	**RA**	**NB**	**XO**	**/**
OPCD	**RT**	**RA**	**WS**	**XO**	**/**
OPCD	**RT**	**///**	**RB**	**XO**	**/**
OPCD	**RT**	**///**	**///**	**XO**	**/**
OPCD	**RS**	**RA**	**RB**	**XO**	**Rc**
OPCD	**RS**	**RA**	**RB**	**XO**	**1**
OPCD	**RS**	**RA**	**RB**	**XO**	**/**
OPCD	**RS**	**RA**	**NB**	**XO**	**/**
OPCD	**RS**	**RA**	**WS**	**XO**	**/**
OPCD	**RS**	**RA**	**SH**	**XO**	**Rc**
OPCD	**RS**	**RA**	**///**	**XO**	**Rc**
OPCD	**RS**	**///**	**RB**	**XO**	**/**
OPCD	**RS**	**///**	**///**	**XO**	**/**
OPCD	**BF** **/** **L**	**RA**	**RB**	**XO**	**/**
OPCD	**BF** **//**	**BFA** **//**	**///**	**XO**	**Rc**
OPCD	**BF** **//**	**///**	**///**	**XO**	**/**
OPCD	**BF** **//**	**///**	**U**	**XO**	**Rc**
OPCD	**BF** **//**	**///**	**///**	**XO**	**/**
OPCD	**TO**	**RA**	**RB**	**XO**	**/**
OPCD	**BT**	**///**	**///**	**XO**	**Rc**
OPCD	**///**	**RA**	**RB**	**XO**	**/**
OPCD	**///**	**///**	**///**	**XO**	**/**
OPCD	**///**	**///**	**E** **//**	**XO**	**/**
0	6	11	16	21	31

Figure A-5. X Instruction Format

A.1.2.6 XL-Form

OPCD	BT		BA		BB	XO	/
OPCD	BC		BI		///	XO	LK
OPCD	BF	//	BFA	//	///	XO	/
OPCD	///		///		///	XO	/
0	6		11		16	21	31

Figure A-6. XL Instruction Format

A.1.2.7 XFX-Form

OPCD	RT	SPRF			XO	/
OPCD	RT	DCRF			XO	/
OPCD	RT	/	FXM	/	XO	/
OPCD	RS	SPRF			XO	/
OPCD	RS	DCRF			XO	/
0	6	11	16		21	31

Figure A-7. XFX Instruction Format

A.1.2.8 XO-Form

OPCD	RT	RA	RB	OE	XO	Rc
OPCD	RT	RA	RB	OE	XO	Rc
OPCD	RT	RA	///	/	XO	Rc
0	6	11	16	21	22	31

Figure A-8. XO Instruction Format

A.1.2.9 M-Form

OPCD	RS	RA	RB	MB	ME	Rc
OPCD	RS	RA	SH	MB	ME	Rc
0	6	11	16	21	26	31

Figure A-9. M Instruction Format

A.2 Alphabetical Summary of Implemented Instructions

~~Table A-1~~*Table A-1* summarizes the PPC440GP instruction set, including required extended mnemonics. All mnemonics are listed alphabetically, without regard to whether the mnemonic is realized in hardware or software. When an instruction supports multiple hardware mnemonics (for example, **b**, **ba**, **bl**, **bla** are all forms of **b**), the instruction is alphabetized under the root form. The hardware instructions are described in detail in ~~Chapter 28, "Instruction Set,"~~*Section 28 Instruction Set* on page 893 which is also alphabetized under the root form. ~~Chapter 28~~*Section 28* also describes the instruction operands and notation.

Programming Note: Bit 4 of the BO instruction field provides a hint about the most likely outcome of a conditional branch. (See ~~"Branch Prediction" on page 4-47~~*Branch Prediction* on page 215 for a detailed description of branch prediction.) Assemblers should set BO_4 = 0 unless a specific reason exists otherwise. In the BO field values specified in Table A-1, BO_4 = 0 has always been assumed. The assembler must enable the programmer to specify branch prediction. To do this, the assembler supports suffixes for the conditional branch mnemonics:

+ Predict branch to be taken.

– Predict branch not to be taken.

For example, **bc** also could be coded as **bc+** or **bc–**, and **bne** also could be coded **bne+** or **bne–**. These alternate codings set BO_4 = 1 only if the requested prediction differs from the standard prediction. See ~~"Branch Prediction" on page 4-47~~*Branch Prediction* on page 215 for more information.

Table B-1. PPC440GP Instruction Syntax Summary

Mnemonic	Operands	Function	Other Registers Changed	Page
add	RT, RA, RB	Add (RA) to (RB). Place result in RT.		28-7
add.			CR[CR0]	
addo			XER[SO, OV]	
addo.			CR[CR0] XER[SO, OV]	
addc	RT, RA, RB	Add (RA) to (RB). Place result in RT. Place carry-out in XER[CA].		28-8
addc.			CR[CR0]	
addco			XER[SO, OV]	
addco.			CR[CR0] XER[SO, OV]	
adde	RT, RA, RB	Add XER[CA], (RA), (RB). Place result in RT. Place carry-out in XER[CA].		28-9
adde.			CR[CR0]	
addeo			XER[SO, OV]	
addeo.			CR[CR0] XER[SO, OV]	
addi	RT, RA, IM	Add EXTS(IM) to (RA|0). Place result in RT.		28-10
addic	RT, RA, IM	Add EXTS(IM) to (RA|0). Place result in RT. Place carry-out in XER[CA].		28-11

Table B-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
addic.	RT, RA, IM	Add EXTS(IM) to (RA|0). Place result in RT. Place carry-out in XER[CA].	CR[CR0]	28-12
addis	RT, RA, IM	Add (IM || ^{16}0) to (RA|0). Place result in RT.		28-13
addme	RT, RA	Add XER[CA], (RA), (-1). Place result in RT. Place carry-out in XER[CA].		28-14
addme.			CR[CR0]	
addmeo			XER[SO, OV]	
addmeo.			CR[CR0] XER[SO, OV]	
addze	RT, RA	Add XER[CA] to (RA). Place result in RT. Place carry-out in XER[CA].		28-15
addze.			CR[CR0]	
addzeo			XER[SO, OV]	
addzeo.			CR[CR0] XER[SO, OV]	
and	RA, RS, RB	AND (RS) with (RB). Place result in RA.		28-16
and.			CR[CR0]	
andc	RA, RS, RB	AND (RS) with ¬(RB). Place result in RA.		28-17
andc.			CR[CR0]	
andi.	RA, RS, IM	AND (RS) with (^{16}0 || IM). Place result in RA.	CR[CR0]	28-18
andis.	RA, RS, IM	AND (RS) with (IM || ^{16}0). Place result in RA.	CR[CR0]	28-19
b	target	Branch unconditional relative. LI ← (target – CIA)$_{6:29}$ NIA ← CIA + EXTS(LI || ^{2}0)		28-20
ba		Branch unconditional absolute. LI ← target$_{6:29}$ NIA ← EXTS(LI || ^{2}0)		
bl		Branch unconditional relative. LI ← (target – CIA)$_{6:29}$ NIA ← CIA + EXTS(LI || ^{2}0)	(LR) ← CIA + 4	
bla		Branch unconditional absolute. LI ← target$_{6:29}$ NIA ← EXTS(LI || ^{2}0)	(LR) ← CIA + 4	

Table B-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
bc	BO, BI, target	Branch conditional relative. BD ← $(target - CIA)_{16:29}$ NIA ← CIA + EXTS(BD || $^{2}0$)	CTR if $BO_2 = 0$	28-21
bca		Branch conditional absolute. BD ← $target_{16:29}$ NIA ← EXTS(BD || $^{2}0$)	CTR if $BO_2 = 0$	
bcl		Branch conditional relative. BD ← $(target - CIA)_{16:29}$ NIA ← CIA + EXTS(BD || $^{2}0$)	CTR if $BO_2 = 0$ (LR) ← CIA + 4	
bcla		Branch conditional absolute. BD ← $target_{16:29}$ NIA ← EXTS(BD || $^{2}0$)	CTR if $BO_2 = 0$ (LR) ← CIA + 4	
bcctr	BO, BI	Branch conditional to address in CTR. Using (CTR) at exit from instruction, NIA ← $CTR_{0:29}$ || $^{2}0$	CTR if $BO_2 = 0$	28-27
bcctrl			CTR if $BO_2 = 0$ (LR) ← CIA + 4	
bclr	BO, BI	Branch conditional to address in LR. Using (LR) at entry to instruction, NIA ← $LR_{0:29}$ || $^{2}0$	CTR if $BO_2 = 0$	28-31
bclrl			CTR if $BO_2 = 0$ (LR) ← CIA + 4	
bctr		Branch unconditionally to address in CTR. *Extended mnemonic for* **bcctr 20,0**		28-27
bctrl		*Extended mnemonic for* **bcctrl 20,0**	(LR) ← CIA + 4	
bdnz	target	Decrement CTR. Branch if CTR ≠ 0 *Extended mnemonic for* **bc 16,0,target**		28-21
bdnza		*Extended mnemonic for* **bca 16,0,target**		
bdnzl		*Extended mnemonic for* **bcl 16,0,target**	(LR) ← CIA + 4	
bdnzla		*Extended mnemonic for* **bcla 16,0,target**	(LR) ← CIA + 4	
bdnzlr		Decrement CTR. Branch if CTR ≠ 0 to address in LR. *Extended mnemonic for* **bclr 16,0**		28-31
bdnzlrl		*Extended mnemonic for* **bclrl 16,0**	(LR) ← CIA + 4	

Table B-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
bdnzf	cr_bit, target	Decrement CTR. Branch if CTR ≠ 0 AND CR_{cr_bit} = 0 *Extended mnemonic for* **bc 0,cr_bit,target**		28-21
bdnzfa		*Extended mnemonic for* **bca 0,cr_bit,target**		
bdnzfl		*Extended mnemonic for* **bcl 0,cr_bit,target**	(LR) ← CIA + 4	
bdnzfla		*Extended mnemonic for* **bcla 0,cr_bit,target**	(LR) ← CIA + 4	
bdnzflr	cr_bit	Decrement CTR. Branch if CTR ≠ 0 AND CR_{cr_bit} = 0 to address in LR. *Extended mnemonic for* **bclr 0,cr_bit**		28-31
bdnzflrl		*Extended mnemonic for* **bclrl 0,cr_bit**	(LR) ← CIA + 4	
bdnzt	cr_bit, target	Decrement CTR. Branch if CTR ≠ 0 AND CR_{cr_bit} = 1. *Extended mnemonic for* **bc 8,cr_bit,target**		28-21
bdnzta		*Extended mnemonic for* **bca 8,cr_bit,target**		
bdnztl		*Extended mnemonic for* **bcl 8,cr_bit,target**	(LR) ← CIA + 4	
bdnztla		*Extended mnemonic for* **bcla 8,cr_bit,target**	(LR) ← CIA + 4	
bdnztlr	cr_bit	Decrement CTR. Branch if CTR ≠ 0 AND CR_{cr_bit} = 1 to address in LR. *Extended mnemonic for* **bclr 8,cr_bit**		28-31
bdnztlrl		*Extended mnemonic for* **bclrl 8,cr_bit**	(LR) ← CIA + 4	
bdz	target	Decrement CTR. Branch if CTR = 0 *Extended mnemonic for* **bc 18,0,target**		28-21
bdza		*Extended mnemonic for* **bca 18,0,target**		
bdzl		*Extended mnemonic for* **bcl 18,0,target**	(LR) ← CIA + 4	
bdzla		*Extended mnemonic for* **bcla 18,0,target**	(LR) ← CIA + 4	

Table B-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
bdzlr		Decrement CTR. Branch if CTR = 0 to address in LR. *Extended mnemonic for* **bclr 18,0**		28-31
bdzlrl		*Extended mnemonic for* **bclrl 18,0**	(LR) ← CIA + 4	
bdzf	cr_bit, target	Decrement CTR. Branch if CTR = 0 AND CR_{cr_bit} = 0 *Extended mnemonic for* **bc 2,cr_bit,target**		28-21
bdzfa		*Extended mnemonic for* **bca 2,cr_bit,target**		
bdzfl		*Extended mnemonic for* **bcl 2,cr_bit,target**	(LR) ← CIA + 4	
bdzfla		*Extended mnemonic for* **bcla 2,cr_bit,target**	(LR) ← CIA + 4	
bdzflr	cr_bit	Decrement CTR. Branch if CTR = 0 AND CR_{cr_bit} = 0 to address in LR. *Extended mnemonic for* **bclr 2,cr_bit**		28-31
bdzflrl		*Extended mnemonic for* **bclrl 2,cr_bit**	(LR) ← CIA + 4	
bdzt	cr_bit, target	Decrement CTR. Branch if CTR = 0 AND CR_{cr_bit} = 1. *Extended mnemonic for* **bc 10,cr_bit,target**		28-21
bdzta		*Extended mnemonic for* **bca 10,cr_bit,target**		
bdztl		*Extended mnemonic for* **bcl 10,cr_bit,target**	(LR) ← CIA + 4	
bdztla		*Extended mnemonic for* **bcla 10,cr_bit,target**	(LR) ← CIA + 4	
bdztlr	cr_bit	Decrement CTR. Branch if CTR = 0 AND CR_{cr_bit} = 1to address in LR. *Extended mnemonic for* **bclr 10,cr_bit**		28-31
bdztlrl		*Extended mnemonic for* **bclrl 10,cr_bit**	(LR) ← CIA + 4	

Table B-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
beq	[cr_field], target	Branch if equal. UseCR[CR0] if cr_field is omitted. *Extended mnemonic for* **bc 12,4∗cr_field+2,target**		28-21
beqa		*Extended mnemonic for* **bca 12,4∗cr_field+2,target**		
beql		*Extended mnemonic for* **bcl 12,4∗cr_field+2,target**	(LR) ← CIA + 4	
beqla		*Extended mnemonic for* **bcla 12,4∗cr_field+2,target**	(LR) ← CIA + 4	
beqctr	[cr_field]	Branch if equal to address in CTR. UseCR[CR0] if cr_field is omitted. *Extended mnemonic for* **bcctr 12,4∗cr_field+2**		28-27
beqctrl		*Extended mnemonic for* **bcctrl 12,4∗cr_field+2**	(LR) ← CIA + 4	
beqlr	[cr_field]	Branch if equal to address in LR. UseCR[CR0] if cr_field is omitted. *Extended mnemonic for* **bclr 12,4∗cr_field+2**		28-31
beqlrl		*Extended mnemonic for* **bclrl 12,4∗cr_field+2**	(LR) ← CIA + 4	
bf	cr_bit, target	Branch if CR_{cr_bit} = 0. *Extended mnemonic for* **bc 4,cr_bit,target**		28-21
bfa		*Extended mnemonic for* **bca 4,cr_bit,target**		
bfl		*Extended mnemonic for* **bcl 4,cr_bit,target**	(LR) ← CIA + 4	
bfla		*Extended mnemonic for* **bcla 4,cr_bit,target**	(LR) ← CIA + 4	
bfctr	cr_bit	Branch if CR_{cr_bit} = 0 to address in CTR. *Extended mnemonic for* **bcctr 4,cr_bit**		28-27
bfctrl		*Extended mnemonic for* **bcctrl 4,cr_bit**	(LR) ← CIA + 4	
bflr	cr_bit	Branch if CR_{cr_bit} = 0 to address in LR. *Extended mnemonic for* **bclr 4,cr_bit**		28-31
bflrl		*Extended mnemonic for* **bclrl 4,cr_bit**	(LR) ← CIA + 4	

Table B-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
bge	[cr_field], target	Branch if greater than or equal. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bc 4,4∗cr_field+0,target**		28-21
bgea		*Extended mnemonic for* **bca 4,4∗cr_field+0,target**		
bgel		*Extended mnemonic for* **bcl 4,4∗cr_field+0,target**	(LR) ← CIA + 4	
bgela		*Extended mnemonic for* **bcla 4,4∗cr_field+0,target**	(LR) ← CIA + 4	
bgectr	[cr_field]	Branch if greater than or equal to address in CTR. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bcctr 4,4∗cr_field+0**		28-27
bgectrl		*Extended mnemonic for* **bcctrl 4,4∗cr_field+0**	(LR) ← CIA + 4	
bgelr	[cr_field]	Branch if greater than or equal to address in LR. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bclr 4,4∗cr_field+0**		28-31
bgelrl		*Extended mnemonic for* **bclrl 4,4∗cr_field+0**	(LR) ← CIA + 4	
bgt	[cr_field], target	Branch if greater than. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bc 12,4∗cr_field+1,target**		28-21
bgta		*Extended mnemonic for* **bca 12,4∗cr_field+1,target**		
bgtl		*Extended mnemonic for* **bcl 12,4∗cr_field+1,target**	(LR) ← CIA + 4	
bgtla		*Extended mnemonic for* **bcla 12,4∗cr_field+1,target**	(LR) ← CIA + 4	
bgtctr	[cr_field]	Branch if greater than to address in CTR. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bcctr 12,4∗cr_field+1**		28-27
bgtctrl		*Extended mnemonic for* **bcctrl 12,4∗cr_field+1**	(LR) ← CIA + 4	
bgtlr	[cr_field]	Branch if greater than to address in LR. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bclr 12,4∗cr_field+1**		28-31
bgtlrl		*Extended mnemonic for* **bclrl 12,4∗cr_field+1**	(LR) ← CIA + 4	

Table B-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
ble	[cr_field], target	Branch if less than or equal. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bc 4,4∗cr_field+1,target**		28-21
blea		*Extended mnemonic for* **bca 4,4∗cr_field+1,target**		
blel		*Extended mnemonic for* **bcl 4,4∗cr_field+1,target**	(LR) ← CIA + 4	
blela		*Extended mnemonic for* **bcla 4,4∗cr_field+1,target**	(LR) ← CIA + 4	
blectr	[cr_field]	Branch if less than or equal to address in CTR. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bcctr 4,4∗cr_field+1**		28-27
blectrl		*Extended mnemonic for* **bcctrl 4,4∗cr_field+1**	(LR) ← CIA + 4	
blelr	[cr_field]	Branch if less than or equal to address in LR. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bclr 4,4∗cr_field+1**		28-31
blelrl		*Extended mnemonic for* **bclrl 4,4∗cr_field+1**	(LR) ← CIA + 4	
blr		Branch unconditionally to address in LR. *Extended mnemonic for* **bclr 20,0**		28-31
blrl		*Extended mnemonic for* **bclrl 20,0**	(LR) ← CIA + 4	
blt	[cr_field], target	Branch if less than. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bc 12,4∗cr_field+0,target**		28-21
blta		*Extended mnemonic for* **bca 12,4∗cr_field+0,target**		
bltl		*Extended mnemonic for* **bcl 12,4∗cr_field+0,target**	(LR) ← CIA + 4	
bltla		*Extended mnemonic for* **bcla 12,4∗cr_field+0,target**	(LR) ← CIA + 4	
bltctr	[cr_field]	Branch if less than to address in CTR. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bcctr 12,4∗cr_field+0**		28-27
bltctrl		*Extended mnemonic for* **bcctrl 12,4∗cr_field+0**	(LR) ← CIA + 4	

Table B-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
bltlr	[cr_field]	Branch if less than to address in LR. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bclr 12,4∗cr_field+0**		28-31
bltlrl		*Extended mnemonic for* **bclrl 12,4∗cr_field+0**	(LR) ← CIA + 4	
bne	[cr_field], target	Branch if not equal. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bc 4,4∗cr_field+2,target**		28-21
bnea		*Extended mnemonic for* **bca 4,4∗cr_field+2,target**		
bnel		*Extended mnemonic for* **bcl 4,4∗cr_field+2,target**	(LR) ← CIA + 4	
bnela		*Extended mnemonic for* **bcla 4,4∗cr_field+2,target**	(LR) ← CIA + 4	
bnectr	[cr_field]	Branch if not equal to address in CTR. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bcctr 4,4∗cr_field+2**		28-27
bnectrl		*Extended mnemonic for* **bcctrl 4,4∗cr_field+2**	(LR) ← CIA + 4	
bnelr	[cr_field]	Branch if not equal to address in LR. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bclr 4,4∗cr_field+2**		28-31
bnelrl		*Extended mnemonic for* **bclrl 4,4∗cr_field+2**	(LR) ← CIA + 4	
bng	[cr_field], target	Branch if not greater than. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bc 4,4∗cr_field+1,target**		28-21
bnga		*Extended mnemonic for* **bca 4,4∗cr_field+1,target**		
bngl		*Extended mnemonic for* **bcl 4,4∗cr_field+1,target**	(LR) ← CIA + 4	
bngla		*Extended mnemonic for* **bcla 4,4∗cr_field+1,target**	(LR) ← CIA + 4	
bngctr	[cr_field]	Branch if not greater than to address in CTR. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bcctr 4,4∗cr_field+1**		28-27
bngctrl		*Extended mnemonic for* **bcctrl 4,4∗cr_field+1**	(LR) ← CIA + 4	

Table B-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
bnglr	[cr_field]	Branch if not greater than to address in LR. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bclr 4,4∗cr_field+1**		28-31
bnglrl		*Extended mnemonic for* **bclrl 4,4∗cr_field+1**	(LR) ← CIA + 4	
bnl	[cr_field], target	Branch if not less than. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bc 4,4∗cr_field+0,target**		28-21
bnla		*Extended mnemonic for* **bca 4,4∗cr_field+0,target**		
bnll		*Extended mnemonic for* **bcl 4,4∗cr_field+0,target**	(LR) ← CIA + 4	
bnlla		*Extended mnemonic for* **bcla 4,4∗cr_field+0,target**	(LR) ← CIA + 4	
bnlctr	[cr_field]	Branch if not less than to address in CTR. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bcctr 4,4∗cr_field+0**		28-27
bnlctrl		*Extended mnemonic for* **bcctrl 4,4∗cr_field+0**	(LR) ← CIA + 4	
bnllr	[cr_field]	Branch if not less than to address in LR. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bclr 4,4∗cr_field+0**		28-31
bnllrl		*Extended mnemonic for* **bclrl 4,4∗cr_field+0**	(LR) ← CIA + 4	
bns	[cr_field], target	Branch if not summary overflow. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bc 4,4∗cr_field+3,target**		28-21
bnsa		*Extended mnemonic for* **bca 4,4∗cr_field+3,target**		
bnsl		*Extended mnemonic for* **bcl 4,4∗cr_field+3,target**	(LR) ← CIA + 4	
bnsla		*Extended mnemonic for* **bcla 4,4∗cr_field+3,target**	(LR) ← CIA + 4	
bnsctr	[cr_field]	Branch if not summary overflow to address in CTR. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bcctr 4,4∗cr_field+3**		28-27
bnsctrl		*Extended mnemonic for* **bcctrl 4,4∗cr_field+3**	(LR) ← CIA + 4	

