BAE SYSTEMS

RAD750$^{\hat{O}}$ Board

Appendix to the Hardware User's Manual

Document Number
TBD

PowerPC

Release Date

December 20, 2000

Notices

Before using this information and the product it supports, be sure to read the general information on the back cover of this book.

Trademarks

The following are trademarks of International Business Machines Corporation in the United States, or other countries, or both:

IBM IBM Logo PowerPC
PowerPC 750

The following are trademarks of BAE SYSTEMS in the United States, or other countries, or both:

RAD750

The following are registered trademarks of PCI Industrial Computer Manufacturing Group in the United States, or other countries, or both:

PICMG CompactPCI

Other company, product, and service names may be trademarks or service marks of others.

Second Edition (Version 2.10, 12/21/00)

This **unpublished document** is the second edition of RAD750™ Board Hardware Users Manual. Make sure you are using the correct edition for the level of the product.

This document contains information on a new product under development by BAE SYSTEMS. BAE SYSTEMS reserves the right to change or discontinue this product without notice.

Preface

Notice for reviewers:

This is an updated release of this document which includes the final register definitions for the Power PCI ASIC.

At this time, some of the information contained in this document is still not in its final state.

Items to still be added:

- An Index at the end of the document
- Add more information on fault tolerance tables with functions
- Add more information on RAD750 clocks as a function of divide down on Power PCI. Add more information on SYSCLK vs. non-sys PCI clocks.
- Add PCI View to RAD750 base address view for BARs
- More hyperlinks
- more info on error/fault detection and recovery from the cores

Items marked TBA or TBD will not be filled in by the preliminary release, but will be filled in for subsequent releases.

When reviewing the document, please keep the following in mind when providing comments:

- Is the document presented in a usable order?
- Since this is intended to be an all-encompassing, standalone document, is there any information that can be found somewhere else that should be included here
- Does the document provide enough information to stand-alone?
- Has too much detail been provided in some areas? What document should that detail be placed?
- What information is missing and should be added?
- Does this appendix belong with the main document or should it remain a separate document?

Table of Contents

List of Figures

List of Tables

1 Scope

This document is intended to be used as a standalone hardware user's manual for the Power PCI ASIC as well as serving as an appendix to the RAD750 3U CompactPCI board User's manual. Since the Power PCI controls the addressing in a processor card design, the address maps presented here are applicable to both board and chip users.

2 Power PCI Overview

The Power PCI ASIC is the single support chip required on the System Flight computer card. It provides the bridge from the 60X processor bus to local memory, SUROM, and a 3.3V 32-bit PCI interface (compatible with the compact PCI standard). It has a 16550 compatible UART, and a 1149.1a JTAG master and slave interfaces.

The Power PCI was designed to support embedded space applications and thus it has features such as timers, programmable discrete I/O, a programmable interrupt controller, and a simple embedded microcontroller for functions including fault handling and recovery.

The Power PCI ASIC operates at 33 MHz for the majority of its functions and its architecture supports a 33 MHz PCI bus operating asynchronous to chip clock frequency.

2.1 Environment

A functional block diagram of the Power PCI chip incorporated onto a processor board is shown in Figure 1.

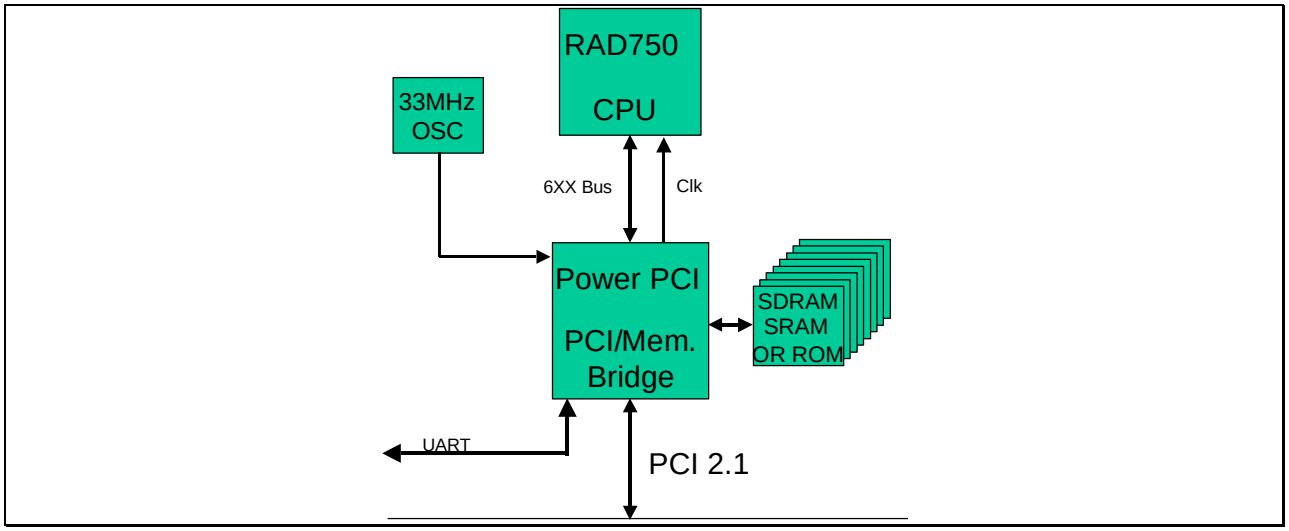


Figure 1: Power PCI Chip on a Processor Board

The main function of the Power PCI ASIC is to act as a bridge and single support chip for a PowerPC processor connected to a PCI bus. Both PowerPC 603 and 740/750 microprocessors are supported by the Power PCI design. The chip architecture is extremely flexible and supports multiple memory types and several ECC schemes.

2.2 Compliance Summary

The Power PCI supports connection to either a PowerPC 603 or a PowerPC 740/750 processor chip. The PCI bus interface is compatible with the 2.2 version of the PCI Local Bus specification and the memory interface is compatible with the JEDEC standard SDRAM Architectural and Operational Features.

2.3 Architecture Overview

The Power PCI functions are partitioned into 10 functional blocks or cores. An On-Chip-Bus (OCB) is used to interconnect all the cores. The OCB is configured as a crossbar switch to prevent the bus from becoming a performance bottleneck. Each core uses a common OCB interface, known as the OCB stub. The ten cores are as follows: PCI, P60X, MCC, EMC, CAT, MISC, JTAG Master (two copies), JTAG Slave, UART, OCB Connection Medium. The P60X, PCI, and EMC cores have both master and slave

capabilities on the OCB and therefore have access to memory and all internal registers excluding the JTAG Slave registers. The JTAG Slave core is master capable only on the OCB and therefore has access to memory and all internal registers. The UART, JTAG Masters, MISC, and CAT cores are slave only on the OCB. All cores except the JTAG's have the capability of generating an interrupt when an error is encountered or service is required. Core interrupts are collected in the MISC Interrupt Collection Register and MISC Interrupt Status Register.

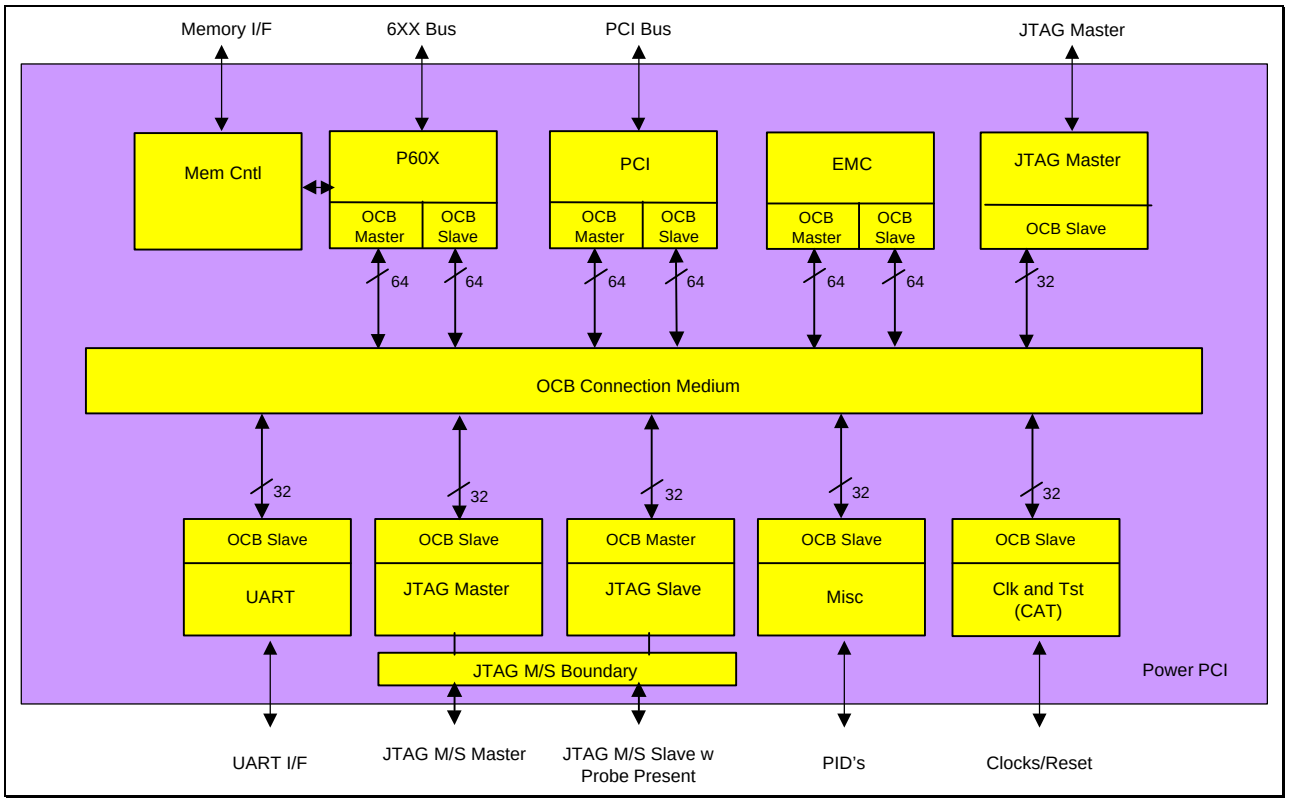


Figure 2: Power PCI Cores

The Power PCI chip will support multiple, asynchronous clock islands as shown in Figure 3. The four clock sources are the system clock which is used for the majority of the logic on the chip including the OCB interconnection interfaces, the PCI bus clock, and the real-time/UART clock, and the slave JTAG input clock.

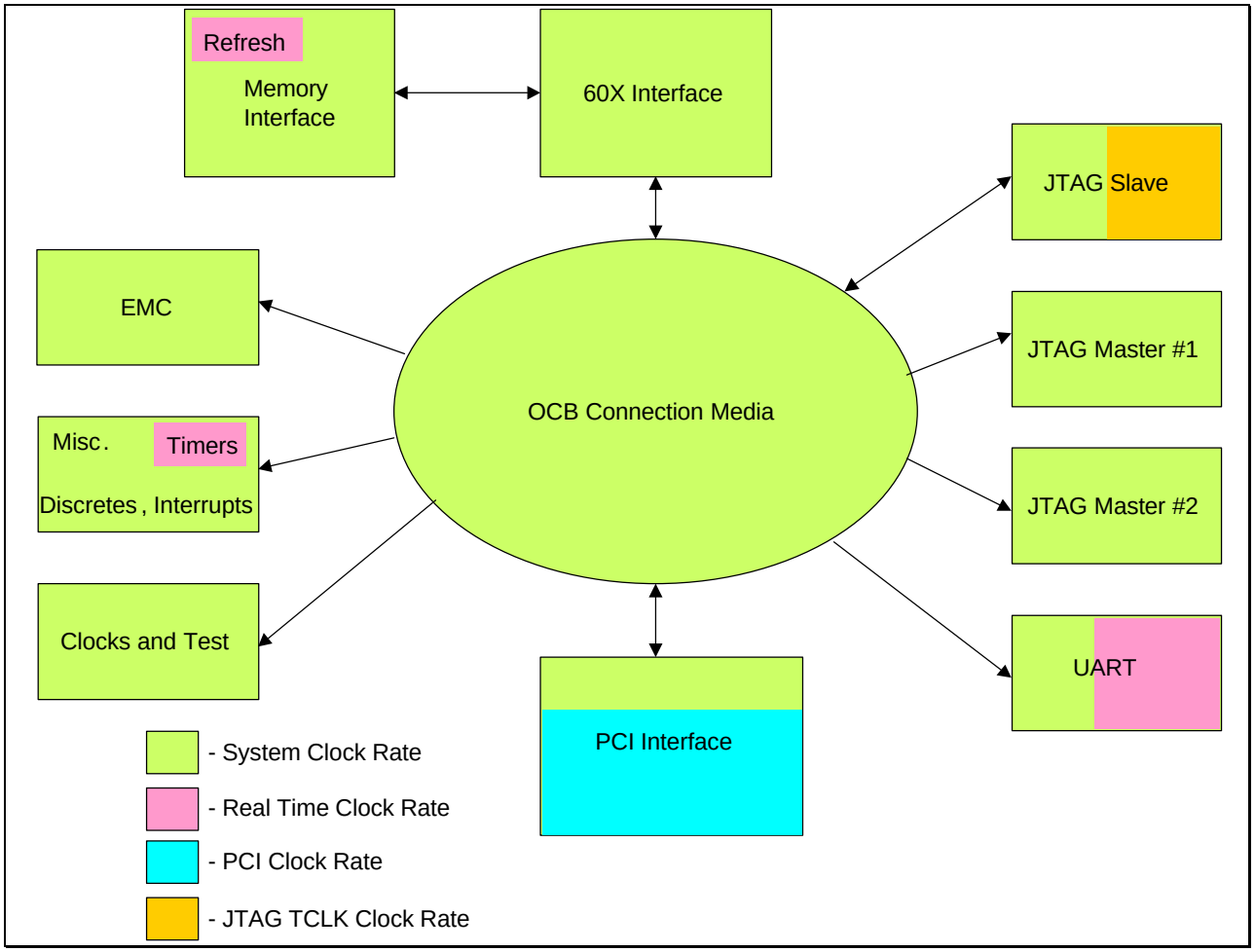


Figure 3: Power PCI Clock Regions

Details on each of the functions in the Power PCI chip can be found in Section 4.

3 Address Maps

The RAD750 board Address map has a different view when being accessed from different bus: Table 1 depicts the memory map when viewed by the RAD750 CPU, and Table 2 depicts the memory map when viewed at the PCI bus / CompactPCI connector.

Table 1 - RAD750 Board Memory Map seen from the CPU

Address	Destination I/F	Destination Address	Notes	Page
0000 0000 - 7FFF FFFF	System Memory	0000 0000 - 7FFF FFFF		
8000 0000 - 8000 0CF7	PCI I/O	0000 0000 - 0000 0CF7		18
8000 0CF8 - 8000 0CFB	PCI	PCI Config ADDR Reg	Config ADDR Port	18
8000 0CFC - 8000 0CFF	PCI Config Or PCI	PCI Config ADDR	Config Data Port PCI Config Cycle or internal PCI register access	18
8000 0D00 - BF7F FFFF	PCI I/O	00D0 0000 - 3F7F FFFF or (1 of 8) I/O Base Addr Regs[0:10]||A[11:31]	MPC-106 Mode or PCI Paging Mode	18
BF80 0000 - BF80 FFFF	System Memory	0000 - FFFF	Configuration Regs	48
BF81 0000 - BF81 FFFF	60x	0000 - FFFF	Configuration Regs	155
BF82 0000 - BF82 FFFF	UART	0000 - FFFF	Configuration Regs	91
BF83 0000 - BF83 FFFF	JTAG(A)	0000 - FFFF	Configuration Regs	99
BF84 0000 - BF84 FFFF	JTAG(B)	0000 - FFFF	Configuration Regs	99
BF85 0000 - BF85 FFFF	Micro-controller	0000 - FFFF	Configuration Regs	
BF86 0000 - BF86 FFFF	Clk and Test	0000 - FFFF	Configuration Regs	117
BF87 0000 - BF87 FFFF	PCI	0000 - FFFF	Configuration Regs	
BF88 0000 - BF88 FFFF	Int/Timers/Disc	0000 - FFFF	Configuration Regs	17
BF89 0000 - BFFF FFFF	Reserved			
C000 0000 - FEFF FFFF	PCI Memory	0000 0000 - 3EFF FFFF or (1 of 8) Memory Base Regs[0:4]||A[5:31]	MPC-106 Mode or PCI Paging Mode	18
FF00 0000 - FF7F FFFF	Reserved			
FF80 0000 - FFFF FFFF	System Memory Or PCI Memory	FF80 0000 - FFFF FFFF	ROM in System Memory Or ROM on PCI	18

The Power PCI Chip contains 2 PCI Memory Base Address registers (BARs):

- **BAR1** defines the base address for system memory. Its size is selectable by external I/O pins on the Power PCI chip as 2 GByte, 1 GByte, 512 MByte, or 256 MByte. The RAD750 board is configured as 256 MByte. Furthermore an internal configuration register **SM_SIZE** determines the number of 8 KByte pages that are mapped to the PCI bus. The number of 8K pages can range from 1 to 32768.
- **BAR2** defines the base address for the internally architected Power PCI registers. It defines a 1 MByte PCI Memory region. Write access to the lower half of the **BAR2** memory region from the PCI Bus is enabled by the BAR2_WE bit stored in the PCI configuration space.

Table 2 - RAD750 Board Memory Map seen from the PCI Bus

Address	Destination I/F	Destination Address	Notes
BAR1	System Memory	0000 0000 - 7FFF FFFF	
BAR2	Architected Regs	BF80 0000 - BF8F FFFF	Write access to BF80 0000 - BF87 FFFF enabled via BAR2_WE
Configuration Cycle	PCI	00 - FF	PCI Configuration registers

3.1 Address Map Deviations from the MPC-106

The Power PCI was architected to be as software compatible with the Motorola MPC-106 PCI bridge chip. This was done intentionally so that software developers could utilize commercial PowerPC based COTS boards to start software prototyping with early in their programs. Since the MPC-106 was designed for desktops, its characteristics are not necessarily compatible with the embedded/space application environment. The following deviations were made:

- Eliminated Address Map B and Emulation Mode Address Map. **Rationale:** These address maps were eliminated in order to reduce the design scope of the Power PCI. They are primarily intended for Apple Mac OS application developers. They provide no value to the space operating environment.
- Eliminated the Discontiguous I/O space mode. **Rationale:** This mode is not needed for space applications.
- Extended system memory into the second GByte of address space (Reserved by MPC-106).
- Added function of the reserved region (0x'BF80 0000' to 0x'BFFF FFEF'). The Power PCI uses this region to map configuration/status registers for added functions (i.e., UART, JTAG, etc.).
- Eliminated the PCI configuration space direct access. **Rationale:** When multiple processors are utilized in a system, this mode could cause damage to the hardware with a SEU hit or a software error.
- Eliminated the PCI Int Ack direct access memory region (0x'BFFF FFF0' - 0x'BFFF FFFF'). **Rationale:** This function is only needed to support ISA Bridge Interrupt controller that reside in desktop machines and will not be present in the RAD750 board configuration. This function can alternatively be performed using the CONFIG ADDR/DATA port.

3.2 Register Reads

Reads within the cache line space (i.e., 64-bits of data), but not defined in the different Register Matrix Tables, return zeros with good parity and will not signal an error. All read accesses that request data from an address that is beyond the padded cache line space, return all zeros with good parity and set the ***Address Out Of Range*** bit in the **Error Status Register**. Also, if the "out of range" access was from the OCB slave interface the RD_D_ERR signal is set active. Furthermore, the register space is able to support cache wrap transfers on read operations.

3.3 Register Writes

Writes within the cache line space where the byte enables are active to a “read only” location or to an undefined location result in the ***Address Out of Range*** bit of the **Error Status Register** being set and the data discarded. However if parts of the enabled data do align with valid locations, this data is written and the invalid data discarded and an error will be signaled. All write accesses outside of the padded cache space result in the ***Address Out of Range*** bit of the **Error Status Register** to be set and the data discarded.

3.4 Register Matrix Conventions

The following conventions or definitions apply to the tables in the remaining sections:

R	Read-only
R/W	Read / Write
R/WC	Read / Write with 0b’1’s to clear bits
W	Write Only - Always read as 0b’0’s
NONE	No Access Rights

4 Power PCI Detailed Description and Register Definitions

4.1 PCI Functional Description

The PCI bus interface in the Power PCI ASIC provides a 32-bit PCI master and target interface, and also includes an optional PCI central resource function. The PCI central resource function is enabled when the SFC is placed in the System Controller slot of a Compact PCI backpanel. The PCI core supports the asynchronous operation of the PCI bus relative to the remainder of the logic on the SFC. Internal buffering of data and command queues is provided.

Some general features the PCI Interface supports are:

- 32-bit PCI address and data path
- 0 to 33 MHz operating frequency
- Asynchronous application interface
- Zero wait-state burst transactions
- All types of PCI abort, retry, and disconnect
- Delayed transactions
- Posted memory writes
- Configurable prefetching of read data
- 64-bit interface to remainder of SFC

A block diagram of the PCI core is shown in Figure 4 and indicates the interconnections between the various key features in the design.

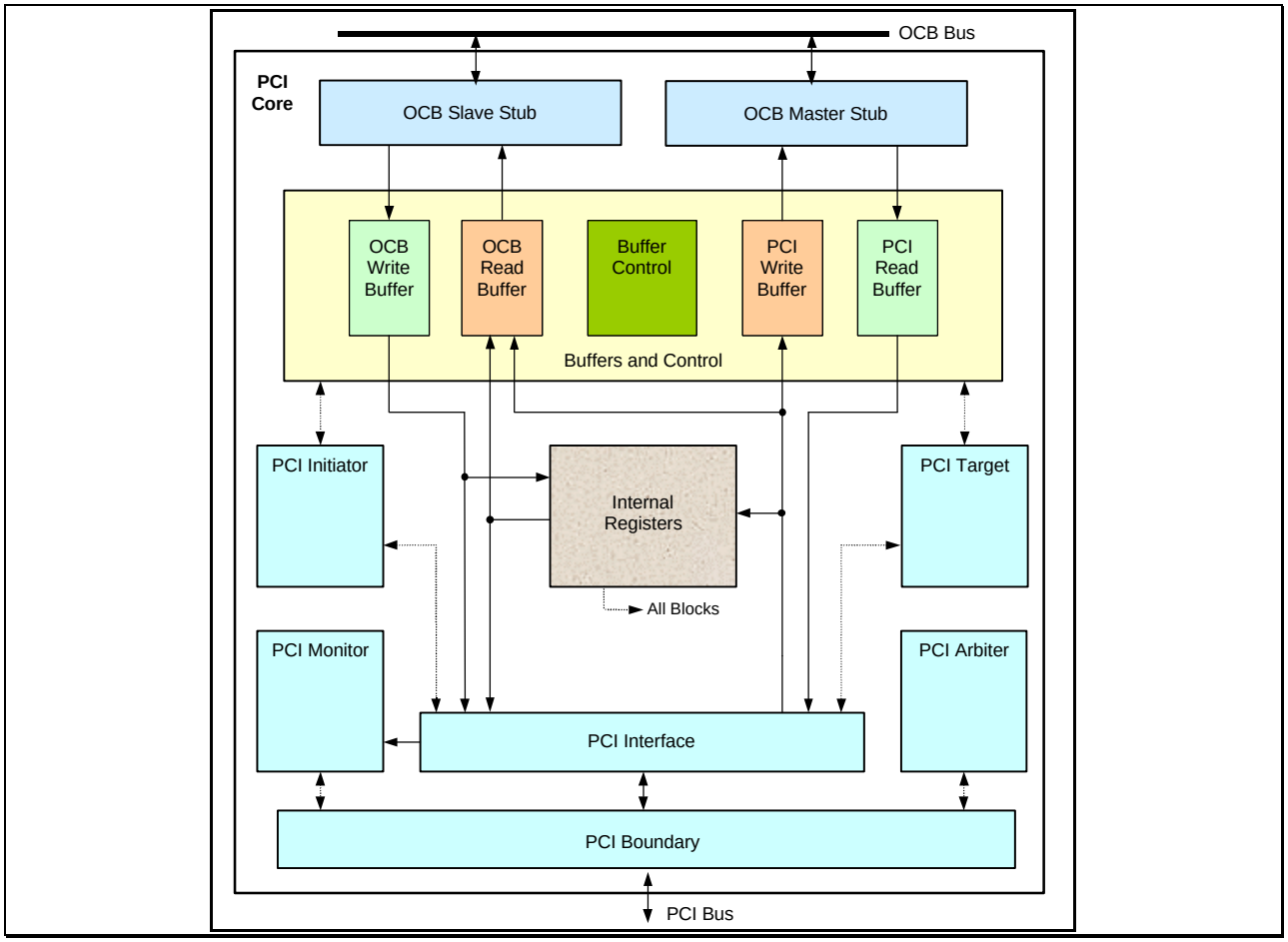


Figure 4: PCI Core Block Diagram

The PCI core contains User Defined registers known as User Defined Registers A, B, and C. These registers supply configuration controls to other areas of the Power PCI chip as described in Table 3. In general, these are used to maintain compatibility with the commercial MPC-106 bridge chip.

Table 3 - PCI User Defined Register Definitions

Bits	Function	Destination	Comments
USER_DEFINED_A(0)	P60x_ERROR_EN	60x Core	60X bus error enable
USER_DEFINED_B(11)	MISC_MCP_ENABLE	Misc Core	Machine Check Interrupt Enable
USER_DEFINED_B(5)	P60x_ENDIAN_MODE	60x Core	Defines endian mode
USER_DEFINED_C(27)	P60x_NO_SNOOP_EN	60x Core	Enables Snoop on P60X bus
USER_DEFINED_C(26)	P60x_FF0_LOCAL	60x Core, OCB	Additional SUROM location control
USER_DEFINED_C(19:18)	P60x_SNOOP_WS	60x Core	Snoop wait states
USER_DEFINED_C(15)	P60x_DBWO_EN	60x Core	Enables DBWO on 60X bus

Bits	Function	Destination	Comments
USER_DEFINED_C(3:2)	P60x_APHASE_WS	60x Core	Address wait states

4.1.1 PCI Arbiter

When the SFC is placed in the system controller slot in a standard Compact PCI backpanel it will act as the central resource for the PCI bus. As the central resource, the Power PCI will provide central arbitration for PCI bus accesses. A three level arbitration scheme is used. It is based on fixed priorities within a level and between levels, but with fairness built in to the two higher priority levels to ensure that a low priority device is not locked out should higher priority devices continuously request the bus. Up to eight PCI masters requesting access to the PCI simultaneously is supported. The Power PCI chip is considered one of these eight masters.

Each of the eight sets of Request/Grant signals can be assigned to one of the three priority levels (or disabled) using the Arbitration Priority Register in the ASIC. If a request/grant pair is disabled, that bus master would never be granted the bus. This violates the PCI specification's arbitration fairness requirement, however it provides the capability for the central resource to turn off an errant master should that be necessary.

The three arbitration priority levels have fixed priorities in relation to each other with level one having the highest priority and level three the lowest. Within each level, priorities are also fixed with the lowest order request having the highest priority (i.e., If **REQ0#** and **REQ1#** are programmed to the same arbitration level, **ARB_REQ0#** has higher priority.)

Level one and level two have fairness built in. This means that in addition to all the request/grant pairs assigned to level one in the **PCI Arbitration Priority Level** register, level one also contains a level two fairness chit. The level two fairness chit has lower priority than all other level one requesters, but enables a lower level requester to be granted the bus before a level one requester that has already used the bus can be granted the bus a second time. Once one lower level requester has used the bus, all level one requesters are given the opportunity to take another turn on the bus before the level two fairness chit can be used again.

Similarly, level two contains a level three fairness chit, allowing one level three device a turn on the bus after all level two requesters have used the bus. In the case where all level two requesters have already used the bus, and all level one requesters have just finished using the bus, then one device from level three is allowed to use the bus since both the level 2 and level 3 chits would be active at that time.

The arbiter always grants the bus to the highest priority 'non-masked' requester. A requester is masked by the arbiter as soon as it becomes owner of the bus. A requester becomes unmasked by the arbiter when there are no longer any unmasked devices at its level requesting the bus – including the fairness chit. If no devices are requesting the bus, then the arbitration algorithm is essentially reset. So at that point, if multiple devices request the bus at the same time, the highest priority requester would always be granted the bus first regardless of previous ownership (or bus parking). Also, note that if only one device is requesting the bus, it will be granted the bus continuously, until a second device requests the bus (even though it is still essentially 'masked' using the previous definition).

In absence of any active bus requests, the arbiter parks the bus at the last used requester. After reset, when there is no 'last used requester', the PCI Core drives **AD(31:0)**, **C/BE(3:0)#**, and **PAR** although the bus is not actually parked (no **GNT(7:0)#** is active).

4.1.2 PCI 2.2 Compliance Summary

The PCI interface is designed to the requirements set forth in the PCI Local Bus Specification, Rev 2.2. Any and all deviations from the PCI Specification are detailed here. This section does not describe any optional PCI features that were not implemented.

PCI Specification Chapter 4: Electrical Specification: The PCI logic in the Power PCI chip does not enforce a minimum length **RST#** when generating PCI reset as a central resource. Software must adhere to this PCI requirement.

PCI Specification Chapter 5: Mechanical Specification: These are expansion card requirements and are dealt with at the card level. The PCI interface contained in the ASIC does not claim compliance to any requirements in Chapter 5 of the PCI Specification.

PCI Specification Chapter 7: 66 MHz PCI Specification: The current version of the Power PCI dies not support a 66 MHz PCI. The **M66EN** signal is not generated by the ASIC.

Transaction Ordering: The PCI core in the Power PCI ASIC can only order transactions with respect to the OCB interface. It has no knowledge of what goes on beyond the OCB. Therefore, it is possible that transaction ordering on the entire ASIC device may be violated assuming that there is buffering beyond the OCB from the PCI core.

SERR#: The PCI Specification defines **SERR#** as a type O/D signal (i.e., open drain). It only discusses **SERR#** as a PCI output. The PCI Core defines this signal as a *bi-directional* open drain signal so that when it is acting as the Central Resource on the bus, it can latch an active **SERR#** input and report it. When not the central resource, the PCI Core ignores the **SERR#** as an input.

Subsystem ID: The PCI Specification requires that the **Subsystem Vendor ID** and **Subsystem ID** registers are loaded with valid information prior to any software accessing them in PCI Configuration Space. The specification suggests responding with Retry on reads to this register until that happens. The PCI Core allows these registers to be written from the OCB interface, but provides no logic to prevent the PCI bus from reading these registers prior to that occurring. The PCI Core relies on system specific software to ensure that this requirement is met.

Transaction Ordering of PCI Configuration Accesses: The PCI Core does not allow PCI configuration access to participate in transaction ordering. That is, PCI configuration reads and writes can occur without regard to whether or not data is posted in any of the buffers. The PCI 2.2 Specification implies that any PCI target read should flush the write buffers, including configuration reads. This 'non-compliancy' precludes the use of the PCI Core internal registers as 'flags' for data completion. For example, if a device were to execute a memory write to the PCI Core which was posted, and then set a 'flag' in one of the PCI Core internal registers, another device could read the 'flag' before the posted write data was written to the OCB. Similarly, a configuration write to a PCI Core internal configuration type register could occur before the posted data was flushed to the OCB, potentially inadvertently modifying the transfer.

Ordering of Delayed Transactions: The PCI Core in the Power PCI ASIC does not support Rule 5 in appendix E which states that a Posted Memory Write must be allowed to pass a Delayed Request to avoid deadlocks. The lack of a retry capability on the OCB interface prohibits adherence to this rule. However, this rule is stated to prevent a particular deadlock scenario that occurs when a PCI-to-PCI bridge that supports delayed transactions is used with a PCI-to-PCI bridge that doesn't support delayed transactions. The PCI Core therefore does not support this configuration.

Memory Write maximum Completion Time Limit: This system level requirement is beyond the scope of the PCI Interface on the Power PCI chip.

Status Register: The PCI Core enhances PCI error reporting by including the **Status 2** register. To compile all error bits into the same register (the **Status 2** register), the PCI Core mirrors bits from the PCI specification defined **Status** register into the **Status 2** register. By mirroring these bits, the PCI Core allows some of the **Status** register bits to be reset by writing '1's to corresponding bit locations in the **Status 2** register. This action may not be considered PCI compliant.

The PCI Specification requires an agent driving data on the PCI bus to drive even parity on **PAR**. The Power PCI violates this requirement when OCB or internal register data intended for the PCI bus contains bad parity. The PCI Core passes the bad OCB or internal register parity on to the PCI. Under normal conditions (no OCB or internal register parity errors), the PCI Core does pass even parity to the PCI as required.

1. The PCI Core does not fully support transaction ordering as a PCI Target in that the core does not prohibit PCI Target reads from completing on the PCI bus when posted OCB Slave write data exists in the OCB Slave Write Buffer and the OCB Slave write completed on the OCB prior to the PCI Target read having completed on the OCB.

4.1.3 PCI Register Matrix

The configuration registers in the PCI Function of the Power PCI are accessible from both the CPU and PCI with the same access rights from both sides. The one exception to this rule is the **Subsystem Vendor ID (0x2C)** and **Subsystem ID (0x2A)** registers that are always read only from the PCI, and may be read/write from the CPU. Refer to Section 4.1.4.10 for details. Writes from either interface to reserved or read only registers will be treated as no-ops; that is, the access will be completed without error and the data will be discarded. Reads to reserved registers will return 0b'0's.

The PCI function does not use the following optional PCI Specification defined Configuration Header registers and treats them as reserved: **BIST**, **Cardbus CIS Pointer**, **Expansion ROM Base Address**, and **Capabilities Pointer**.

Table 4 - PCI Register/Resource Map

Register Name	Address	Access	Reset State	Bits	Type	Page
Vendor ID	0x'0000'	R	0x'11D0'	16	Information	24
Device ID	0x'0002'	R	0x'0030'	16	Information	24
Command	0x'0004'	R/W-R	0x'0080'	16	Configuration	24
Status	0x'0006'	R/WC-R	0x'0280'	16	Status	26
Revision ID	0x'0008'	R	0x'00'	8	Information	27
Class Code	0x'000A'	R	0x'060000'	24	Information	27
Cacheline Size	0x'000C'	R/W	0x'00'	8	Configuration	28
Latency Timer	0x'000D'	R/W	0x'00'	8	Configuration	28
Header Type	0x'000E'	R	0x'00'	8	Information	29
Base Address 1	0x'0010'	R/W - R	0x'0000 0008'	32	Configuration	29
Base Address 2	0x'0018'	R/W - R	0x'0000 0000'	32	Configuration	29
Subsystem Vendor ID	0x'002C'	R	0x'0000'	16	Information	30
Subsystem ID	0x'002E'	R	0x'0000'	16	Information	31
Interrupt Line	0x'003C'	R/W	0x'00'	8	Information	31
Interrupt Pin	0x'003D'	R	Note 1	8	Information	31
MIN_GNT	0x'003E'	R	0x'00	8	Information	32
MAX_LAT	0x'003F'	R	0x'00'	8	Information	32
Status 2	0x'0050'	R/WC	0x'0000 0000'	32	Status	32
Status 2 Mask	0x'0054'	R/W	0x'0000 0000'	32	Intr. Disables	36
BAR1 Size Adjust	0x'0058'	R/W	0x'0000'	16	Configuration	37
PCI Core Config	0x'005C'	R/W	0x'0000 0000'	32	Configuration	38

Register Name	Address	Access	Reset State	Bits	Type	Page
Arbitration Priority	0x'0060'	R/W	0x'0000'	16	Configuration	41
Error Checking	0x'0062'	R/W	0x'0000'	16	Configuration	42
Error Injection	0x'0064'	R/W	0x'00'	8	Action	42
Bus Reset	0x'0065'	W	0x'00'	8	Action	44
PCI Revision ID	0x'0066'	R	0x3001'	16	Information	44
User Defined A Register	0x'0070'	R/W	0x'0000 0000'	32	User Defined	46
O2P I/O Paging 0	0x'0080'	R/W	0x'0000'	16	Configuration	44
O2P I/O Paging 1	0x'0082'	R/W	0x'0000'	16	Configuration	44
O2P I/O Paging 2	0x'0084'	R/W	0x'0000'	16	Configuration	44
O2P I/O Paging 3	0x'0086'	R/W	0x'0000'	16	Configuration	44
O2P I/O Paging 4	0x'0088'	R/W	0x'0000'	16	Configuration	44
O2P I/O Paging 5	0x'008A'	R/W	0x'0000'	16	Configuration	44
O2P I/O Paging 6	0x'008C'	R/W	0x'0000'	16	Configuration	44
O2P I/O Paging 7	0x'008E'	R/W	0x'0000'	16	Configuration	44
O2P Mem Paging 0	0x'0090'	R/W	0x'0000'	16	Configuration	45
O2P Mem Paging 1	0x'0092'	R/W	0x'0000'	16	Configuration	45
O2P Mem Paging 2	0x'0094'	R/W	0x'0000'	16	Configuration	45
O2P Mem Paging 3	0x'0096'	R/W	0x'0000'	16	Configuration	45
O2P Mem Paging 4	0x'0098'	R/W	0x'0000'	16	Configuration	45
O2P Mem Paging 5	0x'009A'	R/W	0x'0000'	16	Configuration	45
O2P Mem Paging 6	0x'009C'	R/W	0x'0000'	16	Configuration	45
O2P Mem Paging 7	0x'009E'	R/W	0x'0000'	16	Configuration	45
User Defined B Register	0x'00A8'	R/W	0x'0000 0000'	32	User Defined	46
User Defined C Register	0x'00AC'	R/W	0x'0000 0000'	32	User Defined	46
PCI Bus Error Status	0x'00C6'	R	0x'0000'	16	Status	47
PCI Error Address	0x'00C8'	R	0x'0000 0000'	32	Status	48

Note 1: The reset value of the **Interrupt Pin** register is dependent on the **INTR_PIN_VALUE** I/O signal.

Note 2. The reset value of the **Bus Reset** register is dependent on the **CENTRAL_RSRC_L** I/O signal.

Table 5 - PCI Register Address Map

31 24	23 16	15 8	7 0	Hex
Device ID		Vendor ID		00
Status		Command		04
Class Code			Revision ID	08

31 24	23 16	15 8	7 0	Hex
Reserved	Header Type	Latency Timer	Cacheline Size	0C
Base Address 1				10
Reserved				14
Base Address 2				18
Reserved				1C - 2B
Subsystem ID		Subsystem Vendor ID		2C
Reserved				30 - 3B
MAX_LAT	MIN_GNT	Interrupt Pin	Interrupt Line	3C
Reserved				40 - 4F
Status 2				50
Status 2 Mask				54
Reserved	BAR1 Size Adjust			58
PCI Configuration				5C
Error Checking		Arbitration Priority		60
PCI Revision ID		Bus Reset	Error Injection	64
Reserved				68 – 6C
User Defined A				70
Reserved				74- 7C
O2P I/O Paging 1		O2P I/O Paging 0		80
O2P I/O Paging 3		O2P I/O Paging 2		84
O2P I/O Paging 5		O2P I/O Paging 4		88
O2P I/O Paging 7		O2P I/O Paging 6		8C
O2P Mem Paging 1		O2P Mem Paging 0		90
O2P Mem Paging 3		O2P Mem Paging 2		94
O2P Mem Paging 5		O2P Mem Paging 4		98
O2P Mem Paging 7		O2P Mem Paging 6		9C
Reserved				A0 – A4
User Defined B				A8
User Defined C				AC
Reserved				B0 - C3
PCI Bus Error Status		Reserved		C4
PCI Error Address				C8
Reserved				CC – FF

4.1.4 PCI Register Descriptions

4.1.4.1 Vendor ID (0x00)

Description: The **Vendor ID** register is, intended to identify the manufacturer of the PCI device. The default value is 0x'11D0' to indicate a BAE SYSTEMS built device. The Vendor ID register is accessible

as a read only register from the PCI interface or the CPU interface. A write to the Vendor ID register does not change its value.

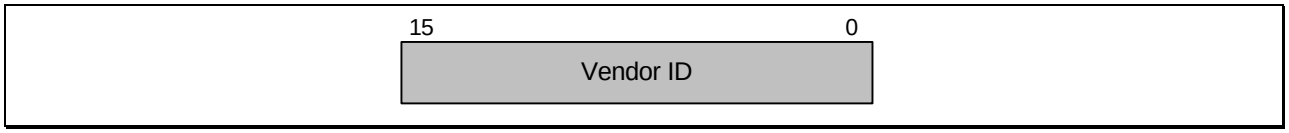


Figure 5: Vendor ID Register Layout

4.1.4.2 Device ID (0x02)

Description: The **Device ID** register is intended to identify this device within a given manufacturer's PCI product line. The value contained in this register is a constant. The **Device ID** register is accessible as a read only register from the PCI interface or the CPU interface. A write to the **Device ID** register does not change its value. The Power PCI has been assigned a 0x'0030' ('0' = BAE SYSTEMS, '03' = Core Based Designs, '0' = First Device of this category).

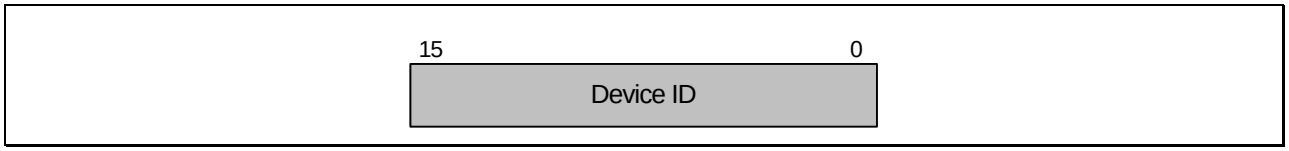


Figure 6: Device ID Register Layout

4.1.4.3 Command Register (0x04)

Description: The **Command** register is used to provide coarse control over the ability to generate and respond to PCI accesses. Fast Back-to-Back transactions as a master, VGA Palette Snooping, Memory Write and Invalidate Command, Special Cycle Monitoring, and I/O Space Enable are not supported and are disabled. The Command register is read/write accessible from the PCI interface or the CPU interface. Some bits are 'read only'. The remainder of the PCI Command register is reset to 0b'0's.

Note: The PCI Specification requires Bus Master to be reset to 0b'0' except when the device is in the boot path.

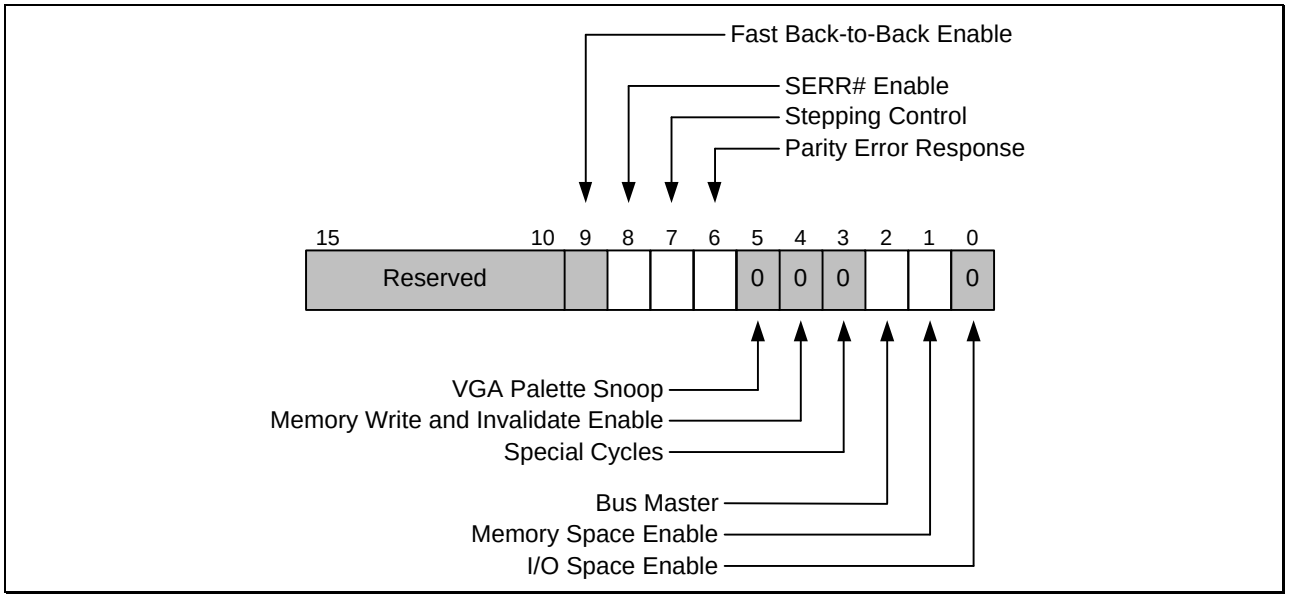


Figure 7: Command Register Layout

Bit/Field Definitions:

Bits 15:10	***Reserved:*** These bits are reserved, and return 0b'0' when read.
Bit 9	***Fast Back-To-Back Enable:*** Writes to the **Command register** do not alter the ***Fast Back-To-Back Enable*** bit. The ***Fast Back-To-Back Enable*** bit is 0b'0' when read. The ability to perform fast back-to-back transactions as a master is not supported.
Bit 8	***SERR# Enable:*** The SERR# output driver is disabled when the ***SERR# Enable*** bit is cleared to 0b'0'. The SERR# output driver is enabled when the ***SERR# Enable*** bit is set to 0b'1'.
Bit 7	***Stepping Control:*** Address/data stepping is used when the ***Stepping Control*** bit is set to 0b'1'. Address/data stepping is not be used when the ***Stepping Control*** bit is cleared to 0b'0'.
Bit 6	***Parity Error Response:*** Parity errors on the PCI bus are ignored (except for setting the ***Detected Parity Error bit*** in the **Status Register (0x06)**) when the ***Parity Error Response*** bit in the **Command register** is set to 0b'0'. Parity errors on the PCI interface are processed normally when the ***Parity Error Response*** bit is set to 0b'1'.
Bit 5	***VGA Palette Snoop:*** Writes to the Command register do not alter the VGA Palette Snoop bit. Reads of the Command register returns 0b'0' for the VGA Palette Snoop bit. VGA Palette Snooping is not supported by the Power PCI.
Bit 4	**Memory Write and Invalidate Enable:** Writes to the Command register do not alter the Memory Write and Invalidate Enable bit. Reads of the Command register returns 0b'0' for the Memory Write and Invalidate Enable bit. The Power PCI does not generate Memory Write and Invalidate commands.
Bit 3	***Special Cycles:*** Writes to the Command register do not alter the Special Cycles bit. Reads of the Command register returns 0b'0' for the Special Cycles bit. The Power PCI ignores all Special Cycle Operations as a PCI target.
Bit 2	***Bus Master:*** Setting the Bus Master bit of the Command register to 0b'0' disables generation of PCI accesses. Setting the Bus Master bit to 0b'1' enables generation of PCI accesses.
Bit 1	***Memory Space Enable:*** The Power PCI does not respond to PCI access of the memory address space when the Command register's Memory Space Enable bit is set to 0b'0'.
Bit 0	***I/O Space Enable:*** Writes to the Command register do not alter the I/O Space Enable bit. Reads of the Command register returns 0b'0' for the I/O Space Enable bit. The Power PCI does not support I/O Space as a target.