Table B-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
bnslr	[cr_field]	Branch if not summary overflow to address in LR. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bclr 4,4∗cr_field+3**		28-31
bnslrl		*Extended mnemonic for* **bclrl 4,4∗cr_field+3**	(LR) ← CIA + 4	
bnu	[cr_field], target	Branch if not unordered. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bc 4,4∗cr_field+3,target**		28-21
bnua		*Extended mnemonic for* **bca 4,4∗cr_field+3,target**		
bnul		*Extended mnemonic for* **bcl 4,4∗cr_field+3,target**	(LR) ← CIA + 4	
bnula		*Extended mnemonic* for **bcla 4,4∗cr_field+3,target**	(LR) ← CIA + 4	
bnuctr	[cr_field]	Branch if not unordered to address in CTR. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bcctr 4,4∗cr_field+3**		28-27
bnuctrl		*Extended mnemonic for* **bcctrl 4,4∗cr_field+3**	(LR) ← CIA + 4	
bnulr	[cr_field]	Branch if not unordered to address in LR. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bclr 4,4∗cr_field+3**		28-31
bnulrl		*Extended mnemonic for* **bclrl 4,4∗cr_field+3**	(LR) ← CIA + 4	
bso	[cr_field], target	Branch if summary overflow. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bc 12,4∗cr_field+3,target**		28-21
bsoa		*Extended mnemonic for* **bca 12,4∗cr_field+3,target**		
bsol		*Extended mnemonic for* **bcl 12,4∗cr_field+3,target**	(LR) ← CIA + 4	
bsola		*Extended mnemonic for* **bcla 12,4∗cr_field+3,target**	(LR) ← CIA + 4	
bsoctr	[cr_field]	Branch if summary overflow to address in CTR. UseCR[CR0] if cr_field is omitted. *Extended mnemonic for* **bcctr 12,4∗cr_field+3**		28-27
bsoctrl		*Extended mnemonic for* **bcctrl 12,4∗cr_field+3**	(LR) ← CIA + 4	

Table B-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
bsolr	[cr_field]	Branch if summary overflow to address in LR. UseCR[CR0] if cr_field is omitted. *Extended mnemonic* for **bclr 12,4∗cr_field+3**		28-31
bsolrl		*Extended mnemonic* for **bclrl 12,4∗cr_field+3**	(LR) ← CIA + 4	
bt	cr_bit, target	Branch if CR_{cr_bit} = 1. *Extended mnemonic for* **bc 12,cr_bit,target**		28-21
bta		*Extended mnemonic for* **bca 12,cr_bit,target**		
btl		*Extended mnemonic for* **bcl 12,cr_bit,target**	(LR) ← CIA + 4	
btla		*Extended mnemonic for* **bcla 12,cr_bit,target**	(LR) ← CIA + 4	
btctr	cr_bit	Branch if CR_{cr_bit} = 1 to address in CTR. *Extended mnemonic for* **bcctr 12,cr_bit**		28-27
btctrl		*Extended mnemonic for* **bcctrl 12,cr_bit**	(LR) ← CIA + 4	
btlr	cr_bit	Branch if CR_{cr_bit} = 1, to address in LR. *Extended mnemonic for* **bclr 12,cr_bit**		28-31
btlrl		*Extended mnemonic for* **bclrl 12,cr_bit**	(LR) ← CIA + 4	
bun	[cr_field], target	Branch if unordered. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bc 12,4∗cr_field+3,target**		28-21
buna		*Extended mnemonic for* **bca 12,4∗cr_field+3,target**		
bunl		*Extended mnemonic for* **bcl 12,4∗cr_field+3,target**	(LR) ← CIA + 4	
bunla		*Extended mnemonic for* **bcla 12,4∗cr_field+3,target**	(LR) ← CIA + 4	
bunctr	[cr_field]	Branch if unordered to address in CTR. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bcctr 12,4∗cr_field+3**		28-27
bunctrl		*Extended mnemonic for* **bcctrl 12,4∗cr_field+3**	(LR) ← CIA + 4	

Table B-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
bunlr	[cr_field]	Branch if unordered, to address in LR. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bclr 12,4∗cr_field+3**		28-31
bunlrl		*Extended mnemonic for* **bclrl 12,4∗cr_field+3**	(LR) ← CIA + 4	
clrlwi	RA, RS, n	Clear left immediate. (n < 32) $(RA)_{0:n-1} \leftarrow {}^{n}0$ *Extended mnemonic for* **rlwinm RA,RS,0,n,31**		28-149
clrlwi.		*Extended mnemonic for* **rlwinm. RA,RS,0,n,31**	CR[CR0]	
clrlslwi	RA, RS, b, n	Clear left and shift left immediate. (n ≤ b < 32) $(RA)_{b-n:31-n} \leftarrow (RS)_{b:31}$ $(RA)_{32-n:31} \leftarrow {}^{n}0$ $(RA)_{0:b-n-1} \leftarrow {}^{b-n}0$ *Extended mnemonic for* **rlwinm RA,RS,n,b−n,31−n**		28-149
clrlslwi.		*Extended mnemonic for* **rlwinm. RA,RS,n,b−n,31−n**	CR[CR0]	
clrrwi	RA, RS, n	Clear right immediate. (n < 32) $(RA)_{32-n:31} \leftarrow {}^{n}0$ *Extended mnemonic for* **rlwinm RA,RS,0,0,31−n**		28-149
clrrwi.		*Extended mnemonic for* **rlwinm. RA,RS,0,0,31−n**	CR[CR0]	
cmp	BF, 0, RA, RB	Compare (RA) to (RB), signed. Results in CR[CRn], where n = BF.		28-35
cmpi	BF, 0, RA, IM	Compare (RA) to EXTS(IM), signed. Results in CR[CRn], where n = BF.		28-36
cmpl	BF, 0, RA, RB	Compare (RA) to (RB), unsigned. Results in CR[CRn], where n = BF.		28-37
cmpli	BF, 0, RA, IM	Compare (RA) to ($^{16}0$ || IM), unsigned. Results in CR[CRn], where n = BF.		28-38
cmplw	[BF,] RA, RB	Compare Logical Word. UseCR[CR0] if BF is omitted. *Extended mnemonic for* **cmpl BF,0,RA,RB**		28-37
cmplwi	[BF,] RA, IM	Compare Logical Word Immediate. UseCR[CR0] if BF is omitted. *Extended mnemonic for* **cmpli BF,0,RA,IM**		28-38

Table B-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
cmpw	[BF,] RA, RB	Compare Word. UseCR[CR0] if BF is omitted. *Extended mnemonic for* **cmp BF,0,RA,RB**		28-35
cmpwi	[BF,] RA, IM	Compare Word Immediate. UseCR[CR0] if BF is omitted. *Extended mnemonic for* **cmpi BF,0,RA,IM**		28-36
cntlzw	RA, RS	Count leading zeros in RS. Place result in RA.		28-39
cntlzw.			CR[CR0]	
crand	BT, BA, BB	AND bit (CR_{BA}) with (CR_{BB}). Place result in CR_{BT}.		28-40
crandc	BT, BA, BB	AND bit (CR_{BA}) with $\neg$(CR_{BB}). Place result in CR_{BT}.		28-41
crclr	bx	Condition register clear. *Extended mnemonic for* **crxor bx,bx,bx**		28-47
creqv	BT, BA, BB	Equivalence of bit CR_{BA} with CR_{BB}. $CR_{BT} \leftarrow \neg(CR_{BA} \oplus CR_{BB})$		28-42
crmove	bx, by	Condition register move. *Extended mnemonic for* **cror bx,by,by**		28-45
crnand	BT, BA, BB	NAND bit (CR_{BA}) with (CR_{BB}). Place result in CR_{BT}.		28-43
crnor	BT, BA, BB	NOR bit (CR_{BA}) with (CR_{BB}). Place result in CR_{BT}.		28-44
crnot	bx, by	Condition register not. *Extended mnemonic for* **crnor bx,by,by**		28-44
cror	BT, BA, BB	OR bit (CR_{BA}) with (CR_{BB}). Place result in CR_{BT}.		28-45
crorc	BT, BA, BB	OR bit (CR_{BA}) with $\neg$(CR_{BB}). Place result in CR_{BT}.		28-46
crset	bx	Condition register set. *Extended mnemonic for* **creqv bx,bx,bx**		28-42
crxor	BT, BA, BB	XOR bit (CR_{BA}) with (CR_{BB}). Place result in CR_{BT}.		28-47
dcba	RA, RB	Treated as a no-op.		28-48
dcbf	RA, RB	Flush (store, then invalidate) the data cache block which contains the effective address (RA|0) + (RB).		28-49
dcbi	RA, RB	Invalidate the data cache block which contains the effective address (RA|0) + (RB).		28-50
dcbst	RA, RB	Store the data cache block which contains the effective address (RA|0) + (RB).		28-51

Table B-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
dcbt	RA, RB	Load the data cache block which contains the effective address (RA|0) + (RB).		28-52
dcbtst	RA,RB	Load the data cache block which contains the effective address (RA|0) + (RB).		28-53
dcbz	RA, RB	Zero the data cache block which contains the effective address (RA|0) + (RB).		28-54
dccci	RA, RB	Invalidate the data cache array.		28-55
dcread	RT, RA, RB	Read tag and data information from the data cache line selected using effective address bits 17:26. The effective address is calculated by (RA|0) + (RB). Place the data word selected by effective address bits 27:29 in GPR RT; place the tag information in DCDBTRH and DCDBTRL.		28-56
divw	RT, RA, RB	Divide (RA) by (RB), signed. Place result in RT.		28-58
divw.			CR[CR0]	
divwo			XER[SO, OV]	
divwo.			CR[CR0] XER[SO, OV]	
divwu	RT, RA, RB	Divide (RA) by (RB), unsigned. Place result in RT.		28-59
divwu.			CR[CR0]	
divwuo			XER[SO, OV]	
divwuo.			CR[CR0] XER[SO, OV]	

dlmzb	RA, RS, RB	d ← (RS) || (RB)
		i, x, y ← 0
		do while (x < 8) ∧ (y = 0)
		  x ← x + 1
		  if $d_{i:i+7}$ = 0 then
		  y ← 1
		  else
		  i ← i + 8
		(RA) ← x
		XER[TBC] ← x
		if Rc = 1 then
		  $CR[CR0]_3$ ←XER[SO]
		  if y = 1 then
		  if x < 5 then
		  $CR[CR0]_{0:2}$ ← 0b010
		  else
		  $CR[CR0]_{0:2}$ ← 0b100
		  else
		  $CR[CR0]_{0:2}$ ← 0b001
			XER[TBC], RA	28-60

dlmzb.			XER[TBC], RA, CR[CR0]	
eqv	RA, RS, RB	Equivalence of (RS) with (RB). (RA) ← ¬((RS) ⊕ (RB))		28-61
eqv.			CR[CR0]	

Table B-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
extlwi	RA, RS, n, b	Extract and left justify immediate. (n > 0) $(RA)_{0:n-1} \leftarrow (RS)_{b:b+n-1}$ $(RA)_{n:31} \leftarrow {}^{32-n}0$ *Extended mnemonic for* **rlwinm RA,RS,b,0,n–1**		28-149
extlwi.		*Extended mnemonic for* **rlwinm. RA,RS,b,0,n–1**	CR[CR0]	
extrwi	RA, RS, n, b	Extract and right justify immediate. (n > 0) $(RA)_{32-n:31} \leftarrow (RS)_{b:b+n-1}$ $(RA)_{0:31-n} \leftarrow {}^{32-n}0$ *Extended mnemonic for* **rlwinm RA,RS,b+n,32–n,31**		28-149
extrwi.		*Extended mnemonic for* **rlwinm. RA,RS,b+n,32–n,31**	CR[CR0]	
extsb	RA, RS	Extend the sign of byte $(RS)_{24:31}$. Place the result in RA.		28-62
extsb.			CR[CR0]	
extsh	RA, RS	Extend the sign of halfword $(RS)_{16:31}$. Place the result in RA.		28-63
extsh.			CR[CR0]	
icbi	RA, RB	Invalidate the instruction cache block which contains the effective address (RA|0) + (RB).		28-64
icbt	RA, RB	Load the instruction cache block which contains the effective address (RA|0) + (RB).		28-62
iccci	RA, RB	Invalidate the instruction cache array.		28-67
icread	RA, RB	Read tag and data information from the instruction cache line selected using effective address bits 17:26. The effective address is calculated by (RA|0) + (RB). Place the instruction selected by effective address bits 27:29 in ICDBDR; place the tag information in ICDBTRH and ICDBTRL.		28-68
inslwi	RA, RS, n, b	Insert from left immediate. (n > 0) $(RA)_{b:b+n-1} \leftarrow (RS)_{0:n-1}$ *Extended mnemonic for* **rlwimi RA,RS,32–b,b,b+n–1**		28-148
inslwi.		*Extended mnemonic for* **rlwimi. RA,RS,32–b,b,b+n–1**	CR[CR0]	
insrwi	RA, RS, n, b	Insert from right immediate. (n > 0) $(RA)_{b:b+n-1} \leftarrow (RS)_{32-n:31}$ *Extended mnemonic for* **rlwimi RA,RS,32–b–n,b,b+n–1**		28-148
insrwi.		*Extended mnemonic for* **rlwimi. RA,RS,32–b–n,b,b+n–1**	CR[CR0]	
isync		Synchronize execution context by flushing the prefetch queue.		28-70

Table B-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
la	RT, D(RA)	Load address. (RA ≠ 0) D is an offset from a base address that is assumed to be (RA). (RT) ← (RA) + EXTS(D) *Extended mnemonic for* **addi RT,RA,D**		28-10
lbz	RT, D(RA)	Load byte from EA = (RA|0) + EXTS(D) and pad left with zeroes, (RT) ← $^{24}0$ || MS(EA,1).		28-71
lbzu	RT, D(RA)	Load byte from EA = (RA|0) + EXTS(D) and pad left with zeroes, (RT) ← $^{24}0$ || MS(EA,1). Update the base address, (RA) ← EA.		28-72
lbzux	RT, RA, RB	Load byte from EA = (RA|0) + (RB) and pad left with zeroes, (RT) ← $^{24}0$ || MS(EA,1). Update the base address, (RA) ← EA.		28-73
lbzx	RT, RA, RB	Load byte from EA = (RA|0) + (RB) and pad left with zeroes, (RT) ← $^{24}0$ || MS(EA,1).		28-74
lha	RT, D(RA)	Load halfword from EA = (RA|0) + EXTS(D) and sign extend, (RT) ← EXTS(MS(EA,2)).		28-75
lhau	RT, D(RA)	Load halfword from EA = (RA|0) + EXTS(D) and sign extend, (RT) ← EXTS(MS(EA,2)). Update the base address, (RA) ← EA.		28-76
lhaux	RT, RA, RB	Load halfword from EA = (RA|0) + (RB) and sign extend, (RT) ← EXTS(MS(EA,2)). Update the base address, (RA) ← EA.		28-77
lhax	RT, RA, RB	Load halfword from EA = (RA|0) + (RB) and sign extend, (RT) ← EXTS(MS(EA,2)).		28-78
lhbrx	RT, RA, RB	Load halfword from EA = (RA|0) + (RB), then reverse byte order and pad left with zeroes, (RT) ← $^{16}0$ || MS(EA+1,1) || MS(EA,1).		28-79
lhz	RT, D(RA)	Load halfword from EA = (RA|0) + EXTS(D) and pad left with zeroes, (RT) ← $^{16}0$ || MS(EA,2).		28-80

Table B-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
lhzu	RT, D(RA)	Load halfword from EA = (RA|0) + EXTS(D) and pad left with zeroes, (RT) ← $^{16}0$ || MS(EA,2). Update the base address, (RA) ← EA.		28-81
lhzux	RT, RA, RB	Load halfword from EA = (RA|0) + (RB) and pad left with zeroes, (RT) ← $^{16}0$ || MS(EA,2). Update the base address, (RA) ← EA.		28-82
lhzx	RT, RA, RB	Load halfword from EA = (RA|0) + (RB) and pad left with zeroes, (RT) ← $^{16}0$ || MS(EA,2).		28-83
li	RT, IM	Load immediate. (RT) ← EXTS(IM) *Extended mnemonic for* **addi RT,0,value**		28-10
lis	RT, IM	Load immediate shifted. (RT) ← (IM || $^{16}0$) *Extended mnemonic for* **addis RT,0,value**		28-13
lmw	RT, D(RA)	Load multiple words starting from EA = (RA|0) + EXTS(D). Place into consecutive registers RT through GPR(31). RA is not altered unless RA = GPR(31).		28-84
lswi	RT, RA, NB	Load consecutive bytes from EA=(RA|0). Number of bytes n=32 if NB=0, else n=NB. Stack bytes into words in CEIL(n/4) consecutive registers starting with RT, to R_{FINAL} ← ((RT + CEIL(n/4) – 1) % 32). GPR(0) is consecutive to GPR(31). RA is not altered unless RA = R_{FINAL}.		28-85
lswx	RT, RA, RB	Load consecutive bytes from EA=(RA|0)+(RB). Number of bytes n=XER[TBC]. Stack bytes into words in CEIL(n/4) consecutive registers starting with RT, to R_{FINAL} ← ((RT + CEIL(n/4) – 1) % 32). GPR(0) is consecutive to GPR(31). RA is not altered unless RA = R_{FINAL}. RB is not altered unless RB = R_{FINAL}. If n=0, content of RT is undefined.		28-87
lwarx	RT, RA, RB	Load word from EA = (RA|0) + (RB) and place in RT, (RT) ← MS(EA,4). Set the Reservation bit.		28-89
lwbrx	RT, RA, RB	Load word from EA = (RA|0) + (RB) then reverse byte order, (RT) ← MS(EA+3,1) || MS(EA+2,1) || MS(EA+1,1) || MS(EA,1).		28-90

Table B-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
lwz	RT, D(RA)	Load word from EA = (RA|0) + EXTS(D) and place in RT, (RT) ← MS(EA,4).		28-91
lwzu	RT, D(RA)	Load word from EA = (RA|0) + EXTS(D) and place in RT, (RT) ← MS(EA,4). Update the base address, (RA) ← EA.		28-92
lwzux	RT, RA, RB	Load word from EA = (RA|0) + (RB) and place in RT, (RT) ← MS(EA,4). Update the base address, (RA) ← EA.		28-93
lwzx	RT, RA, RB	Load word from EA = (RA|0) + (RB) and place in RT, (RT) ← MS(EA,4).		28-94
macchw	RT, RA, RB	$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{0:15}$ $temp_{0:32} \leftarrow prod_{0:31} + (RT)$ $(RT) \leftarrow temp_{1:32}$		28-95
macchw.			CR[CR0]	
macchwo			XER[SO, OV]	
macchwo.			CR[CR0] XER[SO, OV]	
macchwu	RT, RA, RB	$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{0:15}$ $temp_{0:32} \leftarrow prod_{0:31} + (RT)$ $(RT) \leftarrow temp_{1:32}$		28-98
macchwu.			CR[CR0]	
macchwuo			XER[SO, OV]	
macchwuo.			CR[CR0] XER[SO, OV]	
macchws	RT, RA, RB	$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{0:15}$ $temp_{0:32} \leftarrow prod_{0:31} + (RT)$ if $((prod_0 = RT_0) \wedge (RT_0 \neq temp_1))$ then $(RT) \leftarrow (RT_0 \vee {}^{31}(\neg RT_0))$ else $(RT) \leftarrow temp_{1:32}$		28-96
macchws.			CR[CR0]	
macchwso			XER[SO, OV]	
macchwso.			CR[CR0] XER[SO, OV]	
macchwsu	RT, RA, RB	$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{0:15}$ $temp_{0:32} \leftarrow prod_{0:31} + (RT)$ $(RT) \leftarrow (temp_{1:32} \vee {}^{31}temp_0)$		28-97
macchwsu.			CR[CR0]	
macchwsuo			XER[SO, OV]	
macchwsuo.			CR[CR0] XER[SO, OV]	
machhw	RT, RA, RB	$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{0:15}$ $temp_{0:32} \leftarrow prod_{0:31} + (RT)$ $(RT) \leftarrow temp_{1:32}$		28-99
machhw.			CR[CR0]	
machhwo			XER[SO, OV]	
machhwo.			CR[CR0] XER[SO, OV]	

Table B-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
machhwu	RT, RA, RB	$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{0:15}$ $temp_{0:32} \leftarrow prod_{0:31} + (RT)$ $(RT) \leftarrow temp_{1:32}$		28-102
machhwu.			CR[CR0]	
machhwuo			XER[SO, OV]	
machhwuo.			CR[CR0] XER[SO, OV]	
machhws	RT, RA, RB	$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{0:15}$ $temp_{0:32} \leftarrow prod_{0:31} + (RT)$ if $((prod_0 = RT_0) \wedge (RT_0 \neq temp_1))$ then $(RT) \leftarrow (RT_0 \vee {}^{31}(\neg RT_0))$ else $(RT) \leftarrow temp_{1:32}$		28-100
machhws.			CR[CR0]	
machhwso			XER[SO, OV]	
machhwso.			CR[CR0] XER[SO, OV]	
machhwsu	RT, RA, RB	$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{0:15}$ $temp_{0:32} \leftarrow prod_{0:31} + (RT)$ $(RT) \leftarrow (temp_{1:32} \vee {}^{31}temp_0)$		28-101
machhwsu.			CR[CR0]	
machhwsuo			XER[SO, OV]	
machhwsuo.			CR[CR0] XER[SO, OV]	
maclhw	RT, RA, RB	$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{0:15}$ $temp_{0:32} \leftarrow prod_{0:31} + (RT)$ $(RT) \leftarrow temp_{1:32}$		28-103
maclhw.			CR[CR0]	
maclhwo			XER[SO, OV]	
maclhwo.			CR[CR0] XER[SO, OV]	
maclhwu	RT, RA, RB	$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{0:15}$ $temp_{0:32} \leftarrow prod_{0:31} + (RT)$ $(RT) \leftarrow temp_{1:32}$		28-106
maclhwu.			CR[CR0]	
maclhwuo			XER[SO, OV]	
maclhwuo.			CR[CR0] XER[SO, OV]	
maclhws	RT, RA, RB	$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{0:15}$ $temp_{0:32} \leftarrow prod_{0:31} + (RT)$ if $((prod_0 = RT_0) \wedge (RT_0 \neq temp_1))$ then $(RT) \leftarrow (RT_0 \vee {}^{31}(\neg RT_0))$ else $(RT) \leftarrow temp_{1:32}$		28-104
maclhws.			CR[CR0]	
maclhwso			XER[SO, OV]	
maclhwso.			CR[CR0] XER[SO, OV]	
maclhwsu	RT, RA, RB	$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{0:15}$ $temp_{0:32} \leftarrow prod_{0:31} + (RT)$ $(RT) \leftarrow (temp_{1:32} \vee {}^{31}temp_0)$		28-105
maclhwsu.			CR[CR0]	
maclhwsuo			XER[SO, OV]	
maclhwsuo.			CR[CR0] XER[SO, OV]	
mbar		Storage synchronization. All loads and stores that precede the **mbar** instruction complete before any loads and stores that follow the instruction access main storage.		28-107
mcrf	BF, BFA	Move CR field, $(CR[CRn]) \leftarrow (CR[CRm])$ where $m \leftarrow BFA$ and $n \leftarrow BF$		28-108