4.1.4.4 Status Register (0x06)

Description: This register is used to report status information on PCI bus related events. The *Capabilities List*, *66 MHz Capable*, *Fast Back-to-back Capable*, and *DEVSEL Timing* Status register bits are read only. All other non-reserved bits in the Status register can be set by hardware, and cleared to 0b'0' when the register is written and the corresponding bit location is set to 0b'1'. Additionally, the non-reserved bits are cleared to 0b'0' when the most significant byte of the **Status 2** register is written and the corresponding bit locations are set to 0b'1'. The Status register is accessible from both the PCI and the CPU interfaces.

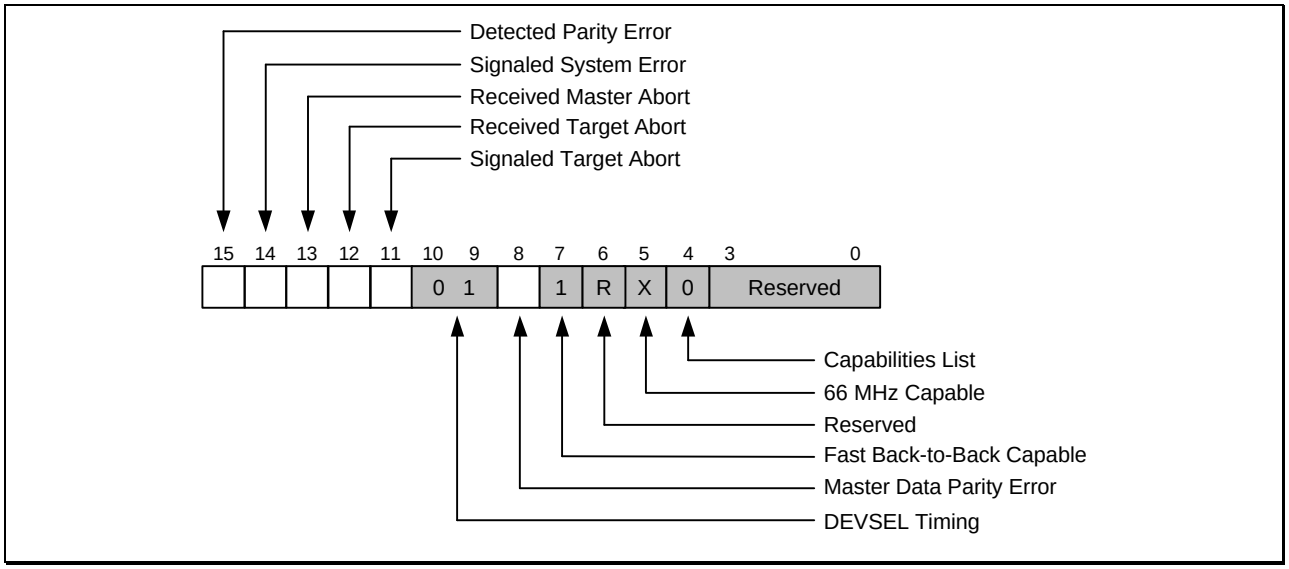


Figure 8: Status Register

Bit/Field Definitions:

Bit 15	***Detected Parity Error:*** Detection of a parity error on the PCI bus causes the Detected Parity Error bit of the Status register to be set to 0b'1'.
Bit 14	***Signaled System Error:*** The Signaled System Error bit is set to 0b'1' whenever the SERR# output is asserted.
Bit 13	***Received Master Abort:*** The Received Master Abort bit is set to 0b'1' whenever the Power PCI is the PCI bus initiator and terminates a transaction with master abort.
Bit 12	***Received Target Abort:*** The Received Target Abort bit is set to 0b'1' whenever the Power PCI is the PCI bus initiator and its transaction is terminated with a target abort.
Bit 11	***Signaled Target Abort:*** The Signaled Target Abort bit is set to 0b'1' whenever the Power PCI is the PCI bus target and terminates a transaction with target abort.
Bit 10-9	***DEVSEL Timing:*** The DEVSEL Timing field of the Status register returns 0b'01' when read, indicating medium timing for assertion of DEVSEL#. Writes to the Status register do not affect the value of the DEVSEL Timing field. See Figure 9.
Bit 8	***Master Data Parity Error:*** The Power PCI sets the Master Data Parity Error bit to 0b'1' when it is the PCI bus master, PERR# has been asserted, and the Parity Error Response bit in the Command register is 0b'1'.
Bit 7	***Fast Back-to-Back Capable:*** This PCI target supports fast back-to-back transactions to different agents. The Fast Back-to-Back Capable bit of the PCI Status Register returns 0b'1' when read. Writes to the Status register do not affect the value of the Fast Back-to-Back Capable bit.
Bit 6	***Reserved***: This bit is reserved, and returns 0b'0' when read.
Bit 5	***66 MHz Capable:*** The 66 MHz Capable feature is not supported by the Power PCI. Therefore, the Capabilities List bit is hardcoded to 0b'0'. Writes to the Status register do not affect the value of the Capabilities List bit.

Bit 4	***Capabilities List:*** The Capabilities List feature is not supported by the Power PCI. Therefore, the Capabilities List bit is hardcoded to 0b'0'. Writes to the Status register do not affect the value of the Capabilities List bit.
Bits 3:0	***Reserved***: These bits are reserved, and return 0b'0' when read.

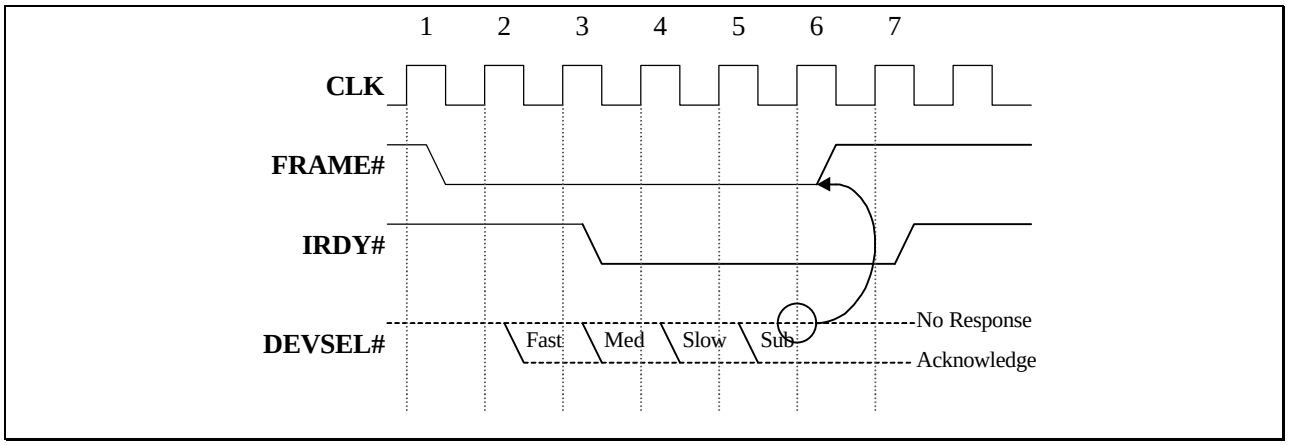


Figure 9: DEVSEL# Assertion

4.1.4.5 Revision ID (0x08)

Description: This register contains the design revision number for the Power PCI. The Revision ID register is read only and is accessible from both the PCI the CPU interfaces. A write to the Revision ID register does not change its value.

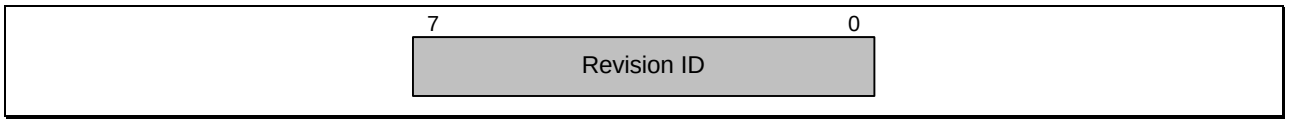


Figure 10: Revision ID Register Layout Class Code (0x09)

Description: This register identifies the generic function of the Power PCI. The default value for the BASE_CLASS is 0x'06'. The default value for both the SUB_CLASS and PROGIF_CLASS is 0x'00'. The Class Code register is read only and is accessible from both the PCI and CPU interfaces. A write to the Class Code register does not change its value.

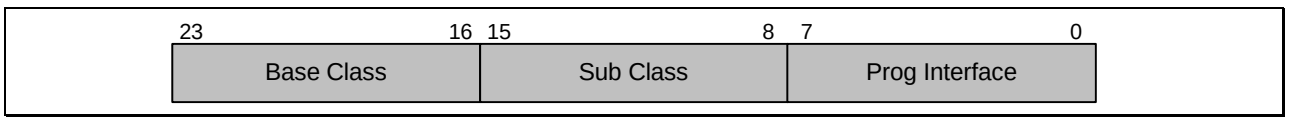


Figure 11: Class Code Register Layout

Bit/Field Definitions:

Bits 23:16	***Base Class:*** The Base Class is used to indicate a broad classification for the type of function the device performs. The Base Class field of the Class Code register returns the value 0x'06'.
Bits 15:8	***Sub Class:*** The Sub Class is used to specifically identify the function of the device. The Sub Class field of the Class Code register returns the value 0x'00'.

Bits 7:0	***Programming Interface:*** Programming Interface identifies a specific register level programming interface (if any) that device independent software can interact with. The Programming Interface field of the Class Code register returns the value 0x'00'.

4.1.4.6 Cacheline Size (0x0C)

Description: This register specifies the system cache line size in units of 32-bit words. The Power PCI only supports a 32-byte cache line, therefore, to meet PCI Specification requirements, this register ignores all writes, except a write with the value 0x'08' or 0x'00'. The Cacheline Size register is read/write accessible from both the PCI and CPU interfaces. Reads return the current value of the register (either 0x'00' or 0x'08'). The Cacheline Size register has a reset value of 0x'00'.

The Power PCI uses the value in the Cacheline Size register to determine cache line boundaries. When this register contains the value 0x'08' the Power PCI enables Cacheline Wrap mode burst ordering on PCI target reads. Additionally, PCI target Memory Read Line commands are enabled when this register is set to 0x'08'. Finally, when this register contains the value 0x'08', the Power PCI enables the generation of Memory Read Line commands as a PCI master on 4-beat CPU reads to PCI Memory space. When this register contains 0x'00', PCI target Cacheline Wrap mode burst ordering is disabled, PCI target Memory Read Line commands are treated as Memory Reads (no prefetch), and CPU 4-beat reads to PCI Memory space do not use the Memory Read Line command.

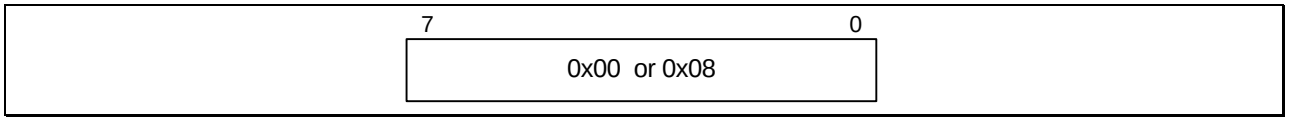


Figure 12: Cacheline Size Register Layout

4.1.4.7 Latency Timer (0x0D)

Description: This register specifies the value of the Latency Counter in PCI bus clock cycles for the Power PCI when it is master of the PCI bus. When the **FRAME#** signal is asserted, the Latency Counter will be loaded from the value stored in this register and then begin counting down. The PCI bus will not be released when **GNT#** has been removed, unless the Latency Counter has reached 0x'00'. Setting this register to 0x'00' causes the PCI bus to be released immediately upon removal of **GNT#**. This register is initialized to 0x'00' on **RST#**. The Latency Timer is read/write accessible from both the PCI and the CPU interfaces. The Power PCI disables the latency timer function when the *Disable Latency Timer* bit is set in the Power PCI Configuration register.

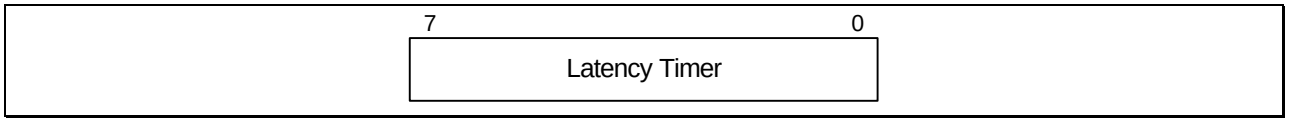


Figure 13: Latency Timer Register Layout

4.1.4.8 Header Type Register (0x0E)

Description: This register indicates the layout of the Power PCI's Configuration Header space (offsets 0x'0000' - 0x'003F'). The Power PCI does not support multiple functions and uses a standard Configuration Space Header layout. The Header Type register is read only and is accessible from both the PCI and the CPU interfaces. A write to the Header Type register does not change its value.

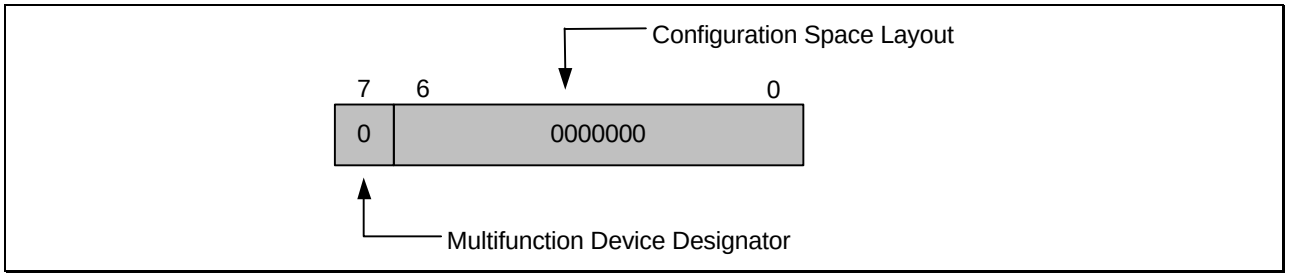


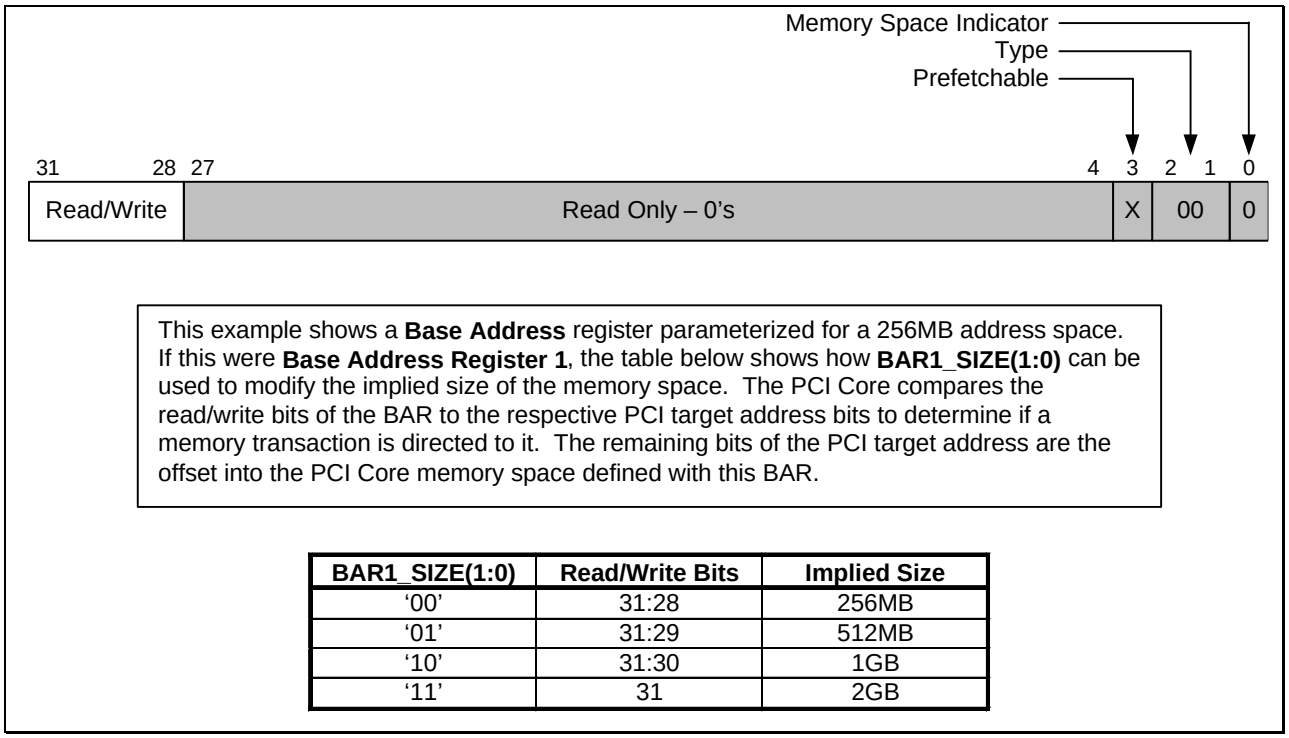
Figure 14: Header Type Register Layout

Bit/Field Definitions:

Bit 7	***Multifunction Device Designator:*** This bit indicates if the device supports multiple functions. The Multifunction Device Designator bit of the Header Type register returns 0b'0' when read.
Bits 6:0	***Configuration Space Layout:*** This field identifies the configuration space header layout. The Header Type register returns 0x'00' when read.

4.1.4.9 Base Address Registers (0x10, 0x18)

Description: The **Base Address registers** identify the PCI address range(s) a PCI target responds to on the PCI bus. The Power PCI provides two **Base Address Registers (BAR1 and BAR2).** The Power PCI does not support I/O Space as a target on the PCI. The high order bits of implemented **Base Address registers (BAR1 and BAR2)** are read/write from both the PCI and CPU interfaces. All other bits are read only.



This example shows a **Base Address** register parameterized for a 256MB address space. If this were **Base Address Register 1**, the table below shows how **BAR1_SIZE(1:0)** can be used to modify the implied size of the memory space. The PCI Core compares the read/write bits of the BAR to the respective PCI target address bits to determine if a memory transaction is directed to it. The remaining bits of the PCI target address are the offset into the PCI Core memory space defined with this BAR.

BAR1_SIZE(1:0)	Read/Write Bits	Implied Size
'00'	31:28	256MB
'01'	31:29	512MB
'10'	31:30	1GB
'11'	31	2GB

Figure 15: Memory Base Address Register Layout

Bit/Field Definitions:

Bits 31:4	***Base Address:*** The Base Address field resets to 0x'000's. The Power PCI uses the 0x'F000 0000' constant for BAR1 and 0x'FFF0 0000' for BAR2 to determine how many high order bits of this Base Address field are read/write. This number also determines the size of the memory space defined by this BAR. For example, if the constant indicates that bits 31:28 of this BAR are read/write, then the Power PCI responds to memory accesses to this BAR (if enabled in the Command register) in a 256Mbyte range of the PCI memory address space starting at the address specified by bits 31:28 of this BAR. The RAD750 board is configured with bits 31:28 as 0x'00'.
Bit 3	***Prefetchable:*** This bit is read only and is hardcoded (and reset) to the value 0b'1' for BAR1 and 0b'0' for BAR2. This is an informational bit and is used to indicate that the memory space has no side effects on reads. The Power PCI does not use this bit internally.
Bits 2:1	***Type:*** These bits are read only and are reset to the 0b'00'.
Bit 0	***Memory Space Indicator:*** This bit is read only and is set to 0b'0' for memory space BAR's. This bit indicates that the Power PCI only responds to memory transactions in this BAR's address range.

4.1.4.10 Subsystem Vendor ID (0x2C)

Description: This register is used to identify the vendor of the add-in board or subsystem where the device that uses the Power PCI resides. The Power PCI uses the two constants defined below to control the behavior of this register. This register is always read only from the PCI and CPU interfaces. A write to the Subsystem Vendor ID register from the PCI or CPU does not change its value. The initial value of the Subsystem Vendor ID register is set to the constant 0x'0000'.

Note: The Power PCI does not contain logic that prohibits the PCI from reading this register prior to it being written by the CPU. It is up to system specific software to ensure this PCI requirement is met.

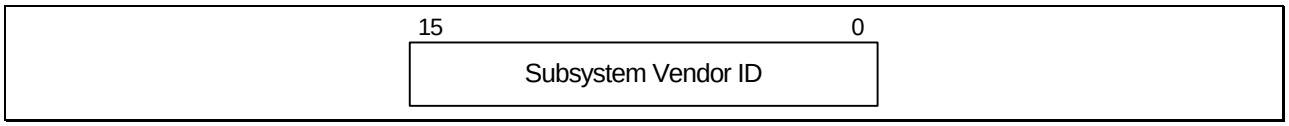


Figure 16: Subsystem Vendor ID Register Layout

4.1.4.11 Subsystem ID (0x2A)

Description: This register is used to identify the add-in board or subsystem where the device that uses the Power PCI resides. This register is always read only from the PCI and CPU interfaces. A write to the Subsystem ID register does not change its value. The value of the Subsystem ID register is 0x'0000'.

Note: The Power PCI does not contain logic that prohibits the PCI from reading this register prior to it being written by the CPU. It is up to system specific software to ensure this PCI requirement is met.

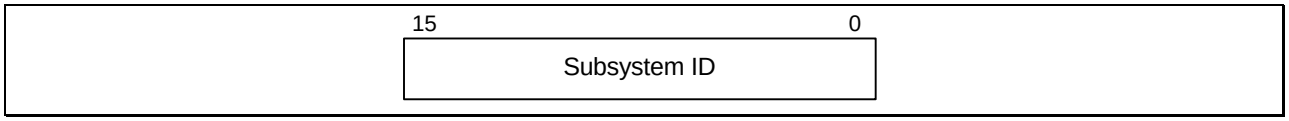


Figure 17: Subsystem ID Register Layout

4.1.4.12 Interrupt Line (0x3C)

Description: Although the Power PCI does not provide any PCI interrupt pins, this read/write register is provided for software use. The Power PCI itself does not use the contents of this register. The Interrupt

Line register is read/write from both the PCI and CPU interfaces. The Interrupt Line register is reset to 0x'00' on any reset. The **UART_IO_OUT2_L** signal is connected to the **PCI_INTA#** signal and when the UART sets **UART_IO_OUT2_L**, software uses the value in this register as a vector. This register is intended for lab use only.

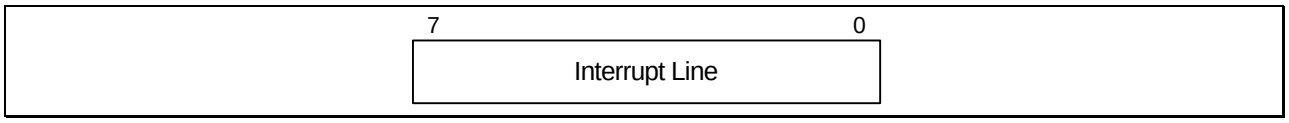


Figure 18: Interrupt Line Register Layout

4.1.4.13 Interrupt Pin (0x3D)

Description: Although the Power PCI does not provide any PCI interrupt pins, this read only register is provided for software use. The Power PCI itself does not use the contents of this register. The Interrupt Pin register is 'read only' from both the PCI and CPU interfaces. The Interrupt Pin register contains the value to 0x'00' when the **INTR_PIN_VALUE** external input signal is 0b'0', and 0x'01', when **INTR_PIN_VALUE** is 0b'1'. For this RAD750 board, **INTR_PIN_VALUE** is 0b'1'.

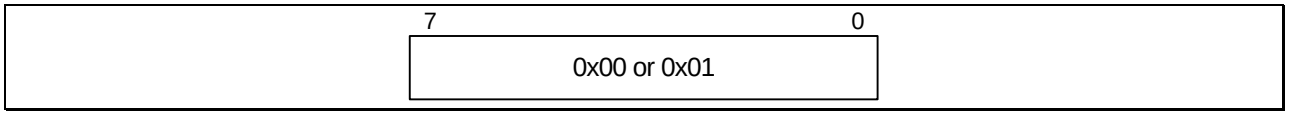


Figure 19: Interrupt Pin Register Layout

4.1.4.14 MIN_GNT (0x3E)

Description: This read only register specifies the setting for how long a burst period the Power PCI requires, assuming a clock rate of 33 MHz. The MIN_GNT register is read only and is accessible from both the PCI and CPU interfaces. A write to the MIN_GNT register does not change its value. The value in the MIN_GNT register specifies how long a burst period, in 0.25 µs intervals the device the Power PCI resides on requires. A value of 0b'0' indicates the device has no major requirements for the settings of Latency Timers. The reset value placed in the MIN_GNT register is 0x'00'.

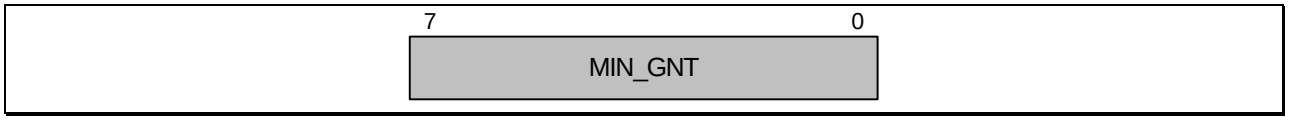


Figure 20: MIN_GNT Register Layout

4.1.4.15 MAX_LAT (0x3F)

Description: This register specifies the setting for how often the Power PCI needs to gain access to the PCI Bus. The MAX_LAT register is read only and is accessible from both the PCI and CPU interfaces. A write to the MAX_LAT register does not change its value. The value in the MAX_LAT register defines in 0.25 µs intervals, how often the Power PCI needs to gain access to the PCI Bus. A value of 0b'0' indicates the Power PCI has no major requirements for the settings of Latency Timers. The value placed in the MAX_LAT register is 0x'00'.

Figure 21: MAX_LAT Register Layout

4.1.4.16 Status 2 Register (0x50)

Description: This register is an extension to the standard PCI **Status** register. It provides additional status on errors detected in the Power PCI. Bits 15:8 (except for read only bits 10:9) of the PCI **Status** register are mirrored in bits 31:24 of the **Status 2** register. Bits 26:25 of the **Status 2** register are reserved. All reserved bits in the **Status 2** register returns 0's on reads and are ignored on writes. The **Status 2** register is accessible from both the PCI and CPU interfaces. The **Status 2** register has a reset value of 0x'0000 0000', however it is only reset on a chip reset. The **Status 2** register is not reset on a PCI reset. Bits in the **Status 2** register are set by various conditions within the Power PCI, as defined below. Bits in the **Status 2** register are cleared to 0b'0' when the register is written, and the corresponding bit location is set to 0b'1'. Bits 31:24 are also cleared to 0b'0' when the **Status 2** register is written, and the corresponding bit location is set to 0b'1'.

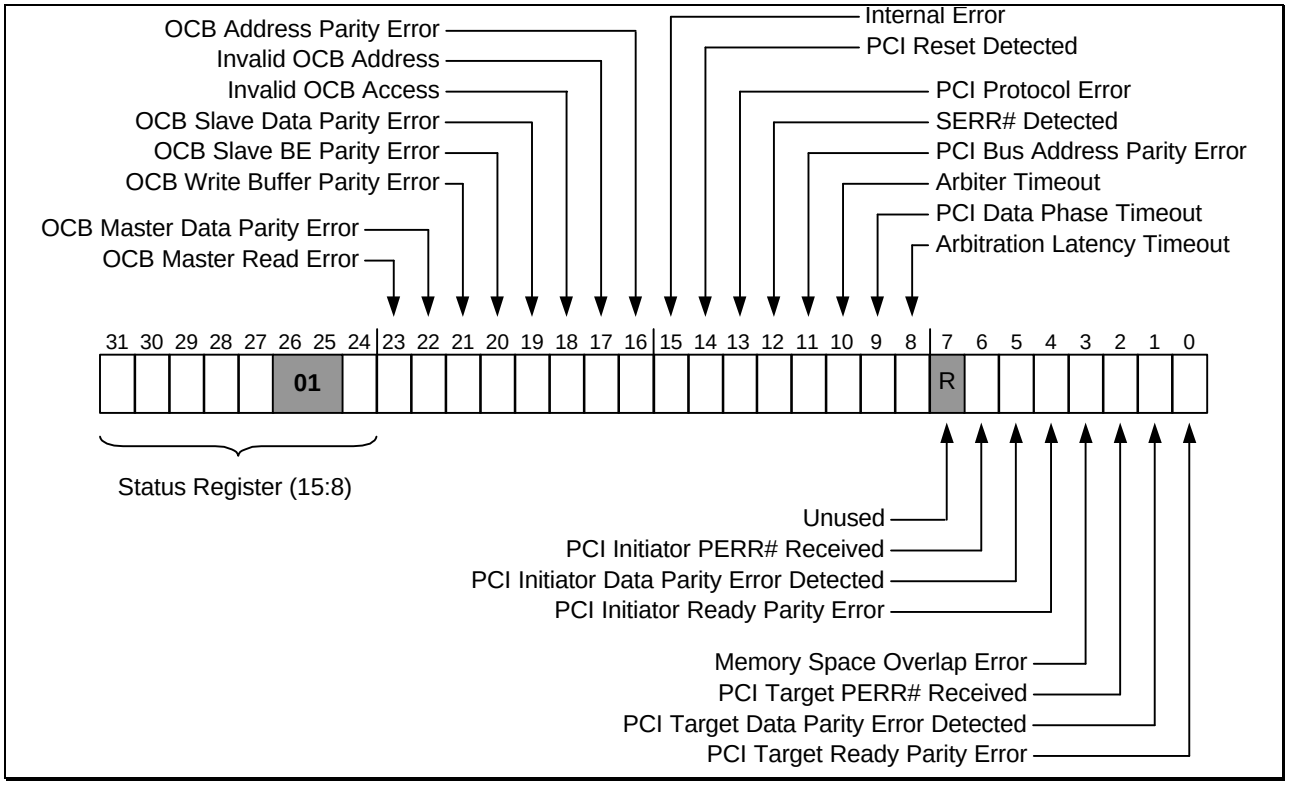


Figure 22: Status 2 Register Layout

Bit/Field Definitions:

Mirrored PCI Status Register bits	
Bit 31	***Detected Parity Error:*** Detection of a parity error on the PCI bus causes the *Detected Parity Error* bit of the **Status** and **Status 2** registers to be set to 0b'1'. Writing this bit with 0b'1' clears it to 0b'0' in both status register locations.
Bit 30	***Signaled System Error:*** The *Signaled System Error* bit is set to 0b'1' in both the **Status** and **Status 2** registers whenever the SERR# output is asserted. Writing this bit with 0b'1' clears it to 0b'0' in both status register locations.
Bit 29	***Received Master Abort:*** The *Received Master Abort* bit is set to 0b'1' in both the **Status** and **Status 2** registers whenever the Power PCI is the PCI bus initiator and terminates a transaction with master abort. Writing this bit with 0b'1' clears it to 0b'0' in both status register locations.

Bit 28	***Received Target Abort:*** The *Received Target Abort* bit is set to 0b'1' in both the **Status** and **Status 2** registers whenever the Power PCI is the PCI bus initiator and its transaction is terminated with a target abort. Writing this bit with 0b'1' clears it to 0b'0' in both status register locations.
Bit 27	***Signaled Target Abort:*** The *Signaled Target Abort* bit is set to 0b'1' in both the **Status** and **Status 2** registers whenever the Power PCI is the PCI bus target and terminates a transaction with target abort. Writing this bit with 0b'1' clears it to 0b'0' in both status register locations.
Bits 26-25	***DEVSEL Timing***: These bits are read-only and return '01'b indicating medium timing for assertion of **DEVSEL#** as a PCI target. Writes to the **Status** register or the **Status 2** register do not affect these
Bit 24	***Master Data Parity Error:*** The Power PCI sets the *Master Data Parity Error* bit to 0b'1' in both the **Status** and **Status 2** registers when it is the PCI bus master, PERR# has been asserted, and the *Parity Error Response* bit in the Command register is 0b'1'. Writing this bit with 0b'1' clears it to 0b'0' in both status register locations.

OCB Error bits	
Bit 23	***OCB Master Read Error:*** This bit is set to 0b'1' when the OCB Master detects M_RD_D_ERR_IN(1:0) active on an OCB master read transaction.
Bit 22	***OCB Master Data Parity Error:*** This bit is set to 0b'1' when the OCB Master detects bad data parity on an OCB master read transaction.
Bit 21	***OCB Write Buffer Parity Error:*** This bit is set to 0b'1' when the OCB Slave detects bad parity on the output of the OCB Write Buffer.
Bit 20	***OCB Slave BE Parity Error:*** This bit is set to 0b'1' when the OCB Slave detects bad byte enable parity on an OCB slave write transaction.
Bit 19	***OCB Slave Data Parity Error:*** This bit is set to 0b'1' when the OCB Slave detects bad data parity on an OCB slave write transaction.
Bit 18	***Invalid OCB Access:*** This bit is set to 0b'1' when the OCB attempts an access other than one to Power PCI's internal register space when PCI RST# is active.
Bit 17	***Invalid OCB Address:*** This bit is set to 0b'1' when the OCB attempts an internal register access with the OCB address offset in the range 0x'0100' through 0x'FFFF'.
Bit 16	***OCB Address Parity Error:*** This bit is set to 0b'1' when bad parity is detected by the OCB slave over the OCB Address, Transfer Size, Command, and Request signals.

Internal Error bits	
Bit 15	***Internal Register Error:*** The *Internal Error* bit of the **Status 2** register is set to 0b'1' when a parity error is detected on any internal configuration register that is protected by parity or when an invalid state is detected in the internal state machines in the PCI interface logic.

General PCI Error/Status Bits	
Bit 14	***PCI Reset Detected:*** This bit is set to '1' when PCI RST# has gone active and the Power PCI is not the central resource.

Bit 13	***PCI Protocol Error:*** This bit is set when the Power PCI logic detects a protocol error on the PCI bus and the *PCI Protocol Checking Enable* is set in the Error Checking register. • The *PCI Protocol Error* bit of the **Status 2** register is set to 0b'1' when this device has been configured to provide PCI bus arbitration, FRAME# is asserted and the Arbiter has not granted the bus to any device. • The *PCI Protocol Error* bit of the **Status 2** register is set to 0b'1' when IRDY#, DEVSEL#, STOP#, or TRDY# are asserted during the address phase of a PCI transaction. • The *PCI Protocol Error* bit of the **Status 2** register is set to 0b'1' when FRAME# and/or IRDY# change after assertion of IRDY# during a data phase, but before the data phase completes. • The *PCI Protocol Error* bit of the **Status 2** register is set to 0b'1' when a master allows the bus to go idle (both FRAME# and IRDY# are de-asserted on the same cycle) during a data phase that a target has claimed (DEVSEL# is active). • The *PCI Protocol Error* bit of the **Status 2** register is set to 0b'1' when IRDY# is not de-asserted on the clock cycle that follows the completion of the last data phase of a transaction. • The *PCI Protocol Error* bit of the **Status 2** register is set to 0b'1' when an initiator does not terminate correctly due to target termination. (That is, when STOP# is asserted during an ongoing transaction (FRAME# is still asserted), the master must de-assert FRAME# on the first cycle thereafter that IRDY# is asserted.) • The *PCI Protocol Error* bit of the **Status 2** register is set to 0b'1' when a target terminates an ongoing transaction with STOP#, but STOP# is de-asserted prior to FRAME# being de-asserted. • The *PCI Protocol Error* bit of the **Status 2** register is set to 0b'1' when a target has asserted TRDY# or STOP# for a data phase, and it changes TRDY#, STOP#, or DEVSEL# before the data phase completes.
Bit 12	***SERR# Detected:*** The *SERR# Detected* bit of the **Status 2** register is set to 0b'1' when the Power PCI detects that SERR# is active. This bit is independent of the setting of the *SERR# Enable* bit in the Command register.
Bit 11	***PCI Bus Address Parity Error:*** The *PCI Bus Address Parity Error* bit of the **Status 2** register is set to 0b'1' when the Power PCI detects bad parity during the address phase of a PCI transaction. This bit is not affected by the *Parity Error Response* bit in the Command register.
Bit 10	***Arbiter Timeout:*** The *Arbiter Timeout* bit is set to 0b'1' when the Power PCI is the central resource, and the bus has been idle for 16 cycles after an ARB_GNT(7:0)# line has been activated and the corresponding ARB_REQ(n)# is still active.
Bit 9	***PCI Data Phase Timeout:*** This bit is set when the Power PCI is the central resource, the *PCI Data Phase Timeout Enable* bit is set in the Error Checking register, and the Power PCI has detected a PCI Data Phase Timeout. This timeout occurs when any data phase in any PCI transaction takes longer to complete than the time allotted by the PCI Data Phase Timer.
Bit 8	***Arbitration Latency Timeout:*** This bit is set when the Power PCI is the central resource, the *Arbitration Latency Timeout Enable* bit is set in the Error Checking register, and the Power PCI has detected an Arbitration Latency Timeout. This timeout occurs when the Power PCI's Arbiter has removed the current PCI master's GNT#, and the master has not released the PCI bus by deasserting both FRAME# and IRDY# within the time allotted by the Arbitration Latency Timer.

PCI Master Error bits	
Bit 7	***Reserved***: These bits are reserved, and return 0b'0' when read.
Bit 6	***PCI Initiator PERR# Received:*** The *PCI Initiator PERR# Received* bit of the **Status 2** register is set to 0b'1' when the Power PCI is master on the PCI bus and it detects an active PERR# that it did not drive. (Basically, this means that a parity error was reported to the Power PCI by a PCI target that the Power PCI was writing to.) This bit is not affected by the *Parity Error Response* bit in the Command register.
Bit 5	***PCI Initiator Data Parity Error Detected:*** The *PCI Initiator Data Parity Error Detected* bit of the **Status 2** register is set to 0b'1' when the Power PCI detects a PCI data parity error when it is the master of a PCI read transaction. This bit is not affected by the *Parity Error Response* bit in the Command register.
Bit 4	***PCI Initiator Ready Parity Error:*** The *PCI Initiator Ready Parity Error* bit of the **Status 2** register is set to 0b'1' when the Power PCI is master, and detects a parity error across IRDY#, TRDY#, and RDY_PAR and this error is enabled by both the *Parity Error Response* bit in the Command register, and the *Ready Parity Enable* bit in the Error Checking register.

PCI Target Error bits	
Bit 3	***Memory Space Overlap Error:*** This bit is set to 0b'1' when the Power PCI detects a PCI target memory transaction that matches multiple Base Address registers configured for memory space.
Bit 2	***PCI Target PERR# Received:*** The *PCI Target PERR# Received* bit of the **Status 2** register is set to 0b'1' when the Power PCI is the target of a transaction on the PCI and it detects PERR# asserted on a read transaction. This bit is not affected by the *Parity Error Response* bit in the Command register.
Bit 1	***PCI Target Data Parity Error Detected:*** The *PCI Target Data Parity Error Detected* bit of the **Status 2** register is set to 0b'1' when the Power PCI detects a PCI data parity error when it is the target of a PCI write transaction. This bit is not affected by the *Parity Error Response* bit in the Command register.
Bit 0	***PCI Target Ready Parity Error:*** The *PCI Target Ready Parity Error* bit of the **Status 2** register is set to 0b'1' when the Power PCI is target, and it detects a parity error across IRDY#, TRDY#, and RDY_PAR, and this error is enabled by both the *Parity Error Response* bit in the Command register, and the *Ready Parity Enable* bit in the Error Checking register.

4.1.4.17 Status 2 Mask Register (0x54)

Description: This register provides an interrupt mask for bits in the **Status 2** register. There is a one to one correspondence between bits in the **Status 2** register and bits in the **Status 2 Mask** register. The **Status 2** Mask register is read/write accessible from both the PCI and CPU interfaces. The **Status 2** Mask register has a reset value of 0x'0000 0000'. When cleared to 0b'0', bits in the **Status 2** Mask register mask off interrupts from the corresponding bits in the **Status 2** register. When set to 0b'1', bits in the **Status 2** Mask register enable interrupts from the corresponding bits in the **Status 2** register.

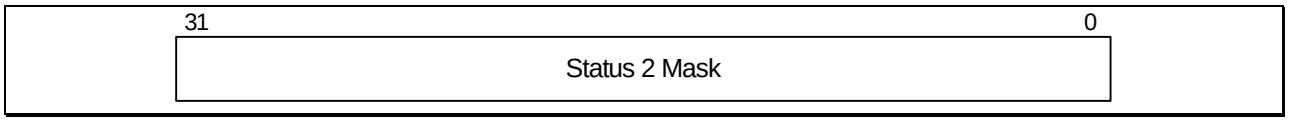


Figure 23: Status 2 Mask Register Layout

Bit/Field Definitions:

Bit 31:0	***Status 2 Mask:*** Each bit in the **Status 2 Mask** register masks or unmasks interrupt generation for the corresponding bit the **Status 2** register. Note that bits 26:25 of the **Status 2** register are read-only and do not participate in interrupt generation and masking.

4.1.4.18 BAR1 Size Adjust Register (0x58)

Description: This register allocates the number of 8K pages mapped to the PCI bus in **Base Address 1** register. The Power PCI only responds to PCI access in BAR1 in the address range allocated by this register. The width of **Bar 1 Size Adjust** can be modified by the **BAR1_SIZE(1:0)** inputs. The bit layouts shown below demonstrate how the memory space defined by BAR1 is 256 MByte, 512 MByte, 1 GByte or 2 GByte, as selected by **BAR1_SIZE(1:0).**

For example, if the BAR1 memory space is 512 MByte, there are a total of 64K, 8K pages within that space. In this case, 16 bits are required to indicate the number of pages. Bits 23:16 would be read only; bits 15:0 would contain the binary number indicating the number of 8K pages the Power PCI responds to within BAR1. Software can write to bits 23:16, but since they are reserved, only 0's will be read back in those bit positions and bits 15:0 will contain a valid number of 8K pages.

The BAR1 Size Adjust register is read/write accessible from both the PCI and CPU interfaces. The BAR1 Size Adjust register has a reset value of 0x'0000' which indicates maximum size.