Table B-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
mcrxr	BF	Move XER[0:3] into field CRn, where n←BF. CR[CRn] ← (XER[SO, OV, CA]) (XER[SO, OV, CA]) ← $^{3}0$		28-109
mfcr	RT	Move from CR to RT, (RT) ← (CR).		28-110
mfdcr	RT, DCRN	Move from DCR to RT, (RT) ← (DCR(DCRN)).		28-111
mfmsr	RT	Move from MSR to RT, (RT) ← (MSR).		28-112

Table B-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
mfccr0 **mfctr** **mfdac1** **mfdac2** **mfdbcr0** **mfdbcr1** **mfdbsr** **mfdccr** **mfdcwr** **mfdvc1** **mfdvc2** **mfesr** **mfiac1** **mfiac2** **mfiac3** **mfiac4** **mficcr** **mficdbdr** **mfivpr** **mflr** **mfpid** **mfpit** **mfpvr** **mfsgr** **mfsprg0** **mfsprg1** **mfsprg2** **mfsprg3** **mfsprg4** **mfsprg5** **mfsprg6** **mfsprg7** **mfsrr0** **mfsrr1** **mfsrr2** **mfsrr3** **mfsu0r** **mftbl** **mftbu** **mftcr** **mftsr** **mfxer**	RT	Move from special purpose register (SPR) SPRN. *Extended mnemonic for* **mfspr RT,SPRN** See Table 29-2, "Special Purpose Registers Sorted by SPR Number," on page 29-4 for listing of valid SPRN values.		28-113
mfspr	RT, SPRN	Move from SPR to RT, (RT) ← (SPR(SPRN)).		28-113

Table B-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
mr	RT, RS	Move register. (RT) ← (RS) *Extended mnemonic for* **or RT,RS,RS**		28-142
mr.		*Extended mnemonic for* **or. RT,RS,RS**	CR[CR0]	
msync		Synchronization. All instructions that precede **msync** complete before any instructions that follow **msync** begin. When **msync** completes, all storage accesses initiated prior to **msync** will have completed.		28-116
mtcr	RS	Move to Condition Register. *Extended mnemonic for* **mtcrf 0xFF,RS**		28-117
mtcrf	FXM, RS	Move some or all of the contents of RS into CR as specified by FXM field, mask ← $^{4}(FXM_0)$ || $^{4}(FXM_1)$ || ... || $^{4}(FXM_6)$ || $^{4}(FXM_7)$. (CR)←((RS) ∧ mask) ∨ (CR) ∧ ¬mask).		28-117
mtdcr	DCRN, RS	Move to DCR from RS, (DCR(DCRN)) ← (RS).		28-118
mtmsr	RS	Move to MSR from RS, (MSR) ← (RS).		28-119

Table B-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
mtccr0 **mtctr** **mtdac1** **mtdac2** **mtdbcr0** **mtdbcr1** **mtdbsr** **mtdccr** **mtdcwr** **mtdvc1** **mtdvc2** **mtesr** **mtiac1** **mtiac2** **mtiac3** **mtiac4** **mticcr** **mticdbdr** **mtivpr** **mtlr** **mtpid** **mtpit** **mtpvr** **mtsgr** **mtsprg0** **mtsprg1** **mtsprg2** **mtsprg3** **mtsprg4** **mtsprg5** **mtsprg6** **mtsprg7** **mtsrr0** **mtsrr1** **mtsrr2** **mtsrr3** **mtsu0r** **mttbl** **mttbu** **mttcr** **mttsr** **mtxer** **mtzpr**	RS	Move to SPR SPRN. *Extended mnemonic for* **mtspr SPRN,RS** See Table 29-2, "Special Purpose Registers Sorted by SPR Number," on page 29-4 for listing of valid SPRN values.		28-120
mtspr	SPRN, RS	Move to SPR from RS, (SPR(SPRN)) $\leftarrow$ (RS).		28-120
mulchw	RT, RA, RB	$(RT)_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{0:15}$ (signed)		28-123
mulchw.			CR[CR0]	
mulchwu	RT, RA, RB	$(RT)_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{0:15}$ (unsigned)		28-124
mulchwu.			CR[CR0]	

Table B-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
mulhhw	RT, RA, RB	$(RT)_{0:31} \leftarrow (RA)_{0:15} \times (RB_{0:15}$ (signed)		28-125
mulhhw.			CR[CR0]	
mulhhwu	RT, RA, RB	$(RT)_{0:31} \leftarrow (RA)_{0:15} \times (RB)_{0:15}$ (unsigned)		28-126
mulhhwu.			CR[CR0]	
mulhw	RT, RA, RB	Multiply (RA) and (RB), signed. Place high-order result in RT. $prod_{0:63} \leftarrow (RA) \times (RB)$ (signed). $(RT) \leftarrow prod_{0:31}$.		28-127
mulhw.			CR[CR0]	
mulhwu	RT, RA, RB	Multiply (RA) and (RB), unsigned. Place high-order result in RT. $prod_{0:63} \leftarrow (RA) \times (RB)$ (unsigned). $(RT) \leftarrow prod_{0:31}$.		28-128
mulhwu.			CR[CR0]	
mullhw	RT, RA, RB	$(RT)_{0:31} \leftarrow (RA)_{16:31} \times (RB_{16:31}$ (signed)		28-129
mullhw.			CR[CR0]	
mullhwu	RT, RA, RB	$(RT)_{16:31} \leftarrow (RA)_{0:15} \times (RB)_{16:31}$ (unsigned)		28-130
mullhwu.			CR[CR0]	
mulli	RT, RA, IM	Multiply (RA) and IM, signed. Place low-order result in RT. $prod_{0:47} \leftarrow (RA) \times$ IM (signed) $(RT) \leftarrow prod_{16:47}$		28-131
mullw	RT, RA, RB	Multiply (RA) and (RB), signed. Place low-order result in RT. $prod_{0:63} \leftarrow (RA) \times (RB)$ (signed). $(RT) \leftarrow prod_{32:63}$.		28-132
mullw.			CR[CR0]	
mullwo			XER[SO, OV]	
mullwo.			CR[CR0] XER[SO, OV]	
nand	RA, RS, RB	NAND (RS) with (RB). Place result in RA.		28-133
nand.			CR[CR0]	
neg	RT, RA	Negative (two's complement) of RA. $(RT) \leftarrow \neg(RA) + 1$		28-134
neg.			CR[CR0]	
nego			XER[SO, OV]	
nego.			CR[CR0] XER[SO, OV]	
nmacchw	RT, RA, RB	$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{0:15}$ $temp_{0:32} \leftarrow -prod_{0:31} + (RT)$ $(RT) \leftarrow temp_{1:32}$		28-135
nmacchw.			CR[CR0]	
nmacchwo			XER[SO, OV]	
nmacchwo.			CR[CR0] XER[SO, OV]	
nmacchws	RT, RA, RB	$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{0:15}$ $temp_{0:32} \leftarrow -prod_{0:31} + (RT)$ if $((prod_0 = RT_0) \wedge (RT_0 \neq temp_1))$ then $(RT) \leftarrow (RT_0 \vee {}^{31}(\neg RT_0))$ else $(RT) \leftarrow temp_{1:32}$		28-136
nmacchws.			CR[CR0]	
nmacchwso			XER[SO, OV]	
nmacchwso.			CR[CR0] XER[SO, OV]	

Table B-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
nmachhw	RT, RA, RB	$prod_{0:31} \leftarrow (RA)_{0:15} \times (RB)_{0:15}$ $temp_{0:32} \leftarrow -prod_{0:31} + (RT)$ $(RT) \leftarrow temp_{1:32}$		28-137
nmachhw.			CR[CR0]	
nmachhwo			XER[SO, OV]	
nmachhwo.			CR[CR0] XER[SO, OV]	
nmachhws	RT, RA, RB	$prod_{0:31} \leftarrow (RA)_{0:15} \times (RB)_{0:15}$ $temp_{0:32} \leftarrow -prod_{0:31} + (RT)$ if $((prod_0 = RT_0) \wedge (RT_0 \neq temp_1))$ then $(RT) \leftarrow (RT_0 \vee {}^{31}(\neg RT_0))$ else $(RT) \leftarrow temp_{1:32}$		28-138
nmachhws.			CR[CR0]	
nmachhwso			XER[SO, OV]	
nmachhwso.			CR[CR0] XER[SO, OV]	
nmaclhw	RT, RA, RB	$prod_{0:31} \leftarrow (RA_{16:31} \times (RB)_{16:31}$ $temp_{0:32} \leftarrow -prod_{0:31} + (RT)$ $(RT) \leftarrow temp_{1:32}$		28-139
nmaclhw.			CR[CR0]	
nmaclhwo			XER[SO, OV]	
nmaclhwo.			CR[CR0] XER[SO, OV]	
nmaclhws	RT, RA, RB	$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{16:31}$ $temp_{0:32} \leftarrow -prod_{0:31} + (RT)$ if $((prod_0 = RT_0) \wedge (RT_0 \neq temp_1))$ then $(RT) \leftarrow (RT_0 \vee {}^{31}(\neg RT_0))$ else $(RT) \leftarrow temp_{1:32}$		28-140
nmaclhws.			CR[CR0]	
nmaclhwso			XER[SO, OV]	
nmaclhwso.			CR[CR0] XER[SO, OV]	
nop		Preferred no-op, triggers optimizations based on no-ops. *Extended mnemonic for* **ori 0,0,0**		28-144
nor	RA, RS, RB	NOR (RS) with (RB). Place result in RA.		28-141
nor.			CR[CR0]	
not	RA, RS	Complement register. $(RA) \leftarrow \neg(RS)$ *Extended mnemonic for* **nor RA,RS,RS**		28-141
not.		*Extended mnemonic for* **nor. RA,RS,RS**	CR[CR0]	
or	RA, RS, RB	OR (RS) with (RB). Place result in RA.		28-142
or.			CR[CR0]	
orc	RA, RS, RB	OR (RS) with ¬(RB). Place result in RA.		28-143
orc.			CR[CR0]	
ori	RA, RS, IM	OR (RS) with $({}^{16}0 \parallel IM)$. Place result in RA.		28-144
oris	RA, RS, IM	OR (RS) with $(IM \parallel {}^{16}0)$. Place result in RA.		28-145
rfci		Return from critical interrupt $(PC) \leftarrow (SRR2)$. $(MSR) \leftarrow (SRR3)$.		28-146

Table B-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
rfi		Return from interrupt. (PC) ← (SRR0). (MSR) ← (SRR1).		28-147
rlwimi	RA, RS, SH, MB, ME	Rotate left word immediate, then insert according to mask. r ← ROTL((RS), SH) m ← MASK(MB, ME) (RA) ← (r ∧ m) ∨ ((RA) ∧ ¬m)		28-148
rlwimi.			CR[CR0]	
rlwinm	RA, RS, SH, MB, ME	Rotate left word immediate, then AND with mask. r ← ROTL((RS), SH) m ← MASK(MB, ME) (RA) ← (r ∧ m)		28-149
rlwinm.			CR[CR0]	
rlwnm	RA, RS, RB, MB, ME	Rotate left word, then AND with mask. r ← ROTL((RS), $(RB)_{27:31}$) m ← MASK(MB, ME) (RA) ← (r ∧ m)		28-151
rlwnm.			CR[CR0]	
rotlw	RA, RS, RB	Rotate left. (RA) ← ROTL((RS), $(RB)_{27:31}$) *Extended mnemonic for* **rlwnm RA,RS,RB,0,31**		28-151
rotlw.		*Extended mnemonic for* **rlwnm. RA,RS,RB,0,31**	CR[CR0]	
rotlwi	RA, RS, n	Rotate left immediate. (RA) ← ROTL((RS), n) *Extended mnemonic for* **rlwinm RA,RS,n,0,31**		28-149
rotlwi.		*Extended mnemonic for* **rlwinm. RA,RS,n,0,31**	CR[CR0]	
rotrwi	RA, RS, n	Rotate right immediate. (RA) ← ROTL((RS), 32–n) *Extended mnemonic for* **rlwinm RA,RS,32–n,0,31**		28-149
rotrwi.		*Extended mnemonic for* **rlwinm. RA,RS,32–n,0,31**	CR[CR0]	
sc		System call exception is generated. (SRR1) ← (MSR) (SRR0) ← (PC) PC ← $EVPR_{0:15}$ || 0x0C00 (MSR[WE, PR, EE, PE, DR, IR]) ← 0		28-152
slw	RA, RS, RB	Shift left (RS) by $(RB)_{27:31}$. n ← $(RB)_{27:31}$. r ← ROTL((RS), n). if $(RB)_{26}$ = 0 then m ← MASK(0, 31 – n) else m ← $^{32}0$ (RA) ← r ∧ m.		28-153
slw.			CR[CR0]	

Table B-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
slwi	RA, RS, n	Shift left immediate. (n < 32) $(RA)_{0:31-n} \leftarrow (RS)_{n:31}$ $(RA)_{32-n:31} \leftarrow {}^{n}0$ *Extended mnemonic for* **rlwinm RA,RS,n,0,31–n**		28-149
slwi.		*Extended mnemonic for* **rlwinm. RA,RS,n,0,31–n**	CR[CR0]	
sraw	RA, RS, RB	Shift right algebraic (RS) by $(RB)_{27:31}$. $n \leftarrow (RB)_{27:31}$. $r \leftarrow ROTL((RS), 32 - n)$. if $(RB)_{26} = 0$ then $m \leftarrow MASK(n, 31)$ else $m \leftarrow {}^{32}0$ $s \leftarrow (RS)_0$ $(RA) \leftarrow (r \wedge m) \vee ({}^{32}s \wedge \neg m)$. $XER[CA] \leftarrow s \wedge ((r \wedge \neg m) \neq 0)$.		28-154
sraw.			CR[CR0]	
srawi	RA, RS, SH	Shift right algebraic (RS) by SH. $n \leftarrow SH$. $r \leftarrow ROTL((RS), 32 - n)$. $m \leftarrow MASK(n, 31)$. $s \leftarrow (RS)_0$ $(RA) \leftarrow (r \wedge m) \vee ({}^{32}s \wedge \neg m)$. $XER[CA] \leftarrow s \wedge ((r \wedge \neg m) \neq 0)$.		28-155
srawi.			CR[CR0]	
srw	RA, RS, RB	Shift right (RS) by $(RB)_{27:31}$. $n \leftarrow (RB)_{27:31}$. $r \leftarrow ROTL((RS), 32 - n)$. if $(RB)_{26} = 0$ then $m \leftarrow MASK(n, 31)$ else $m \leftarrow {}^{32}0$ $(RA) \leftarrow r \wedge m$.		28-156
srw.			CR[CR0]	
srwi	RA, RS, n	Shift right immediate. (n < 32) $(RA)_{n:31} \leftarrow (RS)_{0:31-n}$ $(RA)_{0:n-1} \leftarrow {}^{n}0$ *Extended mnemonic for* **rlwinm RA,RS,32–n,n,31**		28-149
srwi.		*Extended mnemonic for* **rlwinm. RA,RS,32–n,n,31**	CR[CR0]	
stb	RS, D(RA)	Store byte $(RS)_{24:31}$ in memory at EA = (RA|0) + EXTS(D).		28-157
stbu	RS, D(RA)	Store byte $(RS)_{24:31}$ in memory at EA = (RA|0) + EXTS(D). Update the base address, $(RA) \leftarrow EA$.		28-158
stbux	RS, RA, RB	Store byte $(RS)_{24:31}$ in memory at EA = (RA|0) + (RB). Update the base address, $(RA) \leftarrow EA$.		28-159
stbx	RS, RA, RB	Store byte $(RS)_{24:31}$ in memory at EA = (RA|0) + (RB).		28-160

Table B-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
sth	RS, D(RA)	Store halfword $(RS)_{16:31}$ in memory at EA = (RA|0) + EXTS(D).		28-161
sthbrx	RS, RA, RB	Store halfword $(RS)_{16:31}$ byte-reversed in memory at EA = (RA|0) + (RB). MS(EA, 2) $\leftarrow$ $(RS)_{24:31}$ $\|\|$ $(RS)_{16:23}$		28-162
sthu	RS, D(RA)	Store halfword $(RS)_{16:31}$ in memory at EA = (RA|0) + EXTS(D). Update the base address, (RA) $\leftarrow$ EA.		28-163
sthux	RS, RA, RB	Store halfword $(RS)_{16:31}$ in memory at EA = (RA|0) + (RB). Update the base address, (RA) $\leftarrow$ EA.		28-164
sthx	RS, RA, RB	Store halfword $(RS)_{16:31}$ in memory at EA = (RA|0) + (RB).		28-165
stmw	RS, D(RA)	Store consecutive words from RS through GPR(31) in memory starting at EA = (RA|0) + EXTS(D).		28-166
stswi	RS, RA, NB	Store consecutive bytes in memory starting at EA=(RA|0). Number of bytes n=32 if NB=0, else n=NB. Bytes are unstacked from CEIL(n/4) consecutive registers starting with RS. GPR(0) is consecutive to GPR(31).		28-166
stswx	RS, RA, RB	Store consecutive bytes in memory starting at EA=(RA|0)+(RB). Number of bytes n=XER[TBC]. Bytes are unstacked from CEIL(n/4) consecutive registers starting with RS. GPR(0) is consecutive to GPR(31).		28-169
stw	RS, D(RA)	Store word (RS) in memory at EA = (RA|0) + EXTS(D).		28-170
stwbrx	RS, RA, RB	Store word (RS) byte-reversed in memory at EA = (RA|0) + (RB). MS(EA, 4) $\leftarrow$ $(RS)_{24:31}$ $\|\|$ $(RS)_{16:23}$ $\|\|$ $(RS)_{8:15}$ $\|\|$ $(RS)_{0:7}$		28-171

stwcx.	RS, RA, RB	Store word (RS) in memory at EA = (RA|0) + (RB)
		only if reservation bit is set.
		if RESERVE = 1 then
		MS(EA, 4) $\leftarrow$ (RS)
		RESERVE $\leftarrow$ 0
		(CR[CR0]) $\leftarrow$ $^{2}0$ $\|\|$ 1 $\|\|$ XER_{SO}
		else
		(CR[CR0]) $\leftarrow$ $^{2}0$ $\|\|$ 0 $\|\|$ XER_{SO}.
				28-172

Table B-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
stwu	RS, D(RA)	Store word (RS) in memory at EA = (RA|0) + EXTS(D). Update the base address, (RA) ← EA.		28-174
stwux	RS, RA, RB	Store word (RS) in memory at EA = (RA|0) + (RB). Update the base address, (RA) ← EA.		28-175
stwx	RS, RA, RB	Store word (RS) in memory at EA = (RA|0) + (RB).		28-176
sub	RT, RA, RB	Subtract (RB) from (RA). (RT) ← ¬(RB) + (RA) + 1. *Extended mnemonic for* **subf RT,RB,RA**		28-177
sub.		*Extended mnemonic for* **subf. RT,RB,RA**	CR[CR0]	
subo		*Extended mnemonic for* **subfo RT,RB,RA**	XER[SO, OV]	
subo.		*Extended mnemonic for* **subfo. RT,RB,RA**	CR[CR0] XER[SO, OV]	
subc	RT, RA, RB	Subtract (RB) from (RA). (RT) ← ¬(RB) + (RA) + 1. Place carry-out in XER[CA]. *Extended mnemonic for* **subfc RT,RB,RA**		28-178
subc.		*Extended mnemonic for* **subfc. RT,RB,RA**	CR[CR0]	
subco		*Extended mnemonic for* **subfco RT,RB,RA**	XER[SO, OV]	
subco.		*Extended mnemonic for* **subfco. RT,RB,RA**	CR[CR0] XER[SO, OV]	
subf	RT, RA, RB	Subtract (RA) from (RB). (RT) ← ¬(RA) + (RB) + 1.		28-177
subf.			CR[CR0]	
subfo			XER[SO, OV]	
subfo.			CR[CR0] XER[SO, OV]	
subfc	RT, RA, RB	Subtract (RA) from (RB). (RT) ← ¬(RA) + (RB) + 1. Place carry-out in XER[CA].		28-178
subfc.			CR[CR0]	
subfco			XER[SO, OV]	
subfco.			CR[CR0] XER[SO, OV]	

Table B-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
subfe	RT, RA, RB	Subtract (RA) from (RB) with carry-in. (RT) ← ¬(RA) + (RB) + XER[CA]. Place carry-out in XER[CA].		28-179
subfe.			CR[CR0]	
subfeo			XER[SO, OV]	
subfeo.			CR[CR0] XER[SO, OV]	
subfic	RT, RA, IM	Subtract (RA) from EXTS(IM). (RT) ← ¬(RA) + EXTS(IM) + 1. Place carry-out in XER[CA].		28-180
subfme	RT, RA, RB	Subtract (RA) from (–1) with carry-in. (RT) ← ¬(RA) + (–1) + XER[CA]. Place carry-out in XER[CA].		28-181
subfme.			CR[CR0]	
subfmeo			XER[SO, OV]	
subfmeo.			CR[CR0] XER[SO, OV]	
subfze	RT, RA, RB	Subtract (RA) from zero with carry-in. (RT) ← ¬(RA) + XER[CA]. Place carry-out in XER[CA].		28-182
subfze.			CR[CR0]	
subfzeo			XER[SO, OV]	
subfzeo.			CR[CR0] XER[SO, OV]	
subi	RT, RA, IM	Subtract EXTS(IM) from (RA|0). Place result in RT. *Extended mnemonic for* **addi RT,RA,–IM**		28-10
subic	RT, RA, IM	Subtract EXTS(IM) from (RA). Place result in RT. Place carry-out in XER[CA]. *Extended mnemonic for* **addic RT,RA,–IM**		28-11
subic.	RT, RA, IM	Subtract EXTS(IM) from (RA). Place result in RT. Place carry-out in XER[CA]. *Extended mnemonic for* **addic. RT,RA,–IM**	CR[CR0]	28-12
subis	RT, RA, IM	Subtract (IM || $^{16}0$) from (RA|0). Place result in RT. *Extended mnemonic for* **addis RT,RA,–IM**		28-13

Table B-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
tlbre	RT, RA,WS	tlbentry ← TLB[$(RA)_{26:31}$] if WS = 0 $(RT)_{0:27}$ ← tlbentry[EPN,V,TS,SIZE] $(RT)_{28:31}$ ← $^{4}0$ MMUCR[STID] ← tlbentry[TID] else if WS = 1 $(RT)_{0:21}$ ← tlbentry[RPN] $(RT)_{22:27}$ ← $^{6}0$ $(RT)_{28:31}$ ← tlbentry[ERPN] else if WS = 2 $(RT)_{0:15}$ ← $^{16}0$ $(RT)_{16:24}$ ← tlbentry[U0,U1,U2,U3,W,I,M,G,E] $(RT)_{25}$ ← 0 $(RT)_{26:31}$ ← tlbentry[UX,UW,UR,SX,SW,SR] else (RT), MMUCR[STID] ← undefined		28-183

tlbsx	RT,RA,RB	Search the TLB for a valid entry that translates the EA.
		EA = (RA|0) + (RB)
		if Rc = 1
		$CR[CR0]_0$ ← 0
		$CR[CR0]_1$ ← 0
		$CR[CR0]_3$ ← XER[SO}
		if Valid TLB entry matching EA and MMUCR[STID,STS] is in the TLB then
		(RT) ← Index of matching TLB Entry
		if Rc = 1
		$CR[CR0]_2$ ← 1
		else
		(RT) ← Undefined
		if Rc = 1
		$CR[CR0]_2$ ← 0
				28-184

tlbsx.			CR[CR0]	
tlbsync		**tlbsync** does not complete until all previous TLB-update instructions executed by this processor have been received and completed by all other processors. For the PPC440GP, **tlbsync** is a no-op.		28-185
tlbwe	RS, RA,WS	tlbentry ← TLB[$(RA)_{26:31}$] if WS = 0 tlbentry[EPN,V,TS,SIZE] ← $(RS)_{0:27}$ tlbentry[TID] ← MMUCR[STID] else if WS = 1 tlbentry[RPN] ← $(RS)_{0:21}$ tlbentry[ERPN] ← $(RS)_{28:31}$ else if WS = 2 tlbentry[U0,U1,U2,U3,W,I,M,G,E] ← $(RS)_{16:24}$ tlbentry[UX,UW,UR,SX,SW,SR] ← $(RS)_{26:31}$ else tlbentry ← undefined		28-186

Table B-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
trap		Trap unconditionally. *Extended mnemonic for* **tw 31,0,0**		28-187
tweq	RA, RB	Trap if (RA) equal to (RB). *Extended mnemonic for* **tw 4,RA,RB**		
twge		Trap if (RA) greater than or equal to (RB). *Extended mnemonic for* **tw 12,RA,RB**		
twgt		Trap if (RA) greater than (RB). *Extended mnemonic for* **tw 8,RA,RB**		
twle		Trap if (RA) less than or equal to (RB). *Extended mnemonic for* **tw 20,RA,RB**		
twlge		Trap if (RA) logically greater than or equal to (RB). *Extended mnemonic for* **tw 5,RA,RB**		
twlgt		Trap if (RA) logically greater than (RB). *Extended mnemonic for* **tw 1,RA,RB**		
twlle		Trap if (RA) logically less than or equal to (RB). *Extended mnemonic for* **tw 6,RA,RB**		
twllt		Trap if (RA) logically less than (RB). *Extended mnemonic for* **tw 2,RA,RB**		
twlng		Trap if (RA) logically not greater than (RB). *Extended mnemonic for* **tw 6,RA,RB**		
twlnl		Trap if (RA) logically not less than (RB). *Extended mnemonic for* **tw 5,RA,RB**		
twlt		Trap if (RA) less than (RB). *Extended mnemonic for* **tw 16,RA,RB**		
twne		Trap if (RA) not equal to (RB). *Extended mnemonic for* **tw 24,RA,RB**		
twng		Trap if (RA) not greater than (RB). *Extended mnemonic for* *tw 20,RA,RB*		
twnl		Trap if (RA) not less than (RB). *Extended mnemonic for* **tw 12,RA,RB**		

Table B-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
tw	TO, RA, RB	Trap exception is generated if, comparing (RA) with (RB), any condition specified by TO is true.		28-187
tweqi	RA, IM	Trap if (RA) equal to EXTS(IM). *Extended mnemonic for* **wi 4,RA,IM**		28-190
twgei		Trap if (RA) greater than or equal to EXTS(IM). *Extended mnemonic for* **twi 12,RA,IM**		
twgti		Trap if (RA) greater than EXTS(IM). *Extended mnemonic for* **twi 8,RA,IM**		
twlei		Trap if (RA) less than or equal to EXTS(IM). *Extended mnemonic for* **twi 20,RA,IM**		
twlgei		Trap if (RA) logically greater than or equal to EXTS(IM). *Extended mnemonic for* **wi 5,RA,IM**		
twlgti		Trap if (RA) logically greater than EXTS(IM). *Extended mnemonic for* **twi 1,RA,IM**		
twllei		Trap if (RA) logically less than or equal to EXTS(IM). *Extended mnemonic for* **twi 6,RA,IM**		
twllti		Trap if (RA) logically less than EXTS(IM). *Extended mnemonic for* **twi 2,RA,IM**		
twlngi		Trap if (RA) logically not greater than EXTS(IM). *Extended mnemonic for* **twi 6,RA,IM**		
twlnli		Trap if (RA) logically not less than EXTS(IM). *Extended mnemonic for* **twi 5,RA,IM**		
twlti		Trap if (RA) less than EXTS(IM). *Extended mnemonic for* **twi 16,RA,IM**		
twnei		Trap if (RA) not equal to EXTS(IM). *Extended mnemonic for* **twi 24,RA,IM**		
twngi		Trap if (RA) not greater than EXTS(IM). *Extended mnemonic for* **twi 20,RA,IM**		
twnli		Trap if (RA) not less than EXTS(IM). *Extended mnemonic for* **twi 12,RA,IM**		

Table B-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
twi	TO, RA, IM	Trap exception is generated if, comparing (RA) with EXTS(IM), any condition specified by TO is true.		28-190
wrtee	RS	Write value of RS_{16} to MSR[EE].		28-192
wrteei	E	Write value of E to MSR[EE].		28-193
xor	RA, RS, RB	XOR (RS) with (RB). Place result in RA.		28-194
xor.			CR[CR0]	
xori	RA, RS, IM	XOR (RS) with ($^{16}0$ || IM). Place result in RA.		28-195
xoris	RA, RS, IM	XOR (RS) with (IM || $^{16}0$). Place result in RA.		28-196

Table A-1. PPC440GP Instruction Syntax Summary

Mnemonic	Operands	Function	Other Registers Changed	Page
add	RT, RA, RB	Add (RA) to (RB). Place result in RT.		899
add.			CR[CR0]	
addo			XER[SO, OV]	
addo.			CR[CR0] XER[SO, OV]	
addc	RT, RA, RB	Add (RA) to (RB). Place result in RT. Place carry-out in XER[CA].		900
addc.			CR[CR0]	
addco			XER[SO, OV]	
addco.			CR[CR0] XER[SO, OV]	
adde	RT, RA, RB	Add XER[CA], (RA), (RB). Place result in RT. Place carry-out in XER[CA].		901
adde.			CR[CR0]	
addeo			XER[SO, OV]	
addeo.			CR[CR0] XER[SO, OV]	
addi	RT, RA, IM	Add EXTS(IM) to (RA|0). Place result in RT.		902
addic	RT, RA, IM	Add EXTS(IM) to (RA|0). Place result in RT. Place carry-out in XER[CA].		903
addic.	RT, RA, IM	Add EXTS(IM) to (RA|0). Place result in RT. Place carry-out in XER[CA].	CR[CR0]	904
addis	RT, RA, IM	Add (IM || $^{16}0$) to (RA|0). Place result in RT.		905
addme	RT, RA	Add XER[CA], (RA), (-1). Place result in RT. Place carry-out in XER[CA].		906
addme.			CR[CR0]	
addmeo			XER[SO, OV]	
addmeo.			CR[CR0] XER[SO, OV]	

Table A-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
addze	RT, RA	Add XER[CA] to (RA). Place result in RT. Place carry-out in XER[CA].		907
addze.			CR[CR0]	
addzeo			XER[SO, OV]	
addzeo.			CR[CR0] XER[SO, OV]	
and	RA, RS, RB	AND (RS) with (RB). Place result in RA.		908
and.			CR[CR0]	
andc	RA, RS, RB	AND (RS) with ¬(RB). Place result in RA.		909
andc.			CR[CR0]	
andi.	RA, RS, IM	AND (RS) with ($^{16}0$ || IM). Place result in RA.	CR[CR0]	910
andis.	RA, RS, IM	AND (RS) with (IM || $^{16}0$). Place result in RA.	CR[CR0]	911
b	target	Branch unconditional relative. LI ← (target – CIA)$_{6:29}$ NIA ← CIA + EXTS(LI || $^{2}0$)		912
ba		Branch unconditional absolute. LI ← target$_{6:29}$ NIA ← EXTS(LI || $^{2}0$)		
bl		Branch unconditional relative. LI ← (target – CIA)$_{6:29}$ NIA ← CIA + EXTS(LI || $^{2}0$)	(LR) ← CIA + 4	
bla		Branch unconditional absolute. LI ← target$_{6:29}$ NIA ← EXTS(LI || $^{2}0$)	(LR) ← CIA + 4	
bc	BO, BI, target	Branch conditional relative. BD ← (target – CIA)$_{16:29}$ NIA ← CIA + EXTS(BD || $^{2}0$)	CTR if BO_2 = 0	913
bca		Branch conditional absolute. BD ← target$_{16:29}$ NIA ← EXTS(BD || $^{2}0$)	CTR if BO_2 = 0	
bcl		Branch conditional relative. BD ← (target – CIA)$_{16:29}$ NIA ← CIA + EXTS(BD || $^{2}0$)	CTR if BO_2 = 0 (LR) ← CIA + 4	
bcla		Branch conditional absolute. BD ← target$_{16:29}$ NIA ← EXTS(BD || $^{2}0$)	CTR if BO_2 = 0 (LR) ← CIA + 4	
bcctr	BO, BI	Branch conditional to address in CTR. Using (CTR) at exit from instruction, NIA ← CTR$_{0:29}$ || $^{2}0$	CTR if BO_2 = 0	919
bcctrl			CTR if BO_2 = 0 (LR) ← CIA + 4	
bclr	BO, BI	Branch conditional to address in LR. Using (LR) at entry to instruction, NIA ← LR$_{0:29}$ || $^{2}0$	CTR if BO_2 = 0	922
bclrl			CTR if BO_2 = 0 (LR) ← CIA + 4	
bctr		Branch unconditionally to address in CTR. *Extended mnemonic for* **bcctr 20,0**		919
bctrl		*Extended mnemonic for* **bcctrl 20,0**	(LR) ← CIA + 4	

Table A-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
bdnz	target	Decrement CTR. Branch if CTR ≠ 0 *Extended mnemonic for* **bc 16,0,target**		913
bdnza		*Extended mnemonic for* **bca 16,0,target**		
bdnzl		*Extended mnemonic for* **bcl 16,0,target**	(LR) ← CIA + 4	
bdnzla		*Extended mnemonic for* **bcla 16,0,target**	(LR) ← CIA + 4	
bdnzlr		Decrement CTR. Branch if CTR ≠ 0 to address in LR. *Extended mnemonic for* **bclr 16,0**		922
bdnzlrl		*Extended mnemonic for* **bclrl 16,0**	(LR) ← CIA + 4	
bdnzf	cr_bit, target	Decrement CTR. Branch if CTR ≠ 0 AND CR_{cr_bit} = 0 *Extended mnemonic for* **bc 0,cr_bit,target**		913
bdnzfa		*Extended mnemonic for* **bca 0,cr_bit,target**		
bdnzfl		*Extended mnemonic for* **bcl 0,cr_bit,target**	(LR) ← CIA + 4	
bdnzfla		*Extended mnemonic for* **bcla 0,cr_bit,target**	(LR) ← CIA + 4	
bdnzflr	cr_bit	Decrement CTR. Branch if CTR ≠ 0 AND CR_{cr_bit} = 0 to address in LR. *Extended mnemonic for* **bclr 0,cr_bit**		922
bdnzflrl		*Extended mnemonic for* **bclrl 0,cr_bit**	(LR) ← CIA + 4	
bdnzt	cr_bit, target	Decrement CTR. Branch if CTR ≠ 0 AND CR_{cr_bit} = 1. *Extended mnemonic for* **bc 8,cr_bit,target**		913
bdnzta		*Extended mnemonic for* **bca 8,cr_bit,target**		
bdnztl		*Extended mnemonic for* **bcl 8,cr_bit,target**	(LR) ← CIA + 4	
bdnztla		*Extended mnemonic for* **bcla 8,cr_bit,target**	(LR) ← CIA + 4	
bdnztlr	cr_bit	Decrement CTR. Branch if CTR ≠ 0 AND CR_{cr_bit} = 1 to address in LR. *Extended mnemonic for* **bclr 8,cr_bit**		922
bdnztlrl		*Extended mnemonic for* **bclrl 8,cr_bit**	(LR) ← CIA + 4	

Table A-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
bdz	target	Decrement CTR. Branch if CTR = 0 *Extended mnemonic for* **bc 18,0,target**		913
bdza		*Extended mnemonic for* **bca 18,0,target**		
bdzl		*Extended mnemonic for* **bcl 18,0,target**	(LR) ← CIA + 4	
bdzla		*Extended mnemonic for* **bcla 18,0,target**	(LR) ← CIA + 4	
bdzlr		Decrement CTR. Branch if CTR = 0 to address in LR. *Extended mnemonic for* **bclr 18,0**		922
bdzlrl		*Extended mnemonic for* **bclrl 18,0**	(LR) ← CIA + 4	
bdzf	cr_bit, target	Decrement CTR. Branch if CTR = 0 AND CR_{cr_bit} = 0 *Extended mnemonic for* **bc 2,cr_bit,target**		913
bdzfa		*Extended mnemonic for* **bca 2,cr_bit,target**		
bdzfl		*Extended mnemonic for* **bcl 2,cr_bit,target**	(LR) ← CIA + 4	
bdzfla		*Extended mnemonic for* **bcla 2,cr_bit,target**	(LR) ← CIA + 4	
bdzflr	cr_bit	Decrement CTR. Branch if CTR = 0 AND CR_{cr_bit} = 0 to address in LR. *Extended mnemonic for* **bclr 2,cr_bit**		922
bdzflrl		*Extended mnemonic for* **bclrl 2,cr_bit**	(LR) ← CIA + 4	
bdzt	cr_bit, target	Decrement CTR. Branch if CTR = 0 AND CR_{cr_bit} = 1. *Extended mnemonic for* **bc 10,cr_bit,target**		913
bdzta		*Extended mnemonic for* **bca 10,cr_bit,target**		
bdztl		*Extended mnemonic for* **bcl 10,cr_bit,target**	(LR) ← CIA + 4	
bdztla		*Extended mnemonic for* **bcla 10,cr_bit,target**	(LR) ← CIA + 4	
bdztlr	cr_bit	Decrement CTR. Branch if CTR = 0 AND CR_{cr_bit} = 1to address in LR. *Extended mnemonic for* **bclr 10,cr_bit**		922
bdztlrl		*Extended mnemonic for* **bclrl 10,cr_bit**	(LR) ← CIA + 4	

Table A-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
beq	[cr_field], target	Branch if equal. UseCR[CR0] if cr_field is omitted. *Extended mnemonic for* **bc 12,4∗cr_field+2,target**		913
beqa		*Extended mnemonic for* **bca 12,4∗cr_field+2,target**		
beql		*Extended mnemonic for* **bcl 12,4∗cr_field+2,target**	(LR) ← CIA + 4	
beqla		*Extended mnemonic for* **bcla 12,4∗cr_field+2,target**	(LR) ← CIA + 4	
beqctr	[cr_field]	Branch if equal to address in CTR. UseCR[CR0] if cr_field is omitted. *Extended mnemonic for* **bcctr 12,4∗cr_field+2**		919
beqctrl		*Extended mnemonic for* **bcctrl 12,4∗cr_field+2**	(LR) ← CIA + 4	
beqlr	[cr_field]	Branch if equal to address in LR. UseCR[CR0] if cr_field is omitted. *Extended mnemonic for* **bclr 12,4∗cr_field+2**		922
beqlrl		*Extended mnemonic for* **bclrl 12,4∗cr_field+2**	(LR) ← CIA + 4	
bf	cr_bit, target	Branch if CR_{cr_bit} = 0. *Extended mnemonic for* **bc 4,cr_bit,target**		913
bfa		*Extended mnemonic for* **bca 4,cr_bit,target**		
bfl		*Extended mnemonic for* **bcl 4,cr_bit,target**	(LR) ← CIA + 4	
bfla		*Extended mnemonic for* **bcla 4,cr_bit,target**	(LR) ← CIA + 4	
bfctr	cr_bit	Branch if CR_{cr_bit} = 0 to address in CTR. *Extended mnemonic for* **bcctr 4,cr_bit**		919
bfctrl		*Extended mnemonic for* **bcctrl 4,cr_bit**	(LR) ← CIA + 4	
bflr	cr_bit	Branch if CR_{cr_bit} = 0 to address in LR. *Extended mnemonic for* **bclr 4,cr_bit**		922
bflrl		*Extended mnemonic for* **bclrl 4,cr_bit**	(LR) ← CIA + 4	

Table A-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
bge	[cr_field], target	Branch if greater than or equal. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bc 4,4∗cr_field+0,target**		913
bgea		*Extended mnemonic for* **bca 4,4∗cr_field+0,target**		
bgel		*Extended mnemonic for* **bcl 4,4∗cr_field+0,target**	(LR) ← CIA + 4	
bgela		*Extended mnemonic for* **bcla 4,4∗cr_field+0,target**	(LR) ← CIA + 4	
bgectr	[cr_field]	Branch if greater than or equal to address in CTR. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bcctr 4,4∗cr_field+0**		919
bgectrl		*Extended mnemonic for* **bcctrl 4,4∗cr_field+0**	(LR) ← CIA + 4	
bgelr	[cr_field]	Branch if greater than or equal to address in LR. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bclr 4,4∗cr_field+0**		922
bgelrl		*Extended mnemonic for* **bclrl 4,4∗cr_field+0**	(LR) ← CIA + 4	
bgt	[cr_field], target	Branch if greater than. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bc 12,4∗cr_field+1,target**		913
bgta		*Extended mnemonic for* **bca 12,4∗cr_field+1,target**		
bgtl		*Extended mnemonic for* **bcl 12,4∗cr_field+1,target**	(LR) ← CIA + 4	
bgtla		*Extended mnemonic for* **bcla 12,4∗cr_field+1,target**	(LR) ← CIA + 4	
bgtctr	[cr_field]	Branch if greater than to address in CTR. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bcctr 12,4∗cr_field+1**		919
bgtctrl		*Extended mnemonic for* **bcctrl 12,4∗cr_field+1**	(LR) ← CIA + 4	
bgtlr	[cr_field]	Branch if greater than to address in LR. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bclr 12,4∗cr_field+1**		922
bgtlrl		*Extended mnemonic for* **bclrl 12,4∗cr_field+1**	(LR) ← CIA + 4	

Table A-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
ble	[cr_field], target	Branch if less than or equal. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bc 4,4∗cr_field+1,target**		913
blea		*Extended mnemonic for* **bca 4,4∗cr_field+1,target**		
blel		*Extended mnemonic for* **bcl 4,4∗cr_field+1,target**	(LR) ← CIA + 4	
blela		*Extended mnemonic for* **bcla 4,4∗cr_field+1,target**	(LR) ← CIA + 4	
blectr	[cr_field]	Branch if less than or equal to address in CTR. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bcctr 4,4∗cr_field+1**		919
blectrl		*Extended mnemonic for* **bcctrl 4,4∗cr_field+1**	(LR) ← CIA + 4	
blelr	[cr_field]	Branch if less than or equal to address in LR. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bclr 4,4∗cr_field+1**		922
blelrl		*Extended mnemonic for* **bclrl 4,4∗cr_field+1**	(LR) ← CIA + 4	
blr		Branch unconditionally to address in LR. *Extended mnemonic for* **bclr 20,0**		922
blrl		*Extended mnemonic for* **bclrl 20,0**	(LR) ← CIA + 4	
blt	[cr_field], target	Branch if less than. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bc 12,4∗cr_field+0,target**		913
blta		*Extended mnemonic for* **bca 12,4∗cr_field+0,target**		
bltl		*Extended mnemonic for* **bcl 12,4∗cr_field+0,target**	(LR) ← CIA + 4	
bltla		*Extended mnemonic for* **bcla 12,4∗cr_field+0,target**	(LR) ← CIA + 4	
bltctr	[cr_field]	Branch if less than to address in CTR. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bcctr 12,4∗cr_field+0**		919
bltctrl		*Extended mnemonic for* **bcctrl 12,4∗cr_field+0**	(LR) ← CIA + 4	

Table A-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
bltlr	[cr_field]	Branch if less than to address in LR. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bclr 12,4∗cr_field+0**		922
bltlrl		*Extended mnemonic for* **bclrl 12,4∗cr_field+0**	(LR) ← CIA + 4	
bne	[cr_field], target	Branch if not equal. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bc 4,4∗cr_field+2,target**		913
bnea		*Extended mnemonic for* **bca 4,4∗cr_field+2,target**		
bnel		*Extended mnemonic for* **bcl 4,4∗cr_field+2,target**	(LR) ← CIA + 4	
bnela		*Extended mnemonic for* **bcla 4,4∗cr_field+2,target**	(LR) ← CIA + 4	
bnectr	[cr_field]	Branch if not equal to address in CTR. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bcctr 4,4∗cr_field+2**		919
bnectrl		*Extended mnemonic for* **bcctrl 4,4∗cr_field+2**	(LR) ← CIA + 4	
bnelr	[cr_field]	Branch if not equal to address in LR. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bclr 4,4∗cr_field+2**		922
bnelrl		*Extended mnemonic for* **bclrl 4,4∗cr_field+2**	(LR) ← CIA + 4	
bng	[cr_field], target	Branch if not greater than. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bc 4,4∗cr_field+1,target**		913
bnga		*Extended mnemonic for* **bca 4,4∗cr_field+1,target**		
bngl		*Extended mnemonic for* **bcl 4,4∗cr_field+1,target**	(LR) ← CIA + 4	
bngla		*Extended mnemonic for* **bcla 4,4∗cr_field+1,target**	(LR) ← CIA + 4	
bngctr	[cr_field]	Branch if not greater than to address in CTR. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bcctr 4,4∗cr_field+1**		919
bngctrl		*Extended mnemonic for* **bcctrl 4,4∗cr_field+1**	(LR) ← CIA + 4	

Table A-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
bnglr	[cr_field]	Branch if not greater than to address in LR. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bclr 4,4∗cr_field+1**		922
bnglrl		*Extended mnemonic for* **bclrl 4,4∗cr_field+1**	(LR) ← CIA + 4	
bnl	[cr_field], target	Branch if not less than. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bc 4,4∗cr_field+0,target**		913
bnla		*Extended mnemonic for* **bca 4,4∗cr_field+0,target**		
bnll		*Extended mnemonic for* **bcl 4,4∗cr_field+0,target**	(LR) ← CIA + 4	
bnlla		*Extended mnemonic for* **bcla 4,4∗cr_field+0,target**	(LR) ← CIA + 4	
bnlctr	[cr_field]	Branch if not less than to address in CTR. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bcctr 4,4∗cr_field+0**		919
bnlctrl		*Extended mnemonic for* **bcctrl 4,4∗cr_field+0**	(LR) ← CIA + 4	
bnllr	[cr_field]	Branch if not less than to address in LR. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bclr 4,4∗cr_field+0**		922
bnllrl		*Extended mnemonic for* **bclrl 4,4∗cr_field+0**	(LR) ← CIA + 4	
bns	[cr_field], target	Branch if not summary overflow. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bc 4,4∗cr_field+3,target**		913
bnsa		*Extended mnemonic for* **bca 4,4∗cr_field+3,target**		
bnsl		*Extended mnemonic for* **bcl 4,4∗cr_field+3,target**	(LR) ← CIA + 4	
bnsla		*Extended mnemonic for* **bcla 4,4∗cr_field+3,target**	(LR) ← CIA + 4	
bnsctr	[cr_field]	Branch if not summary overflow to address in CTR. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bcctr 4,4∗cr_field+3**		919
bnsctrl		*Extended mnemonic for* **bcctrl 4,4∗cr_field+3**	(LR) ← CIA + 4	