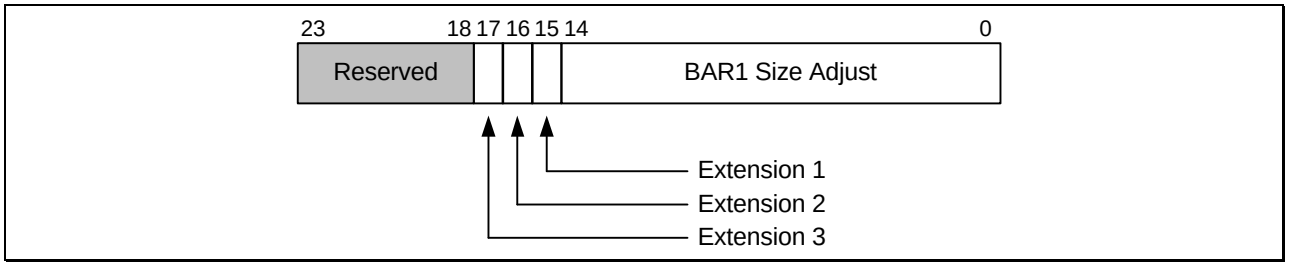


Figure 24: BAR1 Size Adjust Register Layout

Bit/Field Definitions:

Bit 17	***Extension 3:*** This bit provides the MSB for the **BAR1 Size Adjust** register when **BAR1_SIZE(1:0)** is 0b'11' indicating a 2 GByte memory space for BAR1. When **BAR1_SIZE(1:0)** indicates a BAR1 memory space that is less than 2 GByte, this bit is read-only, returning 0b'0' on reads.
Bit 16:	***Extension 2:*** This bit provides the MSB for the **BAR1 Size Adjust** register when **BAR1_SIZE(1:0)** is 0b'10' indicating a 1 GByte memory space for BAR1. This bit is also included in the **BAR1 Size Adjust** when **BAR1_SIZE(1:0)** is 0b'11'. When **BAR1_SIZE(1:0)** indicates a BAR1 memory space that is less than 1 GByte, this bit is read-only, returning 0b'0' on reads.
Bit 15:	***Extension 1:*** This bit provides the MSB for the **BAR1 Size Adjust** register when **BAR1_SIZE(1:0)** is 0b'01' indicating a 512 MByte memory space for BAR1. This bit is included in the **BAR1 Size Adjust** when **BAR1_SIZE(1:0)** is 0b'11' or 0b'10'. When **BAR1_SIZE(1:0)** indicates a BAR1 memory space that is less than 512 MByte, this bit is read-only, returning 0b'0' on reads.
Bits 14:0	***8K Pages:*** The value placed in **BAR1 Size Adjust** Register indicates, in binary, the number of 8k pages the Power PCI responds to in the **Base Address 1** register. A value of 0x'0000' indicates that the entire BAR1 Address range is valid. A value of 0x'0001' indicates that the Power PCI will only respond to addresses in the first 8K of **Base Address 1** register.

4.1.4.19 Power PCI Configuration Register (0x5C)

Description: This register provides control over some of the operational features of the Power PCI. Enabling or disabling some of the features in this section may violate the PCI Specification. Note that the upper half of this register provides control over PCI Target type functions while the lower half provides control over PCI Master type functions. The Power PCI Configuration register is read/write accessible from both the PCI and CPU interfaces. With the exception of the *BAR(n) Prefetch Enable* bits, the Power PCI Configuration register has a reset value of 0x'0000 0000'. *BAR(1) Prefetch Enable* bit is reset to 0b'1' and *BAR(2) Prefetch Enable* bit is reset to 0b'0'.

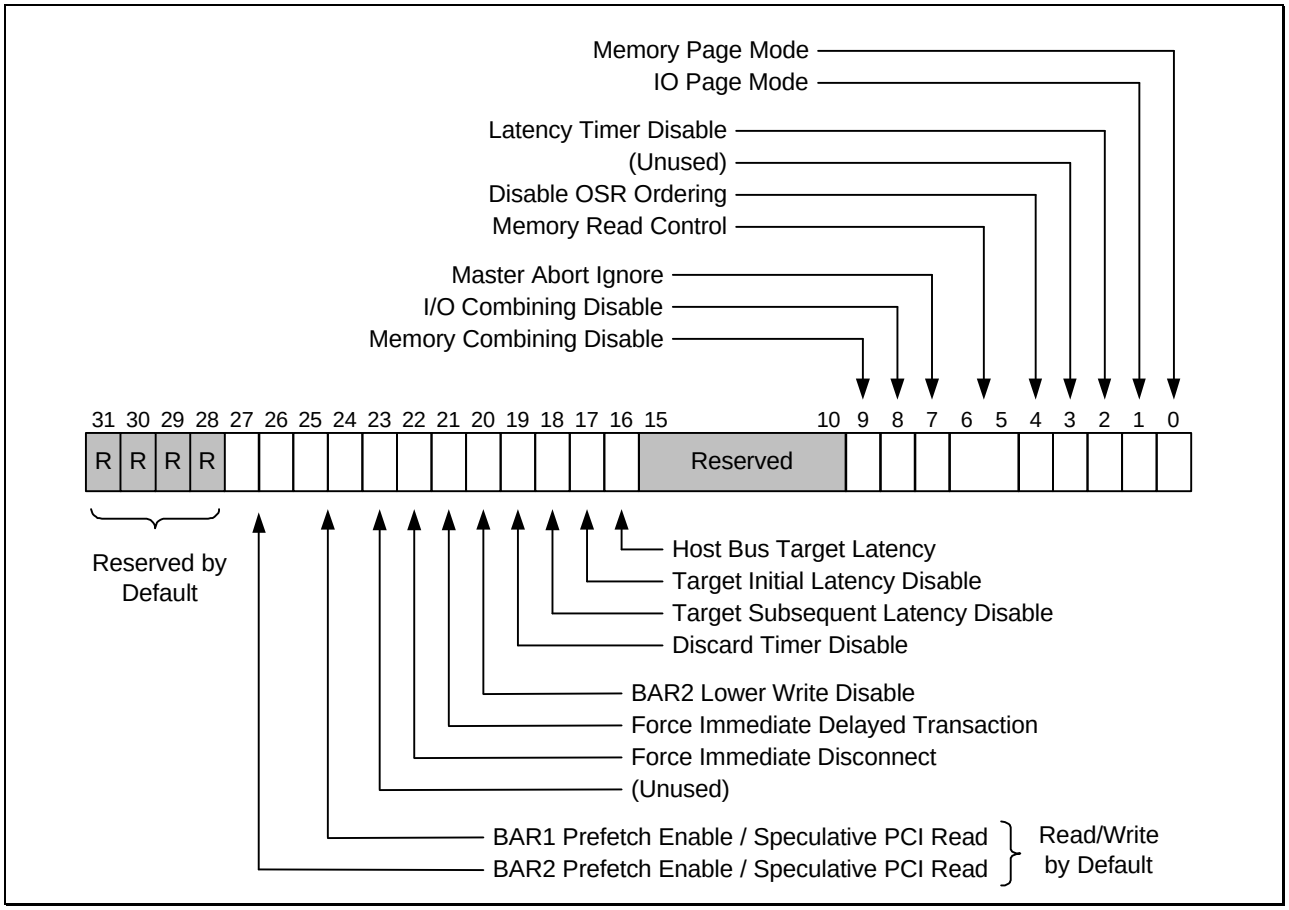


Figure 25: Power PCI Configuration Register Layout

Bit/Field Definitions:

Bit 31:28	***Reserved***: These bits are reserved, and return 0b'0' when read.
Bit 27	***BAR2 Prefetch Enable:*** Power PCI uses this bit in an attempt to treat PCI target Memory Read commands as prefetchable reads. If this bit is set to 0b'1', the ***BAR2 Speculative Read*** bit is reset to 0b'0', and the **Cacheline Size** register is set to 0x'08', the Power PCI treats a Memory Read command as a Memory Read Line command. If this bit is set to 0b'1' and the ***BAR2 Speculative Read*** bit is set to 0b'1', the Power PCI treats a Memory Read command as a Memory Read Multiple command. When the ***BAR2 Prefetch Enable*** is cleared to 0b'0', the Power PCI does not prefetch data on Memory Read commands. The Power PCI resets this bit to the value of the ***Prefetchable*** bit in **Base Address 2** register, which is 0b'0'.

Bit 26	***BAR2 Speculative PCI Read:*** The Power PCI uses this bit in conjunction with the ***BAR2 Prefetch Enable*** bit as described in that bit's description. Additionally, if this bit is set to 0b'1', the Power PCI treats Memory Read Line commands as Memory Read Multiple commands.
Bit 25	***BAR1 Prefetch Enable:*** Power PCI uses this bit in an attempt to treat PCI target Memory Read commands as prefetchable reads. If this bit is set to 0b'1', the ***BAR1 Speculative Read*** bit is reset to 0b'0', and the **Cacheline Size** register is set to 0x'08', the Power PCI treats a Memory Read command as a Memory Read Line command. If this bit is set to 0b'1' and ***the BAR1 Speculative Read*** bit is set to 0b'1', the Power PCI treats a Memory Read command as a Memory Read Multiple command. When the ***BAR1 Prefetch Enable*** is cleared to 0b'0', the Power PCI does not prefetch data on Memory Read commands. The Power PCI resets this bit to the value of the ***Prefetchable*** bit in **Base Address 1** register, which is 0b'1'.
Bit 24	***BAR1 Speculative PCI Read:*** The Power PCI uses this bit in conjunction with the ***BAR1 Prefetch Enable*** bit as described in that bit's description. Additionally, if this bit is set to 0b'1', the Power PCI treats Memory Read Line commands as Memory Read Multiple commands.
Bit 23	***Unused:*** This bit has read/write accessibility but is functionally unused by the Power PCI.
Bit 22	***Force Immediate Disconnect:*** The Power PCI uses this bit on PCI target memory read transactions. When this bit is set to 0b'1', the Power PCI disconnects immediately from an on-going memory read transaction when the PCI Read Buffers go empty rather than wait for the Target Subsequent Latency Timer to expire. On non-prefetchable reads, this will cause the PCI Target to disconnect after a single data phase on attempted memory read burst transactions.
Bit 21	***Force Immediate Delayed Transaction:*** When this bit is set to 0b'1', the Power PCI treats a PCI target memory read transaction immediately as a delayed transaction without first attempting to handle it as a connected transaction.
Bit 20	***BAR2 Lower Write Disable:*** When this bit is set to 0b'1', the Power PCI disables writing the lower half of the memory address space specified with the **Base Address 2** register. Any write to the lower half of this memory space when the BAR2 Lower Write Disable is set results in a target-abort by the Power PCI.
Bit 19	***Discard Timer Disable:*** When the Discard Timer Disable is set to 0b'1', the Power PCI disables the Delayed Transaction Discard Timer and therefore never discards a Delayed Transaction. The Power PCI may not meet PCI Specification requirements if this bit is set.
Bit 18	***Target Subsequent Latency Disable:*** When this bit is set to 0b'1', the Power PCI disables the PCI Target Subsequent Latency Timer. When this timer is disabled, the Power PCI will not disconnect a PCI target transaction solely because it cannot meet the PCI Specification target subsequent latency requirement of 8 cycles. The Power PCI may not meet PCI Specification requirements if this bit is set.
Bit 17	***Target Initial Latency Disable:*** When this bit is set to 0b'1', the Power PCI disables the PCI Target Initial Latency Timer. When this timer is disabled, the Power PCI will not retry a PCI target transaction solely because it cannot meet the PCI Specification target initial latency requirement of 16 cycles (or 32 cycles if the Host Bus Target Latency bit is set). The Power PCI may not meet PCI Specification requirements if this bit is set.

Bit 16	***Host Bus Target Latency:*** When the Host Bus Target Latency bit is set to 0b'1', (and the Target Initial Latency Timer is not disabled), Power PCI allows a worst case target initial latency of 32 cycles. When the Host Bus Target Latency bit is reset to 0b'0', (and the Target Initial Latency Timer is not disabled), Power PCI allows a worst case target initial latency of 16 cycles. This bit should only be set to 0b'1' for Host Bus Bridges in order to meet the PCI Specification requirements.
Bit 15:10	***Reserved***: These bits are reserved, and return 0b'0' when read.
Bit 9	***Memory Combining Disable:*** When the ***Memory Combining Disable*** bit is set to 0b'1', the Power PCI will not combine contiguous OCB Slave transactions to PCI Memory space into a single PCI memory burst transaction. When this bit is cleared to 0b'0', the Power PCI will allow combining of contiguous OCB Slave transactions within the same OCB to PCI Memory space. This bit controls combining for the entire OCB to PCI Memory space.
Bit 8	***I/O Combining Disable:*** When the ***I/O Combining Disable*** bit is set to 0b'1', the Power PCI will not combine contiguous OCB Slave transactions to PCI I/O space into a single PCI I/O burst transaction. When this bit is cleared to 0b'0', the Power PCI will allow combining of contiguous OCB Slave transactions within the same OCB to PCI I/O space. This bit controls combining for the entire OCB to PCI I/O space.
Bit 7	***Master Abort Ignore:*** When the Master Abort Ignore bit is set to 0b'1', the Power PCI does not assert **S_RD_D_ERR_ IN(1:0)** on PCI Master configuration read transactions that terminate with Master-Abort. This bit does not affect the Received Master Abort bit in the Status register.
Bits 6:5	***Memory Read Control:*** This field is used on 4-beat CPU to PCI memory read transactions when not in PCI Memory Page mode to indicate the type of PCI memory read command to perform. The Memory Read Control field is decoded as follows: '0x' Memory Read 0b'10' Memory Read Line (if the Cacheline Size register is 0x'08') 0b'11' Memory Read Multiple
Bit 4	***Disable OSR Ordering:*** When this bit is set to 0b'1', the Power PCI disables the PCI transaction ordering constraint. When this bit is reset to 0b'0', read transaction ordering is as normal. The Power PCI may not meet PCI Specification requirements if this bit is set.
Bit 3	***Unused:*** This bit has read/write access but is functionally unused by the Power PCI.
Bit 2	***Latency Timer Disable:*** When this bit is set to 0b'1', the Power PCI ignores the master Latency Timer which governs a master's tenure on the PCI bus. The Power PCI will, therefore, continue a burst transaction when the Latency Timer has been disabled as long as it can keep the transaction going, regardless of the state of the GNT# signal. The Power PCI may not meet PCI Specification requirements if this bit is set.
Bit 1	***IO Page Mode:*** When IO Page Mode is set to 0b'1', the Power PCI uses the O2P I/O Paging registers to perform paging in PCI I/O space on CPU to PCI I/O space transactions. When this bit is reset to 0b'0', the Power PCI does a direct CPU to PCI address translation on CPU to PCI I/O space transactions.

Bit 0	***Memory Page Mode:*** When Memory Page Mode is set to 0b'1', the Power PCI uses the O2P Mem Paging registers to perform paging in PCI memory space on CPU to PCI memory space transactions. When this bit is reset to 0b'0', the Power PCI does a direct CPU to PCI address translation on CPU to PCI memory space transactions.

4.1.4.20 Arbitration Priority Register (0x60)

Description: When the Power PCI is configured as the central resource, this register specifies the arbitration priority level for each of the eight possible PCI masters. The Arbitration Priority register is read/write accessible from both the PCI and CPU interfaces. The Arbitration Priority register has a reset value of 0x'0000'.

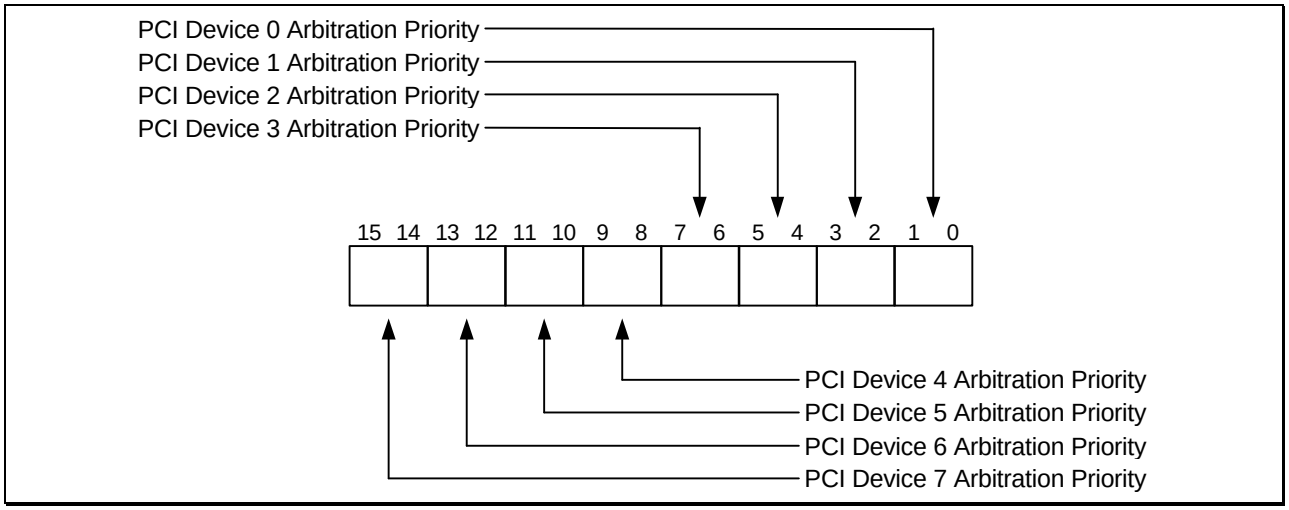


Figure 26: PCI Bus Arbitration Priority Register Layout

The Arbitration Priority Level is encoded as follows for all devices:

Bit Encoding	Arbitration Priority Level
00	Priority Level 1: Devices at this level have highest priority.
01	Priority Level 2: Devices at this level have medium priority.
10	Priority Level 3: Devices at this level have lowest priority.
11	Disabled: Devices at this priority level are disabled from participating in PCI arbitration. (That is, their **GNT#** will never be asserted.)

Bit/Field Definitions:

Bits 15:14	***PCI Device 7 Arbitration Priority Level:*** This field encodes the arbitration Priority Level for the PCI master connected to the PCI request/grant pair ARB_GNT7# and ARB_REQ7#.
Bits 13:12	***PCI Device 6 Arbitration Priority Level:*** This field encodes the arbitration Priority Level for the PCI master connected to the PCI request/grant pair ARB_GNT6# and ARB_REQ6#.
Bits 11:10	***PCI Device 5 Arbitration Priority Level:*** This field encodes the arbitration Priority Level for the PCI master connected to the PCI request/grant pair ARB_GNT5# and ARB_REQ5#.
Bits 9:8	***PCI Device 4 Arbitration Priority Level:*** This field encodes the arbitration Priority Level for the PCI master connected to the PCI request/grant pair ARB_GNT4# and ARB_REQ4#.

Bits 7:6	***PCI Device 3 Arbitration Priority Level:*** This field encodes the arbitration Priority Level for the PCI master connected to the PCI request/grant pair ARB_GNT3# and ARB_REQ3#.
Bits 5:4	***PCI Device 2 Arbitration Priority Level:*** This field encodes the arbitration Priority Level for the PCI master connected to the PCI request/grant pair ARB_GNT2# and ARB_REQ2#.
Bits 3:2	***PCI Device 1 Arbitration Priority Level:*** This field encodes the arbitration Priority Level for the PCI master connected to the PCI request/grant pair ARB_GNT1# and ARB_REQ1#.
Bits 1:0	***PCI Device 0 Arbitration Priority Level:*** This field encodes the arbitration Priority Level for the PCI master connected to the PCI request/grant pair ARB_GNT0# and ARB_REQ0#.

4.1.4.21 Error Checking Register (0x62)

Description: This register provides control for enabling and disabling various error checks within the Power PCI. Additionally, this register provides initial timer values for determining the length of certain timeout periods. The Error Checking register is read/write accessible from both the PCI and CPU interfaces. The Error Checking register has a reset value of 0x'0000'.

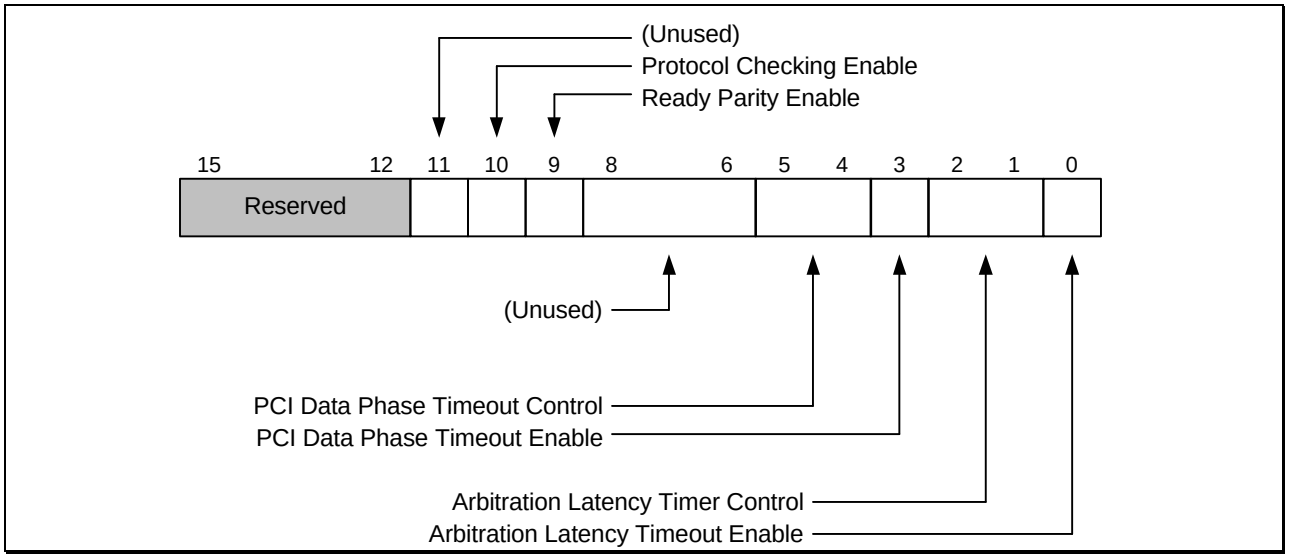


Figure 27: Error Checking Register Layout

Bit/Field Definitions:

Bits 15:12	***Reserved***: These bits are reserved, and return 0b'0' when read.
Bit 11	***Unused:*** This bit has read/write accessibility but is not used functionally by the Power PCI.
Bit 10	***Protocol Checking Enable:*** The Protocol Checking Enable bit, when set to 0b'1', enables the checking and reporting (via the **Status 2** register) of PCI protocol errors.
Bit 9	***Ready Parity Enable:*** The Ready Parity Enable bit, when set to 0b'1', enables the checking and reporting (via the **Status 2** register) of parity errors over IRDY#, TRDY#, and RDY_PAR. Ready Parity checking and reporting is disabled when this bit is cleared to 0b'0'. The Parity Error Response bit of the Command register must also be active for Ready Parity checking to be enabled.

Bits 8:6	***Unused:*** These bits have read/write accessibility but are not used functionally by the Power PCI.
Bits 5:4	***PCI Data Phase Timer Control:*** The PCI Data Phase Timer Control field of the Error Checking register sets the minimum number of clock cycles required for a PCI Data Phase Timeout as follows: 0b'00' = 256 0b'01' = 512 0b'10' = 1024 0b'11' = 2048
Bit 3	***PCI Data Phase Timeout Enable:*** The PCI Data Phase Timeout Enable, when set to 0b'1' enables the PCI Data Phase Timer to count, and enables the reporting of PCI Data Phase Timeout errors via the **Status 2** register.
Bits 2:1	***Arbitration Latency Timer Control:*** The Arbitration Latency Timer Control field of the Error Checking register sets the minimum number of clock cycles required for an Arbitration Latency Timeout as follows: 0b'00' = 256 0b'01' = 512 0b'10' = 1024 0b'11' = 2048
Bit 0	***Arbitration Latency Timeout Enable:*** The Arbitration Latency Timeout Enable, when set to 0b'1' enables the Arbitration Latency Timer to count, and enables the reporting of Arbitration Latency Timeout errors via the **Status 2** register.

4.1.4.22 Error Injection Register (0x64)

Description: This register provides the capability to inject errors into the Power PCI for diagnostic purposes. This register provides the capability to inject continuous (hard) and transient (one-time) errors. The Error Injection register is read/write accessible from both the PCI and CPU interfaces. The Error Injection register has a reset value of 0x'00'.

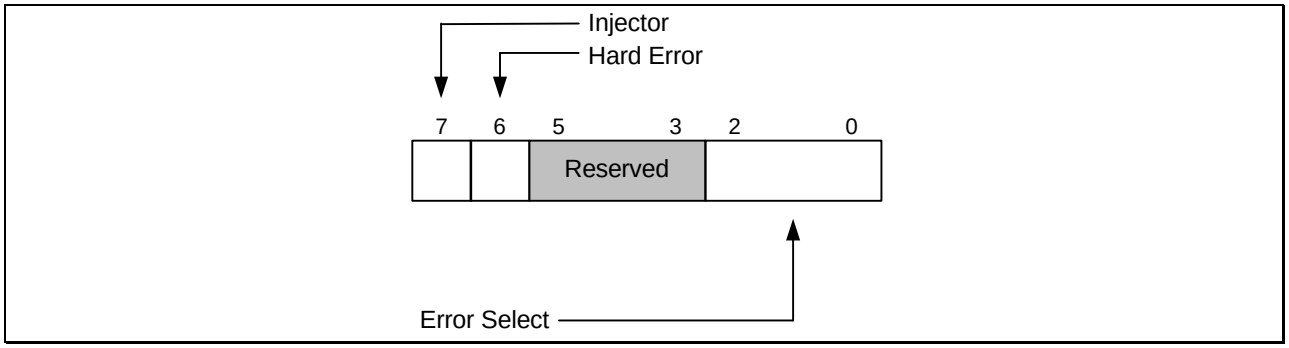


Figure 28: Error Injection Register Layout

Bit/Field Definitions:

Bit 7	***Injector:*** When this bit is set to 0b'1', the error encoded in the Error Select field of this register is injected into the Power PCI PCI function. Some errors will not be injected immediately if an internal condition is required to inject the error properly. When the Hard Error bit in this register is cleared to 0b'0', the injector bit is automatically reset after the error has been injected. When the Hard Error bit in this register is set to 0b'1', the Injector bit is not automatically reset, and therefore causes a hard error to occur.
Bit 6	***Hard Error:*** This bit indicates whether a hard or transient error is to be injected.

Bits 5:3	***Reserved***: These bits are reserved, and return 0b'0' when read.
Bits 2:0	***Error Select:*** This field indicates the type of error to be injected based on the following encoding: 0b'000' assert SERR# 0b'001' detect false PCI Protocol Error 0b'010' output data with bad parity on PCI bus 0b'011' output bad address parity on PCI bus 0b'100' output bad ready parity on PCI bus (when central resource) 0b'101' detect false Internal Register Parity Error 0b'110' output data with bad parity on the Power PCI Internal bus interface 0b'111' (unused, no error is injected)

4.1.4.23 Bus Reset Register (0x65)

Description: When configured as the central resource, this register provides control over the **RST#** signal. The Bus Reset register is a read/write register, and is accessible from both the PCI and CPU interfaces. When the Power PCI is the central resource, bit 0 of this register (*PCI Reset Active*) is used to control PCI RST#. When this bit is a 0b'1', PCI RST# is active. This bit is automatically set under certain error conditions. This register is set to 0x'11' on global reset when the Power PCI is the central resource, or 0x'01' when the Power PCI is not the central resource. This register is not affected by PCI reset. Bit 4 of this register is read –only and indicates that the Power PCI is central resource when it is '1', or that the Power PCI is not central resource when it is '0'. Software must ensure that PCI RST# is active long enough to meet the PCI Specification requirements.

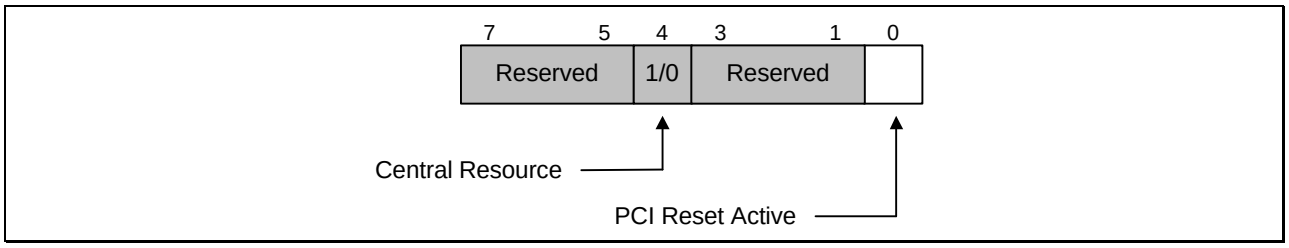


Figure 29: Bus Reset Register Layout

4.1.4.24 Power PCI Revision ID Register (0x66)

Description: This register is hardcoded to the revision ID of this Power PCI which is '3001'X The Power PCI Revision ID register is read only accessible from both the PCI and CPU interfaces.

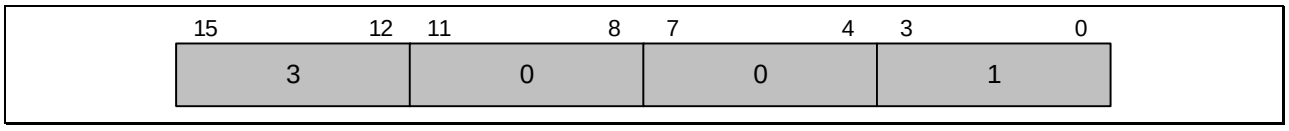


Figure 30: Power PCI Revision ID Register Layout

Bit/Field Definitions:

Bits 15:0	***Power PCI Revision ID:*** These bits are hardcoded to the value 0x'3001'. Writes to this register have no effect.

4.1.4.25 O2P I/O Paging Registers (0x80, 0x82, 0x84, 0x86, 0x88, 0x8A, 0x8C, 0x8E)

Description: This set of eight registers provides a paging capability on PCI I/O transactions.

31 27	26 16	15 11	10 0	Offset
O2P I/O Page 1	Reserved	O2P I/O Page 0	Reserved	0x0080
O2P I/O Page 3	Reserved	O2P I/O Page 2	Reserved	0x0084
O2P I/O Page 5	Reserved	O2P I/O Page 4	Reserved	0x0088
O2P I/O Page 7	Reserved	O2P I/O Page 6	Reserved	0x008C

Figure 31: PCI to OCB I/O Paging Register Layout

Each O2P I/O Paging (n) register is read/write accessible from both the PCI and CPU interfaces. However, bits 26:18 and 10:2 are reserved. Each paging register is two bytes wide. The O2P I/O Paging register reset values each are 0x'0000'.

Bit/Field Definitions:

Bits 31:27	***O2P I/O Page (n+1):*** In I/O Paging Mode, this field provides the PCI base address on Power PCI to PCI transactions for the address range (n+1).
Bits 26:16	***Reserved***: These bits are reserved, and return 0b'0' when read.
Bits 15:11	***O2P I/O Page (n):*** In I/O Paging Mode, this field provides the PCI base address on Power PCI to PCI transactions for the address range 'n'.
Bits 10:0	***Reserved***: These bits are reserved, and return 0b'0' when read.

4.1.4.26 O2P Memory Paging Registers (0x90, 0x92, 0x94, 0x96, 0x98, 0x9A, 0x9C, 0x9E)

Description: This set of eight registers provides a paging capability on PCI memory transactions. Additionally, the ***Memory Read Control*** (MRCn) field of each paging register provides control over the type of PCI read command generated on 4-beat reads (Memory Read, Memory Read Line, or Memory Read Multiple) for each range.

31 27	26 18	17 16	15 11	10 2	1 0	Offset
O2P Mem Page 1	Reserved	MRC1	O2P Mem Page 0	Reserved	MRC0	0x0090
O2P Mem Page 3	Reserved	MRC3	O2P Mem Page 2	Reserved	MRC2	0x0094
O2P Mem Page 5	Reserved	MRC5	O2P Mem Page 4	Reserved	MRC4	0x0098
O2P Mem Page 7	Reserved	MRC7	O2P Mem Page 6	Reserved	MRC6	0x009C

Figure 32: PCI Memory Paging Register Layout

Each O2P Mem Paging(n) register is read/write accessible from both the PCI and CPU interfaces. However, bits 26:18 and 10:2 are reserved.

The O2P Memory Paging register reset values each are 0x'0000'.

Bits 17:16 (or 1:0) of each O2P Memory Paging registers are the ***Memory Read Control*** field and provide memory read command control per the table below. Note that Memory Read Line and Memory

Read Multiple commands are only generated on 4-beat CPU reads, and Memory Read Line commands are only generated when the **Cacheline Size** register is 0x'08'.

Memory Read Control (1:0)	**PCI Memory Read Command**
0x	Memory Read
10	Memory Read Line
11	Memory Read Multiple

Bit/Field Definitions:

Bits 31:27	***O2P Mem Page (n+1):*** In Memory Paging Mode, this field provides the PCI base address on PCI transactions for address range 'n+1'.
Bits 26:18	***Reserved***: These bits are reserved, and return 0b'0' when read.
Bits 17:16	***Memory Read Control(n+1):*** These read/write bits provide PCI memory read command control for address range 'n+1'.
Bits 15:11	***O2P Mem Page (n):*** In Memory Paging Mode, this field provides the PCI base address on PCI transactions for address range 'n'.
Bits 10:2	***Reserved***: These bits are reserved, and return 0b'0' when read.
Bits 1:0	***Memory Read Control(n):*** These read/write bits provide PCI memory read command control for address range 'n'.

4.1.4.27 User Defined A Register

Bytes: 4

Description: The User Defined A Register in the Power PCI is used to provide control signals to the remainder of the chip. These signals are controlled from this register to maintain compatibility with the MPC106. The register is read/write from both the PCI and OCB interfaces. Only bit 0 in this register is used in the Power PCI design.

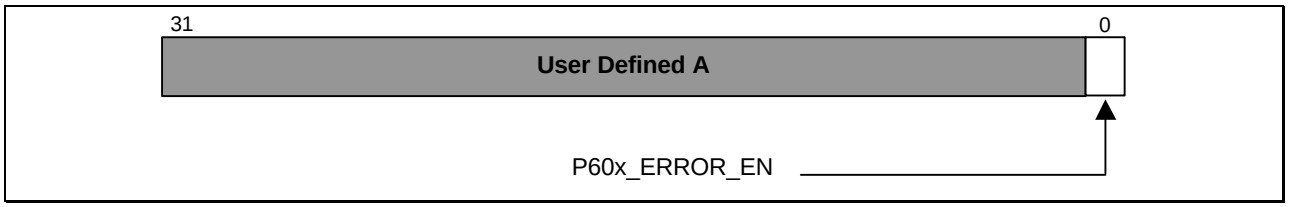


Figure 33: User Defined A Register Layout

4.1.4.28 User Defined B Register

Bytes: 4

Description: The User Defined B Register in the Power PCI is used to provide control signals to the remainder of the chip. These signals are controlled from this register to maintain compatibility with the MPC106. The register is read/write from both the PCI and OCB interfaces. Only bits 5 and 11 in this register are used in the Power PCI design.

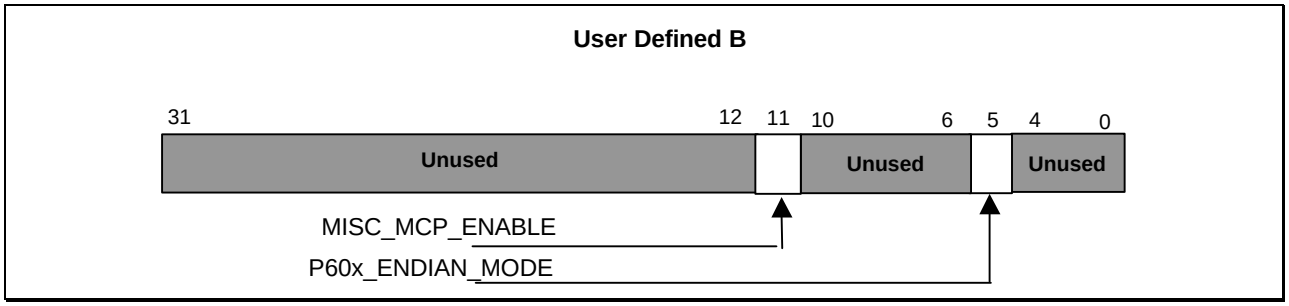


Figure 34: User B Register Layout

4.1.4.29 User Defined C Register

Bytes: 4

Description: The User Defined C Register in the Power PCI is used to provide control signals to the remainder of the chip. These signals are controlled from this register to maintain compatibility with the MPC-106. The register is read/write from both the PCI and OCB interfaces. Only bits 27:26, 19:18, 15, and 3:2 are used in the Power PCI design.

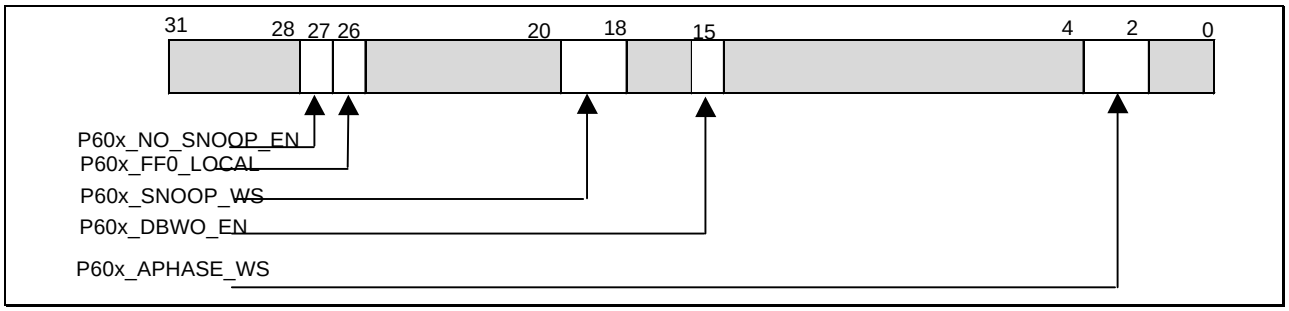


Figure 35: User C Register Layout

4.1.4.30 PCI Bus Error Status Register (0xC6)

Description: This register attempts to provide information on the PCI transaction that was occurring when the Power PCI has detected an error. In the case of multiple errors, this register reflects status of the first error that occurred. If error reporting for a particular error is masked (with a bit in either the **Status 2 Mask** register), detection of that error does not update this register. The PCI Bus Error Status register is read only accessible from both the PCI and CPU interfaces. The PCI Bus Error Status register is reset to 0x'0000' on chip reset. PCI reset does not affect this register. If the Power PCI has detected multiple non-masked errors, this register only corresponds to the initial non-masked errors detected. Reserved bits of this register is unaffected by writes, and returns 0's on reads.

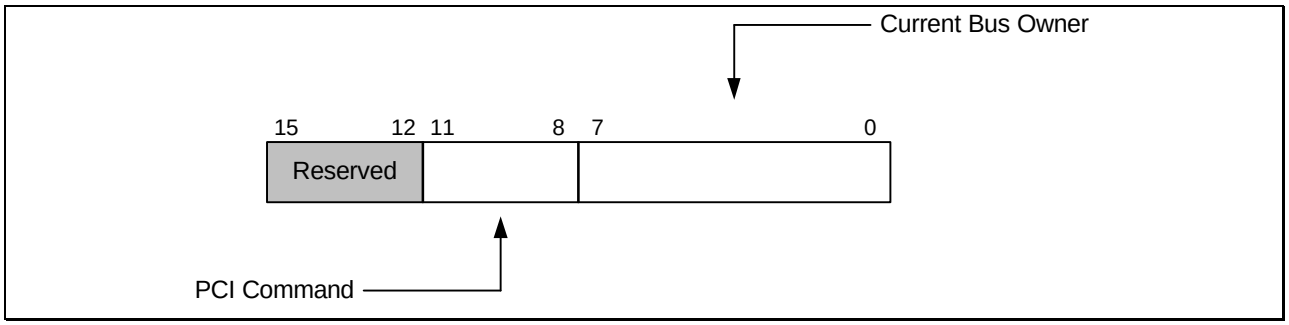


Figure 36: PCI Bus Error Status Register Layout

Bit/Field Definitions:

Bits 15:12	***Reserved***: These bits are reserved, and return 0b'0' when read.
Bits 11:8	***PCI Command:*** The PCI Command field of the PCI Bus Error Status register contains the encoded PCI Command of the PCI initiator transaction that was occurring while the Power PCI detected an error condition. A PCI initiator error is one of: Received Master Abort, Received Target Abort, PCI Initiator PERR# Received, PCI Initiator Data Parity Error Detected, or PCI Initiator Ready Parity Error. This field is updated at the beginning of each PCI initiator transaction when there are no unmasked initiator type errors active, and is frozen when an unmasked initiator error is detected
Bits 7:0	***Current Bus Owner:*** The Current Bus Owner field of the PCI Bus Error Status register contains the **ARB_GNT(7:0)#** value that indicates the PCI device that was the bus master on a PCI transaction when either an Arbiter Timeout, PCI Data Phase Timeout, or Arbitration Latency Timeout error was detected. The owner is indicated by the inverse of the **ARB_GNT(7:0)#** value that was active when the bus was claimed with **FRAME#**. This field is only updated for non-masked timeout errors when all non-masked timeout errors are cleared.

4.1.4.31 PCI Error Address Register (0xC8)

Description: This register provides the PCI address (if possible) of the Power PCI initiator transaction that was occurring when the Power PCI has detected a Power PCI initiator type error. In the case of multiple errors, this register reflects status of the first error that occurred. If error reporting for a particular error is masked (with a bit in the **Status 2 Mask** register), detection of that error does not update this register. The PCI Error Address register is read only accessible from both the PCI and CPU interfaces. The PCI Error Address register is reset to 0x'0000 0000' on chip reset. PCI reset does not affect this register. If the Power PCI has detected multiple non-masked PCI initiator type errors, this register only corresponds to the initial non-masked PCI initiator type error detected. PCI initiator type errors set bits 4, 5, 6, 7, 28, and 29 of the **Status 2** register.

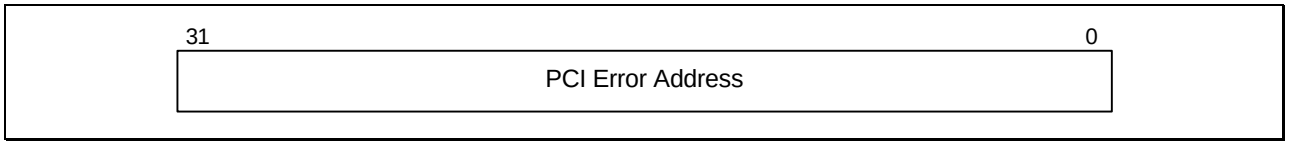


Figure 37: PCI Error Address Register Layout

Bit/Field Definitions:

Bits 31:0	***PCI Error Address:*** The **PCI Error Address** register contains the starting address of the PCI transaction that was occurring when the core detected a PCI initiator error. A PCI initiator error is one of: Received Master Abort, Received Target Abort, PCI Initiator PERR# Received, PCI Initiator Data Parity Error Detected, or PCI Initiator Ready Parity Error. This field is updated at the beginning of each PCI Core initiator transaction when there are no unmasked initiator type errors active, and is frozen when an unmasked initiator error is detected.

4.2 Memory Interface

The Power PCI is capable of interfacing to both local memory and to SUROM for requests from any chip function or interface. The Power PCI accepts both memory read and write commands via its Memory Request Interface (MRI). The request address associated with the command is decoded to determine what type of memory is being accessed: SDRAM, SRAM or ROM (ROM refers to any type of read-only memory, i.e., ROM, PROM, EEPROM, Flash ROM, etc.). The Power PCI then generates the necessary control signals to the external memory interface to execute the memory request. In the case of read

transactions, the read data is queued into a buffer, which is available via the MRI, to complete the data transfer.

The Power PCI is highly configurable via a set of internal registers. These registers are accessed via the Memory Configuration Interface (MCI).

The Power PCI supports up to 8 banks of memory. The 8 banks can be any combination of the three types of memory, although bank 0 will default to be Start-Up ROM (SUROM). Each bank of SDRAM can contain 1 to 512 MBytes of memory. Each bank of SRAM or ROM can contain 64 KB to 32 MBytes of memory. The Power PCI uses a 32-bit address bus; therefore the Power PCI can address the banks anywhere within a 4-gigabyte (GByte) memory address space. (The need to use some memory address space to address internal registers and the limit of number of banks and memory per bank will prevent the Power PCI from being able to access a full 4 GByte of memory however). Figure 38 below shows an example of a memory system using the Power PCI. It contains 4 banks of memory (1 ROM, 1 SRAM and 2 SDRAM). Take note of the different sizes of the banks and the address lines used for each.