Table A-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
bnslr	[cr_field]	Branch if not summary overflow to address in LR. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bclr 4,4∗cr_field+3**		922
bnslrl		*Extended mnemonic for* **bclrl 4,4∗cr_field+3**	(LR) ← CIA + 4	
bnu	[cr_field], target	Branch if not unordered. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bc 4,4∗cr_field+3,target**		913
bnua		*Extended mnemonic for* **bca 4,4∗cr_field+3,target**		
bnul		*Extended mnemonic for* **bcl 4,4∗cr_field+3,target**	(LR) ← CIA + 4	
bnula		*Extended mnemonic* for **bcla 4,4∗cr_field+3,target**	(LR) ← CIA + 4	
bnuctr	[cr_field]	Branch if not unordered to address in CTR. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bcctr 4,4∗cr_field+3**		919
bnuctrl		*Extended mnemonic for* **bcctrl 4,4∗cr_field+3**	(LR) ← CIA + 4	
bnulr	[cr_field]	Branch if not unordered to address in LR. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bclr 4,4∗cr_field+3**		922
bnulrl		*Extended mnemonic for* **bclrl 4,4∗cr_field+3**	(LR) ← CIA + 4	
bso	[cr_field], target	Branch if summary overflow. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bc 12,4∗cr_field+3,target**		913
bsoa		*Extended mnemonic for* **bca 12,4∗cr_field+3,target**		
bsol		*Extended mnemonic for* **bcl 12,4∗cr_field+3,target**	(LR) ← CIA + 4	
bsola		*Extended mnemonic for* **bcla 12,4∗cr_field+3,target**	(LR) ← CIA + 4	
bsoctr	[cr_field]	Branch if summary overflow to address in CTR. UseCR[CR0] if cr_field is omitted. *Extended mnemonic for* **bcctr 12,4∗cr_field+3**		919
bsoctrl		*Extended mnemonic for* **bcctrl 12,4∗cr_field+3**	(LR) ← CIA + 4	

Table A-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
bsolr	[cr_field]	Branch if summary overflow to address in LR. UseCR[CR0] if cr_field is omitted. *Extended mnemonic* for **bclr 12,4∗cr_field+3**		922
bsolrl		*Extended mnemonic* for **bclrl 12,4∗cr_field+3**	(LR) ⟵ CIA + 4	
bt	cr_bit, target	Branch if CR_{cr_bit} = 1. *Extended mnemonic for* **bc 12,cr_bit,target**		913
bta		*Extended mnemonic for* **bca 12,cr_bit,target**		
btl		*Extended mnemonic for* **bcl 12,cr_bit,target**	(LR) ⟵ CIA + 4	
btla		*Extended mnemonic for* **bcla 12,cr_bit,target**	(LR) ⟵ CIA + 4	
btctr	cr_bit	Branch if CR_{cr_bit} = 1 to address in CTR. *Extended mnemonic for* **bcctr 12,cr_bit**		919
btctrl		*Extended mnemonic for* **bcctrl 12,cr_bit**	(LR) ⟵ CIA + 4	
btlr	cr_bit	Branch if CR_{cr_bit} = 1, to address in LR. *Extended mnemonic for* **bclr 12,cr_bit**		922
btlrl		*Extended mnemonic for* **bclrl 12,cr_bit**	(LR) ⟵ CIA + 4	
bun	[cr_field], target	Branch if unordered. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bc 12,4∗cr_field+3,target**		913
buna		*Extended mnemonic for* **bca 12,4∗cr_field+3,target**		
bunl		*Extended mnemonic for* **bcl 12,4∗cr_field+3,target**	(LR) ⟵ CIA + 4	
bunla		*Extended mnemonic for* **bcla 12,4∗cr_field+3,target**	(LR) ⟵ CIA + 4	
bunctr	[cr_field]	Branch if unordered to address in CTR. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bcctr 12,4∗cr_field+3**		919
bunctrl		*Extended mnemonic for* **bcctrl 12,4∗cr_field+3**	(LR) ⟵ CIA + 4	

Table A-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
bunlr	[cr_field]	Branch if unordered, to address in LR. Use CR[CR0] if cr_field is omitted. *Extended mnemonic for* **bclr 12,4∗cr_field+3**		922
bunlrl		*Extended mnemonic for* **bclrl 12,4∗cr_field+3**	(LR) ← CIA + 4	
clrlwi	RA, RS, n	Clear left immediate. (n < 32) $(RA)_{0:n-1} \leftarrow {}^{n}0$ *Extended mnemonic for* **rlwinm RA,RS,0,n,31**		1042
clrlwi.		*Extended mnemonic for* **rlwinm. RA,RS,0,n,31**	CR[CR0]	
clrlslwi	RA, RS, b, n	Clear left and shift left immediate. (n ≤ b < 32) $(RA)_{b-n:31-n} \leftarrow (RS)_{b:31}$ $(RA)_{32-n:31} \leftarrow {}^{n}0$ $(RA)_{0:b-n-1} \leftarrow {}^{b-n}0$ *Extended mnemonic for* **rlwinm RA,RS,n,b−n,31−n**		1042
clrlslwi.		*Extended mnemonic for* **rlwinm. RA,RS,n,b−n,31−n**	CR[CR0]	
clrrwi	RA, RS, n	Clear right immediate. (n < 32) $(RA)_{32-n:31} \leftarrow {}^{n}0$ *Extended mnemonic for* **rlwinm RA,RS,0,0,31−n**		1042
clrrwi.		*Extended mnemonic for* **rlwinm. RA,RS,0,0,31−n**	CR[CR0]	
cmp	BF, 0, RA, RB	Compare (RA) to (RB), signed. Results in CR[CRn], where n = BF.		926
cmpi	BF, 0, RA, IM	Compare (RA) to EXTS(IM), signed. Results in CR[CRn], where n = BF.		927
cmpl	BF, 0, RA, RB	Compare (RA) to (RB), unsigned. Results in CR[CRn], where n = BF.		928
cmpli	BF, 0, RA, IM	Compare (RA) to ($^{16}0$ || IM), unsigned. Results in CR[CRn], where n = BF.		929
cmplw	[BF,] RA, RB	Compare Logical Word. UseCR[CR0] if BF is omitted. *Extended mnemonic for* **cmpl BF,0,RA,RB**		928
cmplwi	[BF,] RA, IM	Compare Logical Word Immediate. UseCR[CR0] if BF is omitted. *Extended mnemonic for* **cmpli BF,0,RA,IM**		929
cmpw	[BF,] RA, RB	Compare Word. UseCR[CR0] if BF is omitted. *Extended mnemonic for* **cmp BF,0,RA,RB**		926

Table A-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
cmpwi	[BF,] RA, IM	Compare Word Immediate. UseCR[CR0] if BF is omitted. *Extended mnemonic for* **cmpi BF,0,RA,IM**		927
cntlzw	RA, RS	Count leading zeros in RS. Place result in RA.		930
cntlzw.			CR[CR0]	
crand	BT, BA, BB	AND bit (CR_{BA}) with (CR_{BB}). Place result in CR_{BT}.		931
crandc	BT, BA, BB	AND bit (CR_{BA}) with $\neg$(CR_{BB}). Place result in CR_{BT}.		932
crclr	bx	Condition register clear. *Extended mnemonic for* **crxor bx,bx,bx**		938
creqv	BT, BA, BB	Equivalence of bit CR_{BA} with CR_{BB}. $CR_{BT} \leftarrow \neg(CR_{BA} \oplus CR_{BB})$		933
crmove	bx, by	Condition register move. *Extended mnemonic for* **cror bx,by,by**		936
crnand	BT, BA, BB	NAND bit (CR_{BA}) with (CR_{BB}). Place result in CR_{BT}.		934
crnor	BT, BA, BB	NOR bit (CR_{BA}) with (CR_{BB}). Place result in CR_{BT}.		935
crnot	bx, by	Condition register not. *Extended mnemonic for* **crnor bx,by,by**		935
cror	BT, BA, BB	OR bit (CR_{BA}) with (CR_{BB}). Place result in CR_{BT}.		936
crorc	BT, BA, BB	OR bit (CR_{BA}) with $\neg$(CR_{BB}). Place result in CR_{BT}.		937
crset	bx	Condition register set. *Extended mnemonic for* **creqv bx,bx,bx**		933
crxor	BT, BA, BB	XOR bit (CR_{BA}) with (CR_{BB}). Place result in CR_{BT}.		938
dcba	RA, RB	Treated as a no-op.		939
dcbf	RA, RB	Flush (store, then invalidate) the data cache block which contains the effective address (RA|0) + (RB).		940
dcbi	RA, RB	Invalidate the data cache block which contains the effective address (RA|0) + (RB).		941
dcbst	RA, RB	Store the data cache block which contains the effective address (RA|0) + (RB).		942
dcbt	RA, RB	Load the data cache block which contains the effective address (RA|0) + (RB).		943
dcbtst	RA,RB	Load the data cache block which contains the effective address (RA|0) + (RB).		944
dcbz	RA, RB	Zero the data cache block which contains the effective address (RA|0) + (RB).		946
dccci	RA, RB	Invalidate the data cache array.		948

Table A-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
dcread	RT, RA, RB	Read tag and data information from the data cache line selected using effective address bits 17:26. The effective address is calculated by (RA|0) + (RB). Place the data word selected by effective address bits 27:29 in GPR RT; place the tag information in DCDBTRH and DCDBTRL.		949
divw	RT, RA, RB	Divide (RA) by (RB), signed. Place result in RT.		951
divw.			CR[CR0]	
divwo			XER[SO, OV]	
divwo.			CR[CR0] XER[SO, OV]	
divwu	RT, RA, RB	Divide (RA) by (RB), unsigned. Place result in RT.		952
divwu.			CR[CR0]	
divwuo			XER[SO, OV]	
divwuo.			CR[CR0] XER[SO, OV]	

dlmzb	RA, RS, RB	d ← (RS) || (RB)
		i, x, y ← 0
		do while (x < 8) ∧ (y = 0)
		  x ← x + 1
		  if $d_{i:i+7}$ = 0 then
		  y ← 1
		  else
		  i ← i + 8
		(RA) ← x
		XER[TBC] ← x
		if Rc = 1 then
		  $CR[CR0]_3$ ←XER[SO]
		  if y = 1 then
		  if x < 5 then
		  $CR[CR0]_{0:2}$ ← 0b010
		  else
		  $CR[CR0]_{0:2}$ ← 0b100
		  else
		  $CR[CR0]_{0:2}$ ← 0b001
			XER[TBC], RA	953

dlmzb.			XER[TBC], RA, CR[CR0]	
eqv	RA, RS, RB	Equivalence of (RS) with (RB). (RA) ← ¬((RS) ⊕ (RB))		954
eqv.			CR[CR0]	
extlwi	RA, RS, n, b	Extract and left justify immediate. (n > 0) $(RA)_{0:n-1}$ ← $(RS)_{b:b+n-1}$ $(RA)_{n:31}$ ← $^{32-n}0$ *Extended mnemonic for* **rlwinm RA,RS,b,0,n−1**		1042
extlwi.		*Extended mnemonic for* **rlwinm. RA,RS,b,0,n−1**	CR[CR0]	
extrwi	RA, RS, n, b	Extract and right justify immediate. (n > 0) $(RA)_{32-n:31}$ ← $(RS)_{b:b+n-1}$ $(RA)_{0:31-n}$ ← $^{32-n}0$ *Extended mnemonic for* **rlwinm RA,RS,b+n,32−n,31**		1042
extrwi.		*Extended mnemonic for* **rlwinm. RA,RS,b+n,32−n,31**	CR[CR0]	

Table A-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
extsb	RA, RS	Extend the sign of byte $(RS)_{24:31}$. Place the result in RA.		955
extsb.			CR[CR0]	
extsh	RA, RS	Extend the sign of halfword $(RS)_{16:31}$. Place the result in RA.		956
extsh.			CR[CR0]	
icbi	RA, RB	Invalidate the instruction cache block which contains the effective address (RA|0) + (RB).		957
icbt	RA, RB	Load the instruction cache block which contains the effective address (RA|0) + (RB).		955
iccci	RA, RB	Invalidate the instruction cache array.		960
icread	RA, RB	Read tag and data information from the instruction cache line selected using effective address bits 17:26. The effective address is calculated by (RA|0) + (RB). Place the instruction selected by effective address bits 27:29 in ICDBDR; place the tag information in ICDBTRH and ICDBTRL.		961
inslwi	RA, RS, n, b	Insert from left immediate. (n > 0) $(RA)_{b:b+n-1} \leftarrow (RS)_{0:n-1}$ *Extended mnemonic for* **rlwimi RA,RS,32−b,b,b+n−1**		1041
inslwi.		*Extended mnemonic for* **rlwimi. RA,RS,32−b,b,b+n−1**	CR[CR0]	
insrwi	RA, RS, n, b	Insert from right immediate. (n > 0) $(RA)_{b:b+n-1} \leftarrow (RS)_{32-n:31}$ *Extended mnemonic for* **rlwimi RA,RS,32−b−n,b,b+n−1**		1041
insrwi.		*Extended mnemonic for* **rlwimi. RA,RS,32−b−n,b,b+n−1**	CR[CR0]	
isync		Synchronize execution context by flushing the prefetch queue.		963
la	RT, D(RA)	Load address. (RA ≠ 0) D is an offset from a base address that is assumed to be (RA). (RT) ← (RA) + EXTS(D) *Extended mnemonic for* **addi RT,RA,D**		902
lbz	RT, D(RA)	Load byte from EA = (RA|0) + EXTS(D) and pad left with zeroes, (RT) ← $^{24}0$ || MS(EA,1).		964
lbzu	RT, D(RA)	Load byte from EA = (RA|0) + EXTS(D) and pad left with zeroes, (RT) ← $^{24}0$ || MS(EA,1). Update the base address, (RA) ← EA.		965
lbzux	RT, RA, RB	Load byte from EA = (RA|0) + (RB) and pad left with zeroes, (RT) ← $^{24}0$ || MS(EA,1). Update the base address, (RA) ← EA.		966
lbzx	RT, RA, RB	Load byte from EA = (RA|0) + (RB) and pad left with zeroes, (RT) ← $^{24}0$ || MS(EA,1).		967
lha	RT, D(RA)	Load halfword from EA = (RA|0) + EXTS(D) and sign extend, (RT) ← EXTS(MS(EA,2)).		968

Table A-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
lhau	RT, D(RA)	Load halfword from EA = (RA|0) + EXTS(D) and sign extend, (RT) ← EXTS(MS(EA,2)). Update the base address, (RA) ← EA.		969
lhaux	RT, RA, RB	Load halfword from EA = (RA|0) + (RB) and sign extend, (RT) ← EXTS(MS(EA,2)). Update the base address, (RA) ← EA.		970
lhax	RT, RA, RB	Load halfword from EA = (RA|0) + (RB) and sign extend, (RT) ← EXTS(MS(EA,2)).		971
lhbrx	RT, RA, RB	Load halfword from EA = (RA|0) + (RB), then reverse byte order and pad left with zeroes, (RT) ← $^{16}0$ || MS(EA+1,1) || MS(EA,1).		972
lhz	RT, D(RA)	Load halfword from EA = (RA|0) + EXTS(D) and pad left with zeroes, (RT) ← $^{16}0$ || MS(EA,2).		973
lhzu	RT, D(RA)	Load halfword from EA = (RA|0) + EXTS(D) and pad left with zeroes, (RT) ← $^{16}0$ || MS(EA,2). Update the base address, (RA) ← EA.		974
lhzux	RT, RA, RB	Load halfword from EA = (RA|0) + (RB) and pad left with zeroes, (RT) ← $^{16}0$ || MS(EA,2). Update the base address, (RA) ← EA.		975
lhzx	RT, RA, RB	Load halfword from EA = (RA|0) + (RB) and pad left with zeroes, (RT) ← $^{16}0$ || MS(EA,2).		976
li	RT, IM	Load immediate. (RT) ← EXTS(IM) *Extended mnemonic for* **addi RT,0,value**		902
lis	RT, IM	Load immediate shifted. (RT) ← (IM || $^{16}0$) *Extended mnemonic for* **addis RT,0,value**		905
lmw	RT, D(RA)	Load multiple words starting from EA = (RA|0) + EXTS(D). Place into consecutive registers RT through GPR(31). RA is not altered unless RA = GPR(31).		977
lswi	RT, RA, NB	Load consecutive bytes from EA=(RA|0). Number of bytes n=32 if NB=0, else n=NB. Stack bytes into words in CEIL(n/4) consecutive registers starting with RT, to R_{FINAL} ← ((RT + CEIL(n/4) – 1) % 32). GPR(0) is consecutive to GPR(31). RA is not altered unless RA = R_{FINAL}.		978

Table A-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
lswx	RT, RA, RB	Load consecutive bytes from EA=(RA|0)+(RB). Number of bytes n=XER[TBC]. Stack bytes into words in CEIL(n/4) consecutive registers starting with RT, to $R_{FINAL} \leftarrow ((RT + CEIL(n/4) - 1) \% 32)$. GPR(0) is consecutive to GPR(31). RA is not altered unless RA = R_{FINAL}. RB is not altered unless RB = R_{FINAL}. If n=0, content of RT is undefined.		980
lwarx	RT, RA, RB	Load word from EA = (RA|0) + (RB) and place in RT, (RT) $\leftarrow$ MS(EA,4). Set the Reservation bit.		982
lwbrx	RT, RA, RB	Load word from EA = (RA|0) + (RB) then reverse byte order, (RT) $\leftarrow$ MS(EA+3,1) || MS(EA+2,1) || MS(EA+1,1) || MS(EA,1).		983
lwz	RT, D(RA)	Load word from EA = (RA|0) + EXTS(D) and place in RT, (RT) $\leftarrow$ MS(EA,4).		984
lwzu	RT, D(RA)	Load word from EA = (RA|0) + EXTS(D) and place in RT, (RT) $\leftarrow$ MS(EA,4). Update the base address, (RA) $\leftarrow$ EA.		985
lwzux	RT, RA, RB	Load word from EA = (RA|0) + (RB) and place in RT, (RT) $\leftarrow$ MS(EA,4). Update the base address, (RA) $\leftarrow$ EA.		986
lwzx	RT, RA, RB	Load word from EA = (RA|0) + (RB) and place in RT, (RT) $\leftarrow$ MS(EA,4).		987
macchw	RT, RA, RB	$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{0:15}$ $temp_{0:32} \leftarrow prod_{0:31} + (RT)$ (RT) $\leftarrow temp_{1:32}$		988
macchw.			CR[CR0]	
macchwo			XER[SO, OV]	
macchwo.			CR[CR0] XER[SO, OV]	
macchwu	RT, RA, RB	$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{0:15}$ $temp_{0:32} \leftarrow prod_{0:31} + (RT)$ (RT) $\leftarrow temp_{1:32}$		991
macchwu.			CR[CR0]	
macchwuo			XER[SO, OV]	
macchwuo.			CR[CR0] XER[SO, OV]	
macchws	RT, RA, RB	$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{0:15}$ $temp_{0:32} \leftarrow prod_{0:31} + (RT)$ if $((prod_0 = RT_0) \wedge (RT_0 \neq temp_1))$ then (RT) $\leftarrow (RT_0 \vee {}^{31}(\neg RT_0))$ else (RT) $\leftarrow temp_{1:32}$		989
macchws.			CR[CR0]	
macchwso			XER[SO, OV]	
macchwso.			CR[CR0] XER[SO, OV]	
macchwsu	RT, RA, RB	$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{0:15}$ $temp_{0:32} \leftarrow prod_{0:31} + (RT)$ (RT) $\leftarrow (temp_{1:32} \vee {}^{31}temp_0)$		990
macchwsu.			CR[CR0]	
macchwsuo			XER[SO, OV]	
macchwsuo.			CR[CR0] XER[SO, OV]	

Table A-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
machhw	RT, RA, RB	$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{0:15}$ $temp_{0:32} \leftarrow prod_{0:31} + (RT)$ $(RT) \leftarrow temp_{1:32}$		992
machhw.			CR[CR0]	
machhwo			XER[SO, OV]	
machhwo.			CR[CR0] XER[SO, OV]	
machhwu	RT, RA, RB	$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{0:15}$ $temp_{0:32} \leftarrow prod_{0:31} + (RT)$ $(RT) \leftarrow temp_{1:32}$		995
machhwu.			CR[CR0]	
machhwuo			XER[SO, OV]	
machhwuo.			CR[CR0] XER[SO, OV]	
machhws	RT, RA, RB	$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{0:15}$ $temp_{0:32} \leftarrow prod_{0:31} + (RT)$ if $((prod_0 = RT_0) \wedge (RT_0 \neq temp_1))$ then $(RT) \leftarrow (RT_0 \vee {}^{31}(\neg RT_0))$ else $(RT) \leftarrow temp_{1:32}$		993
machhws.			CR[CR0]	
machhwso			XER[SO, OV]	
machhwso.			CR[CR0] XER[SO, OV]	
machhwsu	RT, RA, RB	$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{0:15}$ $temp_{0:32} \leftarrow prod_{0:31} + (RT)$ $(RT) \leftarrow (temp_{1:32} \vee {}^{31}temp_0)$		994
machhwsu.			CR[CR0]	
machhwsuo			XER[SO, OV]	
machhwsuo.			CR[CR0] XER[SO, OV]	
maclhw	RT, RA, RB	$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{0:15}$ $temp_{0:32} \leftarrow prod_{0:31} + (RT)$ $(RT) \leftarrow temp_{1:32}$		996
maclhw.			CR[CR0]	
maclhwo			XER[SO, OV]	
maclhwo.			CR[CR0] XER[SO, OV]	
maclhwu	RT, RA, RB	$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{0:15}$ $temp_{0:32} \leftarrow prod_{0:31} + (RT)$ $(RT) \leftarrow temp_{1:32}$		999
maclhwu.			CR[CR0]	
maclhwuo			XER[SO, OV]	
maclhwuo.			CR[CR0] XER[SO, OV]	
maclhws	RT, RA, RB	$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{0:15}$ $temp_{0:32} \leftarrow prod_{0:31} + (RT)$ if $((prod_0 = RT_0) \wedge (RT_0 \neq temp_1))$ then $(RT) \leftarrow (RT_0 \vee {}^{31}(\neg RT_0))$ else $(RT) \leftarrow temp_{1:32}$		997
maclhws.			CR[CR0]	
maclhwso			XER[SO, OV]	
maclhwso.			CR[CR0] XER[SO, OV]	
maclhwsu	RT, RA, RB	$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{0:15}$ $temp_{0:32} \leftarrow prod_{0:31} + (RT)$ $(RT) \leftarrow (temp_{1:32} \vee {}^{31}temp_0)$		998
maclhwsu.			CR[CR0]	
maclhwsuo			XER[SO, OV]	
maclhwsuo.			CR[CR0] XER[SO, OV]	
mbar		Storage synchronization. All loads and stores that precede the **mbar** instruction complete before any loads and stores that follow the instruction access main storage.		1000
mcrf	BF, BFA	Move CR field, (CR[CRn]) ← (CR[CRm]) where m ← BFA and n ← BF		1001
mcrxr	BF	Move XER[0:3] into field CRn, where n←BF. CR[CRn] ← (XER[SO, OV, CA]) (XER[SO, OV, CA]) ← $^{3}0$		1002