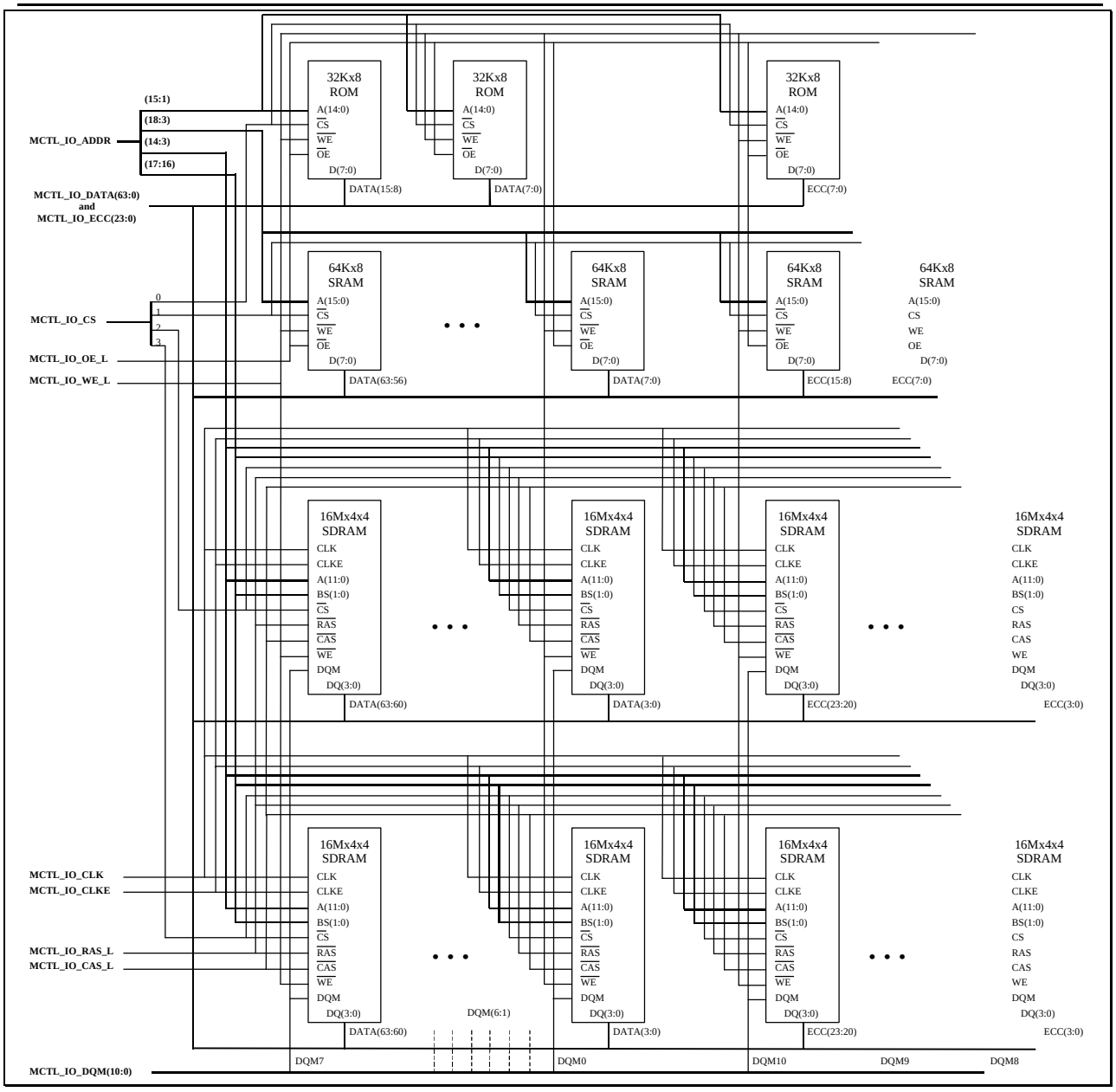


Figure 38: Example Memory System with 1 bank of ROM, 1 bank of SRAM and 2 banks of SDRAM

The Power PCI also supports several fault tolerant capabilities. Memory sparing and error detection and correction are provided to tolerate errors in external memory.

4.2.1 General Memory Support Functions

This section describes the operation and features of the Memory Controller portion of the Power PCI. Its main function is to transfer data to/from external memory as commanded by the Memory Request Interface (MRI). The Power PCI is highly configurable via its internal registers described in Section 4.2.2. The internal registers are accessed via the Memory Configuration Interface (MCI).

4.2.2 SDRAM Support Features

4.2.2.1 SDRAM Addressing

The Power PCI supports banks of SDRAM that contain 8 to 512 MBytes of memory. The RAD750 board is populated with 128 Mbytes. The 13 Row/Column multiplexed address signals, **ADDR(15:3)**, two bank selects, **ADDR(17:16)**, and four addressing modes provide addressability to SDRAM parts of 1 M to 64 M in depth.

4.2.2.2 SDRAM Page Mode Accesses

The Power PCI takes advantage of an open bank when accesses to the same SDRAM row are received consecutively. The Power PCI does not take advantage of bank misses by activating the new row prior to precharging a previously accessed row; only one row will be open at a time. This action applies to both bank misses and sub-bank misses. A bank miss refers to accesses to another SDRAM part; a sub-bank miss refers to a single SDRAM part's bank addresses. Note that page misses and sub-bank misses are determined by the configuration of row and bank select bits, not the geometry of the actual SDRAM part.

4.2.2.3 SDRAM Command Set

The Power PCI performs all accesses to SDRAM using JEDEC Standard SDRAM interface commands. The SDRAM samples the address, control and data inputs on the rising edge of the SDRAM clock, **CLK**. Likewise, the SDRAM data output is valid on the rising edge of the SDRAM clock. The command to execute is determined primarily by the values of the control signals **RAS_L**, **CAS_L** and **WE_L**. However, the signals **CS_L**, **CLKE**, and **ADDR** may also affect the command to execute. Table 6 shows the list of SDRAM commands supported by the Power PCI.

Table 6 - Supported SDRAM Command Set

Command	CS	Previous CLKE	Current CLKE	RAS	CAS	WE	ADDR (17:16)	ADDR (13)	ADDR (15:14 12:3)
Device Deselect	H	X	X	X	X	X	X	X	X
No Operation	L	H	X	H	H	H	X	X	X
Mode Register Set	L	H	X	L	L	L	Mode Register Set Value		
Auto Refresh	L	H	H	L	L	H	X	X	X
Self Refresh Entry	L	H	L	L	L	H	X	X	X
Self Refresh Exit	H	L	H	X	X	X	X	X	X
Precharge one	L	H	X	L	H	L	BS	L	X
Precharge all	L	H	X	L	H	L	X	H	X
Activate	L	H	X	L	H	H	BS	Row	
Write	L	H	X	H	L	L	BS	L	Column
Write with AP	L	H	X	H	L	L	BS	H	Column
Read	L	H	X	H	L	H	BS	L	Column
Read with AP	L	H	X	H	L	H	BS	H	Column

Notes:

1. The Mode Register Set command shows the ***Mode Register Set Value*** extending across all of the address bits. See Section 4.2.11.2.1 for the format of the ***Mode Register Set Value***. The 14-bit value is written to **ADDR(16:3)**.
2. Although not shown in the table, the **DQM** bus is also required for proper communication with the SDRAM parts. The DQM bus is used to mask unwanted writes or reads to memory. The Power PCI does not support byte writes to memory. All byte writes to memory are reformatted internally by the

Power PCI to be Read-Modify-Writes of a full data line (typically this will be a double word). Therefore, the DQM bus is always either all 0b'1's or all 0b'0's. The Power PCI supports a DQM bus to avoid the need for external buffering of the DQM signals.

3. The types of read and write commands used are determined by the ***Auto-Precharge Enable*** bit (see Section 4.2.11.2.1). If the enable bit is set, all read and write commands are Read with Auto-Precharge and Write with Auto-Precharge commands. Otherwise, the Read and Write commands will be used and the Power PCI commands the precharging of SDRAM.

4.2.2.4 SDRAM Initialization

After a system reset, the initialization software and the Power PCI hardware must coordinate the initialization of the SDRAM memory interface. The SDRAM must be initialized prior to being accessed. The list below gives the steps, in the proper sequence and with the correct timing, needed to initialize SDRAM.

Steps required of the Initialization Software:

1. After the reset is deactivated, pause for the amount of time indicated in the SDRAM specification (typically 100-200 μs).
2. Initialize SDRAM configuration registers (this may be done during the pause period of step 1)
   Set the *SDRAM Controller Enable* bit in the **SDRAM Configuration Register (0xE0).**

Steps that are performed by the Power PCI:

3. Issue a Precharge Command to all banks
4. Wait for the Minimum Precharge to Activate Time defined in SDRAM Timing Parameters Register 2 (0xEC).
5. Issue 8 Auto-Refresh Commands (each followed by the Minimum Refresh Command to Activate Time defined in **SDRAM Timing Parameters Register 2 (0xEC)).** If there is more than one enabled SDRAM bank, the Power PCI issues separate Auto-Refresh Commands to each bank if the Refresh Stagger parameter in the **SDRAM Timing Parameters Register 1 (0xE8)** is non-zero.
6. Issue a Mode Register Write Command. If there is more than one enabled SDRAM bank, the Power PCI issues separate Mode Register Write Commands to each bank if the Refresh Stagger parameter in the **SDRAM Timing Parameters Register 1 (0xE8)** is non-zero.
7. Wait for 3 clock cycles
8. SDRAM available for access

4.2.2.5 SDRAM Timing Parameters

System software is responsible for programming the SDRAM Timing Registers, **SDRAM Timing Parameters Register 1 (0xE8)** and **SDRAM Timing Parameters Register 2 (0xEC)**, for optimal system performance. The values written into these registers' fields depend on the timing characteristics of the specific SDRAM devices used in the system.

4.2.2.6 SDRAM Refresh

The Power PCI supports two types of refresh, Auto-Refresh and Self-Refresh. Auto-Refresh is the standard refresh that must be interspersed between SDRAM accesses during normal operation. Self-Refresh is an operation designed to save power when the system enters a power saving mode.

An Auto-Refresh command must be issued to the SDRAM parts when the Power PCI's internal refresh timer expires. At that time, the Power PCI completes the current memory operation and then refreshes all SDRAM banks. The Power PCI only refreshes banks that are designated as SDRAM banks. To minimize instantaneous current draw, the Power PCI can stagger the Auto-Refresh commands by the ***Refresh Stagger*** specified in the **SDRAM Timing Parameters Register 1 (0xE8)**.

The refresh interval is programmable via the ***Refresh Interval*** field in the **SDRAM Timing Parameters Register 1 (0xE8)**. Immediately following reset, the Power PCI loads its internal refresh timer with the value in the ***Refresh Interval*** field. When the timer expires, the Power PCI queues an Auto-Refresh operation and automatically reloads and restarts the internal timer. As mentioned above, the Auto-

Refresh command will be executed immediately following any current memory operation. If the refresh timer expires and an Auto-Refresh operation from the previous timer has not completed, a ***Memory Refresh Overflow*** error is logged in the **Status Register.**

Note: Currently available SDRAM parts do not allow the Power PCI to provide the row address to refresh row address. The SDRAM parts themselves must contain a counter register that sequences through the rows. If an SEU were to upset the SDRAM's counter register, some rows would not get refreshed within the required refresh time, thereby jeopardizing the integrity of the SDRAM's contents.

A Self-Refresh Entry command is executed when the **PWRM_MODE** signals specify Sleep Mode or when the **PWRM_MODE** signals specify Doze or Nap mode and no bank of memory has Active Scrubbing Enabled during Power Management Mode. As with Auto-Refresh, only those banks that are enabled and designated as SDRAM banks receives the Self-Refresh Entry command. Those banks will remain in the Self-Refresh State until the Power PCI receives a memory request from the Memory Request Interface. Upon receiving that request, the Power PCI issues a Self-Refresh Exit command and wait for the ***Minimum Self-Refresh Exit to Activate Time*** designated in the **SDRAM Timing Parameters Register 2 (0xEC)** before performing the memory access requested.

4.2.2.6.1 Refresh Timer

The Power PCI maintains a 12-bit countdown timer to periodically issue an auto-refresh command to the SDRAM memory banks. This counter decrements one count every system clock cycle. When the counter reaches zero, an SDRAM refresh operation is queued. At the same time, the counter is reloaded with the value in the ***Refresh Interval*** field of the **SDRAM Timing Parameters Register 1 (0xE8)**

4.2.2.7 SDRAM Commanded Self-Refresh

In addition to the entry into Self-Refresh state due to a change in the **PWRM_MODE** signals mentioned above, the SDRAMs may also be placed in Self-Refresh by executing an Enter SDRAM Self-Refresh Command (0x08). When this command is received, the Power PCI completes any pending memory operation before issuing the Self-Refresh Entry command to the SDRAMs. As described above, only those memory banks that are enabled and designated as SDRAM receive the Self-Refresh Entry command. At the same time, the Power PCI halts any further scrubbing of SDRAM addresses and the generation of Auto-Refresh commands. (Note that any SDRAM scrub that was in the pipeline when the SDRAM Self-Refresh command was received may generate an ***Address out of Bounds*** error.) Any commands that were intended for SDRAM will fail with an ***Address out of Bounds*** error. After the Self-Refresh Entry command is sent to the SDRAMs, the Power PCI stops the clocks to the SDRAMs if the ***Stop Clocks during Self-Refresh Enable*** bit is set in the **SDRAM Configuration Register (0xE0)**. If the bit is set, the Power PCI waits the ***Minimum Clock Disable to Clock Halted Time*** specified in the **SDRAM Timing Parameters Register 2 (0xEC)** before stopping the clock. When the command has completed, the ***SDRAM Self-Refresh Command Complete*** bit will be set in the **Status Register (0xC0)**.

To exit the Self-Refresh state, an Exit SDRAM Self-Refresh Command (0x0C) must be executed. If the ***Stop Clocks during Self-Refresh Enable*** bit is set in the **SDRAM Configuration Register (0xE0)**, the Power PCI will restart the SDRAM clock and wait for the time specified by the Minimum Clock Restart to Clock Enable Time in the **SDRAM Timing Parameters Register 2 (0xEC)**. Then, regardless of the value of the ***Stop Clocks during Self-Refresh Enable*** bit, the Power PCI will issue the Self-Refresh Exit command to the SDRAM and wait the ***Minimum Self-Refresh Exit to Activate Time*** specified in the **SDRAM Timing Parameters Register 2 (0xEC)** before accepting a memory request. When the command has completed, the ***SDRAM Self-Refresh Command Complete*** bit will be set in the **Status Register (0xC0)**.

If both the **Enter SDRAM Self-Refresh Command (0x08)** and **Exit SDRAM Self-Refresh Command (0x0C)** are received simultaneously (which is possible given the 64-bit MCI data bus), the ***Invalid Command on Memory Configuration Interface*** bit will be set in the **Status Register (0xC0)**. Neither of these commands can be received when the Power PCI not in the Normal Power Management Mode.

4.2.2.8 SDRAM Refresh Prior to Initialization

In an attempt to maintain SDRAM contents during a soft reset, the Power PCI begins refreshing SDRAM banks immediately following any reset. Prior to SDRAM initialization (which occurs when the ***SDRAM Controller Enable*** bit is set in the **SDRAM Configuration Register (0xE0)**), the Power PCI auto-refreshes any bank for which the corresponding **IO_MCTL_SDRAM_BANK** signal and **Memory Bank Enable Register (0x68)** bit are set. Thus, software may disable the refresh of any SDRAM bank prior to initialization by disabling the bank in the **Memory Bank Enable Register (0x68)**. Note that any bank that has its IO_MCTL_SDRAM_BANK signal set will have its enable bit set in the **Memory Bank Enable Register (0x68)** after reset. The auto-refresh of SDRAM prior to initialization may be superseded by an entry into SDRAM self-refresh (either by an **Enter SDRAM Self-Refresh Command (0x08)** or a Power Management Mode change).

Any bank that is specified to be an SDRAM bank by its corresponding **IO_MCTL_SDRAM_BANK** input signal can not be specified as another type of memory in the **Memory Type Register (0x60)**. However, any bank that is not specified as an SDRAM bank by its **IO_MCTL_SDRAM_BANK** input signal may later be specified as SDRAM in the **Memory Type Register (0x60).** Banks specified as SDRAM in the **Memory Type Register (0x60)** that do not have their corresponding IO_MCTL_SDRAM_BANK bit set are not refreshed prior to SDRAM initialization.

As with typical SDRAM auto-refreshes, the auto-refreshes prior to SDRAM initialization uses the ***Refresh Stagger*** and ***Refresh Interval*** parameters from the **SDRAM Timing Parameters Register 1 (0xE8)** to determine the timing of the refreshes. Since the default values of these parameters may adversely affect system performance, software may want to write more appropriate values into the register shortly after reset.

4.2.2.9 SDRAM Timing Diagrams

In the figures that follow, many conventions and abbreviations are used to enhance readability. All of the signal names have their “MCTL_IO_” prefix dropped to save space. The “Command” signal in the figures is a mnemonic form of the values that exist on the RAS_L, CAS_L and WE_L signals. The names of the fields of the SDRAM Timing Parameter Registers are also abbreviated to save space in the figures. Table 7 below shows the command mnemonics and parameter abbreviations used in the figures.

Note that while all signals change on the falling edge of the clock, the SDRAM will latch the signals on the rising edge of the clock to provide a half-cycle cushion on both sides of the sample point. Thus, timing parameters are shown from the rising edge of the clock.

Table 7 - Command Mnemonic and Parameter Abbreviations for SDRAM Timing Figures

Mnemonic	Command
ACT	Row Activate
CBR	Auto-Refresh
MRS	Mode Register Set
PRE	Precharge one bank
PALL	Precharge all banks
RD	Read
RDAP	Read w/Auto-Precharge
SRE	Self-Refresh Enty
WR	Write
WRAP	Write w/Auto-Precharge
<none>	No Operation
???	Any Command

Abbreviation	Parameter
ACT_TO_CMD	Min Activate to Command Time
ACT_TO_PRE	Min Activate to Precharge Time
CL	CAS Latency
DIS_TO_HLT	Min Clock Disable to Clock Halted Time
PRE_TO_ACT	Min Precharge to Activate Time
REF_TO_ACT	Min Refresh Command to Activate Time
REF_STAGGER	Refresh Stagger
RST_TO_ENA	Min Clock Restart to Clock Enable Time
SRX_TO_ACT	Min Self-Refresh Exit to Activate Time
WR_TO_PRE	Min Write Data to Precharge Time

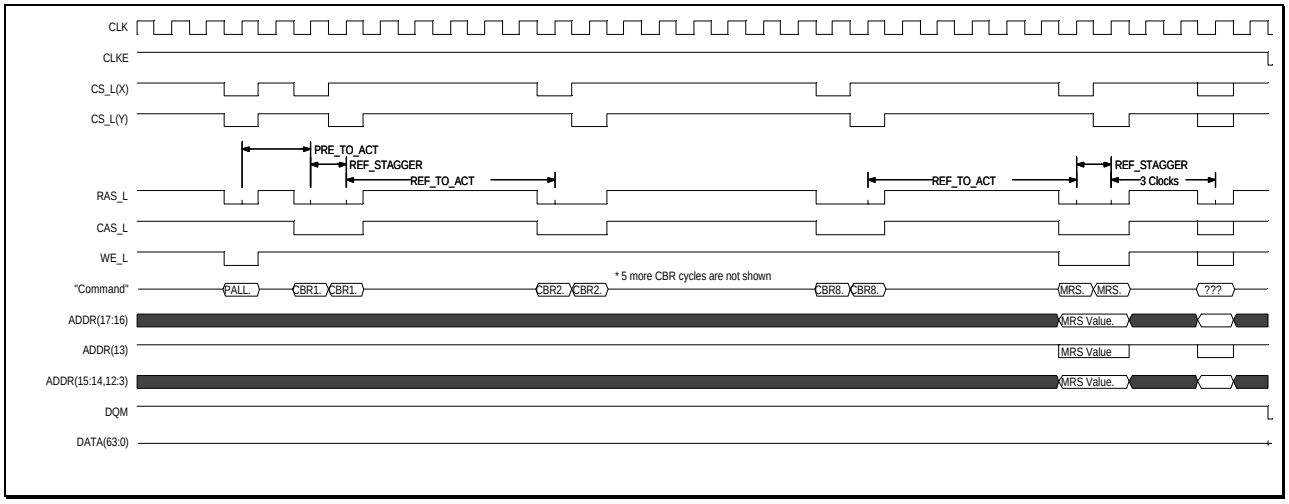


Figure 39: SDRAM Initialization Sequence

Note: The REF_STAGGER parameter is used to separate the Mode Register Set commands as well as the refresh commands.

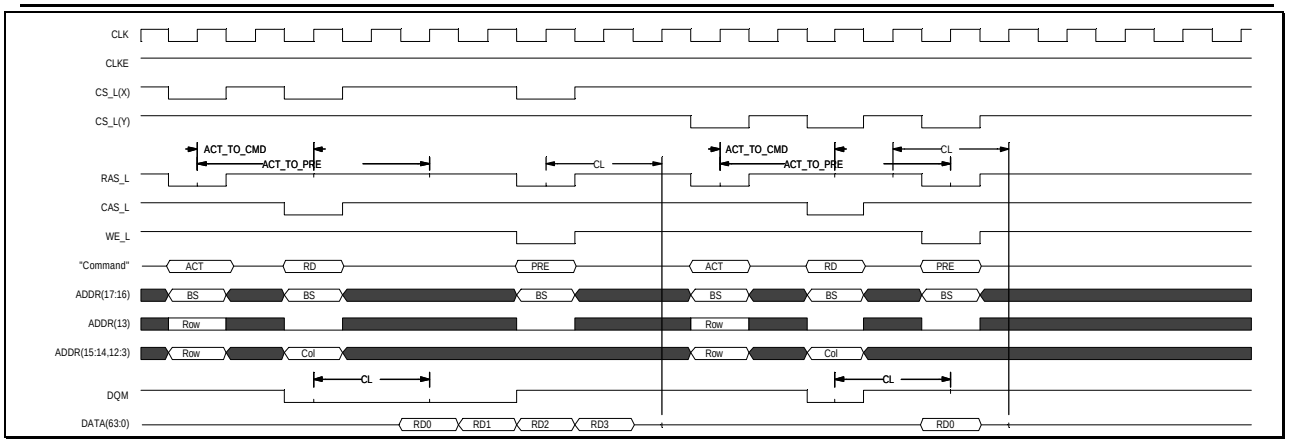


Figure 40: SDRAM Read (without Auto-Precharge) Timing Diagram for a Burst Read and a Single Read

Note 1: For a read to complete successfully two constraints need to be met. *The Minimum Activate to Precharge Time* **must expire before the precharge can be issued and** *CAS Latency* **cycles must extend one cycle past the last data word required. The cycle on which the precharge command is issued is different for the two reads. For the burst read, the timing of the precharge command is determined by the** *CAS Latency* **parameter. For the single read, the timing of the precharge command is determined by the** *Minimum Activate to Precharge Time* **parameter.**

Note 2: The space between the burst read and single read was inserted to enhance readability. The single read could have started 2 cycles earlier for better throughput.

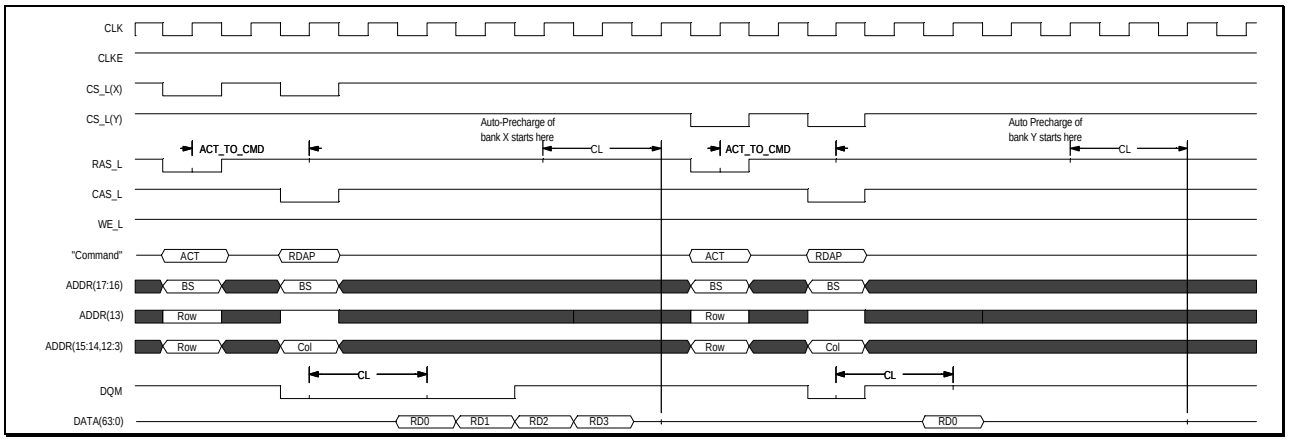


Figure 41: SDRAM Read (with Auto Precharge) Timing Diagram for a Burst Read and a Single Read

Note: The timing of the Auto-Precharge is determined by the value of the Burst Length **and** CAS Latency **parameters.**

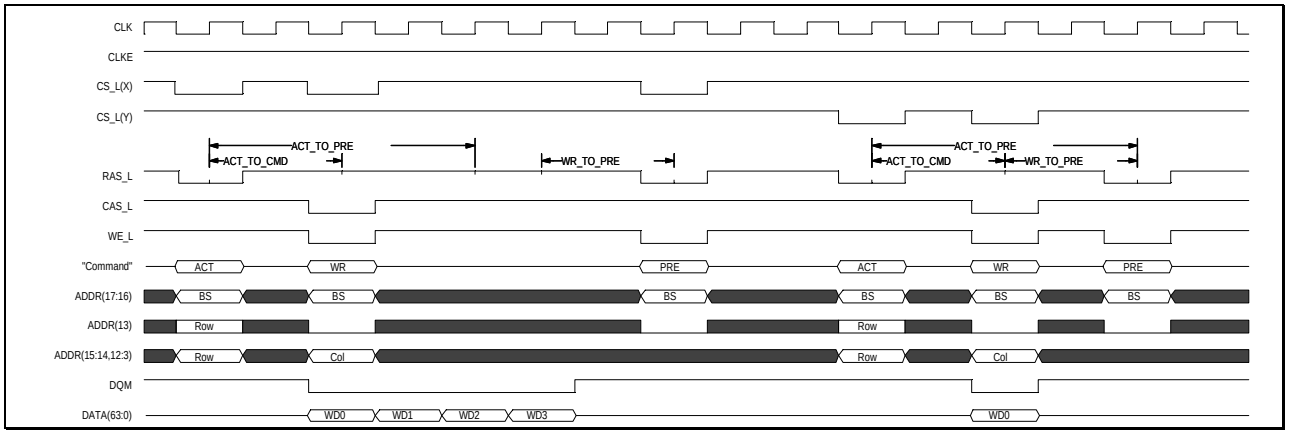


Figure 42: SDRAM Write (without Auto-Precharge) Timing Diagram for a Burst Write and Single Write

Note: For a write to complete successfully the precharge can not be issued until the Minimum Write Data to Precharge Time **has expired following the last write data transmission. By that time the** Minimum Activate to Precharge Time **will typically have expired, but if not then the precharge must be delayed until the** Minimum Activate to Precharge Time **has expired.**

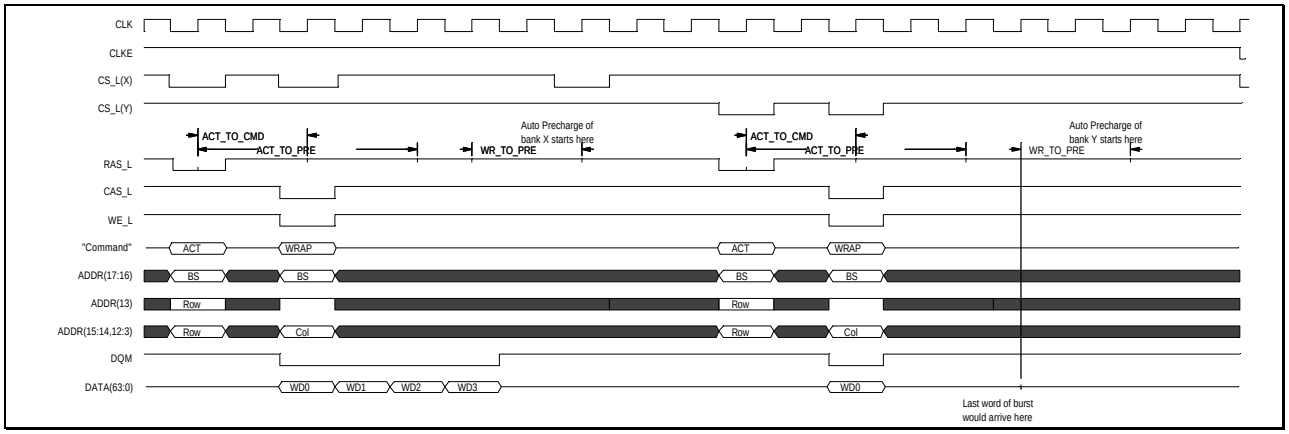


Figure 43: SDRAM Write (with Auto-Precharge) Timing Diagram for a Burst Write and Single Write

4.2.3 ROM Features

The Power PCI supports connection to ROM, including components with ROM-like interfaces, such as PROM, EEPROM and Flash ROM. The Power PCI only supports one set of timing parameters for ROM, thus all ROM-like parts in a system must be the same or conform to one set of timing parameters (see Section 4.2.11.4.1).

The Power PCI supports banks of ROM that contain 64 KB to 32 MByte of memory. The RAD750 board has 256K populated. The lower limit is determined by the bank size granularity. The number of external address bits determines the upper limit.

The Address Bus, **ADDR(24:0)**, contains the byte address to access. With the exception of a 16-bit wide data path for bank 0 (see Section 4.2.3.1 for more information), the 3 LSBs of the address bus are not used. Typically, **ADDR(3)** should be wired to the LSB of the memory parts.

The Power PCI supports burst or non-burst mode accesses to ROM. In burst mode, the first access in a sequence is slower than all subsequent accesses. In non-burst mode, all accesses are treated the same. See Figure 44 below for the protocol for non-burst reads and writes. See Figure 45 below for the protocol for burst reads and writes. The delays ROM_RD_FW, ROM_RD_NW, ROM_WR_FW, ROM_WR_NW,

and HA in the figures correspond to the ***First Wait for Reads***, ***Next Wait for Reads***, ***First Wait for Writes***, ***Next Wait for Writes,*** and ***Hold Address on Write Cycles*** parameters in the **ROM Timing Parameters Register (0xF0)**, respectively. The figures show delays of 3 clock cycles for the First Wait parameters, 2 clock cycles for the Next Wait parameters and 1 cycle for the Hold Address Cycle parameter. Note that for writes, the Power PCI will always have a 1 cycle set up time, SA to avoid unintended writes due to system clock skew. Note also that the set up time for the address, SA, is included in the Wait for Write parameter's delay but the hold time is not. Therefore, the total delay for a write operation will be HA cycles more than the delay specified by the Wait for Write parameters. (The ***Hold Address on Write Cycles*** parameter is intended to provide spacing between writes to ROM in a Page Mode type write to ROM. Software must provide the proper spacing between writes of pages to ROM.)

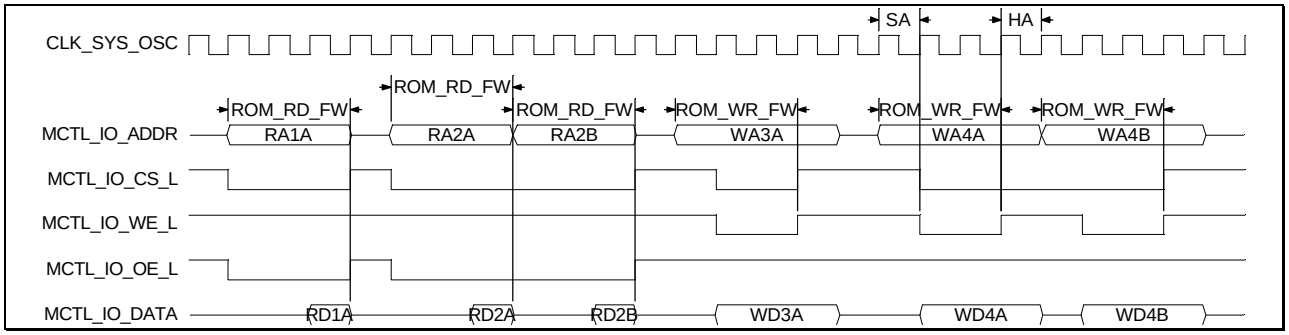


Figure 44: ROM Timing Diagram, Non-Burst Reads and Writes

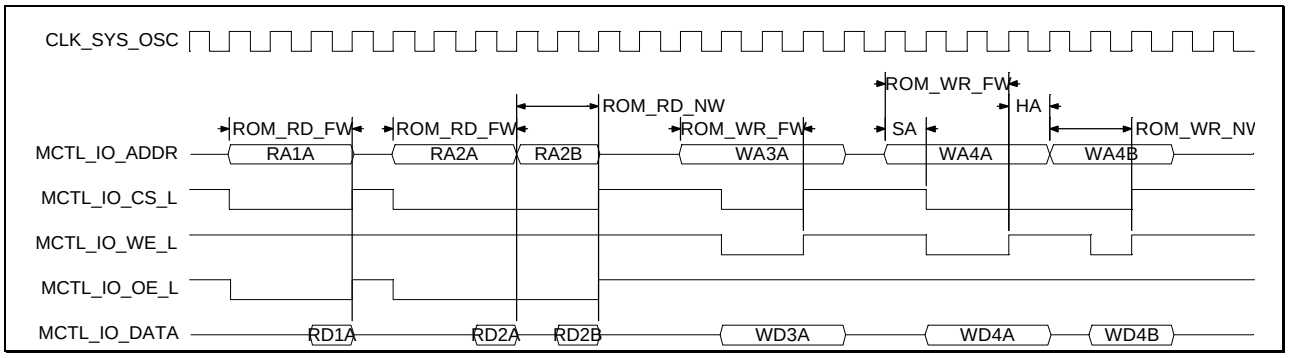


Figure 45: ROM Timing Diagram, Burst Reads and Writes

Note that the **MCTL_IO_OE_L** signal in the ROM timing diagrams above does not go active on the first cycle of a ***First Wait for Read*** access. If a bank's ***First Wait for Read*** parameter is 1, the Power PCI will insert a cycle on a read of that bank if the read follows a write or a read of a different ROM or SRAM bank. This extra cycle will avoid any issues created by two devices driving the data bus simultaneously due to control line signal skew on the board.

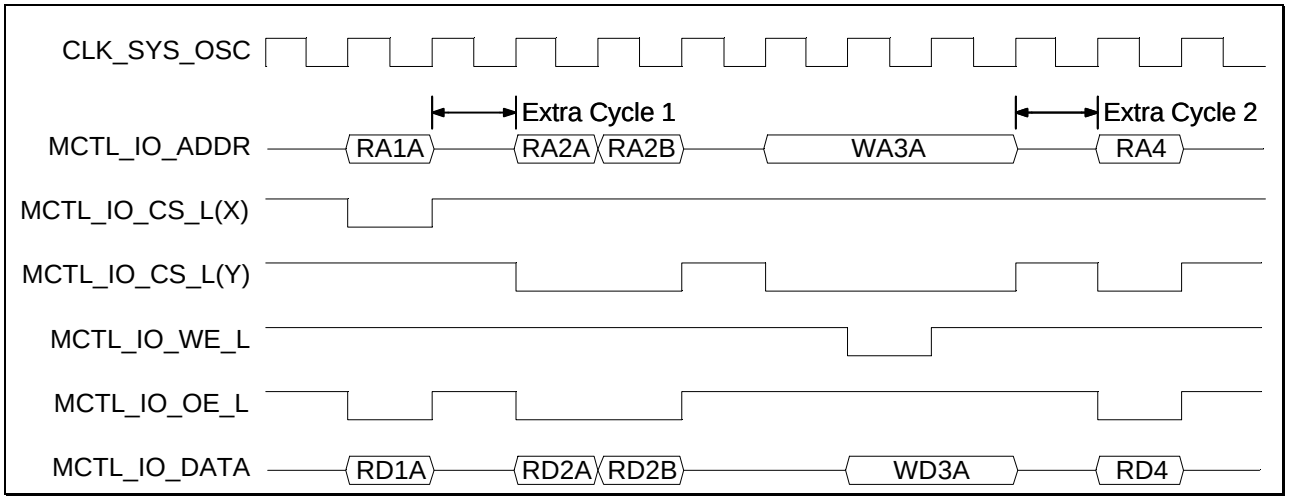


Figure 46: Extra Cycles Inserted when First Read Wait Parameter is 1

Extra Cycle 1 is inserted because previous read was from a different bank

Extra Cycle 2 is inserted because previous operation was a write

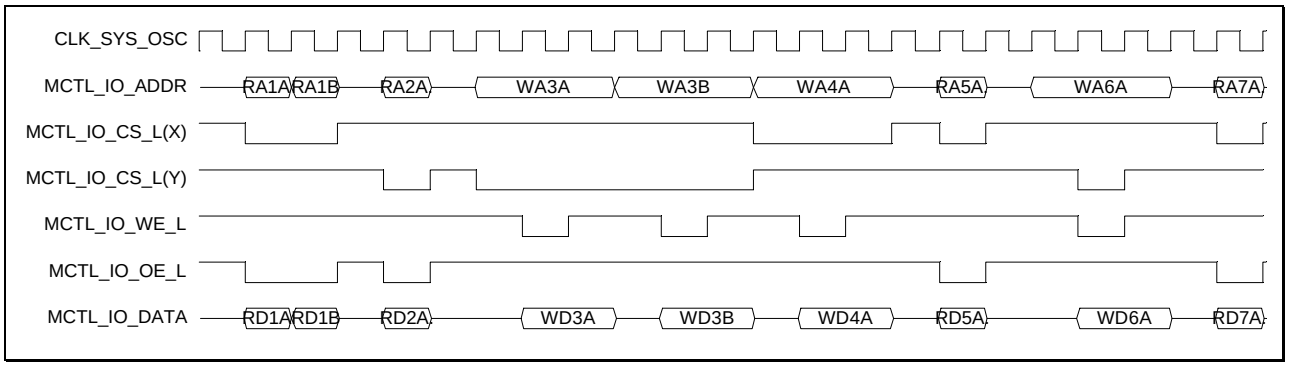


Figure 47: Read and Write Operations to Two Banks (ROM or SRAM) with Minimum Timing

4.2.3.1 SUROM Support Features

As mentioned above, bank 0 defaults to ROM. It is different than the other banks in that some of its configuration parameters (such as Starting Address, Memory Type and Correction Type) are pre-defined at reset. See the individual register descriptions in Section 4.2.11.4 for more information.

The Power PCI supports a 16-bit or 64-bit wide data path to memory for bank 0. If bank 0 is configured as ROM (the default), the configuration signal, **SU_WIDTH**, determines the width of the data path. If software changes bank 0's memory type to something other than ROM, then the data path for bank 0 is 64-bits wide, regardless of the value of the **SU_WIDTH** signal. If a 16-bit wide data path is selected, **ADDR(2:1)** contains the half-word of the double-word address to access. For this RAD750 board, **SU_WIDTH** is tied to 16-bit wide data path. Thus for a 16-bit wide data path, **ADDR(1)** should be wired to the LSB of the memory part. The 16 bits of data should be wired to the 16 LSBs of the data bus, **DATA(15:0)**.

The width of bank 0 is transparent to the Memory Request Interface (MRI). If a 16-bit wide data path is selected, accesses received from the MRI are subdivided into 16-bit transfers to memory. The Power PCI aligns and buffers the data internally into 64-bit groups before forwarding the data to the MRI. All data transfers on the MRI are 64-bits wide.

4.2.3.2 Flash ROM Features

As mentioned above, Power PCI supports connection to Flash ROM. Software, however, is responsible for putting Flash ROM into the proper command mode, following the programming algorithms, etc. The system design is responsible for providing the proper programming voltage to the Flash ROM parts as well as a mechanism for turning off the programming voltage supply (if desired).

4.2.4 SRAM Features

The Power PCI supports connection to SRAM. The memory interface for SRAM is the same as the interface for ROM, except that they use different sets of timing parameters. As with ROM, there is only one set of timing parameters for SRAM, thus all banks of SRAM must conform to the same timing specifications (see Section 4.2.11.5.1).

The Power PCI supports banks of SRAM that contain 64 KB to 32 MByte of memory. SRAM must use a 64-bit wide data bus. The 3 LSBs of the Address Bus are not used, **MCTL_IO_ADDR(3)** should be wired to the LSB of the SRAM memory parts.

The Power PCI supports burst or non-burst mode accesses to SRAM. In burst mode, the first access in a sequence is slower than all subsequent accesses. In non-burst mode, all accesses are treated the same. See Figure 48 below for the protocol for non-burst reads and writes. See Figure 49 below for the protocol for burst reads and writes. The delays SRAM_RD_FW, SRAM_RD_NW, SRAM_WR_FW and SRAM_WR_NW in the figures correspond to the ***First Wait for Reads***, ***Next Wait for Reads***, ***First Wait for Writes***, and ***Next Wait for Writes*** parameters in the **SRAM Timing Parameters Register (0xF4)**, respectively. The figures show delays of 3 clock cycles for the First Wait parameters and 2 clock cycles for the Next Wait parameters. Note that for writes, the Power PCI will always have a 1 cycle set up time, SA, and a 1 cycle hold time, HA, for the address to avoid unintended writes due to system clock skew. Note also that the set up time for the address, SA, is included in the Wait for Write parameter's delay but the hold time is not. Therefore, the total delay for a write operation will be one cycle more than the delay specified by the Wait for Write parameters.

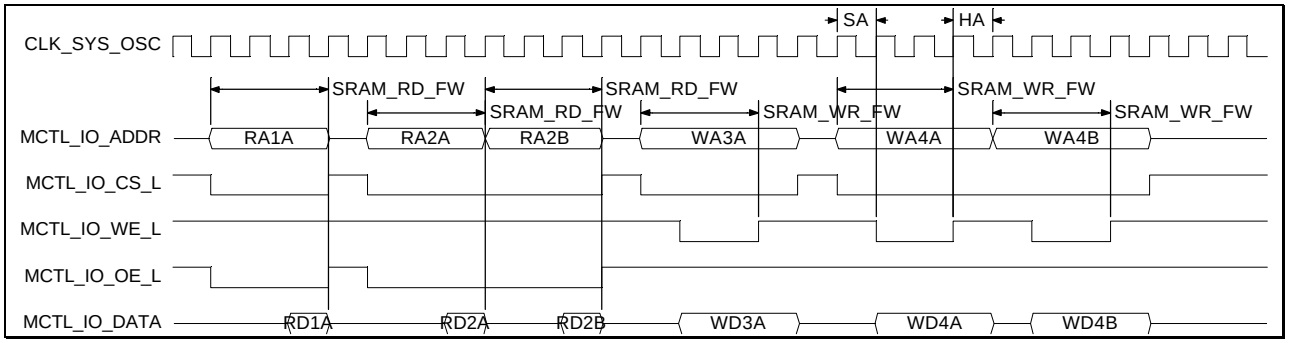


Figure 48: SRAM Timing Diagram, Non-Burst Reads and Writes

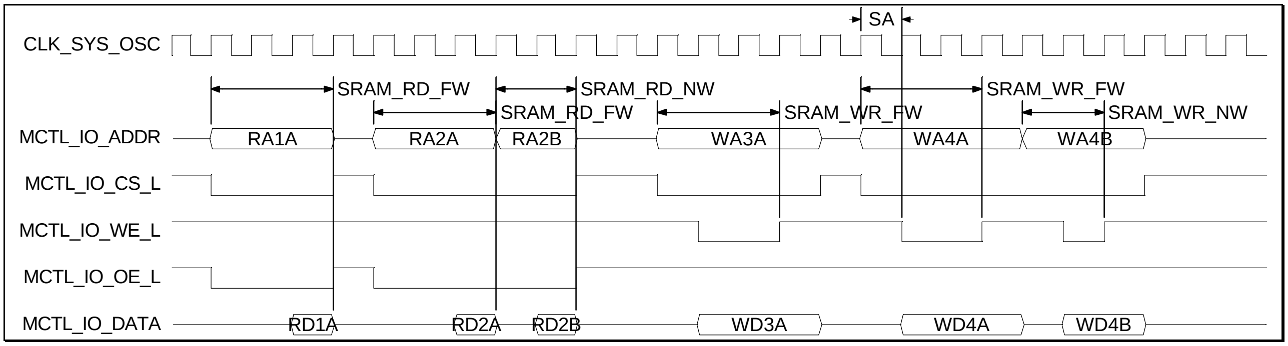


Figure 49: SRAM Timing Diagram, Burst Reads and Writes

4.2.5 Error Monitoring, Control and Status

The Power PCI supports three types of error correction and detection: odd-parity, Single-Error Correct Double-Error Detect (SECDED) and Nibble correct. The odd-parity and SECDED error correction types require 8 ECC bits; the Nibble correction type requires 16 ECC bits. The remaining bits of the 24-bit ECC/Sparing bus are available for sparing. Error correction and detection types are selectable on a bank-by-bank basis via the **Correction Type Register (0x70).** Note that Bank 0 / SUROM only utilizes the SECDED scheme.

4.2.5.1 Detection and Correction Options

4.2.5.1.1 Odd Parity

The simplest form of error detection logic is odd parity checking. For banks that use odd parity for error detection, the Power PCI stores 8 odd parity bits for writes to memory. The Power PCI checks that the 8 parity bits read from memory on memory read operations are correct. If a parity error is detected on a memory read, the Power PCI invalidates the data returned and sets the ***Uncorrectable Data Error*** bit in the **Status Register (0xC0)** and creates an entry in the error log. Note that this type of error coverage does not detect addressing errors and it cannot correct data bit errors. Table 8 below shows the data bits which comprise the ECC bit generation equations for the Odd Parity error detection type. Note that ECC bits 15:8 are not used for this correction type, they are just shown in the table for completeness.

Table 8 - Odd Parity Check Matrix

		Data Bits																																											
AP		63	62	61	60	59	58	57	56		55	54	53	52	51	50	49	48		47	46	45	44	43	42	41	40		39	38	37	36	35	34	33	32		31	30	29	28	27	26	25	24
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																																													

Data Bits																										ECC Bits																	
23	22	21	20	19	18	17	16		15	14	13	12	11	10	09	08		07	06	05	04	03	02	01	00		15	14	13	12	11	10	09	08		07	06	05	04	03	02	01	00

Data Bits																								ECC Bits															
																																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

Note: For each row, the count of designated bits (those marked with an 'x') that are logic 0b'1' should be odd.

Note: The column labeled "AP" signifies address parity (odd parity covering the 32-bit address).

Note: The inclusion of the address parity bit in the ECC bits' calculation is determined by the *Memory Bank n Address Parity Mask* bit in the Correction Type Register (0x70).

4.2.5.1.2 SECDED

The Power PCI supports a Single Error Correction, Double Error Detection (SECDED) code. For banks that use the SECDED error correction type, the Power PCI stores 8 bits of ECC data on write operations to memory. The Power PCI checks that the data returned from memory is correct. If a correctable error is found, the Power PCI corrects the data. If scrubbing is enabled via the **Memory Scrubbing Mode Register (0x78)**, the Power PCI also scrubs the correctable data in memory. If an uncorrectable error is found, the Power PCI invalidates the data returned. If either a correctable or uncorrectable error is found, the Power PCI sets the corresponding bit in the **Status Register (0xC0)** and creates an error log entry. Note that the SECDED code is only guaranteed to correct single-bit errors and detect double-bit errors. An error in 3 or more bits may be detected, but it is not guaranteed. Note that this type of error coverage does detect a single bit addressing error, but those can not be corrected. Table 9 below shows the data bits which comprise the ECC bit generation equations for the SECDED code. Note that ECC bits 15:8 are not used for this correction type, they are just shown in the table for completeness.

Table 9 - Parity check matrix for odd-weight-column (73,65) SECDED code.

	Data Bits																																							
AP	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24
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	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		x		x	x			x		x		x	x								
				x		x	x	x				x		x	x	x				x		x	x	x				x		x	x	x								

Data Bits																								ECC Bits															
23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
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		
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

4.2.5.1.3 Nibble Correct

Note: The MPC-106 does not support nibble correct mode.

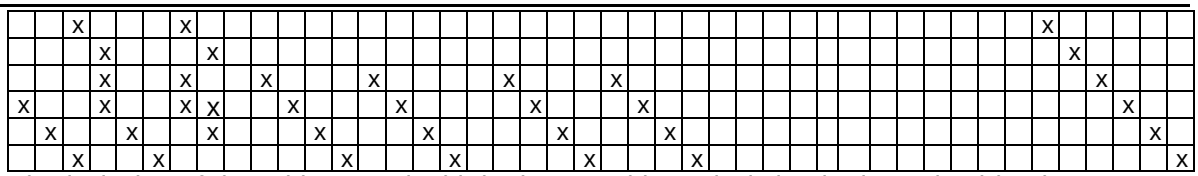
The Power PCI supports a Nibble Correct (S4EC-D4ED) code. The Nibble Correct code is capable of correcting multiple-bit errors within a nibble and detecting multiple bit errors within two nibbles. For banks that use the Nibble Correct error correction type, the Power PCI stores 16 bits of data on write operations to memory. The Power PCI checks that the data returned from memory is correct. If a correctable error is found, the Power PCI corrects the data before outputting that data. If scrubbing is enabled via the **Memory Scrubbing Mode Register (0x78)**, the Power PCI also scrubs the correctable data in memory. If an uncorrectable error is found, the Power PCI invalidates the data. If either a correctable or uncorrectable error is found, the Power PCI sets the corresponding bit in the **Status Register (0xC0)** and creates an error log entry. Note that the Nibble Correct code is only guaranteed to correct multiple-bit errors within one nibble and detect multiple-bit errors in two nibbles. An error in three or more nibbles may be detected, but it is not guaranteed. Note that this type of error coverage does detect a single bit addressing error, but those can not be corrected. Table 10 below shows the data bits which comprise the ECC bit generation equations for the Nibble Correct code.

Note that the nibbles discussed in the paragraph above refer to “aligned” nibbles (i.e., bits 0-3, 4-7, 8-11, etc.). An error that affected bits 7 and 8 would be an error in two nibbles.

Table 10 - Parity check matrix for odd-weight-column (81,65) Nibble Correct (S4EC-D4EC) code.

		Data Bits																																											
AP		63	62	61	60	59	58	57	56		55	54	53	52	51	50	49	48		47	46	45	44	43	42	41	40		39	38	37	36	35	34	33	32		31	30	29	28	27	26	25	24
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				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			x		
x																				x			x		x						x					x					x			x	
x																					x					x						x	x					x							x

Data Bits																											**ECC Bits**																
23	22	21	20	19	18	17	16		15	14	13	12	11	10	09	08		07	06	05	04	03	02	01	00		15	14	13	12	11	10	09	08		07	06	05	04	03	02	01	00
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						



Note: The inclusion of the address parity bit in the ECC bits' calculation is determined by the ***Memory Bank n Address Parity Mask*** bit in the **Correction Type Register (0x70)**.

4.2.6 Memory Function Interrupt and Error Handling

There are 3 interrupt levels in the Power PCI due to Memory related problems. The 3 levels represent varying degrees of criticality. All interrupts are active high and are cleared by writing a 0b'1' to the bit in the **Status Register (0xC0)**.

The Critical Error interrupt represents the highest level of criticality. This interrupt is activated when the Power PCI detects an internal fault. The Power PCI has internal fault checking in the form of odd parity on all programmable registers and one-hot bits on all state machines. If these internal checks detect a fault, the Power PCI's state and ability to respond to system requests is uncertain.

The High Priority interrupt represents a second level of criticality. This interrupt signals some type of error that the Power PCI can not correct. However, unlike the case of the critical error interrupt, the Power PCI's state and ability to respond to system requests are not compromised.

The Low Priority interrupt represents a third level of criticality. This interrupt signals that an event or non-critical (i.e., correctable by the Power PCI) error has occurred.

4.2.6.1 Error Log

The Error Log contains information about memory access errors detected by the Power PCI. The types of memory access errors that are logged are:

- Address Parity Error on MCI
- Data Parity Error on MCI Write
- Byte Enable Parity Error on MCI Write
- Invalid Address Error on MCI
- Invalid Request on MCI
- Address Parity Error on MRI
- Data Parity Error on MRI Write
- Byte Enable Parity Error on MRI Write
- Invalid Request on MRI
- An attempt to write ROM
- Memory Address out of Bounds
- Uncorrectable Address Error
- Uncorrectable Data Error
- Correctable Error - Multi-bit failure
- Correctable Error - Single bit failure

The Power PCI maintains an Error Log queue that contains information for the first 4 memory access errors detected. An Error Log entry consists of two words. The first word contains information about the error (access type, access source, error type, and if applicable, the failing nibble or bit and error syndrome). The second word contains the address of the error. Reading the Error Log address retrieves a complete Error Log entry. The Error Log can be read via 2 addresses, 0x'C8' and 0x'D8'. The first address, 0x'C8', is a non-destructive read of the Error Log (the entry remains in the Error Log). The second address, 0x'D8', is a read and clear (the entry is cleared and the position in the log made available for a new error). The Power PCI discards any error information for a memory access error that is detected when the Error Log queue is full. After both words of an Error Log entry have been read, the entry is available for a new error. For more information, see Section 4.2.11.7.4.

In conjunction with the Error Log, the Power PCI maintains an Error Counter. Any time the Power PCI detects a memory access error, the Error Counter is incremented by one. The Error Counter does not increment past its maximum count (i.e., it does not "roll over"). The Error Counter is reset to zero when it is read.

In the rare case where two (or more) errors occur on the same clock cycle, the Error Log only contains details for one error and the error counter is incremented by 1. The **Status Register (0xC0)** however**,** sets the bits corresponding to both errors.

4.2.7 Sparing Logic

Note: The MPC-106 does not support sparing.

The Power PCI provides the capability to replace up to 2 bytes of failing columns with spare columns. This capability is available on a bank-by-bank basis via the **Sparing, Scrubbing, and Initialization Registers**

Memory Bank n Sparing Register (0x80, 0x84, 0x88, 0x8C, 0x90, 0x94, 0x98, 0x9C).

The signals for the spare bytes of memory are shared with the ECC signals. Thus, the number of spares available to a bank is determined by the type of Error Correction selected and the number of bytes of memory physically available in the system for that bank. (Note that different banks in a system may have a different number of data and ECC bytes). The number of spares available for a memory bank is the number of bytes of ECC memory physically implemented minus the number of bytes used for the ECC algorithm selected.

The process of sparing involves both writes and reads. During writes, data from the byte specified in the **Sparing, Scrubbing, and Initialization Registers**

Memory Bank n Sparing Register (0x80, 0x84, 0x88, 0x8C, 0x90, 0x94, 0x98, 0x9C) is copied onto the spare byte. During reads, data from the spare byte is mapped to the byte specified in the **Sparing, Scrubbing, and Initialization Registers**

Memory Bank n Sparing Register (0x80, 0x84, 0x88, 0x8C, 0x90, 0x94, 0x98, 0x9C). An example of the sparing process is shown graphically in Figure 50.

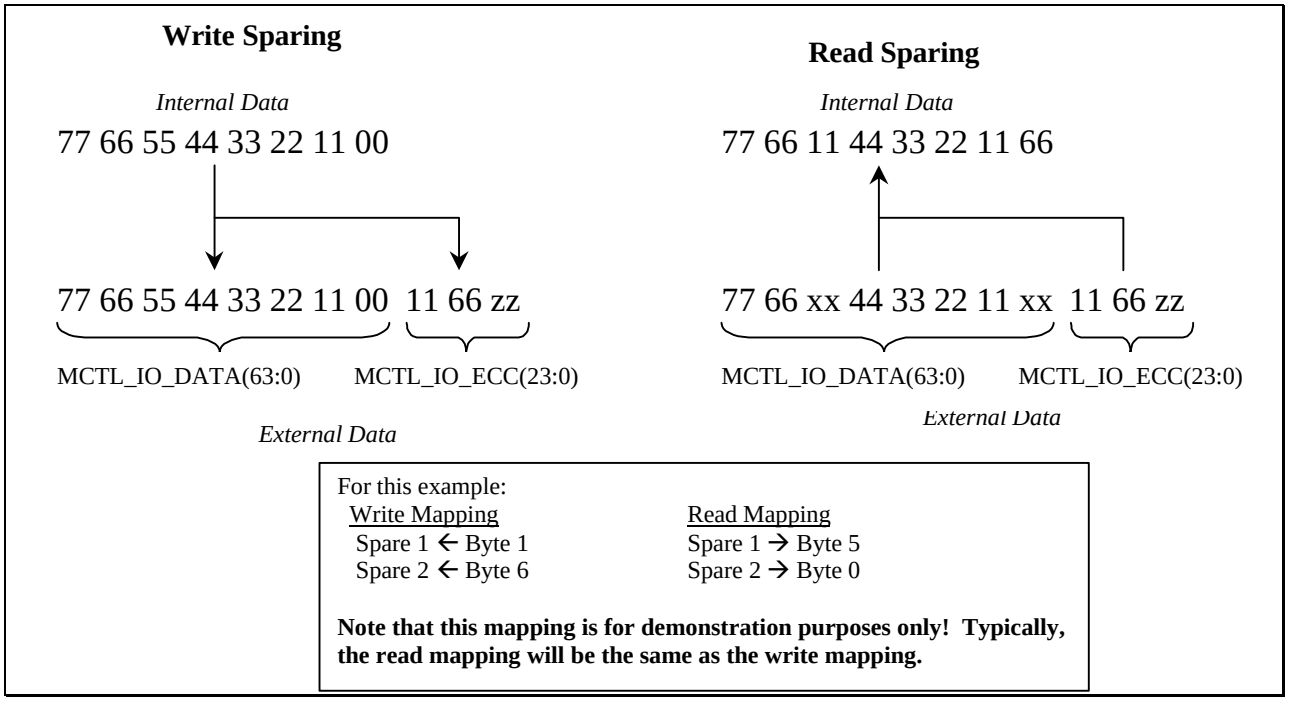


Figure 50: Sparing Example

Sparing only occurs if the respective read or write enable bit is set and the ***Memory Bank n Sparing Type*** field designates that the spare is available. If both of these conditions are met, then sparing can take place as determined by the ***Spare Assignment*** fields in the **Sparing, Scrubbing, and Initialization Registers**

Memory Bank n Sparing Register (0x80, 0x84, 0x88, 0x8C, 0x90, 0x94, 0x98, 0x9C).

4.2.8 Memory Scrubbing Logic

Note: The MPC-106 does not support memory scrubbing.

The Power PCI, when ECC is enabled, will detect correctable errors on reads of memory and pass corrected data to the Power PCI. Memory scrubbing is a feature by which the Power PCI, without processor intervention, that executes a write to also correct the error in external memory.

The Power PCI supports two modes of memory scrubbing, passive and active. The difference between the two modes is the source of the read of memory. In both passive and active modes, the Power PCI corrects any correctable errors detected as the result of a Power PCI requested memory read. In active mode, the Power PCI initiates memory reads during idle cycles to search memory for correctable errors. The rate at which memory reads are initiated in active mode is programmable via the ***Active Scrub Delay*** parameter in the **Memory Scrubbing Parameters Register (0x7C)**.

The Power PCI supports two scopes for active memory scrubbing, normal and not sleeping. In the normal scope, the Power PCI only initiates memory reads when operating in the Normal power management mode. In the not sleeping scope, the Power PCI initiates memory reads when in the Normal, Nap or Doze power management modes.

Scrubbing type and scope is selectable on a bank-by-bank basis; each memory bank can have a different mode and scope of scrubbing selected. Scrubbing is only done on enabled memory banks.

4.2.9 Data Output Selection Logic

The Power PCI always provides data as double-word data. In most cases, data read from external memory will be double-word data. However, in the case of half-word-wide data the Power PCI formats the half-word-wide data into double-word data prior to output. The Power PCI places the half-word from the lowest address (i.e., 3 LSBs of address = 0x'00') in the MSBs of the double word and the half-word from the highest address (i.e., 3 LSBs of address = 0x'11') in the LSBs of the double word.

Memory Contents		Data Returned to RAD750
Addr	Data	^0x DATA Bus(63:0)
11x	6677	
10x	4455	0011 2233 4455 6677
01x	2233	
00x	0011	

Figure 51: Power PCI Data Output Mapping for Half-Word-Wide Memory Banks

In the case of a correctable error, the Power PCI provides the corrected data and, if scrubbing is enabled, queue a write to memory to correct the data. Therefore, the Power PCI will likely have the corrected data a short time prior to the memory being updated.

4.2.10 Memory Initialization Logic

Note: The MPC-106 does not support memory initialization.

The Power PCI provides a built-in memory initialization capability. Using the Memory Initialization Registers (see Section 4.2.11.9) any block of memory can be filled with a double-word pattern. When the initialization is complete, the Power PCI sets a bit in its Status Register. At that point, software can reprogram the registers to initialize another block of memory.

4.2.11 Memory Controller Register Matrix

Table 11 is a listing of all of the Power PCI registers accessible within the Memory Controller function. Each register is 32-bits wide. Since the MCI Data Bus, used to interface from the remainder of the chip to the memory controller, is 64-bits wide, two registers (in the same double word) can be written or read simultaneously. The 32 MSBs of the data are mapped to registers whose address ends with 0b'000' and the 32 LSBs of the data are mapped to registers whose address ends with 0b'100'.

Note: This RAD750 board only populates Memory Banks 0 and 1. All other Memory Bank Configuration Registers are unused, and writing them has no effect.

Table 11 - Memory Function of the Power PCI Register/Resource Map

Memory Function of the Power PCI Register/Resource Map					
Register/Array/Command	**Address**	**Access**	**Reset State**	**Expected Use**	**Page**
General Purpose Registers					
Reserved	0x'0000'	RO	0x'0000 1801'	Diagnostics	
Enter SDRAM Self-Refresh Command	0x'0008'	WO	N/A	Operational	74
Exit SDRAM Self-Refresh Command	0x'000C'	WO	N/A	Operational	74
Initialization Registers					

Memory Function of the Power PCI Register/Resource Map					
Register/Array/Command	**Address**	**Access**	**Reset State**	**Expected Use**	**Page**
Initialization Data Low Register	0x'0010'	R/W	0x'0000 0000'	Initialization	86
Initialization Data High Register	0x'0014'	R/W	0x'0000 0000'	Initialization	87
Initialization Starting Address Register	0x'0018'	R/W	0x'0000 0000'	Initialization	87
Initialization Ending Address Register	0x'001C'	R/W	0x'0000 0000'	Initialization	87
Memory Addressing Registers					
Memory Bank 0 Address Register	0x'0020'	R/W	0x'FFF0 FFFF'	Configuration	68
Memory Bank 1 Address Register	0x'0024'	R/W	0x'8001 8000'	Configuration	68
Memory Bank 2 Address Register	0x'0028'	R/W	0x'8001 8000'	Configuration	68
Memory Bank 3 Address Register	0x'002C'	R/W	0x'8001 8000'	Configuration	68
Memory Bank 4 Address Register	0x'0030'	R/W	0x'8001 8000'	Configuration	68
Memory Bank 5 Address Register	0x'0034'	R/W	0x'8001 8000'	Configuration	68
Memory Bank 6 Address Register	0x'0038'	R/W	0x'8001 8000'	Configuration	68
Memory Bank 7 Address Register	0x'003C'	R/W	0x'8001 8000'	Configuration	68
Memory Type Register	0x'0060'	R/W	See Note	Configuration	69
Memory Address Mode Register	0x'0064'	R/W	0x'0000 0000'	Configuration	69
Memory Bank Enable Register	0x'0068'	R/W	See Note	Configuration	69
Memory Bank Write Enable Register	0x'006C'	R/W	0x'0000 0000'	Configuration	70
Error Correction/Detection Registers					
Correction Type Register	0x'0070'	R/W	0x'0000 FFFD'	Configuration	77
ECC Error Injection Register	0x'0074'	R/W	0x'0000 0000'	Configuration	78
Memory Scrubbing Mode Register	0x'0078'	R/W	0x'0000 0000'	Configuration	86
Memory Scrubbing Parameters Register	0x'007C'	R/W	0x'0000 0000'	Configuration	86
Memory Bank 0 Sparing Register	0x'0080'	R/W	0x'00FF FF00'	Configuration	84
Memory Bank 1 Sparing Register	0x'0084'	R/W	0x'00FF FF00'	Configuration	84
Memory Bank 2 Sparing Register	0x'0088'	R/W	0x'00FF FF00'	Configuration	84
Memory Bank 3 Sparing Register	0x'008C'	R/W	0x'00FF FF00'	Configuration	84
Memory Bank 4 Sparing Register	0x'0090'	R/W	0x'00FF FF00'	Configuration	84
Memory Bank 5 Sparing Register	0x'0094'	R/W	0x'00FF FF00'	Configuration	84
Memory Bank 6 Sparing Register	0x'0098'	R/W	0x'00FF FF00'	Configuration	84
Memory Bank 7 Sparing Register	0x'009C'	R/W	0x'00FF FF00'	Configuration	84
Error Reporting Registers					
Status Register	0x'00C0'	RO	0x'0000 0000'	Status	79
Interrupt Enable Register	0x'00C4'	R/W	0x'0000 0000'	Configuration	81
Error Log Word 1	0x'00C8'	RO	0x'0000 0000'	Status	83
Error Log Word 2	0x'00CC'	RO	0x'0000 0000'	Status	84
Error Counter	0x'00D0'	RO & Clear	0x'0000 0000'	Status	81

Memory Function of the Power PCI Register/Resource Map					
Register/Array/Command	**Address**	**Access**	**Reset State**	**Expected Use**	**Page**
Error Log Word 1	0x‘00D8’	RO & Clear	0x‘0000 0000’	Status	83
Error Log Word 2	0x‘00DC’	RO & Clear	0x‘0000 0000’	Status	84
SDRAM Access Registers					
SDRAM Configuration Register	0x‘00E0’	R/W	0x‘0022 0000’	Configuration	71
SDRAM Timing Parameters Register 1	0x‘00E8’	R/W	0x‘4100 0000’	Configuration	72
SDRAM Timing Parameters Register 2	0x‘00EC’	R/W	0x‘600F 01FF’	Configuration	73
SRAM/ROM Access Registers					
ROM Timing Parameters Register	0x‘00F0’	R/W	0x‘000F FFFF’	Configuration	75
SRAM Timing Parameters Register	0x‘00F4’	R/W	0x‘0000 FFFF’	Configuration	76
Note: The default values of the Memory Type Register (0x60) and the Memory Bank Enable Register (0x68) depend on the value of the **IO_MCTL_SDRAM_BANK** input bus.					

4.2.11.1 Memory Addressing Register Descriptions

The base address for the Memory Controller Function registers is 0x’BF80 0000’.

4.2.11.1.1 Memory Bank n Address Register (0x20, 0x24, 0x28, 0x2C, 0x30, 0x34, 0x38, 0x3C)

There are 8 Memory Bank Address Registers, one for each memory bank. Note that for all banks except for bank 0, the reset value of the Starting Address is greater than the reset value of the Ending Address. Thus, the address ranges for these banks are invalid until reprogrammed by software.

Bank	Addres
Bank 0	0x’20’
Bank 1	0x’24’
Bank 2	0x’28’
Bank 3	0x’2C’
Bank 4	0x’30’
Bank 5	0x’34’
Bank 6	0x’38’
Bank 7	0x’3C’

Bits 31:16	***Memory Bank n Starting Address:*** These bits correspond to the 16 MSBs of the 32-bit address. The lower boundary for memory bank n is equal to <Memory Bank n Starting Address>||0x’0000’. The reset value of these bits are: • 0x’FFF0’ - for bank 0 • 0x’8001’ - for all other banks
Bits 15:0	***Memory Bank n Ending Address:*** These bits correspond to the 16 MSBs of the 32-bit address. The upper boundary for memory bank n is equal to <Memory Bank n Ending Address>||0x’FFFF’. The reset values of these bits are: • 0x’FFFF’ - for bank 0 • 0x’8000’ - for all other banks

4.2.11.1.2 Memory Type Register (0x60)

Bits 31:16	***Reserved***: These bits are reserved, and return 0b'0' when read.
Bits 15:14	***Memory Bank 7 Type:*** '00' - ROM '01' - SRAM '1x' - SDRAM The reset value of these bits is 0b'00', unless the corresponding bit in the **IO_MCTL_SDRAM_BANK** input signal bus is set. In that case the reset value of these bits will be 0b'11' and these bits will be read-only.
Bits 13:12	***Memory Bank 6 Type***
Bits 11:10	***Memory Bank 5 Type***
Bits 9:8	***Memory Bank 4 Type***
Bits 7:6	***Memory Bank 3 Type***
Bits 5:4	***Memory Bank 2 Type***
Bits 3:2	***Memory Bank 1 Type***
Bits 1:0	***Memory Bank 0 Type***

4.2.11.1.3 Memory Address Mode Register (0x64)

These bits define the mapping of address bits to SDRAM memory parts. This register has no effect on ROM or SRAM addressing. (The numbers in the tables below correspond to bits of the Memory Request Interface Address Bus, **P60x_MCTL_ADDR(31:0).** The entry "AP" indicates the Auto-Precharge signal to the SDRAM).

NOTE: The 2 MSBs, **ADDR(17:16)**, signify the address bits used as sub-bank enable bits (if any). Shaded bits are not expected to be used, they are driven with the value indicated to minimize internal muxing. Obviously, all of these address bits won't be used in every design. It is left to the user to select the proper mode and address bits for their SDRAM organization.

Table 12 - SDRAM Address Modes' Bit Mapping

Mode 0	**ADDR Bits**														
	17	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**
Row	25	23	25	23	22	21	20	19	18	17	16	15	14	13	12
Column	25	23	25	28	AP	24	11	10	9	8	7	6	5	4	3

Mode 1	**ADDR Bits**														
	17	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**
Row	25	24	25	23	22	21	20	19	18	17	16	15	14	13	12
Column	25	24	25	28	AP	26	11	10	9	8	7	6	5	4	3

Mode 2	**ADDR Bits**														
	17	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**
Row	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12
Column	26	25	25	28	AP	27	11	10	9	8	7	6	5	4	3

Mode 3	**ADDR Bits**														
	17	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**

Row	24	11	25	23	22	21	20	19	18	17	16	15	14	13	12
Column	24	11	25	28	AP	26	11	10	9	8	7	6	5	4	3

Bits 31:16	***Reserved***: These bits are reserved, and return 0b'0' when read.
Bits 15:14	***Memory Bank 7 Mode:*** '00' - Mode 0; for parts organized as 11x9(2), 11x10(2) '01' - Mode 1; for parts organized as 12x9(4), 13x9(2), 12x10(4), 13x10(2) '10' - Mode 2; for parts organized as 13x9(4), 13x10(4), 13x11(4) '11' - Mode 3; for parts organized as 12x8(4), 13x8(2), 13x8(4) The reset value of these bits is 0b'00'.
Bits 13:12	***Memory Bank 6 Mode***
Bits 11:10	***Memory Bank 5 Mode***
Bits 9:8	***Memory Bank 4 Mode***
Bits 7:6	***Memory Bank 3 Mode***
Bits 5:4	***Memory Bank 2 Mode***
Bits 3:2	***Memory Bank 1 Mode***
Bits 1:0	***Memory Bank 0 Mode***

4.2.11.1.4 Memory Bank Enable Register (0x68)

Bits 31:8	***Reserved***: These bits are reserved, and return 0b'0' when read.
Bits 7:0	***Memory Bank n Enable:*** 0b'0' - Disable 0b'1' - Enable Bit n corresponds to bank n. • The reset value of these bits are 0x'01' or IO**_MCTL_SDRAM_BANK** – Bank 0 and any banks specified as SDRAM by the **IO_MCTL_SDRAM_BANK** input signals will default to enabled. All other banks default to disabled.

4.2.11.1.5 Memory Bank Write Enable Register (0x6C)

Bits 31:8	***Reserved***: These bits are reserved, and return 0b'0' when read.
Bits 7:0	***Memory Bank n Write Enable:*** 0b'0' - Disable (writes not allowed) 0b'1' - Enable (writes allowed) Bit n corresponds to bank n. • The reset value of these bits are 0x'00' (i.e., All banks disabled)

4.2.11.2 SDRAM Access Register Descriptions

All banks of SDRAM are expected to conform to one set of characteristics. Thus, the parameters described in this section apply to all banks of SDRAM.

4.2.11.2.1 SDRAM Configuration Register (0xE0)

Bits 31:30	***Reserved***: These bits are reserved, and return 0b'0' when read.

Bits 29:16	***Mode Register Set Value:*** These bits contain the value that is loaded into the SDRAM parts' Mode Register during initialization. The CAS latency field will also be used by the Power PCI's Memory State Machine. The standard meaning of the fields are shown below, but note that the Power PCI does not support all of the options. **Bits 29:23** - Opcode (typical operation requires 0x'00'. See memory part documentation for details) **Bits 22:20** - CAS Latency 000 = Reserved 001 = Reserved 010 = 2 cycles 011 = 3 cycles 1xx = Reserved **Bit 19** - Burst Type 0 = Sequential 1 = Interleaved (not supported) **Bits 18:16** - Burst Length 000 = 1 (not supported) 001 = 2 (not supported) 010 = 4 011 = 8 (not supported) 100 = Reserved 101 = Reserved 110 = Reserved 111 = Full Page (not supported) The reset value of these bits is 0x'0022'.
Bit 15	***SDRAM Controller Enable:*** 0b'0' - Disable 0b'1' - Enable This bit enables the Power PCI to access SDRAM. This bit should be written to a 0b'1' following a reset after an initial pause of 100-200 µs (see the specific SDRAM part documentation for the exact delay required). Prior to writing this bit to 0b'1', the SDRAM parameter registers in the Power PCI should be configured. Once enabled, the Power PCI initializes the SDRAM part via the power-on sequence and subsequently the SDRAM parts will be available for access. The reset value of this bit is 0b'0'.
Bit 14	***Self-Refresh Enable:*** 0b'0' - Disable 0b'1' - Enable This bit is currently not used. The Power PCI will always put any enabled SDRAM banks into Self-Refresh mode when the chip is going to "Sleep Mode". The reset value of this bit is 0b'0'.
Bit 13	***Stop Clocks during Self-Refresh Enable:*** 0b'0' - Disable 0b'1' - Enable This bit enables the Power PCI to turn off clocks to the SDRAMs during Self-Refresh. This action will save power, but the SDRAMs will require the clocks to be restarted for ***Minimum Clock Restart to Clock Enable Time*** (see the **SDRAM Timing Parameters Register 2**

	(0xEC)) clock cycles before the Clock Enable signal can be activated and SDRAM can be accessed. The reset value of this bit is 0b'0'.
Bit 12	***Auto-Precharge Enable:*** 0b'0' - Disable 0b'1' - Enable This bit determines the type of commands that are used to access SDRAM. If this bit is enabled, all accesses of SDRAM will be Read with Auto-Precharge or Write with Auto-Precharge. The reset value of this bit is 0b'0'.
Bits 11:0	***Reserved***: These bits are reserved, and return 0b'0' when read.

4.2.11.2.2 SDRAM Timing Parameters Register 1 (0xE8)

Bits 31:28	***Refresh Stagger:*** This value represents the number of clock cycles between Auto-Refresh commands to different SDRAM banks during a Refresh Interval. The purpose of this parameter is to allow the user to stagger the refresh commands to the SDRAM banks to minimize the instantaneous current draw. A value of 0b'0' specifies no delay between banks (i.e., all SDRAM banks begin Auto-Refresh on the same cycle). A value of 0b'1' would stagger the start of each banks' Auto-Refresh by one cycle (i.e., SDRAM bank X would begin on cycle A, SDRAM bank Y would begin on cycle A+1, SDRAM bank Z would begin on cycle A+2, etc.). The maximum value for this field is 0b'1111', or 15 clock cycles. The reset value of these bits is 0b'0100'.
Bits 27:16	***Refresh Interval:*** This value represents the number of Real Time clock cycles between SDRAM refresh commands. One row in each bank of SDRAM is refreshed during each refresh cycle. **Note**: Since in most environments a low value could cause excessive cycles to be lost to refresh operations, the Power PCI does not permit a value less than 0x'020' to be written into these bits. Any attempt to write a value less than 0x'020', will result in 0x'020' being written. The reset value of these bits is 0x'100'.
Bits 15:11	***Reserved***: These bits are reserved, and return 0b'0' when read.
Bits 10:8	***SDRAM Clock Skew:*** This value represents the number of delay units that are inserted between the SDRAM clock input to the Memory Interface Function and the SDRAM clock output from the Memory interface Function. The duration of each delay unit is approximately 1.5ns. The reset value of these bits is 0b'000'.
Bits 7:0	***Maximum Bank Active Time:*** This value represents the maximum duration (in units of eight clock cycles) that a page of SDRAM may remain open. The reset value of these bits is 0x'00'.

4.2.11.2.3 SDRAM Timing Parameters Register 2 (0xEC)

Bits 31:30	***Minimum Precharge to Activate Time:*** This value represents the minimum number of clock cycles required between a precharge

	command and the next activate command. 0b‘00’ - Reserved 0b‘01’ - 2 clock cycles 0b‘10’ - 3 clock cycles 0b‘11’ - 4 clock cycles The reset value of these bits is 0b’01’.
Bits 29:28	***Minimum Activate to Command Time:*** This value represents the minimum number of clock cycles required between an activate command and a read or write command. 0x - Reserved 0b‘10’ - 2 clock cycles 0b‘11’ - 3 clock cycles The reset value of these bits is 0b’10’.
Bits 27:24	***Minimum Activate to Precharge Time:*** This value represents the minimum number of clock cycles required between an activate and a precharge command. ‘0001’ - 1 clock cycle ‘0010’ - 2 clock cycles ‘0011’ - 3 clock cycles … ‘1111’ - 15 clock cycles ‘0000’ - 16 clock cycles The reset value of these bits is 0b’0000’.
Bits 23:20	***Reserved***: These bits are reserved, and return 0b’0’ when read.
Bits 19:16	***Minimum Write Data to Precharge Time:*** This value represents the minimum number of clock cycles required between the transmission of the last data word of a write and a Precharge command. If Auto-Precharge is enabled, this value represents the number of clock cycles between the transmission of the last data word of a write and the start of the Auto-Precharge. ‘0000’ - Reserved ‘0001’ - ‘1111’ Binary Decode (1 to 15 clock cycles) The reset value of these bits is 0b’1111’.
Bits 15:12	***Minimum Refresh Command to Activate Time:*** This value represents the minimum number of clock cycles required between a Refresh command and the next Activate command. ‘0001’ - 1 clock cycle ‘0010’ - 2 clock cycles ‘0011’ - 3 clock cycles … ‘1111’ - 15 clock cycles ‘0000’ - 16 clock cycles

	The reset value of these bits is 0b'0000'.
Bits 11:10	***Reserved***: These bits are reserved, and return 0b'0' when read.
Bits 9:8	***Minimum Clock Disable to Clock Halted Time:*** This value represents the minimum number of clock cycles required between clock disable (**CLKE** going low) and when the SDRAM clock, **CLK**, can be stopped. 0b'00' - Reserved 0b'01' - 1 clock cycle 0b'10' - 2 clock cycles 0b'11' - 3 clock cycles The reset value of these bits is 0b'01'.
Bits 7:4	***Minimum Clock Restart to Clock Enable Time:*** This value represents the minimum number of 16 clock cycles required between the SDRAM clock, **CLK**, restart and clock enable (**CLKE** going high). 0b'0000' - Reserved (0 clock cycles) 0b'0001' - 16 clock cycles 0b'0010' - 32 clock cycles … 0b'1111' - 240 clock cycles The reset value of these bits is 0b'1111'.
Bits 3:0	***Minimum Self-Refresh Exit to Activate Time:*** This value represents the minimum number of clock cycles required between a self-refresh exit (**CLKE** going high) and the first activate command. 0b'0000' - Reserved 0b'0001' - 0b'1111' Binary count (1 to 15) The reset value of these bits is 0b'1111'.

4.2.11.3 SDRAM Self-Refresh Control Commands

4.2.11.3.1 Enter SDRAM Self-Refresh Command (0x08)

The data for this command is a don't care. Upon reception of a write to this address with valid data parity the Memory Controller in the Power PCI will place all enabled SDRAM parts in a Self-Refresh state.

4.2.11.3.2 Exit SDRAM Self-Refresh Command (0x0C)

The data for this command is a don't care. Upon reception of a write to this address with valid parity the Memory Controller in the Power PCI will cause all SDRAM parts to exit their Self-refresh states.

4.2.11.4 ROM Access Register Descriptions

4.2.11.4.1 ROM Timing Parameters Register (0xF0)

Bits 31:21	***Reserved***: These bits are reserved, and return 0b'0' when read.
Bit 20	***Burst Mode Enable:*** This bit enables support for burst devices. If enabled the Next Wait Parameters are valid. 0b'0' - Disable

	0b‘1’ - Enable The reset value of this bit is 0b’0’.
Bits 19:16	***Hold Address on Write Cycles:*** This value contains the number of clock cycles to hold the write address on the bus after the Write Enable signal has gone inactive. 0b’0000’ - Reserved 0b’0001’ - 1 clock cycle 0b’0010’ - 2 clock cycles … 0b’1111’ - 15 clock cycles The reset value of these bits is 0b’1111’.
Bits 15:12	***Next Wait for Reads:*** This value contains the number of clock cycles required for reads on burst transfers after the first read data. This parameter is only used if Burst Mode is Enabled. 0b’0000’ - Reserved 0b’0001’ - 1 clock cycle 0b’0010’ - 2 clock cycles … 0b’1111’ - 15 clock cycles The reset value of these bits is 0b’1111’.
Bits 11:8	***Next Wait for Writes:*** This value contains the number of clock cycles required for writes on burst transfers after the first write data. This parameter is only used if Burst Mode is Enabled. 0b’0000’ - Reserved 0b’0001’ - Reserved 0b’0010’ - 2 clock cycles … 0b’1111’ - 15 clock cycles The reset value of these bits is 0b’1111’.
Bits 7:4	***First Wait for Reads:*** If Burst Mode is Enabled, this value contains the number of clock cycles required for the first read of a burst transfer to complete. If Burst Mode is Disabled, this value reflects the number of clock cycles required for any read to complete. 0b’0000’ - Reserved 0b’0001’ - 1 clock cycle 0b’0010’ - 2 clock cycles … 0b’1111’ - 15 clock cycles The reset value of these bits is 0b’1111’.
Bits 3:0	***First Wait for Writes:***

	If Burst Mode is Enabled, this value contains the number of clock cycles required for the first write of a burst transfer to complete. If Burst Mode is Disabled, this value reflects the number of clock cycles required for any write to complete. 0b'0000' - Reserved 0b'0001' - Reserved 0b'0010' - 2 clock cycles … 0b'1111' - 15 clock cycles The reset value of these bits is 0b'1111'.

4.2.11.5 SRAM Access Register Descriptions

4.2.11.5.1 SRAM Timing Parameters Register (0xF4)

Bits 31:17	***Reserved***: These bits are reserved, and return 0b'0' when read.
Bit 16	***Burst Mode Enable:*** This bit enables support for burst devices. If enabled the Next Wait Parameters are valid. 0b'0' - Disable 0b'1' - Enable The reset value of this bit is 0b'0'.
Bits 15:12	***Next Wait for Reads:*** This value contains the number of clock cycles required for reads on burst transfers after the first read data. This parameter is only used if Burst Mode is Enabled. 0b'0000' - Reserved 0b'0001' - 1 clock cycle 0b'0010' - 2 clock cycles … 0b'1111' - 15 clock cycles The reset value of these bits is 0b'1111'.
Bits 11:8	***Next Wait for Writes:*** This value contains the number of clock cycles required for writes on burst transfers after the first write data. This parameter is only used if Burst Mode is Enabled. 0b'0000' - Reserved 0b'0001' - Reserved 0b'0010' - 2 clock cycles … 0b'1111' - 15 clock cycles The reset value of these bits is 0b'1111'.
Bits 7:4	***First Wait for Reads:*** If Burst Mode is Enabled, this value contains the number of clock cycles required for the first read of a burst transfer to complete. If Burst Mode is Disabled, this value reflects the number of clock cycles required for any read

	to complete. 0b'0000' - Reserved 0b'0001' - 1 clock cycle 0b'0010' - 2 clock cycles … 0b'1111' - 15 clock cycles The reset value of these bits is 0b'1111'.
Bits 3:0	***First Wait for Writes:*** If Burst Mode is Enabled, this value contains the number of clock cycles required for the first write of a burst transfer to complete. If Burst Mode is Disabled, this value reflects the number of clock cycles required for any write to complete. 0b'0000' - Reserved 0b'0001' - Reserved 0b'0010' - 2 clock cycles … 0b'1111' - 15 clock cycles The reset value of these bits is 0b'1111'.

4.2.11.6 Error Correction/Detection Register Descriptions

In the register descriptions that follow, the byte numbering correspond to data bits as shown in the table below.

Byte	Data Bits
7	63:56
6	55:48
5	47:40
4	39:32
3	31:24
2	23:16
1	15:8
0	7:0

Byte	ECC bits
10	23:16
9	15:8
8	7:0

4.2.11.6.1 Correction Type Register (0x70)

Bits 31:24	***Reserved***: These bits are reserved, and return 0b'0' when read.
Bits 23:16	***Memory Bank n Address Parity Mask*** These bits determine whether or not the Address Parity bit is included in the ECC code calculation for a given bank. 0 – The Address Parity bit is not included in the ECC calculation 1 – The Address Parity bit is included in the ECC calculation The reset values of these bits is 0x'00'
Bits 15:14	***Memory Bank 7 Correction Type*** 0b'00' - Odd Parity (one parity bit per byte). Each parity bit is driven on **ECC (n)**, where n is the byte number. 0b'01' - SECDED (Single Error Correct, Double Error Detect). This algorithm provides an 8-

	bit code covering the 32 address bits and 64 data bits. Any single bit error on a data bit is correctable. Any single bit error on an address bit, double bit error on two data bits, or double bit error on an address bit and data bit is detectable. The ECC code is transferred on the bus lines **ECC(7:0)**. 0b'10' - Nibble Correct. This algorithm provides a 16-bit code covering the 32 address bits and 64 data bits. Any single bit error on a data bit or any multiple bit error within a nibble is correctable. Any single bit error on an address bit, double bit error on two data bits, or double bit error on an address bit and data bit is detectable. The ECC code is transferred on the bus lines **ECC(15:0)**. 0b'11' - No correction The reset value of these bits is 0b'11'.
Bits 13:12	***Memory Bank 6 Correction Type***
Bits 11:10	***Memory Bank 5 Correction Type***
Bits 9:8	***Memory Bank 4 Correction Type***
Bits 7:6	***Memory Bank 3 Correction Type***
Bits 5:4	***Memory Bank 2 Correction Type***
Bits 3:2	***Memory Bank 1 Correction Type***
Bits 1:0	***Memory Bank 0 Correction Type*** Unlike the other banks, the correction type for bank 0 defaults to the SECDED correction scheme. These bits may be overwritten by software so that bank 0 uses one of the other correction schemes. The reset value of these bits is 0b'01'.

4.2.11.6.2 ECC Error Injection Register (0x74)

Note: The MPC-106 does not support Error Injection.

Bit 31	***Error Injection Enable:*** This bit indicates whether the Error Injection Mask is used on a write to memory. 0b'0' - Disabled 0b'1' - Enabled The reset value of this bit is 0b'0'.
Bits 30:16	***Reserved***: These bits are reserved, and return 0b'0' when read.
Bits 15:0	***Error Injection Mask:*** This field is used to invert Error Correction bits on writes to memory. The value written to memory is the exclusive-or of this field and the "uncorrupted" value intended for the ECC bits. Note: This mask will only invert bits used for ECC or Parity (as determined by the value in the Correction Type Registers). This mask will not alter **ECC** bits used for sparing (provided they are not sparing ECC bytes). The reset value of these bits is 0x'0000'.

4.2.11.7 Error Reporting Register Descriptions

4.2.11.7.1 Status Register (0xC0)

The Status Register contains information about the source of Power PCI memory related interrupts. Only the Power PCI itself can set the bits in this register. The Power PCI is able to clear bits in this register by writing a 0b'1' to the individual bits.

Bits 31:19	***Reserved***: These bits are reserved, and return 0b'0' when read.
Bit 18	***Critical Error Detected:*** This bit signifies that a critical error was detected in the Power PCI's Internal Memory Logic. This error is a result of a parity error on an internal bus or an invalid state on an internal state machine. Note: Only a chip reset can clear this bit. The reset value of this bit is 0b'0'.
Bit 17	***Address Parity Error on Memory Request Interface:*** This bit signifies that a command was received from the Memory Request Interface with a parity error on the address bus. An entry on the **Error Log** is created for this error. Note: A chip reset can only clear this bit. The reset value of this bit is 0b'0'.
Bit 16	***Byte Enable Parity Error on Memory Request Interface Write:*** This bit signifies that a write command was received from the Memory Request Interface with a parity error on the byte enables. An entry on the **Error Log** is created for this error. The reset value of this bit is 0b'0'.
Bit 15	***Invalid Request on Memory Request Interface:*** This bit signifies that an invalid memory access request was received from the Memory Request Interface. Possible invalid memory requests are: ♦ Multi-beat write request, which crosses a cache-line boundary. An entry on the **Error Log** is created for this error. The reset value of this bit is 0b'0'.
Bit 14	***Address Parity Error on Memory Configuration Interface:*** This bit signifies that a command was received from the Memory Configuration Interface with a parity error on the address bus. An entry on the **Error Log** is created for this error. The reset value of this bit is 0b'0'.
Bit 13	***Data Parity Error on Memory Request Interface Write:*** This bit signifies that a write command was received from the Memory Request Interface with a parity error on the write data. An entry on the **Error Log** is created for this error. The reset value of this bit is 0b'0'.
Bit 12	***Byte Enable Parity Error on Memory Configuration Interface Write:*** This bit signifies that a write command was received from the Memory Configuration Interface with a parity error on the byte enables. An entry on the **Error Log** is created for this error. The reset value of this bit is 0b'0'.
Bit 11	***Invalid Address on Memory Configuration Interface:*** This bit signifies that an address was received from the Memory Configuration Interface that was invalid. An address on the MCI is invalid if:

	9. The command is a read and no register exists in the cache-line, i.e., no register has an address that matches the address bits, **P60x_MCTL_MC_ADDR(15:5).** 10. The command is a write and the address does not point to a register or the register it points to is read-only. An entry on the **Error Log** is created for this error. The reset value of this bit is 0b'0'.
Bit 10	***Invalid Command on Memory Configuration Interface:*** This bit signifies that the two MCI control signals, **P60x_MCTL_MC_WR_PUSH** and **P60x_MCTL_MC_RD_PULL,** were active simultaneously. Or that both the **Enter SDRAM Self-Refresh Command** (0x'08') and the **Exit SDRAM Self-Refresh Command** (0x'0C') were received simultaneously. An entry on the **Error Log** is created for this error. The reset value of this bit is 0b'0'.
Bit 9	***Memory Refresh Overflow:*** This bit signifies that a bank of SDRAM was not refreshed within the time specified by the ***Refresh.*** The reset value of this bit is 0b'0'.
Bit 8	***ROM Write Error:*** This bit signifies that a write command was received from the Memory Request Interface for a memory bank that is read-only. An entry on the **Error Log** is created for this error. The reset value of this bit is 0b'0'.
Bit 7	***Address out of Bounds:*** This bit signifies that a read or write command was received from the Memory Request Interface whose address is 1) not within the bounds of any of the enabled memory banks, or 2) within the bounds of more than one enabled memory bank. An entry on the **Error Log** is created for this error. The reset value of this bit is 0b'0'.
Bit 6	***Uncorrectable Address Error:*** This bit signifies that a single-bit error was detected on an address bit during a read from memory. This error may be correctable, but it is not guaranteed, because the error could be the result of an incorrect write to memory. An entry on the **Error Log** is created for this error. The reset value of this bit is 0b'0'.
Bit 5	***Uncorrectable Data Error:*** This bit signifies that a double-bit error was detected during a read from memory. An entry on the **Error Log** is created for this error. The reset value of this bit is 0b'0'.
Bit 4	***Nibble Correctable Data Error:*** This bit signifies that a nibble/byte error with nibble/byte correction enabled was detected during a read from memory. An entry on the **Error Log** is created for this error. The reset value of this bit is 0b'0'.