Table A-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
mfcr	RT	Move from CR to RT, (RT) ← (CR).		1003
mfdcr	RT, DCRN	Move from DCR to RT, (RT) ← (DCR(DCRN)).		1004
mfmsr	RT	Move from MSR to RT, (RT) ← (MSR).		1005
mfccr0 **mfctr** **mfdac1** mfdac2 **mfdbcr0** **mfdbcr1** **mfdbsr** **mfdccr** **mfdcwr** mfdvc1 mfdvc2 **mfesr** **mfiac1** mfiac2 mfiac3 mfiac4 **mficcr** **mficdbdr** **mfivpr** **mflr** mfpid **mfpit** **mfpvr** **mfsgr** **mfsprg0** **mfsprg1** **mfsprg2** **mfsprg3** **mfsprg4** **mfsprg5** **mfsprg6** **mfsprg7** **mfsrr0** **mfsrr1** **mfsrr2** **mfsrr3** mfsu0r **mftbl** **mftbu** **mftcr** **mftsr** **mfxer**	RT	Move from special purpose register (SPR) SPRN. *Extended mnemonic for* **mfspr RT,SPRN** See Table 29-2, "Special Purpose Registers Sorted by SPR Number," on page -1095 for listing of valid SPRN values.		1006

Table A-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
mfspr	RT, SPRN	Move from SPR to RT, (RT) ← (SPR(SPRN)).		1006
mr	RT, RS	Move register. (RT) ← (RS) *Extended mnemonic for* **or RT,RS,RS**		1035
mr.		*Extended mnemonic for* **or. RT,RS,RS**	CR[CR0]	
msync		Synchronization. All instructions that precede **msync** complete before any instructions that follow **msync** begin. When **msync** completes, all storage accesses initiated prior to **msync** will have completed.		1009
mtcr	RS	Move to Condition Register. *Extended mnemonic for* **mtcrf 0xFF,RS**		1010
mtcrf	FXM, RS	Move some or all of the contents of RS into CR as specified by FXM field, mask ← $^4(FXM_0)$ || $^4(FXM_1)$ || ... || $^4(FXM_6)$ || $^4(FXM_7)$. (CR)←((RS) ∧ mask) ∨ (CR) ∧ ¬mask).		1010
mtdcr	DCRN, RS	Move to DCR from RS, (DCR(DCRN)) ← (RS).		1011
mtmsr	RS	Move to MSR from RS, (MSR) ← (RS).		1012

Table A-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
mtccr0 **mtctr** **mtdac1** mtdac2 **mtdbcr0** **mtdbcr1** **mtdbsr** **mtdccr** **mtdcwr** mtdvc1 mtdvc2 **mtesr** **mtiac1** mtiac2 mtiac3 mtiac4 **mticcr** **mticdbdr** **mtivpr** **mtlr** mtpid **mtpit** **mtpvr** **mtsgr** **mtsprg0** **mtsprg1** **mtsprg2** **mtsprg3** **mtsprg4** **mtsprg5** **mtsprg6** **mtsprg7** **mtsrr0** **mtsrr1** **mtsrr2** **mtsrr3** mtsu0r **mttbl** **mttbu** **mttcr** **mttsr** **mtxer** mtzpr	RS	Move to SPR SPRN. *Extended mnemonic for* **mtspr SPRN,RS** See Table 29-2, "Special Purpose Registers Sorted by SPR Number," on page -1095 for listing of valid SPRN values.		1013
mtspr	SPRN, RS	Move to SPR from RS, (SPR(SPRN)) ← (RS).		1013
mulchw mulchw.	RT, RA, RB	$(RT)_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{0:15}$ (signed)	 CR[CR0]	1016

Table A-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
mulchwu	RT, RA, RB	$(RT)_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{0:15}$ (unsigned)		1017
mulchwu.			CR[CR0]	
mulhhw	RT, RA, RB	$(RT)_{0:31} \leftarrow (RA)_{0:15} \times (RB_{0:15}$ (signed)		1018
mulhhw.			CR[CR0]	
mulhhwu	RT, RA, RB	$(RT)_{0:31} \leftarrow (RA)_{0:15} \times (RB)_{0:15}$ (unsigned)		1019
mulhhwu.			CR[CR0]	
mulhw	RT, RA, RB	Multiply (RA) and (RB), signed. Place high-order result in RT. $prod_{0:63} \leftarrow$ (RA) $\times$ (RB) (signed). (RT) $\leftarrow prod_{0:31}$.		1020
mulhw.			CR[CR0]	
mulhwu	RT, RA, RB	Multiply (RA) and (RB), unsigned. Place high-order result in RT. $prod_{0:63} \leftarrow$ (RA) $\times$ (RB) (unsigned). (RT) $\leftarrow prod_{0:31}$.		1021
mulhwu.			CR[CR0]	
mullhw	RT, RA, RB	$(RT)_{0:31} \leftarrow (RA)_{16:31} \times (RB_{16:31}$ (signed)		1022
mullhw.			CR[CR0]	
mullhwu	RT, RA, RB	$(RT)_{16:31} \leftarrow (RA)_{0:15} \times (RB)_{16:31}$ (unsigned)		1023
mullhwu.			CR[CR0]	
mulli	RT, RA, IM	Multiply (RA) and IM, signed. Place low-order result in RT. $prod_{0:47} \leftarrow$ (RA) $\times$ IM (signed) (RT) $\leftarrow prod_{16:47}$		1024
mullw	RT, RA, RB	Multiply (RA) and (RB), signed. Place low-order result in RT. $prod_{0:63} \leftarrow$ (RA) $\times$ (RB) (signed). (RT) $\leftarrow prod_{32:63}$.		1025
mullw.			CR[CR0]	
mullwo			XER[SO, OV]	
mullwo.			CR[CR0] XER[SO, OV]	
nand	RA, RS, RB	NAND (RS) with (RB). Place result in RA.		1026
nand.			CR[CR0]	
neg	RT, RA	Negative (two's complement) of RA. (RT) $\leftarrow \neg$(RA) + 1		1027
neg.			CR[CR0]	
nego			XER[SO, OV]	
nego.			CR[CR0] XER[SO, OV]	
nmacchw	RT, RA, RB	$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{0:15}$ $temp_{0:32} \leftarrow -prod_{0:31} + (RT)$ (RT) $\leftarrow temp_{1:32}$		1028
nmacchw.			CR[CR0]	
nmacchwo			XER[SO, OV]	
nmacchwo.			CR[CR0] XER[SO, OV]	
nmacchws	RT, RA, RB	$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{0:15}$ $temp_{0:32} \leftarrow -prod_{0:31} + (RT)$ if $((prod_0 = RT_0) \wedge (RT_0 \neq temp_1))$ then (RT) $\leftarrow (RT_0 \vee {}^{31}(\neg RT_0))$ else (RT) $\leftarrow temp_{1:32}$		1029
nmacchws.			CR[CR0]	
nmacchwso			XER[SO, OV]	
nmacchwso.			CR[CR0] XER[SO, OV]	

Table A-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
nmachhw	RT, RA, RB	$prod_{0:31} \leftarrow (RA)_{0:15} \times (RB)_{0:15}$ $temp_{0:32} \leftarrow -prod_{0:31} + (RT)$ $(RT) \leftarrow temp_{1:32}$		1030
nmachhw.			CR[CR0]	
nmachhwo			XER[SO, OV]	
nmachhwo.			CR[CR0] XER[SO, OV]	
nmachhws	RT, RA, RB	$prod_{0:31} \leftarrow (RA)_{0:15} \times (RB)_{0:15}$ $temp_{0:32} \leftarrow -prod_{0:31} + (RT)$ if $((prod_0 = RT_0) \wedge (RT_0 \neq temp_1))$ then $(RT) \leftarrow (RT_0 \vee {}^{31}(\neg RT_0))$ else $(RT) \leftarrow temp_{1:32}$		1031
nmachhws.			CR[CR0]	
nmachhwso			XER[SO, OV]	
nmachhwso.			CR[CR0] XER[SO, OV]	
nmaclhw	RT, RA, RB	$prod_{0:31} \leftarrow (RA_{16:31} \times (RB)_{16:31}$ $temp_{0:32} \leftarrow -prod_{0:31} + (RT)$ $(RT) \leftarrow temp_{1:32}$		1032
nmaclhw.			CR[CR0]	
nmaclhwo			XER[SO, OV]	
nmaclhwo.			CR[CR0] XER[SO, OV]	
nmaclhws	RT, RA, RB	$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{16:31}$ $temp_{0:32} \leftarrow -prod_{0:31} + (RT)$ if $((prod_0 = RT_0) \wedge (RT_0 \neq temp_1))$ then $(RT) \leftarrow (RT_0 \vee {}^{31}(\neg RT_0))$ else $(RT) \leftarrow temp_{1:32}$		1033
nmaclhws.			CR[CR0]	
nmaclhwso			XER[SO, OV]	
nmaclhwso.			CR[CR0] XER[SO, OV]	
nop		Preferred no-op, triggers optimizations based on no-ops. *Extended mnemonic for* **ori 0,0,0**		1037
nor	RA, RS, RB	NOR (RS) with (RB). Place result in RA.		1034
nor.			CR[CR0]	
not	RA, RS	Complement register. $(RA) \leftarrow \neg(RS)$ *Extended mnemonic for* **nor RA,RS,RS**		1034
not.		*Extended mnemonic for* **nor. RA,RS,RS**	CR[CR0]	
or	RA, RS, RB	OR (RS) with (RB). Place result in RA.		1035
or.			CR[CR0]	
orc	RA, RS, RB	OR (RS) with ¬(RB). Place result in RA.		1036
orc.			CR[CR0]	
ori	RA, RS, IM	OR (RS) with $({}^{16}0 \parallel IM)$. Place result in RA.		1037
oris	RA, RS, IM	OR (RS) with $(IM \parallel {}^{16}0)$. Place result in RA.		1038
rfci		Return from critical interrupt $(PC) \leftarrow (CSRR0)$. $(MSR) \leftarrow (CSRR1)$.		1039
rfi		Return from interrupt. $(PC) \leftarrow (SRR0)$. $(MSR) \leftarrow (SRR1)$.		1040

Table A-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
rlwimi	RA, RS, SH, MB, ME	Rotate left word immediate, then insert according to mask. r ← ROTL((RS), SH) m ← MASK(MB, ME) (RA) ← (r ∧ m) ∨ ((RA) ∧ ¬m)		1041
rlwimi.			CR[CR0]	
rlwinm	RA, RS, SH, MB, ME	Rotate left word immediate, then AND with mask. r ← ROTL((RS), SH) m ← MASK(MB, ME) (RA) ← (r ∧ m)		1042
rlwinm.			CR[CR0]	
rlwnm	RA, RS, RB, MB, ME	Rotate left word, then AND with mask. r ← ROTL((RS), $(RB)_{27:31}$) m ← MASK(MB, ME) (RA) ← (r ∧ m)		1045
rlwnm.			CR[CR0]	
rotlw	RA, RS, RB	Rotate left. (RA) ← ROTL((RS), $(RB)_{27:31}$) *Extended mnemonic for* **rlwnm RA,RS,RB,0,31**		1045
rotlw.		*Extended mnemonic for* **rlwnm. RA,RS,RB,0,31**	CR[CR0]	
rotlwi	RA, RS, n	Rotate left immediate. (RA) ← ROTL((RS), n) *Extended mnemonic for* **rlwinm RA,RS,n,0,31**		1042
rotlwi.		*Extended mnemonic for* **rlwinm. RA,RS,n,0,31**	CR[CR0]	
rotrwi	RA, RS, n	Rotate right immediate. (RA) ← ROTL((RS), 32−n) *Extended mnemonic for* **rlwinm RA,RS,32−n,0,31**		1042
rotrwi.		*Extended mnemonic for* **rlwinm. RA,RS,32−n,0,31**	CR[CR0]	
sc		System call exception is generated. (SRR1) ← (MSR) (SRR0) ← (PC) PC ← $EVPR_{0:15}$ || 0x0C00 (MSR[WE, PR, EE, PE, DR, IR]) ← 0		1046
slw	RA, RS, RB	Shift left (RS) by $(RB)_{27:31}$. n ← $(RB)_{27:31}$. r ← ROTL((RS), n). if $(RB)_{26}$ = 0 then m ← MASK(0, 31 – n) else m ← $^{32}0$ (RA) ← r ∧ m.		1047
slw.			CR[CR0]	
slwi	RA, RS, n	Shift left immediate. (n < 32) $(RA)_{0:31-n}$ ← $(RS)_{n:31}$ $(RA)_{32-n:31}$ ← $^{n}0$ *Extended mnemonic for* **rlwinm RA,RS,n,0,31−n**		1042
slwi.		*Extended mnemonic for* **rlwinm. RA,RS,n,0,31−n**	CR[CR0]	

Table A-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
sraw	RA, RS, RB	Shift right algebraic (RS) by $(RB)_{27:31}$. n ← $(RB)_{27:31}$. r ← ROTL((RS), 32 – n). if $(RB)_{26}$ = 0 then m ← MASK(n, 31) else m ← $^{32}0$ s ← $(RS)_0$ (RA) ← (r ∧ m) ∨ (^{32}s ∧ ¬m). XER[CA] ← s ∧ ((r ∧ ¬m) ≠ 0).		1048
sraw.			CR[CR0]	
srawi	RA, RS, SH	Shift right algebraic (RS) by SH. n ← SH. r ← ROTL((RS), 32 – n). m ← MASK(n, 31). s ← $(RS)_0$ (RA) ← (r ∧ m) ∨ (^{32}s ∧ ¬m). XER[CA] ← s ∧ ((r ∧ ¬m)≠0).		1049
srawi.			CR[CR0]	
srw	RA, RS, RB	Shift right (RS) by $(RB)_{27:31}$. n ← $(RB)_{27:31}$. r ← ROTL((RS), 32 – n). if $(RB)_{26}$ = 0 then m ← MASK(n, 31) else m ← $^{32}0$ (RA) ← r ∧ m.		1050
srw.			CR[CR0]	
srwi	RA, RS, n	Shift right immediate. (n < 32) $(RA)_{n:31}$ ← $(RS)_{0:31-n}$ $(RA)_{0:n-1}$ ← $^{n}0$ *Extended mnemonic for* **rlwinm RA,RS,32−n,n,31**		1042
srwi.		*Extended mnemonic for* **rlwinm. RA,RS,32−n,n,31**	CR[CR0]	
stb	RS, D(RA)	Store byte $(RS)_{24:31}$ in memory at EA = (RA|0) + EXTS(D).		1051
stbu	RS, D(RA)	Store byte $(RS)_{24:31}$ in memory at EA = (RA|0) + EXTS(D). Update the base address, (RA) ← EA.		1052
stbux	RS, RA, RB	Store byte $(RS)_{24:31}$ in memory at EA = (RA|0) + (RB). Update the base address, (RA) ← EA.		1053
stbx	RS, RA, RB	Store byte $(RS)_{24:31}$ in memory at EA = (RA|0) + (RB).		1054
sth	RS, D(RA)	Store halfword $(RS)_{16:31}$ in memory at EA = (RA|0) + EXTS(D).		1055
sthbrx	RS, RA, RB	Store halfword $(RS)_{16:31}$ byte-reversed in memory at EA = (RA|0) + (RB). MS(EA, 2) ← $(RS)_{24:31}$ ‖ $(RS)_{16:23}$		1056
sthu	RS, D(RA)	Store halfword $(RS)_{16:31}$ in memory at EA = (RA|0) + EXTS(D). Update the base address, (RA) ← EA.		1057
sthux	RS, RA, RB	Store halfword $(RS)_{16:31}$ in memory at EA = (RA|0) + (RB). Update the base address, (RA) ← EA.		1058

Table A-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
sthx	RS, RA, RB	Store halfword $(RS)_{16:31}$ in memory at EA = (RA|0) + (RB).		1059
stmw	RS, D(RA)	Store consecutive words from RS through GPR(31) in memory starting at EA = (RA|0) + EXTS(D).		1060
stswi	RS, RA, NB	Store consecutive bytes in memory starting at EA=(RA|0). Number of bytes n=32 if NB=0, else n=NB. Bytes are unstacked from CEIL(n/4) consecutive registers starting with RS. GPR(0) is consecutive to GPR(31).		1060
stswx	RS, RA, RB	Store consecutive bytes in memory starting at EA=(RA|0)+(RB). Number of bytes n=XER[TBC]. Bytes are unstacked from CEIL(n/4) consecutive registers starting with RS. GPR(0) is consecutive to GPR(31).		1063
stw	RS, D(RA)	Store word (RS) in memory at EA = (RA|0) + EXTS(D).		1064
stwbrx	RS, RA, RB	Store word (RS) byte-reversed in memory at EA = (RA|0) + (RB). $MS(EA, 4) \leftarrow (RS)_{24:31} \parallel (RS)_{16:23} \parallel (RS)_{8:15} \parallel (RS)_{0:7}$		1065

stwcx.	RS, RA, RB	Store word (RS) in memory at EA = (RA|0) + (RB)
		only if reservation bit is set.
		if RESERVE = 1 then
		MS(EA, 4) ← (RS)
		RESERVE ← 0
		$(CR[CR0]) \leftarrow {}^{2}0 \parallel 1 \parallel XER_{SO}$
		else
		$(CR[CR0]) \leftarrow {}^{2}0 \parallel 0 \parallel XER_{SO}$.
				1066

stwu	RS, D(RA)	Store word (RS) in memory at EA = (RA|0) + EXTS(D). Update the base address, (RA) ← EA.		1068
stwux	RS, RA, RB	Store word (RS) in memory at EA = (RA|0) + (RB). Update the base address, (RA) ← EA.		1069
stwx	RS, RA, RB	Store word (RS) in memory at EA = (RA|0) + (RB).		1070
sub	RT, RA, RB	Subtract (RB) from (RA). (RT) ← ¬(RB) + (RA) + 1. *Extended mnemonic for* **subf RT,RB,RA**		1071
sub.		*Extended mnemonic for* **subf. RT,RB,RA**	CR[CR0]	
subo		*Extended mnemonic for* **subfo RT,RB,RA**	XER[SO, OV]	
subo.		*Extended mnemonic for* **subfo. RT,RB,RA**	CR[CR0] XER[SO, OV]	

Table A-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
subc	RT, RA, RB	Subtract (RB) from (RA). (RT) ← ¬(RB) + (RA) + 1. Place carry-out in XER[CA]. *Extended mnemonic for* **subfc RT,RB,RA**		1072
subc.		*Extended mnemonic for* **subfc. RT,RB,RA**	CR[CR0]	
subco		*Extended mnemonic for* **subfco RT,RB,RA**	XER[SO, OV]	
subco.		*Extended mnemonic for* **subfco. RT,RB,RA**	CR[CR0] XER[SO, OV]	
subf	RT, RA, RB	Subtract (RA) from (RB). (RT) ← ¬(RA) + (RB) + 1.		1071
subf.			CR[CR0]	
subfo			XER[SO, OV]	
subfo.			CR[CR0] XER[SO, OV]	
subfc	RT, RA, RB	Subtract (RA) from (RB). (RT) ← ¬(RA) + (RB) + 1. Place carry-out in XER[CA].		1072
subfc.			CR[CR0]	
subfco			XER[SO, OV]	
subfco.			CR[CR0] XER[SO, OV]	
subfe	RT, RA, RB	Subtract (RA) from (RB) with carry-in. (RT) ← ¬(RA) + (RB) + XER[CA]. Place carry-out in XER[CA].		1073
subfe.			CR[CR0]	
subfeo			XER[SO, OV]	
subfeo.			CR[CR0] XER[SO, OV]	
subfic	RT, RA, IM	Subtract (RA) from EXTS(IM). (RT) ← ¬(RA) + EXTS(IM) + 1. Place carry-out in XER[CA].		1074
subfme	RT, RA, RB	Subtract (RA) from (–1) with carry-in. (RT) ← ¬(RA) + (–1) + XER[CA]. Place carry-out in XER[CA].		1075
subfme.			CR[CR0]	
subfmeo			XER[SO, OV]	
subfmeo.			CR[CR0] XER[SO, OV]	
subfze	RT, RA, RB	Subtract (RA) from zero with carry-in. (RT) ← ¬(RA) + XER[CA]. Place carry-out in XER[CA].		1076
subfze.			CR[CR0]	
subfzeo			XER[SO, OV]	
subfzeo.			CR[CR0] XER[SO, OV]	
subi	RT, RA, IM	Subtract EXTS(IM) from (RA|0). Place result in RT. *Extended mnemonic for* **addi RT,RA,–IM**		902

Table A-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
subic	RT, RA, IM	Subtract EXTS(IM) from (RA). Place result in RT. Place carry-out in XER[CA]. *Extended mnemonic for* **addic RT,RA,–IM**		903
subic.	RT, RA, IM	Subtract EXTS(IM) from (RA). Place result in RT. Place carry-out in XER[CA]. *Extended mnemonic for* **addic. RT,RA,–IM**	CR[CR0]	904
subis	RT, RA, IM	Subtract (IM $\|\|$ $^{16}0$) from (RA|0). Place result in RT. *Extended mnemonic for* **addis RT,RA,–IM**		905
tlbre	RT, RA,WS	tlbentry ← TLB[$(RA)_{26:31}$] if WS = 0 $(RT)_{0:27}$ ← tlbentry[EPN,V,TS,SIZE] $(RT)_{28:31}$ ← $^{4}0$ MMUCR[STID] ← tlbentry[TID] else if WS = 1 $(RT)_{0:21}$ ← tlbentry[RPN] $(RT)_{22:27}$ ← $^{6}0$ $(RT)_{28:31}$ ← tlbentry[ERPN] else if WS = 2 $(RT)_{0:15}$ ← $^{16}0$ $(RT)_{16:24}$ ← tlbentry[U0,U1,U2,U3,W,I,M,G,E] $(RT)_{25}$ ← 0 $(RT)_{26:31}$ ← tlbentry[UX,UW,UR,SX,SW,SR] else (RT), MMUCR[STID] ← undefined		1077

tlbsx
tlbsx.
	RT,RA,RB	Search the TLB for a valid entry that translates the EA.
		EA = (RA|0) + (RB)
		if Rc = 1
		$CR[CR0]_0$ ← 0
		$CR[CR0]_1$ ← 0
		$CR[CR0]_3$ ← XER[SO}
		if Valid TLB entry matching EA and MMUCR[STID,STS] is in the TLB then
		(RT) ← Index of matching TLB Entry
		if Rc = 1
		$CR[CR0]_2$ ← 1
		else
		(RT) ← Undefined
		if Rc = 1
		$CR[CR0]_2$ ← 0
			