Bit 3	***Single-Bit Correctable Data Error:*** This bit signifies that a single-bit correctable data error was detected during a read from memory. An entry on the **Error Log** is created for this error. The reset value of this bit is 0b'0'.
Bit 2	***Memory Initialization Complete:*** This bit signifies that the initialization of memory locations (see Section 4.2.11.9) has completed. The reset value of this bit is 0b'0'.
Bit 1	***Scrub Loop Complete:*** This bit signifies that a loop through all of the memory banks enabled for scrubbing has completed. The reset value of this bit is 0b'0'.
Bit 0	***SDRAM Self-Refresh Command Comlete:*** This bit signifies that an SDRAM Self-Refresh command has completed. The reset value of this bit is 0b'0'.

4.2.11.7.2 Interrupt Enable Register (0xC4)

This register controls which status signals propagate to the interrupt interface. The interrupts raised by the various conditions are:

Interrupt Signal	Status Bit	Error Detected
MCTL_CR_ERR	19	Critical Error
	18	Address Parity Error on Memory Request Interface
MCTL_HP_INT	17	Data Parity Error on Memory Request Interface Write
	16	Byte Enable Parity Error on Memory Request Interface Write
	15	Invalid Request on Memory Request Interface
	14	Address Parity Error on Memory Configuration Interface
	13	Data Parity Error on Memory Configuration Interface Write
	12	Byte Enable Parity Error on Memory Configuration Interface Write
	11	Invalid Address on Memory Configuration Interface
	10	Invalid Command on Memory Configuration Interface
	9	Memory Refresh Overflow
	8	ROM Write Error
	7	Address out of Bounds
	6	Uncorrectable Address Error
	5	Uncorrectable Data Error
MCTL_LP_INT	4	Nibble Correctable Data Error
	3	Single-bit Correctable Data Error
	2	Memory Initialization Complete
	1	Scrub Loop Complete
	0	SDRAM Self-Refresh Command Complete

Bits 31:18	***Reserved***: These bits are reserved, and return 0b'0' when read.
Bit 17	***Data Parity Error on Memory Request Interface Write Enable*** 0b'0' - Disable (interrupt is not activated when corresponding status bit is set) 0b'1' - Enable (interrupt is activated when corresponding status bit is set)

Bit 16	***Byte Enable Parity Error on Memory Request Interface Write Enable***
Bit 15	***Invalid Request on Memory Request Interface Enable***
Bit 14	***Address Parity Error on Memory Configuration Interface Enable***
Bit 13	***Data Parity Error on Memory Configuration Interface Write Enable***
Bit 12	***Byte Enable Parity Error on Memory Configuration Interface Write Enable***
Bit 11	***Invalid Address on Memory Configuration Interface Enable***
Bit 10	***Invalid Command on Memory Configuration Interface Enable***
Bit 9	***Memory Refresh Overflow Enable***
Bit 8	***ROM Write Error Enable***
Bit 7	***Address out of Bounds Enable***
Bit 6	***Uncorrectable Address Error Enable***
Bit 5	***Uncorrectable Data Error Enable***
Bit 4	***Nibble/Byte Correctable Data Error Enable***
Bit 3	***Single-Bit Correctable Data Error Enable***
Bit 2	***Memory Initialization Complete Enable***
Bit 1	***Scrub Loop Complete Enable***
Bit 0	***SDRAM Self-Refresh Command Complete Enable***
The reset value of these bits is 0x'00000'.	

4.2.11.7.3 Error Counter (0xD0)

This register is a read-and-clear register.

Bits 31:8	***Reserved***: These bits are reserved, and return 0b'0' when read.
Bits 7:0	***Error Count*** This value is a count of the number of "log-able" errors since this register was last reset or read. A "log-able" error is any error (as specified in the Status Register) which creates an error log entry. The reset value of these bits is 0x'00'.

4.2.11.7.4 Error Log

This address is the access address for the Error Log FIFO. Each read returns a new entry from the FIFO. Note that a read of the Error Log retrieves both **Error Log Word 1 (0xD8)** and **Error Log Word 2 (0xDC).** The FIFO is 4 entries deep; it holds information for the first four errors detected until the FIFO is read. Additional errors are counted in the **Interrupt Enable Register (0xC4),** but will not be stored in the Error Log.

For nondestructive reads of the Error Log (not read and clear), use address 0xC8 for Error Log Word 1 and 0xCC for Error Log Word 2.

4.2.11.7.4.1 Error Log Word 1 (0xD8)

Bit 31	***Entry Valid:*** This bit indicates if this entry is valid. It is set when the entry is created and automatically cleared when the entry is read. 0b'0' - Not Valid 0b'1' - Valid

	The reset value of this bit is 0b'0'.
Bit 30	***Read/Write:*** This bit indicates if the error occurred during a read or write transfer. 0b'0' - Read 0b'1' - Write The reset value of this bit is 0b'0'.
Bit 29	***Source of Address:*** This bit indicates the source of the address on which the error occurred. 0b'0' - Memory Controller portion of the Power PCI Interface 0b'1' - Internal to the Memory Controller portion of the Power PCI (i.e., such as a memory scrub operation) The reset value of this bit is 0b'0'.
Bits 28:24	***Error Type:*** These bits indicate the type of error that occurred. 0x'12' - Address Parity Error on Memory Request Interface 0x'11' - Data Parity Error on Memory Request Interface Write 0x'10' - Byte Enable Parity Error on Memory Request Interface Write 0x'0F' - Invalid Request on Memory Request Interface 0x'0E' - Address Parity Error on Memory Configuration Interface 0x'0D' - Data Parity Error on Memory Configuration Interface Write 0x'0C' - Byte Enable Parity Error on Memory Configuration Interface Write 0x'0B' - Invalid Address on Memory Configuration Interface 0x'0A' - Invalid Command on Memory Configuration Interface 0x'08' - Attempt to write ROM 0x'07' - Address out of Bounds 0x'06' - Uncorrectable Address Error 0x'05' - Uncorrectable Data Error 0x'04' - Correctable Error (multi-bit failure, i.e., failure within a Nibble) 0x'03' - Correctable Error (single-bit failure) All other values are reserved. The reset value of these bits is 0x'00'.
Bits 23:21	***Reserved***: These bits are reserved, and return 0b'0' when read.
Bits 20:16	***Error Location:*** The format of these bits depends on the Error Type. **Address Parity Error on Memory Request Interface, or** **Invalid Request on Memory Request Interface:** **Bits 20:18** - Reserved **Bits 17:16** - Transfer Size **Correctable Error (multi-bit or single-bit failure):** These bits indicate the nibble that contained the correctable error. Values 15-0 refer to the 16 nibbles that comprise data bits 63:0, respectively. Values 21-16 refer to the 6 nibbles that comprise ECC bits 23:0, respectively. (For example, a value of 12 would indicate a failure in the nibble that contains data bits 51:48). This value will reflect the physical data nibble that is in error. Thus, if a spared nibble is the failing nibble, this value will reflect the spare nibble column, not the nibble column that was being spared. For example, if ECC bits 23:16 were being used to spare data bits 47:40 and the nibble

	with ECC bits 19:16 (thus data bits 43:40) was found to contain an error, this field would contain a value of 0x'20', not 0x'10'. **All other Error Types:** These bits are reserved, and are 0b'0' when read.
Bits 15:0	***Error Syndrome:*** The format of these bits depends on the Error Type. **Address out of Bounds:** Bits 15:8 - These bits are reserved, and are 0b'0' when read. Bits 7:0 - These bits indicates the memory banks selected (bit n set signifies that memory bank n was selected). **Uncorrectable or Correctable Error:** Bits 15:0 - These bits contain the Error Syndrome (the XOR of the actual and expected values of the ECC bus). For ECC correction types that only use 8 bits (i.e., – SECDED), only the 8 LSB's will be significant.. **All other Error Types:** Bits 15:0 - These bits are reserved, and are 0b'0' when read. The reset value of these bits is 0x'0000'.

4.2.11.7.4.2 Error Log Word 2 (0xDC)

Bits 31:0	***Address of Error:*** These bits indicate the address of the error. For errors on the MCI, only the 16 LSBs are significant, the 16 MSBs is 0x'0000'. For the Data Parity and Byte Enable Parity Errors on the Memory Request Interface, this address is the address of the data beat that the error was on (which in the case of a multi-beat transfer is not necessarily the starting address of the transfer). The reset value of these bits is 0x'0000 0000'.

4.2.11.8 Sparing, Scrubbing, and Initialization Registers

4.2.11.8.1 Memory Bank n Sparing Register (0x80, 0x84, 0x88, 0x8C, 0x90, 0x94, 0x98, 0x9C)

Sparing is available via **ECC** bits not used by the correction type. The **ECC** bits that are used for the 2 spares are shown in the table below. Note that the order of the bit assignment is inverted. Thus, for a bank that uses a 1-byte correction type and 1 spare column, **ECC** bits 23:16 and 7:0 would be used, but bits 15:8 could be left unpopulated.

Spare	ECC Bits
1	23:16
2	15:8

This register supports a read and write column for each spare. The read and write column will typically be the same, except in the case when a spare is being "swapped in" for a column. Then the read spare will be disabled or point to a "reserved" column (i.e., no sparing, the read data will come from the "bad" column) and the write spare will point to the column being spared.

Bits 31:30	***Memory Bank n Sparing Type:*** '00' - No sparing (available with all Correction Types) '01' - 1 spare column (available with Correction Types 00, 01, or 10) '10' - 2 spare columns (available with Correction Types 00 or 01)

	'11' - Reserved The reset value of these bits is 0b'00'.
Bits 29:24	***Memory Bank n Spare Enables*** Bit 29 - Spare 1 Read Enable Bit 28 - Spare 1 Write Enable Bit 27 - Spare 2 Read Enable Bit 26 - Spare 2 Write Enable Bit 25 - Reserved Bit 24 - Reserved '0' - Disabled (no sparing takes place) '1' - Enabled (the spare uses spare assignment bit for byte mapping) The reset value of these bits is 0b'000000'.
Bits 23:20	***Memory Bank n Spare 1 Read Assignment*** 0x'0' → 0x'9' - Byte to spare (data from the spare column is MUXed into this byte). 0x'A' → 0x'F' - Reserved (no sparing will occur) The reset value of these bits is 0x'F'.
Bits 19:16	***Memory Bank n Spare 1 Write Assignment*** 0x'0' → 0x'9' - Byte to spare (data from this byte is written to the spare column). 0x'A' → 0x'F' - Reserved (no sparing will occur) The reset value of these bits is 0x'F'.
Bits 15:12	***Memory Bank n Spare 2 Read Assignment*** 0x'0' → 0x'8' - Byte to spare (data from the spare column is MUXed into this byte). 0x'9' → 0x'F' - Reserved (no sparing will occur) The reset value of these bits is 0x'F'.
Bits 11:8	***Memory Bank n Spare 2 Write Assignment*** 0x'0' → 0x'8' - Byte to spare (data from this byte is written to the spare column). 0x'9' → 0x'F' - Reserved (no sparing will occur) The reset value of these bits is 0x'F'.
Bits 7:0	***Reserved***: These bits are reserved, and return 0b'0' when read.

4.2.11.8.2 Memory Scrubbing Mode Register (0x78)

Scrubbing is a feature that allows the Power PCI to correct errors in memory. In order for a memory to be scrubbed, the Power PCI must be able to:

1. detect errors (either SECDED or Nibble Correct must be used as the ECC code), and
2. write memory (with a corrected set of data and ECC bits).

Bits 31:16	***Reserved***: These bits are reserved, and return 0b'0' when read.
Bits 15:14	***Memory Bank 7 Scrubbing Mode:*** These bits control the scrubbing mode for the memory bank. 0b'00' - Scrubbing Disabled 0b'01' - Passive Scrubbing Enabled (the Power PCI will scrub Memory Request Interface

	Reads). 0b'10' - Active Scrubbing Enabled (the Power PCI will scrub all Memory Request Interface Reads and will insert additional reads to memory when the external memory interface is idle and the chip is in Normal power mode). 0b'11' - Active Scrubbing Enabled during Power Management Mode (the Power PCI will scrub all Memory Request Interface Reads and will insert reads to memory when running in Normal, Doze or Nap Mode). The reset value of these bits is 0b'00'.
Bits 13:12	***Memory Bank 6 Scrubbing Mode***
Bits 11:10	***Memory Bank 5 Scrubbing Mode***
Bits 9:8	***Memory Bank 4 Scrubbing Mode***
Bits 7:6	***Memory Bank 3 Scrubbing Mode***
Bits 5:4	***Memory Bank 2 Scrubbing Mode***
Bits 3:2	***Memory Bank 1 Scrubbing Mode***
Bits 1:0	***Memory Bank 0 Scrubbing Mode***

4.2.11.8.3 Memory Scrubbing Parameters Register (0x7C)

Bits 31:8	***Reserved***: These bits are reserved, and return 0b'0' when read.
Bits 7:0	***Active Scrub Delay:*** During active scrubbing, the Memory Scrub Logic generates addresses for the Power PCI to read. This value represents the number of clock cycles between each read request generated by the Memory Scrub Logic. Note that for small delays, this value will appear to have no effect because the memory access times will determine how quickly memory can be scrubbed. The reset value of these bits is 0x'00'.

4.2.11.9 Initialization Register Descriptions

4.2.11.9.1 Initialization Data Low Register (0x10)

Bits 31:0	***Initialization Data Low:*** This register contains the value that is written to the 32 LSBs of all memory locations between the Initialization Starting and Ending Addresses, inclusive. The reset value of these bits is 0x'0000 0000'.

4.2.11.9.2 Initialization Data High Register (0x14)

Bits 31:0	***Initialization Data High:*** This register contains the value that is written to the 32 MSBs of all memory locations between the Initialization Starting and Ending Addresses, inclusive. The reset value of these bits is 0x'0000 0000'.

4.2.11.9.3 Initialization Starting Address Register (0x18)

Bits 31:5	***Initialization Start Address:*** This value is the first memory location to initialize with the Initialization Data. Note that the minimum granularity for initialization is 32 bytes.

	The reset value of these bits is 0x'0000 0000'.
Bits 4:0	***Reserved***: These bits are reserved, and return 0b'0' when read.

4.2.11.9.4 Initialization Ending Address Register (0x1C)

Bits 31:5	***Initialization Ending Address:*** This value is the last memory location to initialize with the Initialization Data. Note that the minimum granularity for initialization is 32 bytes. The reset value of these bits is 0x'0000 0000'.
Bits 4:1	***Reserved***: These bits are reserved, and return 0b'0' when read.
Bit 0	***Initialization Begin / Initialization in Progress Flag:*** Writing this bit to a 0b'1' causes initialization of the memory locations between the Initialization Starting and Ending Addresses to begin. Once this bit is set to a 0b'1', it remains set until initialization completes. Any subsequent writes to this register are ignored. Power PCI resets this bit to a 0b'0' when initialization completes. 0b'0' - Initialization Complete 0b'1' - Begin Initialization / Initialization in Progress The reset value of this bit is 0b'0'.

4.3 60x Bus Functional Description

The 60x Bus Interface core contained in the Power PCI (shown in Figure 52) transmits and receives data between the 60X Bus used by the RAD750, a memory controller interface internal to the Power PCI, and the Power PCI's internal On Chip Bus (OCB). It provides the bridge from the processor bus to the memory core as well as the bridge from On Chip Bus to the memory core. Any functional unit master on the OCB (PCI, JTAG Slave, EMC) has the capability of accessing the memory core via the 60X core OCB interface. The 60X core ensures processor cache coherency for these accesses by snooping on the processor bus. The 60X core also supports functions such as BIST, and boundary scan capabilities.

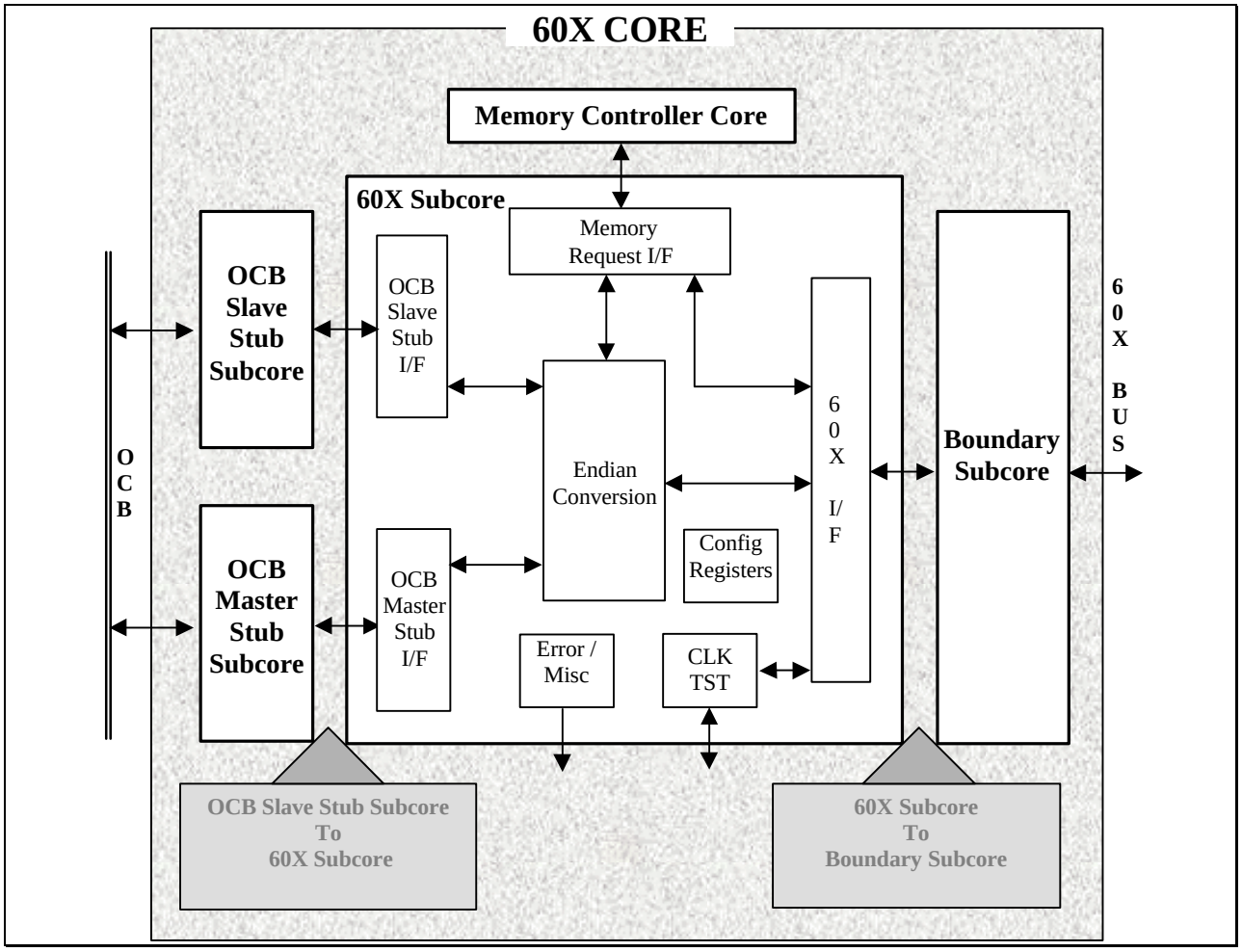


Figure 52: 60X Core functional block diagram

The major functions in the 60X core are:

- Separate Master and Slave interfaces to the internal Power PCI OCB
- 60x Subcore
- 60x Bus Interface
- 60x Bus Master
- 60x Bus Slave
- 60x Bus Arbitrator
- Memory Request Interface
- Configuration Register Block
- Error Interface
- Endian Conversion Logic (see Section 4.9 for details)
- Clock and Test Logic Interface

4.3.1 60x Bus Function Register Matrix

The Power PCI contains a functional capability of communicating with the 60x bus and the RAD750. This function contains a few configuration and status registers for controlling and gatherings status of the 60x bus interface. It also has the responsibility of communicating with the Memory Configuration Interface (see Section 4.2).

Table 13 is a listing of all registers related to the 60x interface in the Power PCI. All defined registers in the Power PCI exist in a single cache line.

Table 13 - 60x Interface Register/Resource Map

Register/ Array/ Command	Address	Access	Reset State	Expected Use	Page
Reserved	0x'0000'	RO	0x'0000 1001'	Diagnostics	
60x Bus Error Status Register	0x'0008'	RO	0x'0000 0000'	Error Analysis	87
60x Bus Error Address Register	0x'000C'	RO	0x'0000 0000'	Error Analysis	87
Error Status Register	0x'0010'	R/W	0x'0000 0000'	Error Analysis	86
Error Status Mask Register	0x'0014'	R/W	0x'0000 0000'	Error Analysis	86
Error Injection Register	0x'0018'	R/W	0x'0000 0000'	Error Analysis	90

4.3.2 60x Bus Function Register Descriptions

The base address for the 60x Bus registers is 0x'BF81 0000'.

4.3.2.1 60x Bus Error Status Address Register (0x08)

Bits 31:8	***Reserved***: These bits are reserved, and return 0b'0' when read.
Bits 7:3	***60x Bus Transfer Type (0:4):*** These bits are a copy of **TT (0:4)** when a 60x Address error is detected. These bits are latched until the **60x MCP Error** bit in the **Error Status Register (0x10)** is cleared. The reset value of these bits is 0b'00000'.
Bits 2:0	***60x Bus Transfer Size (0:2):*** These bits are a copy of **TSIZ (0:2)** when a 60x Address error is detected. These bits are latched until the **60x MCP Error** bit in the **Error Status Register (0x10)** is cleared. The reset value of these bits is 0b'000'.

4.3.2.2 60x Bus Error Address Register (0x0C)

Bits 31:0	***60x Bus Error Address:*** This register contains a copy of **A (0:31)** when a 60x Address error is detected. These bits are latched until the **60x MCP Error** bit in the **Error Status Register (0x10)** is cleared. The reset value of these bits is 0x'0000 0000'.

4.3.2.3 Error Status Register (0x10)

This register contains the error status information. A read from this register does not clear the bits in this register, a write of a 0b'1' to each location is necessary to clear the bit. If an error condition still exists, then the bit will be set again.

Bits 31:5	***Reserved***: These bits are reserved, and return 0b'0' when read.
Bit 4	***OCB MCP:*** This bit signifies that a Machine Check Error has occurred due to an error condition from the Power PCI Internal bus. This bit causes the internal signal **P60x_MCP** to go active to the machine check logic in the Power PCI's Miscellaneous function. See Section 4.8.5.2. The reset value of this bit is 0b'0'.
Bit 3	***60x MCP:***

	This bit signifies that a Machine Check Error has occurred due to an error condition from the 60x Bus. This bit causes internal signal **P60x_MCP** to go active to the machine check logic in the Power PCI's Miscellaneous function. See Section 4.8.5.2. The reset value of this bit is 0b'0'.
Bit 2	***Power PCI Internal Bus Error:*** This bit signifies that an error occurred on the Power PCI Internal Bus Interface by a parity error on a valid transaction on the bus. This bit causes the internal signal **P60x_INT_OUT** to go active to the Interrupt controller in the Power PCI's Miscellaneous function. See Section 4.8.8.1. The reset value of this bit is 0b'0'.
Bit 1	***Address Out of Range:*** This bit is set to a 0b'1' when the Power PCI Internal bus has tried to access an address that is not in the Power PCI function's range of addresses. This bit causes the internal signal **P60x_INT_OUT** to go active to the Interrupt controller in the Power PCI's Miscellaneous function. See Section 4.8.8.1. The reset value of this bit is 0b'0'.
Bit 0	***Memory Controller Interface Error:*** This bit signifies that a data parity error was detected or an error signal was asserted during a valid transaction on the Memory Controller Interface. This bit causes the internal signal **P60x_INT_OUT** to go active to the Interrupt controller in the Power PCI's Miscellaneous function. See Section 4.8.8.1. The reset value of this bit is 0b'0'.

4.3.2.4 Error Status Mask Register (0x14)

This register contains the masks for the **Error Status Register (0x10)**. A 0b'1' in the corresponding bit location masks the status bit from setting its associated output active. The bit will still be set in the **Error Status Register (0x10)**. The Reset Condition is that nothing will be masked.

Bits 31:5	***Reserved***: These bits are reserved, and return 0b'0' when read.
Bit 4	***OCB MCP Mask:*** This bit masks the ***OCB MCP*** bit in the **Error Status Register (0x10)**. Setting this bit to a 0b'1' causes the ***OCB MCP*** bit from causing **P60x_MCP** to go active. The reset value of this bit is 0b'0'.
Bit 3	***60x MCP Mask:*** This bit masks the ***60x MCP*** bit in the **Error Status Register (0x10)**. Setting this bit to a 0b'1' causes the ***60x MCP*** bit from causing **P60x_MCP** to go active. The reset value of this bit is 0b'0'.
Bit 2	***OCB Interface Error Mask:*** This bit masks the ***OCB Interface Error*** bit in the **Error Status Register (0x10)**. Setting this bit to a 0b'1' causes the ***OCB Interface Error*** bit from causing **P60x_INT_OUT** to go active. The reset value of this bit is 0b'0'.

Bit 1	***Address Out of Range Mask:***: This bit masks the ***Address Out of Range*** bit in the **Error Status Register (0x10)**. Setting this bit to a 0b'1' causes the ***Address Out of Range*** bit from causing **P60x_INT_OUT** to go active. The reset value of this bit is 0b'0'.
Bit 0	***Memory Controller Interface Error Mask:*** This bit masks the ***Memory Controller Interface Error*** bit in the **Error Status Register (0x10)**. Setting this bit to a 0b'1' causes the ***Memory Controller Interface Error*** bit from causing **P60x_INT_OUT** to go active. The reset value of this bit is 0b'0'.

4.3.2.5 Error Injection Register (0x18)

Bits 31:2	***Reserved***: These bits are reserved, and return 0b'0' when read.
Bit 1	***60x Data Bus Error Injection:*** When this bit is set in conjunction with the **P60x_INJECT_EN** signal, the parity bit transmitted by the Power PCI on the 60x Data Bus is inverted. The reset value of this bit is 0b'0'.
Bit 0	***60x Address Bus Error Injection:*** When this bit is set in conjunction with the **P60x_INJECT_EN** signal, the parity bit transmitted by the Power PCI on the 60x Address Bus is inverted. The reset value of this bit is 0b'0'.

4.4 UART Operation

Note: The MPC-106 does not support a UART.

The Power PCI contains a 16550 compatible UART interface. The ASIC itself support the full suite of I/O signals including: Serial Data In, Serial Data Out, Request to Send (RTS), Clear to Send (CTS), Data Set Ready (DSR), Data Carrier Detect (DCD), Ring Indicator (RI), Data Terminal Ready (DTR) and two general purpose output signals. However, four of these signals (DSR, DCD, RI, and DTR) are not brought off the SFC card.

Details on control and usage of the UART function in the Power PCI ASIC can be found in the X2000 SFC Software User's Manual.

4.4.1 Baud Rate Generation

The UART function in the Power PCI contains a Baud Rate Generator that takes a clock input and divide it by any divisor from 2 to 2^{16}-1. This clock is the RTC input to the chip, and for the SFC is the 33 MHz system clock divided by 4, 8, 16, or 32. The output frequency of the Baud Generator is 16 x the Baud Rate [divisor # = (frequency input) / (baud rate x 16)]. The divisor is stored in two 8-bit latches called Divisor High Register and Divisor Low. These divisor latches must be loaded during initialization to ensure proper operation of the Baud Rate Generator. A divisor of zero is not recommended. Upon loading either of these registers, a 16-bit Baud rate counter is immediately loaded. Table 14 below shows an example of desired baud rates and their necessary divisors for the four potential input clock frequencies. The accuracy of the desired baud rate is dependent on the input frequency chosen and the baud rate. The formula for percent error is as follows: (Input frequency / Divisor *16) / Desired Baud Rate = Percent Error.

Table 14 - UART Baud Rate Programming

BAUD RATE	8.25 MHz		4.125 MHz		2.0625 MHz		1.03125 MHz	
	Divisor (Decimal)	% Error	Divisor (Decimal)	% Error	Divisor (Decimal)	% Error	Divisor (Decimal)	% Error
50	10312	0.005	5156	0.005	2578	0.005	1289	0.005
75	6875	0.000	3437	0.015	1719	0.015	859	0.044
110	4687	0.011	2344	0.011	1172	0.011	586	0.011
134.5	3834	0.009	1917	0.009	958	0.043	479	0.043
150	3437	0.015	1719	0.015	859	0.044	430	0.073
300	1719	0.015	859	0.044	430	0.073	215	0.073
600	859	0.044	430	0.073	215	0.073	107	0.394
1200	430	0.073	215	0.073	107	0.394	54	0.535
1800	286	0.160	143	0.160	72	0.535	36	0.535
2000	258	0.073	129	0.073	64	0.708	32	0.708
2400	215	0.073	107	0.394	54	0.535	27	0.535
3600	143	0.160	72	0.535	36	0.535	18	0.535
4800	107	0.394	54	0.535	27	0.535	13	3.290
7200	72	0.535	36	0.535	18	0.535	9	0.535
9600	54	0.535	27	0.535	13	3.290	7	4.088
19200	27	0.535	13	3.290	7	4.088	3	11.898
28800	18	0.535	9	0.535	4	11.898	2	11.898
38400	13	3.290	7	4.088	3	11.898	2	16.077
56000	9	2.307	5	7.924	2	15.095	1	15.095
128000	4	0.708	2	0.708	1	0.708	n/a	n/a
256000	2	0.708	1	0.708	n/a	n/a	n/a	n/a

4.4.2 UART Register Matrix

Table 15 lists all registers internal to the UART with address information, etc. The upper 16 bits of the address are parameterizable, for the Power PCI the base address for the UART registers is 0x'BF82 0000'. The lower 16 bits are decoded as the physical address. All of these registers are parity protected.

Table 15 - UART Register/Resource Map

Register/ Array/ Command	Address	Access	Reset State	Expected Use	Page
Receiver Buffer Register (*DLAB = '0')	0x' 0000'	RO	0b'0000 0000'	Read FIFO	93
Transmitter Holding Register (*DLAB = '0')	0x'0000'	WO	0b'0000 0000'	Write FIFO	93
Interrupt Enable Register (*DLAB = '0')	0x'0001'	R/W	0b'0000 0000'	Configuration	93
Divisor Low Register (*DLAB = '1')	0x'0000'	R/W	0b'0000 0000'	Configuration	98
Divisor High Register (*DLAB = '1')	0x'0001'	R/W	0b'0000 0000'	Configuration	98
Interrupt Identification Register	0x'0002'	RO	0b'0000 0001'	Configuration	94
FIFO Control Register	0x'0002'	WO	0b'0000 0000'	Configuration	95
Line Control Register	0x'0003'	R/W	0b'0000 0000'	Configuration	95
MODEM Control Register	0x'0004'	R/W	0b'0000 0000'	Configuration	96

Register/ Array/ Command	Address	Access	Reset State	Expected Use	Page
Line Status Register	0x'0005'	RO	0b'0110 0000'	Status	97
MODEM Status Register	0x'0006'	RO	0b'0000 0000'	Status	98
Scratch Pad Register	0x'0007'	R/W	0b'0000 0000'	Status	99
Error Status Register	0x'0008'	RO	0b'0000 1000'	Status	99

**DLAB = Line Control Register bit(7)*

4.4.3 UART Register Descriptions

4.4.3.1 Receiver Buffer Register (0x00)

The UART supports an 8-bit register containing the oldest data value of in the Receiver FIFO. This register is read only and shares the same address as the Transmitter Holding Register when the Divisor Latch Access bit in the Line Control Register is 0b'0'. The bit definitions of the Receiver Buffer register is as follows:

Bits 7:0	***Receiver Buffer:*** These bits represent the oldest 8-bit data value that stored in the Receiver FIFO.

4.4.3.2 Transmitter Holding Register (0x00)

The UART supports an 8-bit register containing the value of data that is currently being written out of the Transmitter FIFO. This register is written only and shares the same address as the Receiver Buffer Register when the Divisor Latch Access bit in the Line Control Register is 0b'0'. The bit definitions of the Transmitter Holding Register is as follows:

Bits 7:0	***Transmitter Holding Register***: These bits represent the 8-bit data value that is being written out of the Transmitter FIFO.

4.4.3.3 Interrupt Enable Register (0x01)

The UART controls the assertion of interrupts with the Interrupt Enable Register. The UART masks the interrupt associated with a particular condition if the corresponding enable bit is set to a 0b'0'. The UART asserts an interrupt for a particular condition if the corresponding enable bit is set to a 0b'1'. The bit definitions is as follows:

Bits 7:6	***Reserved***: These bits are reserved, and return 0b'0' when read.
Bit 5	***All Error Status:*** This bit controls the assertion of an interrupt when the Error Status bit is flagged in the Interrupt Identification Register.
Bit 4	***Error Status(No FIFO Empty or Full):*** This bit controls the assertion of an interrupt when the Error Status bit is flagged in the Interrupt Identification Register. All but the Transmitter FIFO Full and Receive FIFO Empty will trigger this interrupt.
Bit 3	***MODEM Status:*** This bit controls the assertion of an interrupt when the MODEM Status bit is flagged in the Interrupt Identification Register.

Bit 2	***Receiver Line Status:*** This bit controls the assertion of an interrupt when the Receiver Line Status bit is flagged in the Interrupt Identification Register.
Bit 1	***Transmitter Holding Register Empty:*** This bit controls the assertion of an interrupt when the Transmitter Holding Register bit is flagged in the Interrupt Identification Register.
Bit 0	***Received Data Available:*** This bit controls the assertion of an interrupt when the Received Data Available bit is flagged in the Interrupt Identification Register.

4.4.3.4 Interrupt Identification Register (0x02)

The UART prioritizes interrupts into four levels and records these in the Interrupt Identification Register. The five levels of interrupt conditions in order of priority are Error Status, Receiver Line Status; Received Data Ready; Transmitter Holding Register Empty, and MODEM status. When this register is accessed, the UART freezes all interrupts and indicates the highest priority pending interrupt. This means while the Power PCI internal read access is occurring, the UART records new interrupts, but does not change its current indication until the indicated interrupt is serviced. This register is read only and shares the same address as the FIFO Control Register.

Bits 7:6	***FIFO Mode:*** These bits are reserved, and are 0b'1' when the **FIFO Control Register** bit 0 is a 0b'1'.
Bits 5:4	***Reserved***: These bits are reserved, and return 0b'0' when read.
Bits 3:0	***Interrupt Identification:*** When: '0001' - no interrupts '1110' - Interrupt of Most Highest Priority - ***Error Status*** - when an error in the Error Status Register has been flagged. The interrupt is reset when the Error Status Register has been read or the error has been cleared. '0110' - Interrupt of Highest Priority - ***Receiver Line Status*** - when an Overrun Error, Parity Error, Framing Error, or Break Interrupt occur. The interrupt is reset when the Line Status Register has been read. '0100' - Interrupt of Second Priority - ***Received Data Available*** - when the Receiver Data is available or Trigger Level Reached. The interrupt is reset by reading the Receiver Buffer Register or when the FIFO drops below the Trigger Level. '1100' - Interrupt of Second Priority - ***Character Timeout Indication*** - when no characters have been removed from or input to the receiver FIFO during the last 4 character times and there is at least 1 character in it during this time. The interrupt is reset by reading the Receiver Buffer Register. '0010' - Interrupt of Third Priority - ***Transmitter Holding Register Empty*** - when the Transmitter Holding Register is empty. Reading the Interrupt Identification Register or writing the Transmitter Holding Register resets the interrupt. '0000' - Interrupt of Fourth Priority- ***MODEM Status*** - when CTS (Clear to Send), DSR (Data Set Ready), RI (Ring Indicator), or DCD (Data Carrier Detected) have been flagged. Reading the MODEM Status Register resets the interrupt.

4.4.3.5 FIFO Control Register (0x02)

The UART supports an 8-bit register that is used to enable the FIFOs, clear the FIFOs, set the Receiver FIFO trigger level, and select the type of DMA signaling. This register is write only and shares the same address as the Interrupt Identification Register. The bit definitions of the FIFO Control Register is as follows:

Bits 7:6	***Trigger Level:*** The decode of these bits represents the Receiver FIFO Trigger Level. Every time that location has the right amount of valid data an interrupt will be flagged if the Received Data Available bit in the Interrupt Enable Register is set to logic 0b'1'. '00' - Trigger Level is 1 '01' - Trigger Level is 4 '10' - Trigger Level is 8 '11' - Trigger Level is 14
Bits 5:3	***Reserved***: These bits are reserved, and return 0b'0' when read.
Bit 2	***Transmit FIFO Clear:*** Writing this bit to logic 0b'1' clears all bytes in the Transmit FIFO and reset its counter logic to 0b'0'. The transmit shift register will not be cleared. This bit is self-clearing.
Bit 1	***Receiver FIFO Clear:*** Writing this bit to logic 0b'1' clears all bytes in the Receiver FIFO and reset its counter logic to 0b'0'. The receiver shift register will not be cleared. This bit is self-clearing.
Bit 0	***FIFO Enable:*** Writing this bit to logic 0b'1' enables both Transmit and Receiver FIFOs. Writing this bit back to logic 0b'0' will clear all bytes in both FIFOs. This bit is a 0b'1' when writing all other bits in this register, otherwise they will not be programmed.

4.4.3.6 Line Control Register (0x03)

The UART supports an 8-bit register that specifies the format of the asynchronous data communications exchange and set the Divisor Latch Access bit. The bit definitions of the Line Control register is as follows:

Bit 7	***Register Access:*** This bit is the Divisor Latch Access bit (DLAB). It must be set to logic 0b'1' to access the Divisor Latches of the Baud Rate Generator during a read or write operation. It must be set to logic 0b'0' to access the Receiver Buffer, the Transmitter Holding Register, or the Interrupt Enable Register.
Bit 6	***Break Control:*** This bit is the Break Control bit. It causes a break condition to be transmitted to the peripheral device. When it is set to a logic 0b'1', the serial out (SOUT) port is forced to the spacing (logic 0b'0') state. Setting this bit to logic 0b'0' disables the break. The break control bit acts only on SOUT and has no affect on the Transmitter Logic.
Bit 5	***Stick Parity:*** This bit represents Stick Parity. When bits 3, 4, and 5 are logic 0b'1', the parity bit is transmitted and checked as logic 0b'0'. If bits 3 and 5 are logic 0b'1' and bit 4 is logic 0b'0', then the parity bit is transmitted and checked as logic 0b'1'. When this bit is logic 0b'0', Stick Parity is disabled.

Bit 4	***Even Parity:*** This bit represents Even Parity. When bit 3 is logic 0b'1' and bit 4 is logic 0b'0', an odd number of logic 0b'1's' is transmitted or checked in the data word bits and parity bit. When bit 3 is logic 0b'1' and bit 4 is logic 0b'1', an even number of logic 0b'1's' is transmitted or checked.
Bit 3	***Enable Parity:*** This bit represents Parity Enable. When bit 3 is a logic 0b'1' a parity bit is generated or checked between the last data word bit and the transmitted Stop bit of the serial data.
Bit 2	***Number of Stop Bits:*** This bit represents the number of stop bits. If bit 2 is logic 0b'0', one stop bit is generated in the transmitted data. If bit 2 is logic 0b'1' when a 5-bit word length is selected via bits 1 and 0, one and a half stop bits is generated. Otherwise when bit 2 is logic 0b'1', two stop bits is generated. The Receiver checks the first stop bit only, regardless of how many stop bits are selected.
Bits 1:0	***Character Length:*** These bits represent the length of the transmitted or received word. The decoding of these bits is as follows: '00' - 5 bits '01' - 6 bits '10' - 7 bits '11' - 8 bits

4.4.3.7 MODEM Control Register (0x04)

The UART supports a register that will control the interface with the MODEM or peripheral device. The bit definitions of the MODEM Control Register are as follows.

Bits 7:5	***Reserved***: These bits are reserved, and return 0b'0' when read.
Bit 4	***Diagnostic Mode:*** This bit provides a local loopback feature for diagnostic testing of the UART. When this bit is a logic 0b'1', the serial out (SOUT) port is set to logic 0b'1', the serial in (SIN) port is disconnected, the output of the Transmitter Shift Register is "looped back" into the Receiver Shift Register input, the four MODEM control inputs (DSR, CTS, RI, DCD) is disconnected, the four outputs is internally connected to the four MODEM control inputs (as described in Section 4.4.3.9), and the MODEM control output pins is forced to there inactive high state. In diagnostic mode, data that is transmitted will automatically be received.
Bit 3	***Output 2 Enable:*** This bit controls the Output 2 (OUT2) signal, which is an auxiliary user-designate output. When this bit is logic 0b'1', OUT2 is forced to logic 0b'0'. Inversely, when this bit is logic 0b'0', OUT2 is forced to logic 0b'1'. This signal drives the INTA# on the CompactPCI bus.
Bit 2	***Output 1 Enable:*** This bit controls the Output 1 (OUT1) signal, which is an auxiliary user-designate output. When this bit is logic 0b'1', OUT1 is forced to logic 0b'0'. Inversely, when this bit is logic 0b'0', OUT1 is forced to logic 0b'1.
Bit 1	***Request To Send:*** This bit controls RTS, Request to Send. When this bit is logic 0b'1', the RTS output is forced to logic 0b'0'. When this bit is reset to 0b'0', RTS is set to a 0b'1'.

Bit 0	***Data Terminal Ready:*** This bit controls DTR, Data Terminal Ready. When this bit is logic 0b'1', the DTR output is forced to logic 0b'0'. When this bit is reset to 0b'0', DTR is set to a 0b'1'.

4.4.3.8 Line Status Register (0x05)

The UART supports an 8-bit register that provides status information to the Power PCI internal status concerning the data transfer. The bit definitions of the Line Status register is as follows:

Bit 7	***Parity Frame Break Error:*** This bit is set to logic 0b'1' when there is at least one Parity Error, Frame Error, or Break Indication. This bit resets when the Line Status Register is read and there are no more subsequent errors in the FIFO.
Bit 6	***Transmitter Empty:*** This bit is set to logic 0b'1' to indicate that the Transmitter Holding Register and Transmitter Shift Register are empty. This bit resets when either register contains a data character.
Bit 5	***Transmitter Holding Register Empty:*** This bit is set to logic 0b'1' when the Transmit FIFO is empty. This bit resets when at least 1 byte is written to the FIFO.
Bit 4	***Break Interrupt Indicator:*** This bit is set to logic 0b'1' whenever the received data input is held in the Spacing state for longer than a full word transmission time (start bit + data bits + parity + stop bits). This bit resets when the Line Status Register is read or data has been pulled from the FIFO. When this register is read, this bit corresponds with the oldest data value in the Receive FIFO. When a break occurs, only one zero character is loaded into the FIFO. The next character transfer is enabled after SIN goes to the marking state and receives the next valid start bit.
Bit 3	***Framing Error:*** This bit indicates that the received data did not have a valid stop bit. This bit is set to logic 0b'1' if the Stop bit following the last data bit or parity bit was detected as a logic 0b'0' bit. This bit resets whenever the Line Status Register is read or data has been pulled from the FIFO. When this register is read, this bit corresponds with the oldest data value in the Receive FIFO. The UART will attempt to resynchronize after a framing error by assuming that the framing error was due to the next start bit; thus sampling the start bit twice and then proceeds with the data.
Bit 2	***Parity Error:*** This bit indicates that the received data did not have the correct even or odd parity, as selected by the even-odd parity bit in bit 4 of the Line Control Register. This bit is set to logic 0b'1' when it has detected bad parity and is reset when the Line Status Register has been read or data has been pulled from the FIFO. When this register is read, this bit corresponds with the oldest data value in the Receive FIFO.
Bit 1	***Overrun Error:*** This bit indicates that the data was received and the FIFO is full, thus overrunning the data. This bit is set to logic 0b'1' when the FIFO has been filled and the next character is completely received in the shift register. The data in the shift register will be over written. When this register is read, this bit is set when the error occurs. This bit is reset when the Line Status Register is read or the FIFO is read.

Bit 0	***Data Ready:*** This bit indicates that valid data was stored in the FIFO. This bit is logic 0b'1' when a complete incoming character has been received and transfers to the Receive FIFO. This bit resets by reading all of the data in the Receiver Buffer Register or FIFO.

4.4.3.9 MODEM Status Register (0x06)

This register provides the current state of the control lines from the MODEM to the CPU.

Bit 7	***Data Carrier Detect:*** This bit is the inverse of the Data Carrier Detect (DCD) signal. If bit 4 in the MODEM Control Register is set to logic 0b'1' then this bit is equivalent to the OUT2 in the Modem Control Register.
Bit 6	***Ring Indicator:*** This bit is the inverse of the Ring Indicator (RI) signal. If bit 4 in the MODEM Control Register is set to logic 0b'1' then this bit is equivalent to the OUT1 in the Modem Control Register.
Bit 5	***Data Set Ready:*** This bit is the inverse of the Data Set Ready (DSR) signal. If bit 4 in the MODEM Control Register is set to a logic 0b'1' then this bit is equivalent to the Data Terminal Ready (DTR) in the Modem Control Register.
Bit 4	***Clear to Send:*** This bit is the inverse of the Clear to Send (CTS) signal. If bit 4 in the MODEM Control Register is set to a logic 0b'1' then this bit is equivalent to the Request to Send (RTS) in the Modem Control Register.
Bit 3	***Delta Data Carrier Detect:*** This bit is the Delta Data Carrier Detect (DDCD). This bit is set when DCD has changed states since the last time it was read.
Bit 2	***Trailing Edge of Ring Indicator:*** This bit is the Trailing Edge of the Ring Indicator (TERI). This bit is set when RI has changed from a low to high state.
Bit 1	***Delta Data Set Ready:*** This bit is the Delta Data Set Ready (DDSR). This bit is set when that DSR has changed states since the last time it was read.
Bit 0	***Delta Clear to Send:*** This bit is the Delta Clear to Send (DCTS). This bit is set when CTS has changed states since the last time it was read.

4.4.3.10 Divisor Low Register (0x00)

The UART supports an 8-bit register that will represent the Least Significant Byte of the Baud Rate Generator Divisor Latch.

Bits 7:0	***Divisor Low Register:*** These bits represent the low order 8 bits of the baud rate generator divisor latches.

4.4.3.11 Divisor High Register (0x01)

The UART supports an 8-bit register that will represent the Most Significant Byte of the Baud Rate Generator Divisor Latch.

Bits 7:0	***Divisor High Register:*** These bits represent the high order 8 bits of the baud rate generator divisor latches.

4.4.3.12 Scratch Pad Register (0x07)

The UART supports an 8-bit register to be used as a scratch pad.

Bits 7:0	***Scratch Pad Register:*** These bits represent a scratch pad for the programmer to store data temporarily.

4.4.3.13 Error Status Register (0x08)

In addition to registers defined in the 16550, the UART supports a register that reports other status errors.

Bit 7	***Address Phase Parity Error:*** This bit is set to a 0b'1' when there is a parity error across the address, command, and transfer size inputs. This bit is cleared when the register is read.
Bit 6	***Write Data Parity Error:*** This bit is set to a 0b'1' when there is a parity error across the write data in bus and write data error bit or a parity error across the byte enables. This bit is cleared when the register is read.
Bit 5	***Transmit FIFO Full:*** This bit is set to a 0b'1' when the Transmit FIFO is full. This bit is cleared when the FIFO is no longer full.
Bit 4	***Transmit FIFO Overrun:*** This bit is set to a 0b'1' when the Transmit FIFO has overrun, similar to the Receive FIFO overrun described in Section 4.4.3.8. This bit is cleared when the register is read.
Bit 3	***Receive FIFO Empty:*** This bit is set to a 0b'1' when the Receive FIFO is empty. This bit is cleared when the FIFO is no longer Empty.
Bit 2	***Read From Empty Receive FIFO:*** This bit is set to a 0b'1' when there has been an extra read to an empty Receive FIFO. This bit is cleared when the register is read.
Bit 1	***Address Out of Range:*** This bit is set to a 0b'1' when the Power PCI internally has tried to access an address that is not in the UART's range of addresses. This bit is cleared when the register is read.
Bit 0	***Critical Error Status:*** This bit stores the inverse of the critical error signal value (UART_CR_ERR). This bit is set to 0b'0' on reset.

4.5 JTAG Master and Slave Interface Functions

The PowerPCI contains two JTAG master and one JTAG slave functions. One of these masters is capable of being combined off chip with the JTAG slave function for testing with a JTAG probe as shown in Figure 53. Other than the combining of this interface with the JTAG slave, the two JTAG Master cores are identical in functionality and implementation, as described in this section. The JTAG Master function provides a Test Access Port (TAP) and associated logic to support a fully compliant IEEE Standard 1149.1a JTAG Master interface.

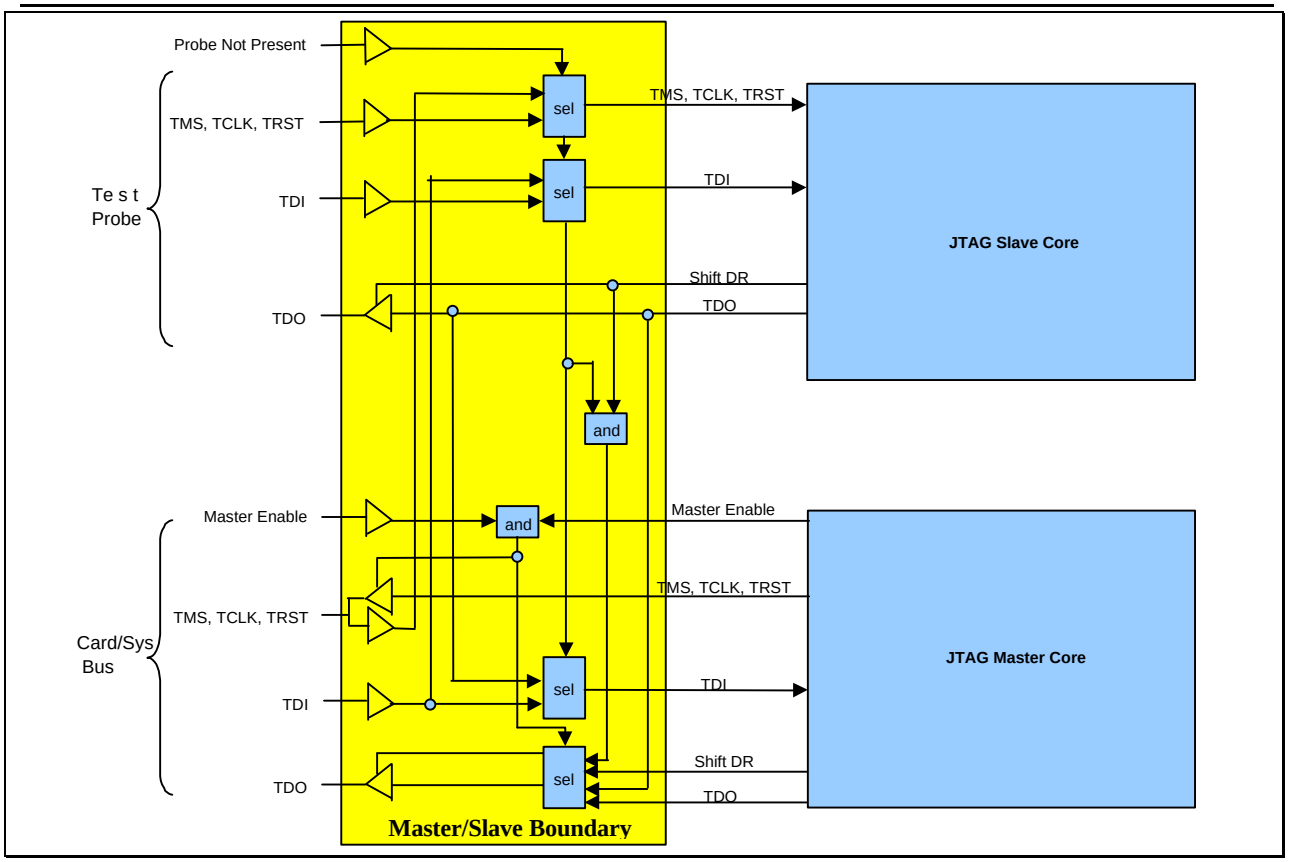


Figure 53: PowerPCI Combination JTAG Master/Slave Port with Probe Support

4.5.1 JTAG Master Function

The JTAG Master Function drives a JTAG 1149.1a Standard Master interface. The JTAG Master Function is shown in Figure 54 and contains a TAP Controller state machine, JTAG interface control registers.

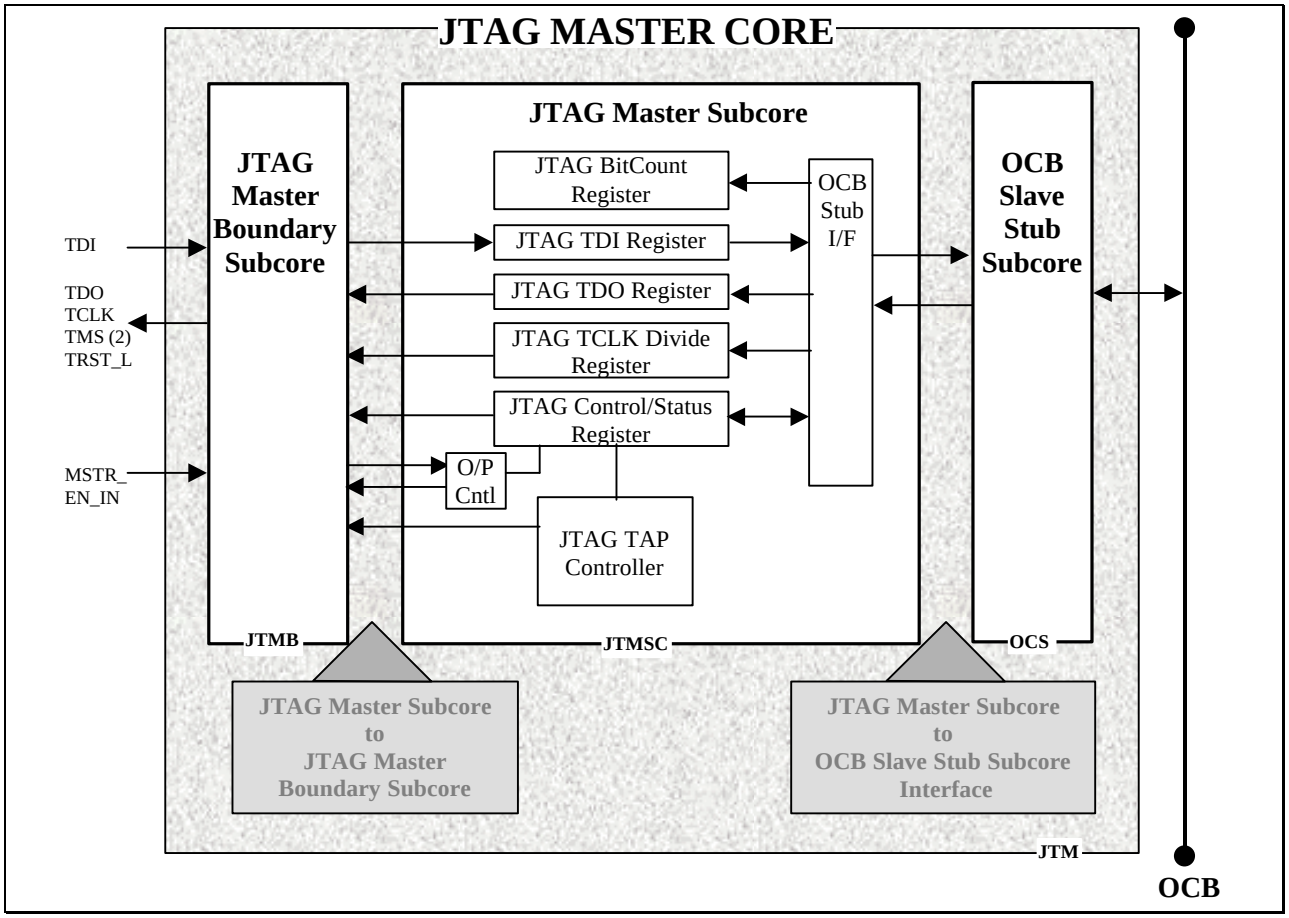


Figure 54: JTAG Master Function Functional Block Diagram

The TAP Controller state machine conforms to the JTAG IEEE 1149.1a Standard and drives the JTAG bus signals according to the state transitions defined by that standard and shown in Figure 55.

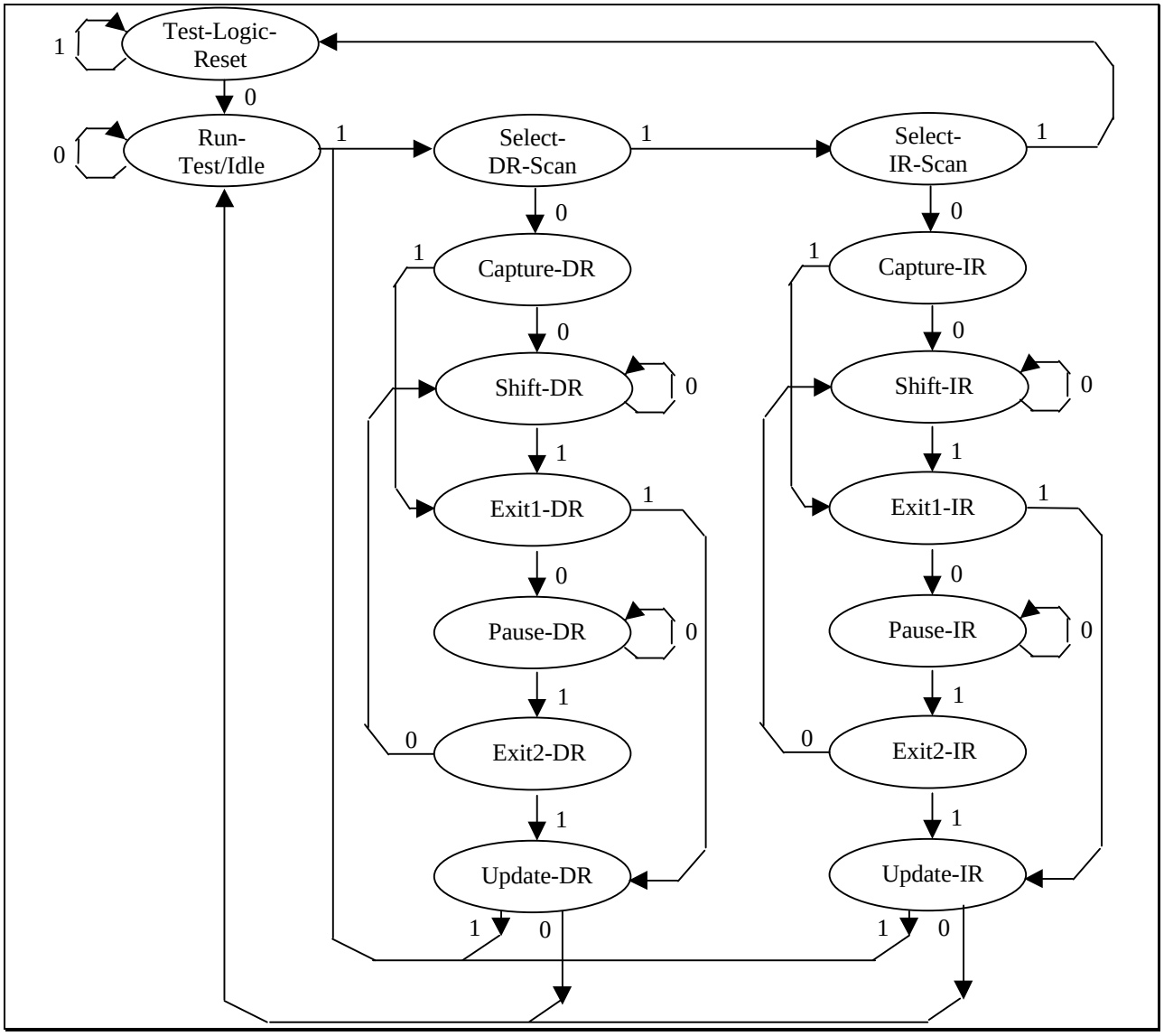


Figure 55: TAP Controller State Diagram

The following five 32-bit JTAG interface control registers are provided to control the JTAG Master operation: the **JTAG TCLK Divide Register (0x10)**, **JTAG *Bit Count* Register (0x04), JTAG TDO Register (0x0C), JTAG TDI Register (0x08),** and the **JTAG Control/Status Register (0x00).** These are further defined in Section 4.5.3.

Two **JTMSC_JTMB_TMS** signals are provided to allow multiple drops on the JTAG string, and they are under control of the ***Interface Select*** bit of the **JTAG *Bit Count* Register (0x04).**

4.5.1.1 JTAG Master Operations

A JTAG Scan operation consists of a number of bits to be scanned on the JTAG bus as defined by the ***Bit Count*** field. Since the maximum data bits held by the **JTAG TDI Register (0x08)** and **JTAG TDO Register (0x0C)** is 32, it may take several write/read operations of these registers to exhaust the ***Bit Count***. The following sequence describes the proper handshakes between a Power PCI and the JTAG Master Function in order to complete a JTAG scan operation.

State	Description
Tap Reset	When the Power PCI sets the ***Tap Reset*** bit of the **JTAG Control/Status Register (0x00),** the JTAG Master function activates **JTM_TRST_L** and drives **JTM_TMS**(n) to a logic 0b'1', thus forcing the JTAG bus to the Test-Logic-Reset state. The JTAG Master function also clears the ***Bit Count Exhausted, Scan in Process, Handshake Error,*** and ***OCB Access/Data Error*** bits of the **JTAG Control/Status Register (0x00)** indicating that any scan activity that may have been in process has been asynchronously stopped. When the ***Tap Reset*** bit is cleared, the JTAG Master function releases **JTM_TRST_L,** but holds the JTAG bus in the Test-Logic-Reset state until the next write to the **JTAG TDO Register (0x0C).**
Initialization	Before starting any JTAG scan operation the Power PCI ensures that it has properly set up the **JTAG TCLK Divide Register (0x10)** and the **JTAG *Bit Count* Register (0x04)**.
Write JTAG TDO Register (0x0C) (*Bit Count* > 0)	The Power PCI writes to the **JTAG TDO Register (0x0C)** the data it wants to scan out. This write causes the JTAG Master function to set the ***Scan in Process*** bit, clear the ***Bit Count Exhausted*** bit, and sequence the JTAG bus to the Shift-DR or Shift-IR state (depending on the ***Data/Instruction Select*** bit). The JTAG Master function then scans data from the **JTAG TDO Register (0x0C)** onto **JTM_TDO** and scans data in from **JTM_TDI** into the **JTAG TDI Register (0x08).** For each bit scanned, the JTAG Master function decrements the ***Bit Count*** field of the **JTAG *Bit Count* Register (0x04)**. The JTAG Master function continues scanning bits until 32 have been scanned or the ***Bit Count*** field reaches zero, whichever occurs first.
Poll JTAG Control/Status Register (0x00)	After writing the **JTAG TDO Register (0x0C)** the Power PCI polls the **JTAG Control/Status Register (0x00)**. The JTAG Master function clears the ***Scan in Process*** bit once it scans all 32 bits of **JTAG TDI Register (0x08)** and **JTAG TDO Register (0x0C)** or the ***Bit Count*** field is exhausted, whichever occurs first. If the ***Bit Count*** has been exhausted the JTAG Master function also sets the ***Bit Count Exhausted*** bit.
Read JTAG TDI Register (0x08)	When the JTAG Master function clears the ***Scan in Process*** bit, it indicates that the **JTAG TDI Register (0x08)** contains valid data to be read.
Bit count not exhausted	If the ***Bit Count*** field has not been decremented to zero, as indicated by a clear ***Bit Count Exhausted*** bit, the Power PCI can continue scanning more data by returning to the control sequence "**Write JTAG TDO Register (0x0C)**", above.
Bit count exhausted	If the ***Bit Count*** field has been decremented to zero, as indicated by a set ***Bit Count Exhausted*** bit, the current scan operation is complete and the controlling agent can begin another scan by returning to the control sequence "**Initialization**" above.
Special Case: Write JTAG TDO Register (0x0C) (*Bit Count* = 0)	If a write occurs to the **JTAG TDO Register (0x0C),** and the ***Bit Count*** field is zero, the JTAG Master function a) sets the ***Scan in Process*** bit and clears the ***Bit Count Exhausted*** bit, b) sequences the JTAG bus through the Capture and Update states, skipping the Shift state (thus scanning no data), and c) sequences the JTAG bus back to the Run-Test-Idle state, clearing the ***Scan in Process*** bit and setting the ***Bit Count Exhausted*** bit.
Self Checking Error indicated	If the Power PCI detects that the ***Self Checking Error*** bit is set while polling the **JTAG Control/Status Register (0x00),** it should set the ***Tap Reset*** bit and re-

State	Description
	initialize all the JTAG Interface Registers before continuing scan operations.

4.5.1.2 JTAG Master Bus States

This section shows the JTAG defined states that the JTAG Master function sequences the JTAG bus through while performing a scan operation. Reference the IEEE 1149.1a Standard state diagram in Figure 9. If the **JTMSC_JTMB_MASTER_EN** signal is not active and the ***Scan in Process*** bit is set, the JTAG Master TAP Controller does not sequence states from Test-Logic-Reset to Run-Test-Idle and Run-Test-Idle to Select-DR-Scan until the JTAG Master is enabled.

- From Test-Logic-Reset or Run-Test-Idle states with ***Bit Count*** field > 0

The JTAG Master function begins any scan operation from either the Test-Logic-Reset or Run-Test-Idle states (Test-Logic-Reset if Tap Reset has been active). After a write to the **JTAG TDO Register (0x0C),** when the JTAG bus is in the Test-Logic-Reset or Run-Test-Idle states, the JTAG Master function sequences the JTAG bus to the Shift state, provided the ***Bit Count*** field is > 0 as shown below:

Data Selected	**Instruction Selected**
Run-Test-Idle	Run-Test-Idle
Select-DR-Scan	Select-DR-Scan
Capture-DR	Select-IR-Scan
Shift-DR	Capture-IR
	Shift-IR

Keep in mind that the write to the **JTAG TDO Register (0x0C)** caused the ***Scan in Process*** bit to be set.

- From Shift state with **JTAG *Bit Count* Register (0x04)** > 32

Sequencing from the Shift state depends on the ***Bit Count*** field value. If the ***Bit Count*** field is > 32 then the JTAG Master function repeats the Shift state 32 times and sequences to the Pause state as shown below:

Data Selected	**Instruction Selected**
Shift-DR (32x)	Shift-IR (32x)
Exit1-DR	Exit1-IR
Pause-DR	Pause-IR

At this point the JTAG Master function has shifted all the data out of the **JTAG TDO Register (0x0C)** but has not exhausted the ***Bit Count*** field. Thus the ***Scan in Process*** bit is cleared and the JTAG Master function waits for more data to be written to the **JTAG TDO Register (0x0C)**.

- From Shift state with ***Bit Count*** field <= 32

Sequencing from the Shift state depends on the ***Bit Count*** field value. If the ***Bit Count*** field is <= 32, then the JTAG Master function repeats the Shift state ***Bit Count*** times and sequences to the Run-Test-Idle state as shown below:

Data Selected	**Instruction Selected**
Shift-DR (BCx)	Shift-IR (BCx)
Exit1-DR	Exit1-IR
Update-DR	Update-IR
Run-Test-Idle	Run-Test-Idle

The JTAG Master function has exhausted the ***Bit Count*** field and thus completed the scan operation. It clears the ***Scan in Process*** bit and sets the ***Bit Count Exhausted*** bit, returning to the Run-Test-Idle state to wait for another scan operation.

- From Pause state

After a write to the **JTAG TDO Register (0x0C)** when the JTAG bus is in the Pause state, the JTAG Master function sequences to the Shift state as shown below:

Data Selected	Instruction Selected
Pause-DR	Pause-IR
Exit2-DR	Exit2-IR
Shift-DR	Shift-IR

- From Test-Logic-Reset or Run-Test-Idle states with ***Bit Count*** field = 0

If a write to the **JTAG TDO Register (0x0C)** occurs with the ***Bit Count*** field set to zero, then the JTAG Master function sequences the JTAG bus through the Capture and Update states, returning to the Run-Test-Idle state as shown below:

Data Selected	Instruction Selected
Run-Test-Idle	Run-Test-Idle
Select-DR-Scan	Select-DR-Scan
Capture-DR	Select-IR-Scan
Exit1-DR	Capture-IR
Update-DR	Exit1-IR
Run-Test-Idle	Update-IR
	Run-Test-Idle

Note that no data is shifted, but the JTAG Master function has exhausted the ***Bit Count*** field and thus completed the scan operation. It clears the ***Scan in Process*** bit and sets the ***Bit Count Exhausted*** bit returning to the Run-Test-Idle state to wait for another scan operation.

4.5.2 JTAG Master Register Matrix

The following JTAG Interface Control registers. The base 64K address (upper 16 address bits) is parameterizable; the lower 16 bits are decoded as the physical address. All of the registers are parity protected except for the status portions of the **JTAG Control/Status Register (0x00)** and the **JTAG TDI Register (0x08)**. When a byte write occurs to one of the parity protected registers, parity is regenerated across the whole word. All registers perform as per Table 16.

Table 16 - JTAG Master Function Register/Resource Map

Register/ Array/ Command	Address	Access	Reset State	Expected Use	Page
JTAG Control/Status Register	0x‘0000’	R/W	0x’0000 0000’	Control/Status	106
JTAG Bit Count Register	0x‘0004’	R/W	0x’0000 0000’	Control	107
JTAG TDI Register	0x‘0008’	R/W	0x’0000 0000’	Read data	108
JTAG TDO Register	0x‘000C’	R/W	0x’0000 0000’	Write data	108
JTAG TCLK Divide Register	0x’0010’	R/W	0x’0000 0003’	Configuration	108
JTAG Function ID	0x‘0014’	R	Hard-coded	Configuration	109

Register/ Array/ Command	Address	Access	Reset State	Expected Use	Page
Reserved	0x‘0018’ – ‘001C’	R	0x‘0000 0000’	Padding to end of cache line	N/A

Reserved addresses return 0x‘0000 0000’ on reads and are considered valid addresses.

4.5.3 JTAG Master Register Descriptions

From the RAD750 view, the Base Address of the JTAG Master A is 0x’BF83 0000’ and Master B is 0x’BF84 0000’.

4.5.3.1 JTAG Control/Status Register (0x00)

The **JTAG Control/Status Register (0x00)** provides error and scan progress information. Occurrence of a Self Checking Error causes a Critical Error Interrupt to occur. Parity is generated across only the control information in this register (bits 8:9).

Bits 31:10	***Reserved***: These bits are reserved, and return 0b’0’ when read.
Bit 9	***Tap Reset:*** This bit discretely controls the **JTM_TRST_L** output and when set, causes the following actions: • **JTAG_TRST_L** active (logic 0) • **JTM_TMS**(x) held at logic 1 (assuming ***Master En*** is active), where x = selected interface • JTAG placed in Test-Logic-reset state • ***Power PCI Internal Access/Data Error, Handshake Error, Bit Count Exhausted***, and ***Scan in Process*** bits of this register cleared • 0b’0’ – No TAP reset (default) • 0b’1’ – TAP reset
Bit 8	***Master En:*** When set, this bit enables the JTAG Master interface. When cleared, **JTM_TDO**, **JTM_TRST_L**, **JTM_TCLK**, and **JTM_TMS(1:0)** are tri-stated, thus allowing another (external) device to drive the JTAG bus. This bit is ANDed with the Primary Input **JTM_MSTR_EN_IN**. 0b’0’ – JTAG Master Function disabled (default) 0b’1’ – JTAG Master Function enabled
Bits 7: 5	***Reserved***: These bits are reserved, and return 0b’0’ when read.
Bit 4	***Power PCI Internal Access/Data Error:*** This bit is set by the JTAG Master Function when it has detected: • an Power PCI Internal Access (Read or Write) to an undefined JTAG Address, or • an Power PCI Internal Write Data Parity Error, or • an Power PCI Internal Write to a Read-only register, or • an Power PCI Internal Byte Enable Parity Error This bit is cleared by a) writing a 0b‘1’ to this bit position, or b) setting the ***Tap Reset*** bit.

	0b'0' – No Power PCI Internal Access/Data error (default) 0b'1' – Power PCI Internal Access/Data error
Bit 3	***Self Checking Error:*** This bit is set by the JTAG Master Function when it has detected: • an internal self checking error, or • an invalid Power PCI Internal state transition, or • a Power PCI Internal Address parity error. When set, the JTAG Master function issues a Critical Error interrupt on **JTM_GBL_CRIT_ERROR**, tri-states the JTAG Primary Outputs, and transitions the TAP Controller state machine to an error state. The JTAG Master function clears this bit when the Power PCI is reset. 0b'0' – No Self checking error (default) 0b'1' – Self Checking error
Bit 2	***Handshake Error:*** This bit is set by the JTAG Master Function when it has detected a write access to a protected control register while busy scanning, including the **JTAG TCLK Divide Register (0x10), JTAG *Bit Count* Register (0x04)**, **JTAG TDO Register (0x0C)**, or the **JTAG TDI Register (0x08)**, while the ***Scan in Process*** bit is set. This bit is cleared by a) writing a 0b'1' to this bit position, or b) setting the ***Tap Reset*** bit. 0b'0' – No handshake error (default) 0b'1' – Handshake error
Bit 1	***Bit Count Exhausted:*** This bit is set by the JTAG Master Function and indicates that the JTAG Master function has completed a scan operation due to the ***Bit Count*** field being zero. The JTAG Master function clears this bit when the ***Tap Reset*** bit is set or whenever a write has occurred to the **JTAG TDO Register (0x0C)**. 0b'0' – Bit count not exhausted (default) 0b'1' – Bit count exhausted
Bit 0	***Scan in Process:*** This bit indicates that the JTAG Master Function is busy scanning the **JTAG TDI Register (0x08)** and **JTAG TDO Register (0x0C)** on the JTAG bus. The JTAG Master Function sets this bit whenever a write has occurred to the **JTAG TDO Register (0x0C),** and clears this bit when a) the ***TAP Reset*** bit is set or b) when all 32 bits from the **JTAG TDI Register (0x08)** and **JTAG TDO Register (0x0C)** have been scanned, or the ***Bit Count*** has been exhausted. 0b'0' – No scan in process (default) 0b'1' – Scan in process

4.5.3.2 JTAG Bit Count *Register (0x04)*

The **JTAG *Bit Count* Register (0x04)** is used to program the total number of bits to scan for a given JTAG operation, along with the type of scan operation, Data or Instruction, and JTAG Interface. The JTAG Master Function decrements the ***Bit Count*** field once for each bit scanned onto the JTAG bus.

When this field reaches zero, the JTAG Master stops scanning and sequences the JTAG bus to the Run-Test-Idle state. Any attempt to write this register while the ***Scan in Process*** bit is set causes the JTAG Master Function to ignore the write data and set the ***Handshake Error*** bit of the **JTAG Control/Status Register (0x00)**.

Bits 31	***Data/Instruction Select:*** This bit determines whether the JTAG Master Function sequences the JTAG through the Data Register scan or Instruction Register scan states. 0b'0' – Instruction Register scan 0b'1' – Data Register scan
Bits 30	***Interface Select:*** This bit determines whether the JTAG Master Function drives **JTM_TMS(1)** or **JTM_TMS(0)** active; the inactive **JTM_TMS** bit remains in its last state prior to switching interfaces. 0b'0' – **JTM_TMS(0)** 0b'1' – **JTM_TMS(1)**
Bits 29:16	***Reserved***: These bits are reserved, and return 0b'0' when read.
Bits 15:0	***Bit Count:*** This field represents the total number of bits to scan for a given operation. This field is decremented for each bit scanned onto the JTAG bus.

4.5.3.3 JTAG TDI Register (0x08)

The **JTAG TDI Register (0x08)** holds the data scanned in from the **JTM_TDI**. The first bit scanned in from the JTAG bus is stored in the least significant bit (0). Any attempt to write this register while the ***Scan In Process*** bit is set causes the JTAG Master Function to ignore the write data and set the ***Handshake Error*** bit of the **JTAG Control/Status Register (0x00)**. There is no parity protection on this register.

Bits 31:0	***TDI Data:*** These bits hold the data scanned in from **JTAG_TDI**

4.5.3.4 JTAG TDO Register (0x0C)

The **JTAG TDO Register (0x0C)** holds the data scanned onto **JTM_TDO**. The first bit scanned onto the JTAG bus is the least significant bit (0). A successful write of this register causes the ***Scan In Process*** bit of the **JTAG Control/Status Register (0x00)** to be set and the ***Bit Count Exhausted*** bit to be cleared. Any attempt to write this register while the ***Scan In Process*** bit is set causes the JTAG Master Function to ignore the write data and set the ***Handshake Error*** bit of the **JTAG Control/Status Register (0x00)**.

Bits 31:0	***TDO Data***: These bits hold the data to be scanned onto **JTM_TDO.**

4.5.3.5 JTAG TCLK Divide Register (0x10)

The **JTAG TCLK Divide Register (0x10)** provides the selection of the frequency of **JTM_TCLK** as a function of the input oscillator, **SYS_CLK_OSC**. Any attempt to write this register while the ***Scan in Process*** bit is set causes the JTAG Master Function to ignore the write data and set the ***Handshake Error*** bit of the **JTAG Control/Status Register (0x00)**.

Bits 31:3	***Reserved***: These bits are reserved, and return 0b'0' when read.
Bits 2:0	***TCLK Select***

	These bits determine the frequency of the **JTM_TCLK** as follows: 0b'000' – Divide OSC by 2 0b'001' – Divide OSC by 4 0b'010' – Divide OSC by 8 0b'011' – Divide OSC by 16 (default) 0b'100' – Divide OSC by 32 0b'101' – Divide OSC by 64 0b'110' – Divide OSC by 128 0b'111' – Divide OSC by 256

4.5.3.6 JTAG Function ID (0x14)

The **JTAG Revision ID** is a hard-coded value (no actual register present) that represents the Function ID and the version number of the JTAG Master function.

Bits 31:16	***Reserved***: These bits are reserved, and return 0b'0' when read.
Bits 15:8	***Function ID:*** Is hardcoded as 0x'28'.
Bits 7:0	***Revision Number:*** These bits represent the JTAG Master function revision number; first revision = 0x'01'

4.6 Embedded Micro Controller (EMC)

The Embedded Micro controller (EMC) I a small processing engine located within the Power PCI. It's primary function is to help system level startup, provide increased error monitoring and decision making, and to provide increased fault tolerance at the system level. The EMC can also perform variety of other functions. The following sections describe the EMC from an error detection and reporting standpoint. Further discussion of the EMC and the details of the instruction set and programming tools can be found in the Software User's manual.

4.6.1 EMC Vector Operations

The EMC provides a means to vector program control in order to service critical events in a timely manner. The **Vector Control Register** contains 8 Vector Pending bits, 8 Vector Enable bits, and 8 Stop on Vector bits. A Vector Interrupt occurs when one of the Vector Pending bits is set and the corresponding Vector Enable is set. The EMC enters the **Stop** state if the corresponding Stop on Vector bit is also set.

When a Vector Interrupt occurs the EMC saves the current contents of the **Program Counter** into the **Vector Interrupt Address** register and then begin fetching instructions from an indexed location from the address stored in the **Vector Table Anchor** register. The **Vector Table Anchor** register should point to 8 consecutive branch instructions, one for each vector. The index for Vector Interrupt 0 is 0, while the index for Vector Interrupt 7 is 0x'1C'.

Multiple Vector Interrupts is handled on a priority basis with Vector Interrupt 7 being the highest and Vector Interrupt 0 being the lowest. When the EMC processes a Vector Interrupt it automatically clears the Vector Enables for all the maskable Vector Interrupts, thus disabling all Vector Interrupts.

4.6.1.1 Vector Interrupt 7

Vector Pending bit 7 is set by a positive edge on **EMC_VECTOR_INT7**. Vector Enable bit 7 does not exist and no mechanism exists to disable Vector Pending bit 7, i.e., it is a non-maskable interrupt to the EMC. The EMC automatically clears Vector Pending bit 7 upon beginning the processing of Vector Interrupt 7 (this keeps the Vector Interrupt from being recursive).

While **EMC_VECTOR_INT7** is active and the EMC is not in the **Stop** state, the EMC decrements an internal **Watchdog Timer** (see also 4.6.4.6 Watchdog Timer Register (0x3C)). When the **Watchdog Timer** expires the EMC activates **EMC_WATCHDOG_EXPIRED**. This indicates that the EMC was not able to service the Vector Interrupt 7, possibly due to hardware failure, and that external corrective action is required such as chip reset.

4.6.1.2 Vector Interrupt 6

Vector Interrupt 6 is reserved for EMC processing exceptions. Vector Pending bit 6 is set by one of the following:

- Attempt to execute an invalid instruction
- Attempt to execute an instruction fetched with **RD_D_ERR** active on the Power PCI Internal bus
- Load instruction data returned with **RD_D_ERR** active or a parity error on the PCI Internal bus

Vector Enable bit 6 does not exist and no mechanism exists to disable Vector Pending bit 6, (i.e., it is a non-maskable interrupt to the EMC). The EMC automatically clears Vector Pending bit 6 upon beginning the processing of Vector Interrupt 6 (this keeps the Vector Interrupt from being recursive).

4.6.1.3 Vector Interrupt (5:0)

Vector Pending bits (5:0) is set by an active level on **EMC_VECTOR_INT(5:0)**. Vector Enable bits (5:0) allow for masking of Vector Interrupts (5:0). Vector Pending bits (5:0) are also set by writing to the **Vector Control** Register via the Power PCI Internal bus.

4.6.2 EMC Instruction Set

The following lists the microinstructions supported by the EMC sequencer. The valid assignments for the Source and Destination fields of the instructions are the 11 General Purpose Registers, the Condition/Status Register, Vector Interrupt Address Register, Vector Control Register, Vector Anchor Register, Watch Dog Timer, and a zero value.

OR: Dest= Src1 OR Src 2

AND: Dest= Src1 AND Src 2

XOR: Dest= Src1 XOR Src 2

ADD: Dest= Src1 + Src2

ADD Immediate: Dest= Src1 + Immed (immediate data is sign extended to 32 bits)

SUB: Dest= Src1 – Src2

SUBI (Sub Immediate): Dest= Src1 – Immed (Immediate data is sign extended to 32 bits)

LD: 32 bit Load to Dest

LDB: 8 bit load, byte determined by 2 lsb's of effective address

LDIU: Load Immediate Upper: Dest(31:16) = Immed

LDIL: Load Immediate Lower: Dest (15:0) = Immed

LDBS: 32 bit load, data byte swapped

LD8: Eight 32-bit loads from 8 consecutive locations into GPR0-7

LD8BS: Eight 32-bit loads from 8 consecutive locations into GPR0-7; Data is byte swapped

ST: 32 bit store

STB: 8 bit store, byte determined by 2 lsb's of effective address

STBS: 32 bit store, data is byte swapped

ST8: Eight 32-bit stores to 8 consecutive locations from GPR0-7

ST8BS: Eight 32-bit stores to 8 consecutive locations in GPR0-7, data is byte swapped

BCI: Branch immediate if condition is true

BNCI: Branch immediate if condition is false

BRI: Branch immediate unconditional

BAL: Branch and Link

Monitor: Enter Monitor State

Stop: Enter Stop State

4.6.3 EMC Register Definition

The EMC core contains the registers as described in Table 17.

Table 17 - EMC Core Register/Resource Map

Register	Physical Address	OCB Access Rights	Reset State	Notes	Page
GPR 0	0x'0000'	R/W[1]	0x'0000 0000'		112
GPR 1	0x'0004'	R/W[1]	0x'0000 0000'		112
GPR 2	0x'0008'	R/W[1]	0x'0000 0000'		112
GPR 3	0x'000C'	R/W[1]	0x'0000 0000'		112
GPR 4	0x'0010'	R/W[1]	0x'0000 0000'		112
GPR 5	0x'0014'	R/W[1]	0x'0000 0000'		112
GPR 6	0x'0018'	R/W[1]	0x'0000 0000'		112
GPR 7	0x'001C'	R/W[1]	0x'0000 0000'		112
GPR 8	0x'0020'	R/W[1]	0x'0000 0000'		112
GPR 9	0x'0024'	R/W[1]	0x'0000 0000'		112
GPR 10	0x'0028'	R/W[1]	0x'0000 0000'		112
Condition/Status Register	0x'002C'	R/W[1]	0x'0000 0000'		112
Vector Interrupt Address	0x'0030'	R/W[1]	0x'0000 0000'		113
Vector Control Register	0x'0034'	R/W	0x'0000 0000'	See register description for write access	113
Vector Anchor Register	0x'0038'	R/W[1]	0x'FFF0 0000'		114
Watchdog Timer Register	0x'003C'	R/W[1]	0x'0000 FFFF'		114
Program Counter	0x'0040'	R/W[1]	0x'FFF0 0020'		114
Breakpoint Address	0x'0044'	R/W[1]	0x'0000 0000'		114

Register	Physical Address	OCB Access Rights	Reset State	Notes	Page
Debug Command/Status	0x‘0048’	R/W	0x'0000 0000’	Writeable only while **EMC_DEBUG_WE** active	115
Error Interrupt Status	0x'004C’	R/W	0x'0000 0000’	Collects OCB slave access errors	116
Revision ID	0x'0050’	R	Hard Wired		128
Zero Pad	0x'0054’ – ‘005C’	R	0x'0000 0000’	Padding to 32 byte address boundary	N/A
Reserved	0x'0060’ – ‘FFFC’	N/A	N/A		N/A

1 Writeable only while in the **Stop** state.

4.6.4 Register Descriptions

4.6.4.1 GPR 0 – GPR 10 (0x00-0x28)

These registers represent the 11 General Purpose Registers of the EMC.

Bits 31:0	General Purpose Registers

4.6.4.2 Condition/Status Register (CR) (0x2C)

This register provides the condition flags used by the conditional branch instructions. The condition flags are updated by the logical and arithmetic instructions.

This register also provides status for EMC processing exceptions.

Bit 31	***Carry Flag:*** This bit represents the value of the carry from the execution unit adder. It is updated by the ADD, ADDI, SUB, and SUBI instructions. 0b'0' – No Carry 0b'1' – Carry
Bit 30	***Greater Than Flag:*** 0b'0' – Not greater than zero 0b'1' – Greater than zero
Bit 29	***Less Than Flag:*** 0b'0' – Not less than zero 0b'1' – Less than zero
Bit 28	***Zero Flag:*** 0b'0' – Not zero 0b'1' – Zero
Bits 27:3	***Reserved***: These bits are reserved, and return 0b’0’ when read.
Bit 2	***Load Error:*** The EMC sets this bit when a Load instruction fails due to **RD_D_ERR** being active or a data parity error detected on the OCB. This bit is cleared by writing a 1 to this bit position

	0b'0' – No load error 0b'1' – Load error
Bit 1	***Instruction Error:*** The EMC sets this bit when an attempt is made to execute an instruction that was fetched with **RD_D_ERR** active on the OCB. This bit is cleared by writing a 0b'1' to this bit position. 0b'0' – No instruction error 0b'1' – Instruction error
Bit 0	***Invalid Opcode:*** The EMC sets this bit when an attempt is made to execute an invalid opcode. This bit is cleared by writing a 0b'1' to this bit position. 0b'0' – No invalid opcode 0b'1' – Invalid opcode

4.6.4.3 Vector Interrupt Address (0x30)

This register is loaded from the **Program Counter** when a Vector Interrupt occurs.

Bits 31:2	***Vector Interrupt Address.***
Bits 1:0	***Reserved***: These bits are reserved, and return 0b'0' when read.

4.6.4.4 Vector Control Register (0x34)

This register provides the Vector Pending, Vector Enable, and Stop on Vector control bits.