			CR[CR0]
				1079

tlbsync		**tlbsync** does not complete until all previous TLB-update instructions executed by this processor have been received and completed by all other processors. For the PPC440GP, **tlbsync** is a no-op.		1080
tlbwe	RS, RA,WS	tlbentry ← TLB[$(RA)_{26:31}$] if WS = 0 tlbentry[EPN,V,TS,SIZE] ← $(RS)_{0:27}$ tlbentry[TID] ← MMUCR[STID] else if WS = 1 tlbentry[RPN] ← $(RS)_{0:21}$ tlbentry[ERPN] ← $(RS)_{28:31}$ else if WS = 2 tlbentry[U0,U1,U2,U3,W,I,M,G,E] ← $(RS)_{16:24}$ tlbentry[UX,UW,UR,SX,SW,SR] ← $(RS)_{26:31}$ else tlbentry ← undefined		1081

Table A-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
trap		Trap unconditionally. *Extended mnemonic for* **tw 31,0,0**		
tweq	RA, RB	Trap if (RA) equal to (RB). *Extended mnemonic for* **tw 4,RA,RB**		
twge		Trap if (RA) greater than or equal to (RB). *Extended mnemonic for* **tw 12,RA,RB**		
twgt		Trap if (RA) greater than (RB). *Extended mnemonic for* **tw 8,RA,RB**		
twle		Trap if (RA) less than or equal to (RB). *Extended mnemonic for* **tw 20,RA,RB**		
twlge		Trap if (RA) logically greater than or equal to (RB). *Extended mnemonic for* **tw 5,RA,RB**		
twlgt		Trap if (RA) logically greater than (RB). *Extended mnemonic for* **tw 1,RA,RB**		
twlle		Trap if (RA) logically less than or equal to (RB). *Extended mnemonic for* **tw 6,RA,RB**		1082
twllt		Trap if (RA) logically less than (RB). *Extended mnemonic for* **tw 2,RA,RB**		
twlng		Trap if (RA) logically not greater than (RB). *Extended mnemonic for* **tw 6,RA,RB**		
twlnl		Trap if (RA) logically not less than (RB). *Extended mnemonic for* **tw 5,RA,RB**		
twlt		Trap if (RA) less than (RB). *Extended mnemonic for* **tw 16,RA,RB**		
twne		Trap if (RA) not equal to (RB). *Extended mnemonic for* **tw 24,RA,RB**		
twng		Trap if (RA) not greater than (RB). *Extended mnemonic for* *tw 20,RA,RB*		
twnl		Trap if (RA) not less than (RB). *Extended mnemonic for* **tw 12,RA,RB**		
tw	TO, RA, RB	Trap exception is generated if, comparing (RA) with (RB), any condition specified by TO is true.		1082

Table A-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
tweqi	RA, IM	Trap if (RA) equal to EXTS(IM). *Extended mnemonic for* **wi 4,RA,IM**		1085
twgei		Trap if (RA) greater than or equal to EXTS(IM). *Extended mnemonic for* **twi 12,RA,IM**		
twgti		Trap if (RA) greater than EXTS(IM). *Extended mnemonic for* **twi 8,RA,IM**		
twlei		Trap if (RA) less than or equal to EXTS(IM). *Extended mnemonic for* **twi 20,RA,IM**		
twlgei		Trap if (RA) logically greater than or equal to EXTS(IM). *Extended mnemonic for* **wi 5,RA,IM**		
twlgti		Trap if (RA) logically greater than EXTS(IM). *Extended mnemonic for* **twi 1,RA,IM**		
twllei		Trap if (RA) logically less than or equal to EXTS(IM). *Extended mnemonic for* **twi 6,RA,IM**		
twllti		Trap if (RA) logically less than EXTS(IM). *Extended mnemonic for* **twi 2,RA,IM**		
twlngi		Trap if (RA) logically not greater than EXTS(IM). *Extended mnemonic for* **twi 6,RA,IM**		
twlnli		Trap if (RA) logically not less than EXTS(IM). *Extended mnemonic for* **twi 5,RA,IM**		
twlti		Trap if (RA) less than EXTS(IM). *Extended mnemonic for* **twi 16,RA,IM**		
twnei		Trap if (RA) not equal to EXTS(IM). *Extended mnemonic for* **twi 24,RA,IM**		
twngi		Trap if (RA) not greater than EXTS(IM). *Extended mnemonic for* **twi 20,RA,IM**		
twnli		Trap if (RA) not less than EXTS(IM). *Extended mnemonic for* **twi 12,RA,IM**		
twi	TO, RA, IM	Trap exception is generated if, comparing (RA) with EXTS(IM), any condition specified by TO is true.		1085
wrtee	RS	Write value of RS_{16} to MSR[EE].		1088
wrteei	E	Write value of E to MSR[EE].		1089
xor	RA, RS, RB	XOR (RS) with (RB). Place result in RA.		1090
xor.			CR[CR0]	

Table A-1. PPC440GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
xori	RA, RS, IM	XOR (RS) with ($^{16}0$ || IM). Place result in RA.		1091
xoris	RA, RS, IM	XOR (RS) with (IM || $^{16}0$). Place result in RA.		1092

A.3 Allocated Instruction Opcodes

Allocated instructions are provided for purposes that are outside the scope of PowerPC Book-E architecture, and are for implementation-dependent and application-specific use, including use within auxiliary processors.

Table A-2 lists the blocks of opcodes which have been allocated by PowerPC Book-E for these purposes. In the table, the character "u" designates a secondary opcode bit which can be set to any value. In some cases, the decimal value of a secondary opcode is shown in parentheses after the binary value.

Table A-2. Allocated Opcodes

Primary Opcode	Extended Opcodes	PPC440GP Usage
0	All instruction encodings (bits 6:31) except 0x00000000 (the instruction encoding of 0x00000000 is and always will be reserved-illegal)	None
4	All instruction encodings (bits 6:31)	Various (see ~~Table A-5 on page A-43~~*Table A-5* on page 1577)
19	Secondary opcodes (bits 21:30) = 0buuuuu0u11u	None
31	Secondary opcodes (bits 21:30) = 0buuuuu0u11u Secondary opcode (bits 21:30) = 0b0101010110 (342) Secondary opcode (bits 21:30) = 0b0101110110 (374) Secondary opcode (bits 21:30) = 0b1100110110 (822)	Various (see ~~Table A-5 on page A-43~~*Table A-5* on page 1577)
59	Secondary opcodes (bits 21:30) = 0buuuuu0u10u	None
63	Secondary opcodes (bits 21:30) = 0buuuuu0u10u (except secondary opcode decimal 12, which is the **fsrp** defined instruction)	None

All of the allocated opcodes listed in the table above are available for use by auxiliary processors attached to the PPC440GP, except for those which have already been implemented within the PPC440GP for certain implementation-specific purposes. As indicated in the table above, this is the case for certain secondary opcodes within primary opcodes 4 and 31. These opcodes are identified in ~~Table A-5 on page A-43~~*Table A-5* on page 1577, along with all of the defined, preserved, and reserved-nop class opcodes which are implemented within the PPC440GP.

A.4 Preserved Instruction Opcodes

The preserved instruction class is provided to support backward compatibility with the PowerPC Architecture, and/or earlier versions of the PowerPC Book-E architecture. This instruction class includes opcodes which were defined for these previous architectures, but which are no longer defined for PowerPC Book-E.

Table A-3 lists the reserved opcodes designated by PowerPC Book-E. The decimal value of the secondary opcode is shown in parentheses after the binary value.

Table A-3. Preserved Opcodes

Primary Opcode	Extended Opcode	Preserved Mnemonic	PPC440GP Usage
31	0b00110 10010 (210)	mtsr	
31	0b00111 10010 (242)	mtsrin	
31	0b01011 10010 (370)	tlbia	
31	0b01001 10010 (306)	tlbie	
31	0b01011 10011 (371)	mftb	Yes
31	0b10010 10011 (595)	mfsr	
31	0b10100 10011 (659)	mfsrin	
31	0b01001 10110 (310)	eciwx	
31	0b01101 10110 (438)	ecowx	

As indicated in the table above, the only preserved opcode which is implemented within the PPC440GP is the **mftb** instruction. See ~~"Preserved Instruction Class" on page 4-37~~*Preserved Instruction Class* on page 205 for more information on PPC440GP support for this instruction. All other preserved instructions are treated as reserved by PPC440GP and will cause Illegal Instruction exception type Program interrupts if their execution is attempted.

The preserved opcode for **mftb** is included in ~~Table A-5 on page A-43~~*Table A-5* on page 1577, along with all of the defined, allocated, and reserved-nop class opcodes which are implemented within the PPC440GP.

A.5 Reserved Instruction Opcodes

This class of instructions consists of all instruction primary opcodes (and associated extended opcodes, if applicable) which do not belong to either the defined, allocated, or preserved instruction classes.

Reserved instructions are available for future versions of PowerPC Book-E architecture. That is, future versions of PowerPC Book-E may define any of these instructions to perform new functions or make them available for implementation-dependent use as allocated instructions. There are two types of reserved instructions: reserved-illegal and reserved-nop.

Table A-4 lists the reserved-nop opcodes designated by PowerPC Book-E. In the table, the character "u" designates a secondary opcode bit which can be set to any value. All other reserved opcodes are in the reserved-illegal class.

Table A-4. Reserved-nop Opcodes

Primary Opcode	Extended Opcode
31	0b10uuu 10010

As shown in the table, there are a total of eight (8) secondary opcodes in the reserved-nop class. The PPC440GP implements all of the reserved-nop instruction opcodes as true no-ops. These opcodes are included in ~~Table A-5 on page A-43~~*Table A-5* on page 1577, along with all of the defined, allocated, and preserved class opcodes which are implemented within the PPC440GP.

A.6 Implemented Instructions Sorted by Opcode

~~Table A-5 on page A-43~~*Table A-5* on page 1577 lists all of the instructions which have been implemented within the PPC440GP, sorted by primary and secondary opcode. These include defined, allocated, preserved, and reserved-nop class instructions (see ~~"Instruction Classes" on page 4-35~~*Instruction Classes* on page 203 for a more detailed description of each of these instruction classes). Opcodes which are *not* implemented in the PPC440GP are *not* shown in the table, and consist of the following:

- Defined instructions

  These include the floating-point operations (which may be implemented in an auxiliary processor and executed via the AP interface), as well as the 64-bit operations and the **tlbiva** and **mfapidi** instructions, all of which are handled as reserved-illegal instructions by the PPC440GP.

- Allocated instructions

  These include all of the allocated opcodes identified in ~~Table A-2 on page A-40~~*Table A-2* on page 1575 which are not already implemented within the PPC440GP. If not implemented within an attached auxiliary processor, these instructions will be handled as reserved-illegal by the PPC440GP.

- Preserved instructions

  These include all of the preserved opcodes identified in ~~Table A-3 on page A-41~~*Table A-3* on page 1576 except for the **mftb** opcode (which *is* implemented and thus included in Table A-5). These instructions will be handled as reserved-illegal by the PPC440GP.

- Reserved instructions

  These include all of the reserved opcodes as defined by ~~"Reserved Instruction Opcodes," on page A-41~~*Appendix A.5* on page 1576, except for the reserved-nop opcodes identified in ~~Table A-4 on page A-42~~*Table A-4* on page 1576. These instructions by definition are all in the reserved-illegal class and will be handled as such by the PPC440GP.

All PowerPC Book-E instructions are four bytes long and word aligned. All instructions have a primary opcode field (shown as field OPCD in Figure A-1 through ~~Figure A-9, beginning on page A-3~~*Figure A-9* on page 1544) in bits 0:5. Some instructions also have a secondary opcode field (shown as field XO in Figure A-1 through Figure A-9).

The "Form" indicated in the table refers to the arrangement of valid field combinations within the four-byte instruction. See ~~"Instruction Formats," on page A-1~~*Appendix A.1* on page 1539, for the field layouts of each form.

Form X has a 10-bit secondary opcode field, while form XO uses only the low-order 9-bits of that field. Form XO uses the high-order secondary opcode bit (the tenth bit) as a variable; therefore, every form XO instruction really consumes two secondary opcodes from the 10-bit secondary-opcode space. The implicitly consumed secondary opcode is listed in parentheses for form XO instructions in the table below.

Table B-2. PPC440GP Instructions by Opcode

Primary Opcode	Secondary Opcode	Form	Mnemonic	Operands	Page
3		D	**twi**	TO, RA, IM	28-190
4	8	X	**mulhhwu**	RT, RA, RB	28-126
			mulhhwu.		

Table B-2. PPC440GP Instructions by Opcode (continued)

Primary Opcode	Secondary Opcode	Form	Mnemonic	Operands	Page
4	12 (524)	XO	**machhwu**	RT, RA, RB	28-102
			machhwu.		
			machhwuo		
			machhwuo.		
4	40	X	**mulhhw**	RT, RA, RB	28-125
			mulhhw.		
4	44 (556)	XO	**machhw**	RT, RA, RB	28-99
			machhw.		
			machhwo		
			machhwo.		
4	46 (558)	XO	**nmachhw**	RT, RA, RB	28-137
			nmachhw.		
			nmachhwo		
			nmachhwo.		
4	76 (588)	XO	**machhwsu**	RT, RA, RB	28-101
			machhwsu.		
			machhwsuo		
			machhwsuo.		
4	108 (620)	XO	**machhws**	RT, RA, RB	28-100
			machhws.		
			machhwso		
			machhwso.		
4	110 (622)	XO	**nmachhws**	RT, RA, RB	28-138
			nmachhws.		
			nmachhwso		
			nmachhwso.		
4	136	X	**mulchwu**	RT, RA, RB	28-124
			mulchwu.		
4	140 (652)	XO	**macchwu**	RT, RA, RB	28-98
			macchwu.		
			macchwuo		
			machhwuo.		
4	168	X	**mulchw**	RT, RA, RB	28-123
			mulchw.		
4	172 (684)	XO	**macchw**	RT, RA, RB	28-95
			macchw.		
			macchwo		
			macchwo.		

Table B-2. PPC440GP Instructions by Opcode (continued)

Primary Opcode	Secondary Opcode	Form	Mnemonic	Operands	Page
4	174 (686)	XO	**nmacchw**	RT, RA, RB	28-135
			nmacchw.		
			nmacchwo		
			nmacchwo.		
4	204 (716)	XO	**macchwsu**	RT, RA, RB	28-97
			macchwsu.		
			macchwsuo		
			macchwsuo.		
4	236 (748)	XO	**macchws**	RT, RA, RB	28-96
			macchws.		
			macchwso		
			macchwso.		
4	238 (750)	XO	**nmacchws**	RT, RA, RB	28-136
			nmacchws.		
			nmacchwso		
			nmacchwso.		
4	392	X	**mullhwu**	RT, RA, RB	28-130
			mullhwu.		
4	396 (908)	XO	**maclhwu**	RT, RA, RB	28-106
			maclhwu.		
			maclhwuo		
			maclhwuo.		
4	424	X	**mullhw**	RT, RA, RB	28-129
			mullhw.		
4	428 (940)	XO	**maclhw**	RT, RA, RB	28-103
			maclhw.		
			maclhwo		
			maclhwo.		
4	430 (942)	XO	**nmaclhw**	RT, RA, RB	28-139
			nmaclhw.		
			nmaclhwo		
			nmaclhwo.		
4	460 (972)	XO	**maclhwsu**	RT, RA, RB	28-105
			maclhwsu.		
			maclhwsuo		
			maclhwsuo.		
4	492 (1004)	XO	**maclhws**	RT, RA, RB	28-104
			maclhws.		
			maclhwso		
			maclhwso.		

Table B-2. PPC440GP Instructions by Opcode (continued)

Primary Opcode	Secondary Opcode	Form	Mnemonic	Operands	Page
4	494 (1006)	XO	**nmaclhws**	RT, RA, RB	28-140
			nmaclhws.		
			nmaclhwso		
			nmaclhwso.		
7		D	**mulli**	RT, RA, IM	28-131
8		D	**subfic**	RT, RA, IM	28-180
10		D	**cmpli**	BF, 0, RA, IM	28-38
11		D	**cmpi**	BF, 0, RA, IM	28-36
12		D	**addic**	RT, RA, IM	28-11
13		D	**addic.**	RT, RA, IM	28-12
14		D	**addi**	RT, RA, IM	28-10
15		D	**addis**	RT, RA, IM	28-13
16		B	**bc**	BO, BI, target	28-21
			bca		
			bcl		
			bcla		
17		SC	**sc**		28-152
18		I	**b**	target	28-20
			ba		
			bl		
			bla		
19	0	XL	**mcrf**	BF, BFA	28-108
19	16	XL	**bclr**	BO, BI	28-31
			bclrl		
19	33	XL	**crnor**	BT, BA, BB	28-44
19	50	XL	**rfi**		28-147
19	51	XL	**rfci**		28-146
19	129	XL	**crandc**	BT, BA, BB	28-41
19	150	XL	**isync**		28-70
19	193	XL	**crxor**	BT, BA, BB	28-47
19	225	XL	**crnand**	BT, BA, BB	28-43
19	257	XL	**crand**	BT, BA, BB	28-40
19	289	XL	**creqv**	BT, BA, BB	28-42
19	417	XL	**crorc**	BT, BA, BB	28-46
19	449	XL	**cror**	BT, BA, BB	28-45
19	528	XL	**bcctr**	BO, BI	28-27
			bcctrl		
20		M	**rlwimi**	RA, RS, SH, MB, ME	28-148
			rlwimi.		

Table B-2. PPC440GP Instructions by Opcode (continued)

Primary Opcode	Secondary Opcode	Form	Mnemonic	Operands	Page
21		M	**rlwinm**	RA, RS, SH, MB, ME	28-149
			rlwinm.		
23		M	**rlwnm**	RA, RS, RB, MB, ME	28-151
			rlwnm.		
24		D	**ori**	RA, RS, IM	28-144
25		D	**oris**	RA, RS, IM	28-145
26		D	**xori**	RA, RS, IM	28-195
27		D	**xoris**	RA, RS, IM	28-196
28		D	**andi.**	RA, RS, IM	28-18
29		D	**andis.**	RA, RS, IM	28-19
31	0	X	**cmp**	BF, 0, RA, RB	28-35
31	4	X	**tw**	TO, RA, RB	28-187
31	8 (520)	XO	**subfc**	RT, RA, RB	28-178
			subfc.		
			subfco		
			subfco.		
31	10 (522)	XO	**addc**	RT, RA, RB	28-8
			addc.		
			addco		
			addco.		
31	11 (523)	XO	**mulhwu**	RT, RA, RB	28-128
			mulhwu.		
31	19	X	**mfcr**	RT	28-110
31	20	X	**lwarx**	RT, RA, RB	28-89
31	22	X	**icbt**	RA, RB	28-65
31	23	X	**lwzx**	RT, RA, RB	28-94
31	24	X	**slw**	RA, RS, RB	28-153
			slw.		
31	26	X	**cntlzw**	RA, RS	28-39
			cntlzw.		
31	28	X	**and**	RA, RS, RB	28-16
			and.		
31	32	X	**cmpl**	BF, 0, RA, RB	28-37
31	40 (552)	XO	**subf**	RT, RA, RB	28-177
			subf.		
			subfo		
			subfo.		
31	54	X	**dcbst**	RA, RB	28-53
31	55	X	**lwzux**	RT, RA, RB	28-93

Table B-2. PPC440GP Instructions by Opcode (continued)

Primary Opcode	Secondary Opcode	Form	Mnemonic	Operands	Page
31	60	X	**andc**	RA, RS, RB	28-17
			andc.		
31	75 (587)	XO	**mulhw**	RT, RA, RB	28-127
			mulhw.		
31	78	X	**dlmzb**	RA, RS, RB	28-60
			dlmzb.		
31	83	X	**mfmsr**	RT	28-112
31	86	X	**dcbf**	RA, RB	28-49
31	87	X	**lbzx**	RT, RA, RB	28-74
31	104 (616)	XO	**neg**	RT, RA	28-134
			neg.		
			nego		
			nego.		
31	119	X	**lbzux**	RT, RA, RB	28-73
31	124	X	**nor**	RA, RS, RB	28-141
			nor.		
31	131	X	**wrtee**	RS	28-192
31	136 (648)	XO	**subfe**	RT, RA, RB	28-179
			subfe.		
			subfeo		
			subfeo.		
31	138 (650)	XO	**adde**	RT, RA, RB	28-9
			adde.		
			addeo		
			addeo.		
31	144	XFX	**mtcrf**	FXM, RS	28-117
31	146	X	**mtmsr**	RS	28-119
31	150	X	**stwcx.**	RS, RA, RB	28-172
31	151	X	**stwx**	RS, RA, RB	28-176
31	163	X	**wrteei**	E	28-193
31	183	X	**stwux**	RS, RA, RB	28-175
31	200 (712)	XO	**subfze**	RT, RA, RB	28-182
			subfze.		
			subfzeo		
			subfzeo.		
31	202 (714)	XO	**addze**	RT, RA	28-15
			addze.		
			addzeo		
			addzeo.		
31	215	X	**stbx**	RS, RA, RB	28-160

Table B-2. PPC440GP Instructions by Opcode (continued)

Primary Opcode	Secondary Opcode	Form	Mnemonic	Operands	Page
31	232 (744)	XO	**subfme**	RT, RA, RB	28-181
			subfme.		
			subfmeo		
			subfmeo.		
31	234 (746)	XO	**addme**	RT, RA	28-14
			addme.		
			addmeo		
			addmeo.		
31	235 (747)	XO	**mullw**	RT, RA, RB	28-132
			mullw.		
			mullwo		
			mullwo.		
31	246	X	**dcbtst**	RA,RB	28-53
31	247	X	**stbux**	RS, RA, RB	28-158
31	262	X	**icbt**	RA, RB	28-65
31	266 (778)	XO	**add**	RT, RA, RB	28-7
			add.		
			addo		
			addo.		
31	278	X	**dcbt**	RA, RB	28-51
31	279	X	**lhzx**	RT, RA, RB	28-83
31	284	X	**eqv**	RA, RS, RB	28-61
			eqv.		
31	311	X	**lhzux**	RT, RA, RB	28-82
31	316	X	**xor**	RA, RS, RB	28-194
			xor.		
31	323	XFX	**mfdcr**	RT, DCRN	28-111
31	339	XFX	**mfspr**	RT, SPRN	28-113
31	343	X	**lhax**	RT, RA, RB	28-78
31	371	XFX	**mftb**	RT, SPRN	A-41
31	375	X	**lhaux**	RT, RA, RB	28-77
31	407	X	**sthx**	RS, RA, RB	28-165
31	412	X	**orc**	RA, RS, RB	28-143
			orc.		
31	439	X	**sthux**	RS, RA, RB	28-164
31	444	X	**or**	RA, RS, RB	28-142
			or.		
31	451	XFX	**mtdcr**	DCRN, RS	28-118
31	454	X	**dccci**	RA, RB	28-55

Table B-2. PPC440GP Instructions by Opcode (continued)

Primary Opcode	Secondary Opcode	Form	Mnemonic	Operands	Page
31	459 (971)	XO	**divwu**	RT, RA, RB	28-59
			divwu.		
			divwuo		
			divwuo.		
31	467	XFX	**mtspr**	SPRN, RS	28-120
31	470	X	**dcbi**	RA, RB	28-50
31	476	X	**nand**	RA, RS, RB	28-133
			nand.		
31	486	X	**dcread**	RT, RA, RB	28-56
31	491 (1003)	XO	**divw**	RT, RA, RB	28-58
			divw.		
			divwo		
			divwo.		
31	512	X	**mcrxr**	BF	28-109
31	530		Reserved-nop		A-41
31	533	X	**lswx**	RT, RA, RB	28-87
31	534	X	**lwbrx**	RT, RA, RB	28-90
31	536	X	**srw**	RA, RS, RB	28-156
			srw.		
31	562		Reserved-nop		A-41
31	566	X	**tlbsync**		28-185
31	594		Reserved-nop		A-41
31	597	X	**lswi**	RT, RA, NB	28-85
31	598	X	**sync**		28-116
31	626		Reserved-nop		A-41
31	658		Reserved-nop		A-41
31	661	X	**stswx**	RS, RA, RB	28-169
31	662	X	**stwbrx**	RS, RA, RB	28-171
31	690		Reserved-nop		A-41
31	722		Reserved-nop		A-41
31	725	X	**stswi**	RS, RA, NB	28-166
31	754		Reserved-nop		A-41
31	758	X	**dcba**	RA, RB	28-48
31	790	X	**lhbrx**	RT, RA, RB	28-79
31	792	X	**sraw**	RA, RS, RB	28-154
			sraw.		
31	824	X	**srawi**	RA, RS, SH	28-155
			srawi.		
31	854	X	**eieio**		28-107

Table B-2. PPC440GP Instructions by Opcode (continued)

Primary Opcode	Secondary Opcode	Form	Mnemonic	Operands	Page
31	914	X	**tlbsx**	RT,RA,RB	28-184
			tlbsx.		
31	918	X	**sthbrx**	RS, RA, RB	28-162
31	922	X	**extsh**	RA, RS	28-63
			extsh.		
31	946	X	**tlbre**	RT, RA,WS	28-183
31	954	X	**extsb**	RA, RS	28-62
			extsb.		
31	966	X	**iccci**	RA, RB	28-67
31	978	X	**tlbwe**	RS, RA,WS	28-186
31	982	X	**icbi**	RA, RB	28-64
31	998	X	**icread**	RA, RB	28-68
31	1014	X	**dcbz**	RA, RB	28-54
32		D	**lwz**	RT, D(RA)	28-91
33		D	**lwzu**	RT, D(RA)	28-92
34		D	**lbz**	RT, D(RA)	28-71
35		D	**lbzu**	RT, D(RA)	28-72
36		D	**stw**	RS, D(RA)	28-170
37		D	**stwu**	RS, D(RA)	28-174
38		D	**stb**	RS, D(RA)	28-157
39		D	**stbu**	RS, D(RA)	28-158
40		D	**lhz**	RT, D(RA)	28-80
41		D	**lhzu**	RT, D(RA)	28-81
42		D	**lha**	RT, D(RA)	28-75
43		D	**lhau**	RT, D(RA)	28-76
44		D	**sth**	RS, D(RA)	28-161
45		D	**sthu**	RS, D(RA)	28-163
46		D	**lmw**	RT, D(RA)	28-84
47		D	**stmw**	RS, D(RA)	28-166

Table A-5. PPC440GP Instructions by Opcode

Primary Opcode	Secondary Opcode	Form	Mnemonic	Operands	Page
3		D	**twi**	TO, RA, IM	1085
4	8	X	mulhhwu	RT, RA, RB	1019
			mulhhwu.		
4	12 (524)	XO	machhwu	RT, RA, RB	995
			machhwu.		
			machhwuo		
			machhwuo.		

Table A-5. PPC440GP Instructions by Opcode (continued)

Primary Opcode	Secondary Opcode	Form	Mnemonic	Operands	Page
4	40	X	mulhhw	RT, RA, RB	1018
			mulhhw.		
4	44 (556)	XO	machhw	RT, RA, RB	992
			machhw.		
			machhwo		
			machhwo.		
4	46 (558)	XO	nmachhw	RT, RA, RB	1030
			nmachhw.		
			nmachhwo		
			nmachhwo.		
4	76 (588)	XO	machhwsu	RT, RA, RB	994
			machhwsu.		
			machhwsuo		
			machhwsuo.		
4	108 (620)	XO	machhws	RT, RA, RB	993
			machhws.		
			machhwso		
			machhwso.		
4	110 (622)	XO	nmachhws	RT, RA, RB	1031
			nmachhws.		
			nmachhwso		
			nmachhwso.		
4	136	X	mulchwu	RT, RA, RB	1017
			mulchwu.		
4	140 (652)	XO	macchwu	RT, RA, RB	991
			macchwu.		
			macchwuo		
			machhwuo.		
4	168	X	mulchw	RT, RA, RB	1016
			mulchw.		
4	172 (684)	XO	macchw	RT, RA, RB	988
			macchw.		
			macchwo		
			macchwo.		
4	174 (686)	XO	nmacchw	RT, RA, RB	1028
			nmacchw.		
			nmacchwo		
			nmacchwo.		
4	204 (716)	XO	macchwsu	RT, RA, RB	990
			macchwsu.		
			macchwsuo		
			macchwsuo.		

Table A-5. PPC440GP Instructions by Opcode (continued)

Primary Opcode	Secondary Opcode	Form	Mnemonic	Operands	Page
4	236 (748)	XO	macchws	RT, RA, RB	989
			macchws.		
			macchwso		
			macchwso.		
4	238 (750)	XO	nmacchws	RT, RA, RB	1029
			nmacchws.		
			nmacchwso		
			nmacchwso.		
4	392	X	mullhwu	RT, RA, RB	1023
			mullhwu.		
4	396 (908)	XO	maclhwu	RT, RA, RB	999
			maclhwu.		
			maclhwuo		
			maclhwuo.		
4	424	X	mullhw	RT, RA, RB	1022
			mullhw.		
4	428 (940)	XO	maclhw	RT, RA, RB	996
			maclhw.		
			maclhwo		
			maclhwo.		
4	430 (942)	XO	nmaclhw	RT, RA, RB	1032
			nmaclhw.		
			nmaclhwo		
			nmaclhwo.		
4	460 (972)	XO	maclhwsu	RT, RA, RB	998
			maclhwsu.		
			maclhwsuo		
			maclhwsuo.		
4	492 (1004)	XO	maclhws	RT, RA, RB	997
			maclhws.		
			maclhwso		
			maclhwso.		
4	494 (1006)	XO	nmaclhws	RT, RA, RB	1033
			nmaclhws.		
			nmaclhwso		
			nmaclhwso.		
7		D	**mulli**	RT, RA, IM	1024
8		D	**subfic**	RT, RA, IM	1074
10		D	**cmpli**	BF, 0, RA, IM	929
11		D	**cmpi**	BF, 0, RA, IM	927
12		D	**addic**	RT, RA, IM	903
13		D	**addic.**	RT, RA, IM	904
14		D	**addi**	RT, RA, IM	902
15		D	**addis**	RT, RA, IM	905

Table A-5. PPC440GP Instructions by Opcode (continued)

Primary Opcode	Secondary Opcode	Form	Mnemonic	Operands	Page
16		B	**bc**	BO, BI, target	913
			bca		
			bcl		
			bcla		
17		SC	**sc**		1046
18		I	**b**	target	912
			ba		
			bl		
			bla		
19	0	XL	**mcrf**	BF, BFA	1001
19	16	XL	**bclr**	BO, BI	922
			bclrl		
19	33	XL	**crnor**	BT, BA, BB	935
19	50	XL	**rfi**		1040
19	51	XL	**rfci**		1039
19	129	XL	**crandc**	BT, BA, BB	932
19	150	XL	**isync**		963
19	193	XL	**crxor**	BT, BA, BB	938
19	225	XL	**crnand**	BT, BA, BB	934
19	257	XL	**crand**	BT, BA, BB	931
19	289	XL	**creqv**	BT, BA, BB	933
19	417	XL	**crorc**	BT, BA, BB	937
19	449	XL	**cror**	BT, BA, BB	936
19	528	XL	**bcctr**	BO, BI	919
			bcctrl		
20		M	**rlwimi**	RA, RS, SH, MB, ME	1041
			rlwimi.		
21		M	**rlwinm**	RA, RS, SH, MB, ME	1042
			rlwinm.		
23		M	**rlwnm**	RA, RS, RB, MB, ME	1045
			rlwnm.		
24		D	**ori**	RA, RS, IM	1037
25		D	**oris**	RA, RS, IM	1038
26		D	**xori**	RA, RS, IM	1091
27		D	**xoris**	RA, RS, IM	1092
28		D	**andi.**	RA, RS, IM	910
29		D	**andis.**	RA, RS, IM	911
31	0	X	**cmp**	BF, 0, RA, RB	926
31	4	X	**tw**	TO, RA, RB	1082

Table A-5. PPC440GP Instructions by Opcode (continued)

Primary Opcode	Secondary Opcode	Form	Mnemonic	Operands	Page
31	8 (520)	XO	**subfc**	RT, RA, RB	1072
			subfc.		
			subfco		
			subfco.		
31	10 (522)	XO	**addc**	RT, RA, RB	900
			addc.		
			addco		
			addco.		
31	11 (523)	XO	**mulhwu**	RT, RA, RB	1021
			mulhwu.		
31	19	X	**mfcr**	RT	1003
31	20	X	**lwarx**	RT, RA, RB	982
31	22	X	**icbt**	RA, RB	958
31	23	X	**lwzx**	RT, RA, RB	987
31	24	X	**slw**	RA, RS, RB	1047
			slw.		
31	26	X	**cntlzw**	RA, RS	930
			cntlzw.		
31	28	X	**and**	RA, RS, RB	908
			and.		
31	32	X	**cmpl**	BF, 0, RA, RB	928
31	40 (552)	XO	**subf**	RT, RA, RB	1071
			subf.		
			subfo		
			subfo.		
31	54	X	**dcbst**	RA, RB	944
31	55	X	**lwzux**	RT, RA, RB	986
31	60	X	**andc**	RA, RS, RB	909
			andc.		
31	75 (587)	XO	**mulhw**	RT, RA, RB	1020
			mulhw.		
31	78	X	**dlmzb**	RA, RS, RB	953
			dlmzb.		
31	83	X	**mfmsr**	RT	1005
31	86	X	**dcbf**	RA, RB	940
31	87	X	**lbzx**	RT, RA, RB	967
31	104 (616)	XO	**neg**	RT, RA	1027
			neg.		
			nego		
			nego.		

Table A-5. PPC440GP Instructions by Opcode (continued)

Primary Opcode	Secondary Opcode	Form	Mnemonic	Operands	Page
31	119	X	**lbzux**	RT, RA, RB	966
31	124	X	**nor**	RA, RS, RB	1034
			nor.		
31	131	X	**wrtee**	RS	1088
31	136 (648)	XO	**subfe**	RT, RA, RB	1073
			subfe.		
			subfeo		
			subfeo.		
31	138 (650)	XO	**adde**	RT, RA, RB	901
			adde.		
			addeo		
			addeo.		
31	144	XFX	**mtcrf**	FXM, RS	1010
31	146	X	**mtmsr**	RS	1012
31	150	X	**stwcx.**	RS, RA, RB	1066
31	151	X	**stwx**	RS, RA, RB	1070
31	163	X	**wrteei**	E	1089
31	183	X	**stwux**	RS, RA, RB	1069
31	200 (712)	XO	**subfze**	RT, RA, RB	1076
			subfze.		
			subfzeo		
			subfzeo.		
31	202 (714)	XO	**addze**	RT, RA	907
			addze.		
			addzeo		
			addzeo.		
31	215	X	**stbx**	RS, RA, RB	1054
31	232 (744)	XO	**subfme**	RT, RA, RB	1075
			subfme.		
			subfmeo		
			subfmeo.		
31	234 (746)	XO	**addme**	RT, RA	906
			addme.		
			addmeo		
			addmeo.		
31	235 (747)	XO	**mullw**	RT, RA, RB	1025
			mullw.		
			mullwo		
			mullwo.		
31	246	X	**dcbtst**	RA,RB	944

Table A-5. PPC440GP Instructions by Opcode (continued)

Primary Opcode	Secondary Opcode	Form	Mnemonic	Operands	Page
31	247	X	**stbux**	RS, RA, RB	1052
31	262	X	**icbt**	RA, RB	958
31	266 (778)	XO	**add**	RT, RA, RB	899
			add.		
			addo		
			addo.		
31	278	X	**dcbt**	RA, RB	942
31	279	X	**lhzx**	RT, RA, RB	976
31	284	X	**eqv**	RA, RS, RB	954
			eqv.		
31	311	X	**lhzux**	RT, RA, RB	975
31	316	X	**xor**	RA, RS, RB	1090
			xor.		
31	323	XFX	**mfdcr**	RT, DCRN	1004
31	339	XFX	**mfspr**	RT, SPRN	1006
31	343	X	**lhax**	RT, RA, RB	971
31	371	XFX	**mftb**	RT, SPRN	1575
31	375	X	**lhaux**	RT, RA, RB	970
31	407	X	**sthx**	RS, RA, RB	1059
31	412	X	**orc**	RA, RS, RB	1036
			orc.		
31	439	X	**sthux**	RS, RA, RB	1058
31	444	X	**or**	RA, RS, RB	1035
			or.		
31	451	XFX	**mtdcr**	DCRN, RS	1011
31	454	X	**dccci**	RA, RB	948
31	459 (971)	XO	**divwu**	RT, RA, RB	952
			divwu.		
			divwuo		
			divwuo.		
31	467	XFX	**mtspr**	SPRN, RS	1013
31	470	X	**dcbi**	RA, RB	941
31	476	X	**nand**	RA, RS, RB	1026
			nand.		
31	486	X	**dcread**	RT, RA, RB	949
31	491 (1003)	XO	**divw**	RT, RA, RB	951
			divw.		
			divwo		
			divwo.		
31	512	X	**mcrxr**	BF	1002

Table A-5. PPC440GP Instructions by Opcode (continued)

Primary Opcode	Secondary Opcode	Form	Mnemonic	Operands	Page
31	530		Reserved-nop		1576
31	533	X	**lswx**	RT, RA, RB	980
31	534	X	**lwbrx**	RT, RA, RB	983
31	536	X	**srw**	RA, RS, RB	1050
			srw.		
31	562		Reserved-nop		1576
31	566	X	**tlbsync**		1080
31	594		Reserved-nop		1576
31	597	X	**lswi**	RT, RA, NB	978
31	598	X	**sync**		1009
31	626		Reserved-nop		1576
31	658		Reserved-nop		1576
31	661	X	**stswx**	RS, RA, RB	1063
31	662	X	**stwbrx**	RS, RA, RB	1065
31	690		Reserved-nop		1576
31	722		Reserved-nop		1576
31	725	X	**stswi**	RS, RA, NB	1060
31	754		Reserved-nop		1576
31	758	X	**dcba**	RA, RB	939
31	790	X	**lhbrx**	RT, RA, RB	972
31	792	X	**sraw**	RA, RS, RB	1048
			sraw.		
31	824	X	**srawi**	RA, RS, SH	1049
			srawi.		
31	854	X	**eieio**		1000
31	914	X	**tlbsx**	RT,RA,RB	1079
			tlbsx.		
31	918	X	**sthbrx**	RS, RA, RB	1056
31	922	X	**extsh**	RA, RS	956
			extsh.		
31	946	X	**tlbre**	RT, RA,WS	1077
31	954	X	**extsb**	RA, RS	955
			extsb.		
31	966	X	**iccci**	RA, RB	960
31	978	X	**tlbwe**	RS, RA,WS	1081
31	982	X	**icbi**	RA, RB	957
31	998	X	**icread**	RA, RB	961
31	1014	X	**dcbz**	RA, RB	946
32		D	**lwz**	RT, D(RA)	984
33		D	**lwzu**	RT, D(RA)	985
34		D	**lbz**	RT, D(RA)	964

Table A-5. PPC440GP Instructions by Opcode (continued)

Primary Opcode	Secondary Opcode	Form	Mnemonic	Operands	Page
35		D	**lbzu**	RT, D(RA)	965
36		D	**stw**	RS, D(RA)	1064
37		D	**stwu**	RS, D(RA)	1068
38		D	**stb**	RS, D(RA)	1051
39		D	**stbu**	RS, D(RA)	1052
40		D	**lhz**	RT, D(RA)	973
41		D	**lhzu**	RT, D(RA)	974
42		D	**lha**	RT, D(RA)	968
43		D	**lhau**	RT, D(RA)	969
44		D	**sth**	RS, D(RA)	1055
45		D	**sthu**	RS, D(RA)	1057
46		D	**lmw**	RT, D(RA)	977
47		D	**stmw**	RS, D(RA)	1060

Appendix B. PPC440GP Compiler Optimizations

This appendix describes some potential optimizations for compilers.

1. Place target addresses (subroutine entry points) on cache line boundaries (32-bytes)
2. Up to five instructions between a load and a use of the load result. Assuming a data cache hit, the worst case scenario for the PPC440GP is five instructions between a load-use, in order to avoid any bubbles. The five instructions are:
   - One dispatch, together with the load
   - Two the cycle after
   - Two the cycle after that

   In the next cycle, the use of the load result can dispatch. Therefore, the compiler should try to schedule as many as five instructions between the load and use of the load result. However, if some of the instruction pairs between the load-use have pipeline dependencies (such that they cannot dispatch together), there is no benefit in including the extra instructions between the load-use, and other scheduling optimizations could be made.

   In the worst case of instruction pairings, the maximum performance can be achieved with only two instructions between the load and use of the load result. This is the case when the load instruction pairs with the instruction before it (instead of after it), and then the next two instructions require the same pipe, so only one can dispatch during the cycle after the load, and then third instruction after the load needs the same pipe as the second, so they cannot dispatch together either. In such a case, the third instruction after the load might as well be the use of the load result. See item 3 for information about which instruction pairings can dispatch together.
3. Pair instructions for dual dispatch. The rules for instruction dispatch in the PPC440GP are as follows: loads and stores can only use the L-Pipe. Branches, CR-updates, XER-updates ("o" forms of arithmetic instructions), multiply, divide, system instructions (such as **rfi** and **sc**), and any SPR accesses (**mtspr**, **mfspr**) can only use the I-Pipe. All other instructions (primarily non-CR-updating and non-XER-updating arithmetic and logic instructions) can use either the J-Pipe or the I-Pipe. Instructions should be paired so that they can dispatch as pairs. For example, pair loads and stores with any other instructions. Pair CR-updates with non-CR-updating instructions and so on.
4. Do *not* bother to try to schedule instructions between CR-updates and branches that are conditional on those CR-updates (with some exceptions).

   The exceptions are for CR-updates caused by multiply, divide, multiply-accumulate, **mtcrf**, **tlbsx.**, and **stwcx.** instructions. If a branch depends on the CR result of one of these instructions, one or more instructions should be scheduled (if possible) between the CR update and the branch. Of course, it is also the general case (as pointed out in item 3) that the compiler should schedule instructions so they can issue in pairs, and a CR-update and a branch both issue to the I-Pipe, so they cannot issue together. (The compiler should try to set things up so a CR-update and a following branch (regardless of any CR-dependency by the branch) can issue in pairs.) This can mean the CR-update can get paired with the instruction before it, and the branch with the instruction after it, such that there is dual issue in both cycles. However, if this pairing is not possible, an instruction should be inserted (if possible, of course; do not create no-ops for no reason) between the CR-update and the branch to allow the dual issue.

   The point of this item is to explain that there is no need to separate the CR-update and the branch simply for the sake of the CR-dependency. That is, there is no extra cycle penalty associated with the CR-update/branch CR-dependency, beyond the "standard" penalty of the inability to dual issue, unless the CR-update is one of the types mentioned above.

If the CR-update is MAC or a 16 × 32 multiply, 1 to 3 instructions should be scheduled between the CR-update and the branch (0 or 1 instruction, depending on whether the CR-update pairs with the instruction before or after, or 1 to 2 instructions to issue between the issue of the CR-update and the issue of the branch, depending on whether there is a single-issue or dual-issue opportunity for the instruction(s) which are scheduled between the CR-update and the branch).

Similarly, if the CR-update is 32 × 32 multiply, divide, **tlbsx.**, or **stwcx.**, schedule 3 to 5 instructions between the CR-update and the branch (two issue cycles of 2 to 4 instructions between, plus the 0 to 1 issuing with the CR-update).

Finally, if the CR-update is **mtcrf**, schedule 5 to 7 instructions between (3 cycles of issue between them).

5. Avoid the use of string/multiple instructions (with some exceptions).

   The exceptions have to do with cache effects (more cache misses due to more instructions if you use separate loads/stores instead of a string/multiple), and the specialized behavior of a string, where the bytes are inserted into the more-significant portion of the GPR, in preparation for a "string compare" operation to determine which string is "greater" than another. If the string/multiple is for a relatively small number of registers (or the expansion into discrete loads/stores is known to not have an overall detrimental cache impact), and if a string is being used only for a copy operation and the size is known, performance can be improved by using discrete loads/stores. Essentially, due to hazard determination within the processor, string/multiples impose a couple of cycles of extra, "false" penalty on both the front-end and the back-end. On the other hand, if this penalty is amortized over a large number of registers (say 16 or so), the impact of the extra stalls is probably negligible.
6. Insert 10 or so instructions within a **bdnz** loop (loop unrolling).
7. Put 4 to 8 instructions between **mtlr**/**mtctr** and **blr**/**bctr**
8. Put 1 to 3 instructions between 16 × 32 multiply and the use of the result.
9. Put 2 to 5 instructions between 32 × 32 multiply and the use of the result.
10. Use the "without allocate" attribute appropriately on block copy operations, such as calls to the library **memcpy** function, or implicit structure copies.
11. Block move operations. If moving a block of memory using a series of load/store operations, perform the load/store operations in the following order: L1-L2-L3-S1-S2-S3, and repeat. Having the second and third loads between the first load and the first store fills the two-cycle load-use penalty.

Index

A

B

C

D

E

F

G

J

L

M

N

O

P

R

S

T

U

U, V, W

W

X

Z

Revision Log

Revision Date	Contents of Modification
07/30/02	migration to division template with additional functional updates