Bit 31	***Vector Pending 7:*** The EMC sets this bit upon a positive edge on **EMC_VECTOR_INT7**. The EMC clears this bit upon beginning to process a Vector Interrupt 7. The OCB has bit write set access while in the **Stop** state else write access ignored. The EMC execution unit does not have write access. 0b'0' – No vector pending 0b'1' – Vector pending
Bit 30	***Vector Pending 6:*** The EMC sets this bit due to a processing exception. The EMC clears this bit upon beginning to process a Vector Interrupt 6. The OCB has bit write set access while in the **Stop** state else write access ignored. The EMC execution unit does not have write access. 0b'0' – No vector pending 0b'1' – Vector pending
Bits 29:24	***Vector Pending (5:0):*** The EMC sets the corresponding Vector Pending (5:0) bit upon an active level on **EMC_VECTOR_INT(5:0)**. The OCB has bit write set access to these bits while the EMC execution unit has bit write clear access (the bit is cleared when a 0b'1' is written). 0b'0' – No vector pending 0b'1' – Vector pending
Bit 23	***Reserved.*** Vector Pending 7 can not be disabled.
Bit 22	***Reserved***. Vector Pending 6 can not be disabled.
Bits 21:16	***Vector Enable (5:0):***

	These bits enable the corresponding Vector Pending (5:0) bits to generate a Vector Interrupt. When taking a Vector Interrupt, the EMC clears all Vector Enables (5:0). The OCB has write access while in the **Stop** state else write access ignored. The EMC execution unit has write access. 0b'0' – Disabled 0b'1' – Enabled
Bits 15:8	***Stop on Vector (7:0):*** These bits cause the EMC to enter the **Stop** state when the corresponding Vector Interrupt is active (provided the Stop on Vector Enable is set). The OCB has write access only while **EMC_DEBUG_WE** is active else write access ignored. The EMC execution unit does not have write access. 0b'0' – Do not stop on vector 0b'1' – Stop on vector
Bit 7	***Stop on Vector Enable:*** This bit globally enables the Stop on Vector(7:0) bits. It is self correcting (triple redundant) thus preventing a SEU from causing the EMC to enter the Stop state. The OCB has write access only while **EMC_DEBUG_WE** is active else write access ignored. The EMC execution unit does not have write access. 0b'0' – Disabled 0b'1' – Enabled
Bits 6:0	***Reserved***: These bits are reserved, and return 0b'0' when read.

4.6.4.5 Vector Anchor Register (0x38)

This register contains the location of the Vector Table. The Vector Table should contain 8 consecutive branch instructions, 1 for each vector. The Vector Table is located on a 32 byte address boundary.

Bits 31:5	***Vector Table Address.***
Bits 4:0	***Reserved***: These bits are reserved, and return 0b'0' when read.

4.6.4.6 Watchdog Timer Register (0x3C)

This register contains the EMC's Watchdog Timer. The Watchdog Timer decrements once for each clock cycle that **EMC_VECTOR_INT7** is active and the EMC is not in the **Stop** state.

Bits 31:16	***Reserved***: These bits are reserved, and return 0b'0' when read.
Bits 15:0	***Watchdog Timer.***

4.6.4.7 Program Counter (0x40)

This register contains the address of the currently executing instruction. Instructions are always 4 byte aligned.

Bits 31:2	***Program Counter.***
Bits 1:0	***Reserved***: These bits are reserved, and return 0b'0' when read.

4.6.4.8 Breakpoint Address (0x44)

This register contains the address to be compared against the **Program Counter** to determine if a breakpoint has been reached. A breakpoint only occurs when the Go Till Address command has been issued in the **Debug Command/Status** register.

Bits 31:2	***Breakpoint Address.***
Bits 1:0	***Reserved***: These bits are reserved, and return 0b'0' when read.

4.6.4.9 Debug Command/Status (0x48)

This register contains facilities for debug of the EMC execution.

Bits 31:30	***Command:*** These bits controls state transitions of the EMC when the Command Request bit is set as follows: 0b'00' – Go Forever. From the **Stop** state, this command causes the EMC to enter the **Sequence** state. 0b'01' – Single Step. From the **Stop** state, this command causes the EMC to enter the **Sequence** state. From the **Sequence** state the EMC enters the **Stop** state after executing the next instruction, provided it is not Monitor. 0b'10' – Go Till Address. From the **Stop** state, this command causes the EMC to enter the **Sequence** state. From the **Sequence** state the EMC enters the **Stop** state after executing the instruction for which the **Program Counter** matches the **Breakpoint Address**. 0b'11' – Stop. From the **Sequence** state the EMC enters the **Stop** state after executing the next instruction. From the **Monitor** state the EMC enters the **Stop** state.
Bit 29	***Critical Error Injection:*** This bit when set causes the EMC to enter the **Error** state and activate **EMC_CRITICAL_ERROR**. This bit is only writeable when both **EMC_DEBUG_WE** and **EMC_CE_INJECT_EN** are active. 0b'0' – No critical error injection 0b'1' – Inject critical error
Bits 28:5	***Reserved***: These bits are reserved, and return 0b'0' when read.
Bit 4	***Command Request:*** This bit validates the Command field. The EMC clears this bit upon completing the requested command. This bit is self correcting (triple redundant) thus preventing a SEU from causing the EMC to enter the Stop state. 0b'0' – No command request 0b'1' – Command request
Bit 3	***Stopped due to Vector:*** This bit indicates that the EMC entered the **Stop** state due to a Vector Interrupt operation in which the Stop on Vector bit was set. The EMC clears this bit upon exiting the **Stop** state. The OCB does not have write access to this bit. 0b'0' – No stop due to vector 0b'1' – Stopped due to vector
Bit 2	***Single Step/Stop Command:*** This bit indicates that the EMC entered the **Stop** state due to a Single Step or Stop

	command. The EMC clears this bit upon exiting the **Stop** state. The OCB does not have write access to this bit. 0b'0' – No single step or stop commanded 0b'1' – Single step or stop commanded
Bit 1	***Breakpoint Reached:*** This bit indicates that the EMC entered the **Stop** state due to a Go Till Address command with an address match between the **Program Counter** and **Breakpoint Address**. The EMC clears this bit upon exiting the **Stop** state. The OCB does not have write access to this bit. 0b'0' – No breakpoint reached 0b'1' – Breakpoint reached
Bit 0	***Stop Instruction:*** This bit indicates that the EMC entered the **Stop** state due to a Stop instruction. The EMC clears this bit upon exiting the **Stop** state. The OCB does not have write access to this bit. 0b'0' – No stop instruction 0b'1' – Stop instruction

4.6.4.10 Error Interrupt Status (0x4C)

This register collects error conditions detected as a slave on the OCB. **EMC_ERROR_INT** represents the logical OR of the bits in this register.

Bit 31	***Write Access Error:*** This bit is set when one of the following OCB slave write errors occurs: • Write to read only register • Write to register protected by the **Stop** state when not in the **Stop** state • Write to register protected by **EMC_DEBUG_WE** when inactive • Write to reserved address space This bit is cleared by writing a 0b'1' to this bit position. 0b'0' – No write access error 0b'1' – Write access error
Bits 30	***Address Phase Parity Error:*** This bit is set by an OCB slave address phase parity error. This bit is cleared by writing a 0b'1' to this bit position. 0b'0' – No address phase parity error 0b'1' – Address phase parity error
Bit 29	***Write Data Parity Error:*** This bit is set by an OCB slave write data parity error. This bit is cleared by writing a 0b'1' to this bit position. 0b'0' – No write data parity error 0b'1' – Write data parity error
Bit 28	***Address Out of Range:*** This bit is set by an OCB slave access to the Reserved address area. This bit is cleared by writing a 0b'1' to this bit position. 0b'0' – No address out of range error

	0b'1' – Address out of range error
Bits 27:0	***Reserved***: These bits are reserved, and return 0b'0' when read.

4.6.4.11 Revision ID (0x50)

This register contains the hardwired Revision ID for the EMC.

Bits 31:16	***Reserved***: These bits are reserved, and return 0b'0' when read.
Bits 15:8	***Core ID:*** Hardwired as 0x'48'
Bits 7:0	***Revision Number:*** 0x'01' – Initial release

4.7 Clock and Test (CAT) Functions

4.7.1 Clock Generation

The Power PCI chip contains clock generation logic capable of generating CPU, Real Time Clock (RTC), and PCI clocks. The input source for the clock generation function is either the **SYS_OSC** input signal when **OSC_SEL** is set to 0b'1' or from the **EXT_OSC** input signal when **OSC_SEL** is set to 0b'0'. **EXT_OSC** is connected to the CompactPCI bus. A block diagram of the clock generation function is shown in Figure 56.

In the figure, R represents a receiver, D represents driver, and SEL represents a selection block on a chip. DELAYs are for matching clock skews and will normally never be adjusted. The RAD750 board defaults after reset to the lowest speed mode on all clocks with the PLL Disabled.

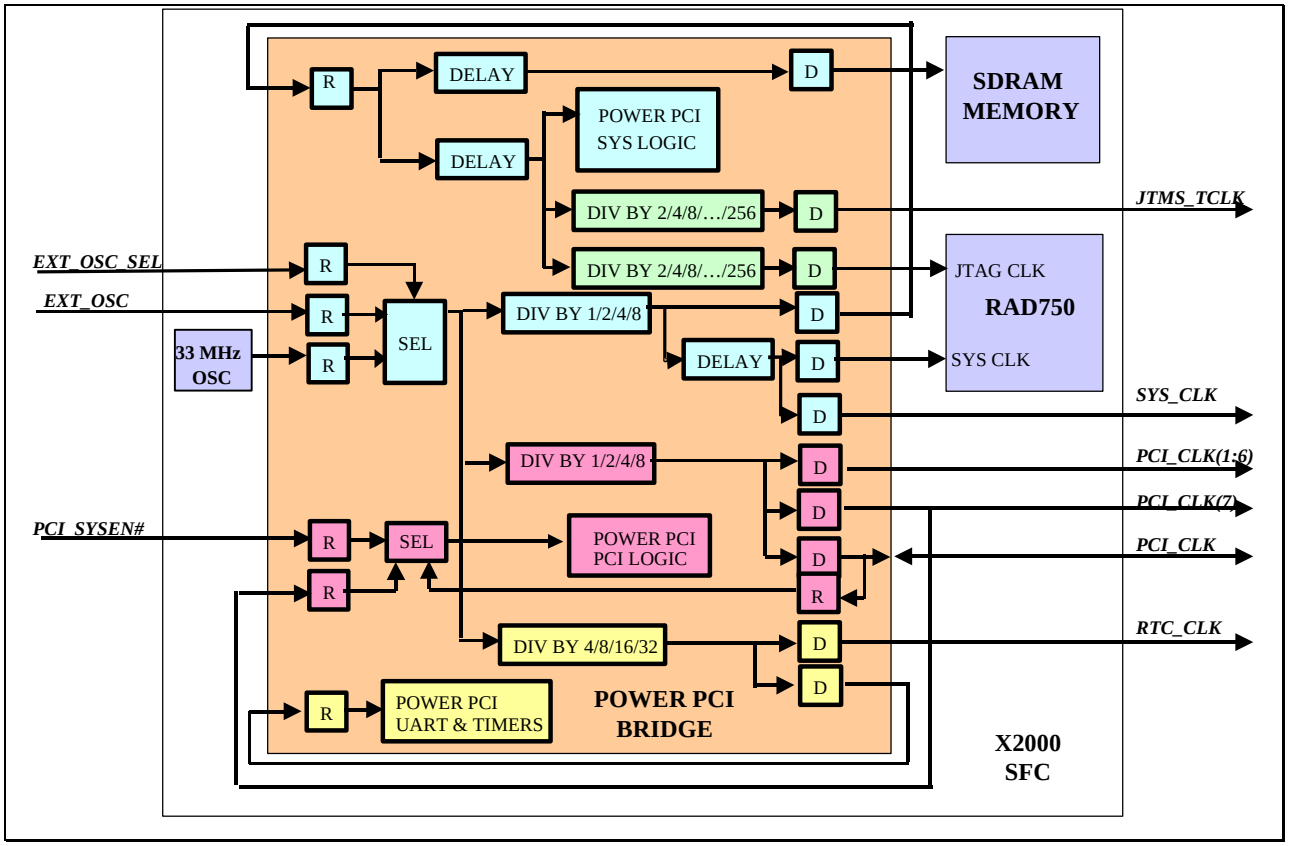


Figure 56: Power PCI Clock Generation Logic

The Power PCI chip generates three distinct clock sets off of its input clock oscillator. These are the PCI, CPU/System, and RTC clocks. These are shown in pink, light blue, and yellow in the figure. Software is capable of configuring the frequency of each clock using the clock generation divide registers. Changes in the divide frequency of any clock register causes the specified clock set to adjust in a glitch-free-manner to the new rate.

The CPU/System clock set generated by the Power PCI consists of three identical signals, each of which represents the input oscillator frequency divided by either 1, 2, 4, or 8 as determined by the CPU/System Clock divide register. On a RAD750 board which requires reduced clock frequency power savings modes, one of these signals would be connected to the oscillator input to the CPU chip while a second signal would be wrapped back to the system clock input to the Power PCI. The third signal is available to provide a monitor point for lab debug use. These clocks are active in all power saving modes. When the Power PCI is being used to generate the CPU/System clocks, it is also capable of generating the required control signal sequence to the PowerPC chip to safely control it's transitions between clock frequencies and modes. This is described in more detail in Section 4.7.1.1.

The RTC Clock set generated by the Power PCI consists of two identical signals, each of which represents the input oscillator frequency divided by either 4, 8, 16, or 32 as determined by the RTC Clock divide register. One of these pins may be wrapped back to the Power PCI RTC clock input to control the internal timers and UART function in place of an external oscillator. The second signal is available to provide a monitor point for lab debug use. These clocks are active in all power saving modes.

The PCI Clock set generated by the Power PCI consists of eight identical signals, each of which represents the input oscillator frequency divided by either 1, 2, 4, or 8 as determined by the PCI Clock divide register. One of these pins may be wrapped back to the PCI clock input to the Power PCI chip if the Power PCI is the PCI clock source in the system. These clocks are active in Full, Doze, and Nap

modes of operation. In sleep mode, the clocks are capable of being disabled via the Sleep disable bit in the PCI Clock divide register.

The JTAG clocks are generated off of the internal Power PCI system clock and can be divided by between 2 and 256.

4.7.1.1 PowerPC PLL/Clock Mode Control Function

For designs which require the Power PCI to supply clocks to the PowerPC processor chip and which require the RAD750 chip to be operated with its internal PLL bypassed, the Power PCI is capable of generating the required sequence of operations to the CPU described below. The actual generation of this sequence will be controlled via software. Due to the potentially critical result of an inadvertent change in clocking modes to the CPU, the setting of the PLL configuration bus requires a two step operation by software to execute (see the exact sequence below).

- Place CPU in sleep mode and wait for CPU to enter Sleep
- Activate Reset to the CPU
- Alter the PLL configuration discrete bits from the Power PCI to the CPU to the required state (see RAD750 spec for state definitions)
- Write to the PLL Configuration Discrete activation register to send the new discrete value to the CPU
- Set CPU clock control registers to their appropriate values for the frequency clock desired
- Deactivate Reset
- Wake-up CPU and perform required reinitialization

4.7.2 Power Management

The RAD750 and the Power PCI Bridge support a multi-level power management scheme that allows the devices to operate in reduced power states during times of inaction. This section describes how the two devices interoperate under these conditions and how users might benefit from this feature. The first section provides an overview of the RAD750 power management which will help in understanding the Power PCI power management modes. This information can also be found from the Power PC online information sources, some of which are listed in Section 4.9.

4.7.2.1 RAD750 Chip Level Scenarios

The RAD750 microprocessor is specifically designed for low-power operation. It provides both automatic and program-controlled power reduction modes for progressive reduction of power consumption. From a power management perspective, the RAD750 will operate identically to the commercial PowerPC 750.

4.7.2.1.1 Dynamic Power Management

Dynamic power management (DPM) automatically powers up and down the individual execution units of the RAD750, based upon the contents of the instruction stream. For example, if no floating-point instructions are being executed, the floating-point unit is automatically powered down. Power is not actually removed from the execution unit; instead, each execution unit has an independent clock input, which is automatically controlled on a clock-by-clock basis. Since CMOS circuits consume negligible power when they are not switching, stopping the clock to an execution unit effectively eliminates its power consumption. The operation of DPM is completely transparent to software or any external hardware. Dynamic power management is enabled by setting HID0[DPM] to 1.

4.7.2.1.2 Programmable Power Modes

The RAD750 provides four programmable power states: full power, doze, nap, and sleep. Software selects these modes by setting one (and only one) of the three power saving mode bits in the HID0 register. Hardware can enable a power management state through external asynchronous interrupts. Such a hardware interrupt causes the transfer of program flow to interrupt handler code that then invokes the appropriate power saving mode. The RAD750 provides a separate interrupt and interrupt vector for

power management: the system management interrupt (SMI). The RAD750 also contains a decrementer which allows it to enter the nap or doze mode for a predetermined amount of time and then return to full power operation through a decrementer interrupt. Note that the RAD750 cannot switch from one power management mode to another without first returning to full-power mode. The sleep mode disables bus snooping; therefore, a hardware handshake is provided to ensure coherency before the RAD750 enters this power management mode. Table 18 summarizes the four power states.

Table 18 - RAD750 Microprocessor Programmable Power Modes

PM Mode	Functioning Units	Activation Method	Full-Power Wake Up Method
Full power	All units active	--	--
Full power (with DPM)	Requested logic by demand	Controlled by software	--
Doze	• Bus snooping • Data cache as needed • Decrementer timer	Controlled by software	External asynchronous exceptions* Decrementer interrupt Performance monitor interrupt Thermal management interrupt Hard or soft reset
Nap	• Bus snooping - enabled by deassertion of QACK • Decrementer timer	Controlled by hardware and software	External asynchronous exceptions* Decrementer interrupt Hard or soft reset
Sleep	None	Controlled by hardware and software	External asynchronous exceptions* Hard or soft reset

Note: * Exceptions are referred to as interrupts in the architecture specification.

The following sections describe the characteristics of the RAD750's power management modes, the requirements for entering and exiting the various modes, and the system capabilities provided by the RAD750 while the power management modes are active.

4.7.2.1.2.1 Full-Power Mode with DPM Disabled

This is the default power state of the RAD750. The RAD750 is fully powered and the internal functional units are operating at the full processor clock speed. Full-power mode with DPM disabled is selected when the DPM enable bit (bit 11) in HID0 is cleared.

- Default state following power-up and HRESET
- All functional units are operating at full processor speed at all times.

4.7.2.1.2.2 Full-Power Mode with DPM Enabled

When the dynamic power management mode is enabled, functional units that are idle will automatically enter a low-power state without affecting performance, software execution, or external hardware. Full-power mode with DPM enabled (HID0[DPM] = 1) provides on-chip power management without affecting the functionality or performance of the RAD750.

- Required functional units are operating at full processor speed.
- Functional units are clocked only when needed.
- No software or hardware intervention is required after mode is set.
- Software/hardware and performance transparent

4.7.2.1.2.3 Doze Mode

All the functional units of the RAD750 are disabled except for the time base/decrementer registers and the bus snooping logic. When the processor is in doze mode, an external asynchronous interrupt, a system management interrupt, a decrementer exception, a hard or soft reset, or machine check brings the RAD750 into the full-power state. The RAD750 in doze mode maintains the PLL in a fully powered state and locked to the system external clock input (SYSCLK) so a transition to the full-power state takes only a few processor clock cycles. Doze mode disables most functional units but maintains cache coherency by enabling the bus interface unit and snooping. A snoop hit causes the RAD750 to enable the data cache, copy the data back to memory, disable the cache, and fully return to the doze state.

- Most functional units disabled
- Bus snooping and time base/decrementer still enabled
- Doze mode sequence
- Set doze bit (HID0[8] = 1), clear nap and sleep bits (HID0[9] and HID0[10] = 0)
- The RAD750 enters doze mode after several processor clocks
- Several methods of returning to full-power mode
- Assert INT, SMI, MCP, decrementer, performance monitor, machine check, or thermal management interrupts
- Assert hard reset or soft reset
- Transition to full-power state takes no more than a few processor cycles
- PLL running and locked to SYSCLK

4.7.2.1.2.4 Nap Mode

The nap mode further reduces power consumption by disabling bus snooping, leaving only the time base register and the PLL in a powered state. The RAD750 returns to the full-power state upon receipt of an external asynchronous interrupt, a system management interrupt, a decrementer exception, a hard or soft reset, or a machine check input (MCP). A return to full-power state from a nap state takes only a few processor clock cycles. When the processor is in nap mode, if QACK is negated, the processor is put in doze mode to support snooping.

The nap mode disables the RAD750 but still maintains the phase-locked loop (PLL), delay locked loop (DLL), L2CLK_OUTA and L2CLK_OUTB output signals, and the time base/ decrementer. The time base can be used to restore the RAD750 to full-power state after a programmed amount of time. To maintain data coherency, bus snooping is disabled for nap and sleep modes through a hardware handshake sequence using the quiesce request (QREQ) and quiesce acknowledge (QACK) signals. The RAD750 asserts the QREQ signal to indicate that it is ready to disable bus snooping. When the system has ensured that snooping is no longer necessary, it will assert QACK and the RAD750 will enter the nap mode. If the system determines that a bus snoop cycle is required, QACK is deasserted to the RAD750 for at least eight bus clock cycles, and the RAD750 will then be able respond to a snoop cycle. Assertion of QACK following the snoop cycle will again disable the RAD750's snoop capability. The RAD750's power dissipation while in nap mode with QACK deasserted is the same as the power dissipation while in doze mode.

The RAD750 also allows dynamic switching between nap and doze modes to allow the use of nap mode without sacrificing hardware snoop coherency. For this operation, negating QACK at any time for at least 8 bus cycles guarantees that the RAD750 has transitioned from nap mode to doze mode in order to snoop. Reasserting QACK then allows the RAD750 to return to nap mode. This sequencing could be used by the system at any time without knowledge of what power management mode, if any, that the RAD750 is currently in.

Note that when in nap mode the DLL should be kept locked to enable a quick recovery to full-power mode without having to wait for the DLL to re-lock. Additionally, an L2ZZ signal is provided by the RAD750's L2 cache interface to drive external SRAM into a low power mode when the nap or sleep modes are invoked. The L2ZZ signal is enabled by setting the L2CR[CTL] bit to 1. Note that if bus snooping is to be performed through deassertion of the QACK signal, the L2CR[CTL] bit should always be cleared to 0.

- Time base/decrementer still enabled
- Thermal management unit enabled
- Most functional units disabled
- All nonessential input receivers disabled
- Nap mode sequence
- Set nap bit (HID0[9] = 1), clear doze and sleep bits (HID0[8] and HID0[10] = 0)
- The RAD750 asserts quiesce request (QREQ) signal
- System asserts quiesce acknowledge (QACK) signal
- The RAD750 enters nap mode after several processor clocks
- Nap mode bus snoop sequence
- System deasserts QACK signal for eight or more bus clock cycles
- The RAD750 snoops address tenure(s) on bus
- System asserts QACK signal to restore full nap mode
- Several methods of returning to full-power mode
- Assert INT, SMI, MCP, machine check, or decrementer interrupts
- Assert hard reset or soft reset
- Transition to full-power takes no more than a few processor cycles
- PLL and DLL running and locked to SYSCLK.

4.7.2.1.2.5 Sleep Mode

Sleep mode minimizes power consumption by disabling all internal functional units, after which external system logic may disable the PLL and SYSCLK. Returning the RAD750 to the full-power state requires the enabling of the PLL and SYSCLK, followed by the assertion of an external asynchronous interrupt, a system management interrupt, a hard or soft reset, or a machine check input (MCP) signal after the time required to relock the PLL.

Sleep mode consumes the least amount of power of the four modes since all functional units are disabled. To conserve the maximum amount of power, the PLL may be disabled by placing the PLL_CFG signals in the PLL bypass mode, and disabling SYSCLK. Note that forcing the SYSCLK signal into a static state does not disable the RAD750's PLL, which will continue to operate internally at an undefined frequency unless placed in PLL bypass mode. Additionally, if the PLL is not disabled, the L2 cache interface DLL will remain locked and the L2CLK_OUTA and L2CLK_OUTB signals will remain active. The DLL is disabled by clearing the L2CR[L2E] bit to 0.

Due to the fully static design of the RAD750, internal processor state is preserved when no internal clock is present. Because the time base and decrementer are disabled while the RAD750 is in sleep mode, the RAD750's time base contents will have to be updated from an external time base after exiting sleep mode if maintaining an accurate time-of-day is required. Before entering the sleep mode, the RAD750 asserts the QREQ signal to indicate that it is ready to disable bus snooping. When the system has ensured that snooping is no longer necessary, it asserts QACK and the RAD750 will enter sleep mode.

- All functional units disabled (including bus snooping and time base)
- All nonessential input receivers disabled
- Internal clock regenerators disabled
- PLL and DLL still running (see below)
- Sleep mode sequence
- Set sleep bit (HID0[10] = 1), clear doze and nap bits (HID0[8] and HID0[9])
- The RAD750 asserts quiesce request (QREQ)
- System asserts quiesce acknowledge (QACK)
- The RAD750 enters sleep mode after several processor clocks
- Several methods of returning to full-power mode
- Assert INT, SMI, or MCP interrupts
- Assert hard reset or soft reset
- PLL and DLL may be disabled and SYSCLK may be removed while in sleep mode

- Return to full-power mode after PLL and SYSCLK are disabled in sleep mode
- Enable SYSCLK
- Reconfigure PLL into desired processor clock mode
- System logic waits for PLL startup and relock time (100 μsec)
- System logic asserts one of the sleep recovery signals (for example, INT or SMI)
- Reconfigure DLL, wait for DLL relock (640 L2 clock cycles) and re-enable L2 cache through the L2CR

4.7.2.1.3 Power Management Software Considerations

Since the RAD750 is a dual-issue processor with out-of-order execution capability, care must be taken in how the power management mode is entered. Furthermore, nap and sleep modes require all outstanding bus operations to be completed before these power management modes are entered. Normally, during system configuration time, one of the power management modes would be selected by setting the appropriate HID0 mode bit. Later on, the power management mode is invoked by setting the MSR[POW] bit. To ensure a clean transition into and out of a power management mode, set the MSR[EE] bit to '1' and execute the following code sequence:

sync

mtmsr[POW = 1]

isync

continue

4.7.2.2 Power PCI Bridge Chip Level Scenarios

The Power PCI provides the system designer hardware resources to flexibly reduce system power consumption through the use of software and system hardware power control mechanisms. The Power PCI provides hardware support for four levels of power reduction: the doze, nap, and sleep modes are invoked by register programming. Note the MPC-106 also supports a "Suspend" mode which is not implemented on the Power PCI because of the fatalistic results of entering it. The design of the Power PCI is fully static, allowing internal logic states to be preserved during all power saving modes. The power management mode is selected by setting the appropriate bits in the Power Management Control Register (configuration offset 70). The bits from this register that are defined by the Power PCI are:

- Bit 7 PM – Power Management Enable
- Bit 5 Doze
- Bit 4 Nap
- Bit 3 Sleep

The power management operational modes for the Power PCI are shown in Table 19. The Power PCI disables clocks to those internal functions that are not required to be operational during a selected power savings mode. Each of the operational units in the Power PCI supplies the internal clock function with status signals indicating when they are busy performing operations and are not capable of entering reduced power mode. The clock function uses these signals to ensure that the disabling of the clocks to the shut-off functions does not occur until all on-going operations have completed.

The difference between Doze and Nap is slight, with the only difference being that the chip if the chip is in Nap mode it will return to full power mode after a 60x bus request. In Doze mode it would return to Doze rather than going back to full power.

When the Power PCI chip is placed in Nap or Doze mode, various elements have their clocks stopped internal to the chip. Included in the elements which are stopped are:

- 60x logic, except for the portion monitoring for bus requests
- Memory Interface core logic, except SDRAM refresh and scrubbing
- PCI functions ONLY if the source of the PCI clocks in the system disables the clocks

In addition, the lack of transactions occurring from the CPU (and likely the PCI bus) will also cause a decrease in power simply due to the reduction in internal logic switching factor.

Table 19 - Power PCI Power Management Operational Modes

PM Mode	Functioning Units	Activation	Wake Up	Return Mode
Full Power	All	--	--	--
Doze	• Same as Nap	Doze bit set	• Same as Nap	• Doze
Nap	• Same as Sleep • PCI target • PCI arbiter • PCI clocks • DRAM Scrub • DRAM refresh (active or self refresh supported)	Nap bit set and assertion of QREQ	• Same as Sleep • PCI target transaction	• 60x bus request returns to full power • All others return to Doze
Sleep	• DRAM refresh (Self-refresh supported) • 60x bus request monitoring • NMI monitoring • Interrupt/Discrete logic • Machine Check monitoring • JTAG slave • Timers • EMC Vector Interrupt monitoring • Checkstop Monitoring • PCI Clocks (Configurable as on or off)	Sleep bit set and assertion of QREQ	• Reset • 60x bus request • NMI • Interrupt to CPU • Machine Check to CPU • JTAG slave OCB access • Internal EMC Vector Interrupt • Checkstop from CPU	• 60x bus request returns to full power • All others return to Sleep

4.7.3 Clock and Test (CAT) Register Definition

The Clock and Test (CAT) function of the Power PCI provides the following functions:

- BIST Function
  - — BIST Support
  - — Manufacturing Pin Self Test Support
  - — Flush Reset logic
  - — Manufacturing LSSD Test Support
- System Clock Divide
- Power Management

4.7.4 Clock and Test Register Matrix

Table 26 is a listing of all registers related to the Clock and Test function of the Power PCI. All defined registers in the Power PCI exist in a single cache line.

Table 20 - CAT Function Register/Resource Map

Register/Array/Command	Address	Access	Reset State	Expected Use	Page
Clock Control	0x'0000'	R/W	0x'0000 003F'	Control	125
PLL Setup	0x'0004'	R/W	0x'0000 0000'	Control	126
PLL Configuration	0x'0008'	R/W	0x'0000 0000'	Control	127
PM Control	0x'000C'	R/W	0x'0000 0000'	Control	127
Error Interrupt Status	0x'0010'	R/W	0x'0000 0000'	Status	128
Reserved	0x'0014'	R	0x'0000 0000'	N/A	128
Zero Pad	0x'0018'	R	0x'0000 0000'	N/A	
Zero Pad	0x'001C'	R	0x'0000 0000'	N/A	
BIST MISR	0x'0020 - 0024'	R/W	N/A	Test/Initialization	128
BIST Control	0x'0028'	R/W	0x'0000 0000'	Test/Initialization	129
BIST Status	0x'002C'	R/W	N/A	Test/Initialization	129
BIST ID	0x'0030'	R	N/A	Test/Initialization	131
Zero Pad	0x'0034 – 003C'	R	0x'0000 0000'		
BIST Soft Reset	0x'0040'	W	N/A	Test/Initialization	131
BIST Stop Clocks	0x'0044'	W	N/A	Test/Initialization	131
BIST Start Clocks	0x'0048'	W	N/A	Test/Initialization	131
BIST Single Cycle	0x'004C'	W	N/A	Test/Initialization	131
BIST Start LBIST	0x'0050'	W	N/A	Test/Initialization	131
BIST Start ABIST	0x'0054'	W	N/A	Test/Initialization	131
BIST Start Labscan	0x'0058'	W	N/A	Test/Initialization	131
BIST Cycle Count	0x'005C'	R/W	0x'0000 0000'	Test/Initialization	131
Reserved	0x'0060' – 'FFFC'	N/A	N/A	Address out of Range error	

4.7.5 CAT Register Descriptions

The base address for the CAT function registers is 0x'BF86 0000'.

4.7.5.1 Clock Control (0x00)

This register contains clock control bits.

Bits 31:28	***Tap Delay In Select(3:0):***
	These bits select the number of delays on **IO_CAT_SYS_CLK_IN** prior to becoming the Power PCI system clock.
	0b'0000' - No delays
	0b'0001' - 1 delay
	0b'0010' - 2 delays

	… 0b'1111' - 15 delays
Bits 27:24	***Tap Delay Out Select(3:0):*** These bits select the number of delays on **CAT_IO_SYS_CLK_OUT(2:1)**. 0b'0000' - No delays 0b'0001' - 1 delay 0b'0010' - 2 delays … 0b'1111' - 15 delays
Bits 23:16	***PCI Clock Disable(7:0):*** These bits force each of the corresponding **CAT_IO_PCI_CLK_OUT(7:0)** high at the next rising edge, provided **PCI_CENTRAL_RESOURCE** is active. 0b'0' - Enabled 0b'1' - Disabled
Bits 15	***Reserved***: These bits are reserved, and return 0b'0' when read.
Bits 14:11	***Reserved:*** These bits can be read/written but are not used functionally in the Power PCI.
Bit 10	***UART Clock Disable:*** This bit forces the signal **UART_CLK** to a logic 0b'1' at the next rising edge. 0b'0' - Enabled 0b'1' - Disabled
Bits 9:8	***RTC Clock Disable(1:0):*** These bits force each of the corresponding **CAT_IO_RTC_CLK_OUT(1:0)** high at the next rising edge. 0b'0' - Enabled 0b'1' - Disabled
Bits 7:6	***Sys Clock Disable(2:1):*** These bits force each of the corresponding **CAT_IO_SYS_CLK_OUT(2:1)** high at the next rising edge. 0b'0' - Enabled 0b'1' - Disabled
Bits 5:4	***PCI Clock Divide Select(1:0):*** These bits select the clock divide for **CAT_IO_PCI_CLK_OUT(7:0)**. 0b'00' - Divide by 1 0b'01' - Divide by 2 0b'10' - Divide by 4 0b'11' - Divide by 8
Bits 3:2	***RTC Clock Divide Select(1:0):*** These bits select the clock divide for **CAT_IO_RTC_CLK_OUT(1:0)**. 0b'00' - Divide by 4

	0b'01' - Divide by 8 0b'10' - Divide by 16 0b'11' - Divide by 32
Bits 1:0	***Sys Clock Divide Select(1:0):*** These bits select the clock divide for **CAT_IO_SYS_CLK_OUT(2:0)**. 0b'00' - Divide by 1 0b'01' - Divide by 2 0b'10' - Divide by 4 0b'11' - Divide by 8

4.7.5.2 PLL Setup (0x04)

This register controls the setup for the PLL configuration of the CPU.

Bits 31:5	***Reserved***: These bits are reserved, and return 0b'0' when read.
Bit 4	***PLL Enable Setup:*** This bit is loaded into the PLL Enable bit of the **PLL Configuration** register upon a write access to the **PLL Configuration** register. 0b'0' - Disabled 0b'1' - Enabled
Bits 3:0	***PLL Setup(3:0):*** These bits are loaded into the PLL Configuration bits of the **PLL Configuration** register upon a write access to the **PLL Configuration** register.

4.7.5.3 PLL Configuration (0x08)

Bits 31:5	***Reserved***: These bits are reserved, and return 0b'0' when read.
Bit 4	***PLL Enable:*** A write access to the **PLL Configuration** register causes this bit to be loaded from the **PLL Setup** register. This bit enables the driving of the **CAT_IO_PLL_CONFIG(3:0)**. 0b'0' - Disabled (3-state) 0b'1' - Enabled
Bits 3:0	***PLL Configuration(3:0):*** A write access to the **PLL Configuration** register causes these bits to be loaded from the **PLL Setup** register. These bits are driven onto **CAT_IO_PLL_CONFIG(3:0),** provided the PLL Enable bit is also set.

4.7.5.4 Power Management Control (0x0C)

This register provides control of the PowerPC architecture Power Management modes.

The CAT function enters the Power Management Doze Mode when the Doze bit and Power Management Enable bit of the **Power Management Control** register are set and all power management busy inputs have been inactive for 64 cycles. While in Doze Mode, the **SYS_CLK** signal is held inactive (logic 1).

The CAT function returns to Full Power Mode when any of the power management busy inputs becomes active. Note that the Power Management Enable bit is not altered allowing the CAT function to return to Doze Mode.

Bit 31	***Power Management Enable:*** This bit enables entry into one of the Power Management modes. This bit is cleared whenever the Nap or Sleep Power Management modes are selected, the Power Management Enable is set, and the **P60X_INTERFACE_BUSY** becomes active. 0b'0' - Disabled 0b'1' - Enabled
Bits 30:28	***Power Management Select(3:0):*** These bits selects the Power Management mode provided the Power Management Enable is also set. 0b'001' - Doze 0b'010' - Nap 0b'100' - Sleep others - Full Power
Bits 27:2	***Reserved***: These bits are reserved, and return 0b'0' when read.
Bit 1	**QREQ:** This bit represents an inverted free running status of the **IO_CAT_QREQ_L** input. This bit is not writeable. 0 – **IO_CAT_QREQ_L** inactive (logic 1) 1 – **IO_CAT_QREQ_L** active (logic 0)
Bit 0	**QACK:** This bit represents an inverted free running status of the **IO_CAT_QACK_L** input. This bit is not writeable. 0 – **IO_CAT_QACK_L** inactive (logic 1) 1 – **IO_CAT_QACK_L** active (logic 0)

4.7.5.5 Error Interrupt Status (0x10)

This register contains status as to the source of the interrupt from the Clock and Test function to the internal interrupt handler.

Bit 31	***Write Access Error:*** This bit is set by a slave OCB write to a read only register. This bit is cleared by writing a 0b'1' to this bit position. 0b'0' - No write access error 0b'1' - Write access error
Bit 30	***Address Phase Parity Error:*** This bit is set by a slave OCB address phase parity error. This bit is cleared by writing a 0b'1' to this bit position. This bit activates **CAT_CRITICAL_ERROR**. 0b'0' - No address phase parity error 0b'1' - Address phase parity error
Bit 29	***Write Data Parity Error:*** This bit is set by an OCB write access with bad parity. This bit is cleared by writing a 0b'1' to this bit position. 0b'0' - No write data parity error

	0b'1' - Write data parity error
Bit 28	***Address out of Range:*** This bit is set by a slave OCB access to an address out of range. This bit is cleared by writing a 0b'1' to this bit position. 0b'0' - No address out of range 0b'1' - Address out of range
Bits 27:0	***Reserved***: These bits are reserved, and return 0b'0' when read.

4.7.5.6 Revision ID (0x14)

Bits 31:16	***Reserved***: These bits are reserved, and return 0b'0' when read.
Bits 15:8	***Function ID:*** Hardwired as 0x'58'
Bits 7:0	***Revision Number:*** 0b' '01' - Initial release

4.7.5.7 BIST MISR Low (0x20)

Bits 31:0	***MISR Low*** This register contains the low order 32 bits of the Multiple Input Signature generated during Logic BIST, Array BIST, and Pin Self Test. A write to this register will reset both the MISR Low and MISR High registers to 0x'0000 0000'.

4.7.5.8 BIST MISR High (0x24)

Bits 31:0	***MISR High*** This register contains the high order 32 bits of the Multiple Input Signature generated during Logic BIST, Array BIST, and Pin Self Test. A write to the MISR Low register will reset both the MISR Low and MISR High registers to 0x'0000 0000'.

4.7.5.9 BIST Control Register (0x28)

Bits 31:3	***Reserved***: These bits are reserved, and return 0b'0' when read.
Bit 2:	***OCD Enable*** This bit controls the On Chip Drivers during Labscan. 0b'0' – On Chip Drivers are tristated 0b'1' – On Chip Drivers are enabled
Bit 1:	***Scan Enable*** This bit enables Labscan operations 0b'0' – Labscan is disabled 0b'1' – Labscan is enabled
Bit 0	***Test Logic Enable:*** This bit is used to conserve power by disabling logic in the BIST when not in use. 0b'0' – BIST is inactive, with the exception of the Reset logic and OCB interface. 0b'1' – BIST is active

4.7.5.10 BIST Status Register (0x2C)

The following register contains status on the Built in Self Test function in the Power PCI. Note that some of the bits will never be set for the Power PCI since there are no arrays (and thus no array self-test) in the design.

Bits 31:16	***Reserved***: These bits are reserved, and return 0b'0' when read.
Bit 15:	***Control Register Parity Error*** This bit indicates the presence of a parity error in the Control Register Data. The Control register is checked continually for odd parity. 0b'0' – No parity error 0b'1' – Parity Error
Bit 14:	***State Machine Error*** This bit indicates the presence of a state machine error within the Array BIST Logical Register Data. The Control register is checked continually for odd parity. 0b'0' – No state machine error 0b'1' – State machine error
Bit 13:	***Array BIST Complete*** This bit is set following the completion of Array BIST. 0b'0' – Array BIST not complete. 0b'1' – Array BIST complete
Bit 12:	***Array BIST In Progress*** This bit indicates that Array BIST is currently in progress. 0b'0' – Array BIST not in progress. 0b'1' – Array BIST in progress
Bit 11:	***Logic BIST Complete:*** This bit is set following the completion of Logic BIST. 0b'0' – Logic BIST not complete. 0b'1' – Logic BIST complete.
Bit 10:	***Logic BIST In Progress:*** This bit indicates that Logic BIST is currently in progress. 0b'0' – Logic BIST not in progress. 0b'1' – Logic BIST in progress.
Bit 9:	***Scan Operation In Progress:*** This bit indicates that a scan operation is currently in progress. 0b'0' – Scan operation not in progress. 0b'1' – Scan operation in progress.
Bit 8:	**BISTable Clocks Stopped:** This bit indicates that BISTable clocks have been stopped, either through a Stop Clocks command, or by the Logic BIST state machine. 0b'0' – BISTable clocks not stopped. 0b'1' – BISTable clocks not stopped.

Bits 7:5:	***Reserved***: These bits are reserved, and return 0b'0' when read.
Bit 4:	**Logic BIST Reset:** This bit is set by the Logic BIST state machine to indicate that BISTable logic has been reset following Logic BIST. This bit is cleared by writing a 1 to this bit position, writing a 1 to bit position 0, or the occurrence of a Power On Reset. 0b'0' – No Logic BIST Reset. 0b'1' – Logic BIST Reset.
Bit 3:	**JTAG Software Reset:** This bit is set by a JTAG Software Reset command. This bit is cleared by writing a 0b'1' to this bit position, writing a 0b'1' to bit position 0, or the occurrence of a Power On Reset. 0b'0' – No JTAG Software Reset. 0b'1' – JTAG Software Reset.
Bit 2:	***OCB Software Reset:*** This bit is set by an OCB Software Reset command. This bit is cleared by writing a 0b'1' to this bit position, writing a 0b'1' to bit position 0, or the occurrence of a Power On Reset. 0b'0' – No OCB Software Reset. 0b'1' – OCB Software Reset.
Bit 1:	***Internal Hardware Reset:*** This bit is set by an internal hardware reset. This bit is cleared by writing a 0b'1' to this bit position, writing a 0b'1' to bit position 0, or the occurrence of a Power On Reset. 0b'0' – No Internal Hardware Reset. 0b'1' – Internal Hardware Reset.
Bit 0:	***Power On Reset:*** This bit is set by a Power On Reset. This bit is cleared by writing a 0b'1' to this bit position. 0b'0' – No Power On Reset. 0b'1' – Power On Reset.

4.7.5.11 BIST ID (0x30)

Bits 31:16	***Reserved***: These bits are reserved, and return 0b'0' when read.
Bit 15:8:	***Core ID*** Hardwired as 0x'40'
Bit 7:0	***Revision ID*** Hardwired as 0x'01' – Initial release

4.7.5.12 BIST Soft Reset (0x40)

A write access to this location will issue a soft reset.

4.7.5.13 BIST Stop Clocks (0x44)

A write access to this location issues a Stop Clocks command, causing BISTable clocks to be stopped. BISTable clocks will be stopped in frequency order, from fastest to slowest, as designated by the user.

4.7.5.14 BIST Start Clocks (0x48)

A write access to this location issues a Start Clocks command, causing stopped BISTable clocks to be started. BISTable clocks will be started in frequency order, from fastest to slowest, as designated by the user.

4.7.5.15 BIST Single Cycle Clocks (0x4C)

A write access to this location issues a Single Cycle Clocks command, causing stopped BISTable clocks to be single cycled. BISTable clocks are single cycled in frequency order, from fastest to slowest, as designated by the user.

4.7.5.16 BIST Start LBIST (0x50)

A write access to this location issues a Start Logic BIST command.

4.7.5.17 BIST Start ABIST (0x54)

A write access to this location issues a Start Array BIST command.

4.7.5.18 BIST Start Labscan (0x58)

A write access to this location issues a Start Labscan command.

4.7.6 BIST Cycle Count Register (0x5C)

This register contains the cycle count, which determines the number of cycles of Logic BIST or Array BIST to be executed. It also determines the number of scan clocks executed during Labscan.

4.8 Miscellaneous Function Register Definition

The Power PCI provides many miscellaneous functions, including:

- Embedded Programmable Interrupt Controller
  - 32 external interrupts (shared with Programmable I/O Discretes)
  - 16 internal interrupts
  - 32 Multiprocessor signals
- Timers
  - 3 general purpose
  - 1 watchdog
- 32 Programmable I/O Discretes (PIDs) (shared with external interrupts)
- Multiprocessing support
  - 32 hardware semaphores
  - 32 signaling interrupts
- Critical Error Collection
- CPU Control and Error Collection
  - HRESET
  - Checkstop

4.8.1 Embedded Programmable Interrupt Controller (EPIC)

The EPIC subfunction provides for the collection of interrupts from internal and external sources. It collapses these interrupts into a single interrupt to the RAD750, **MISC_IO_INT_L**. Figure 57 depicts the Interrupt related signals from the Power PCI that connect to the RAD750 and the other CompactPCI connector.

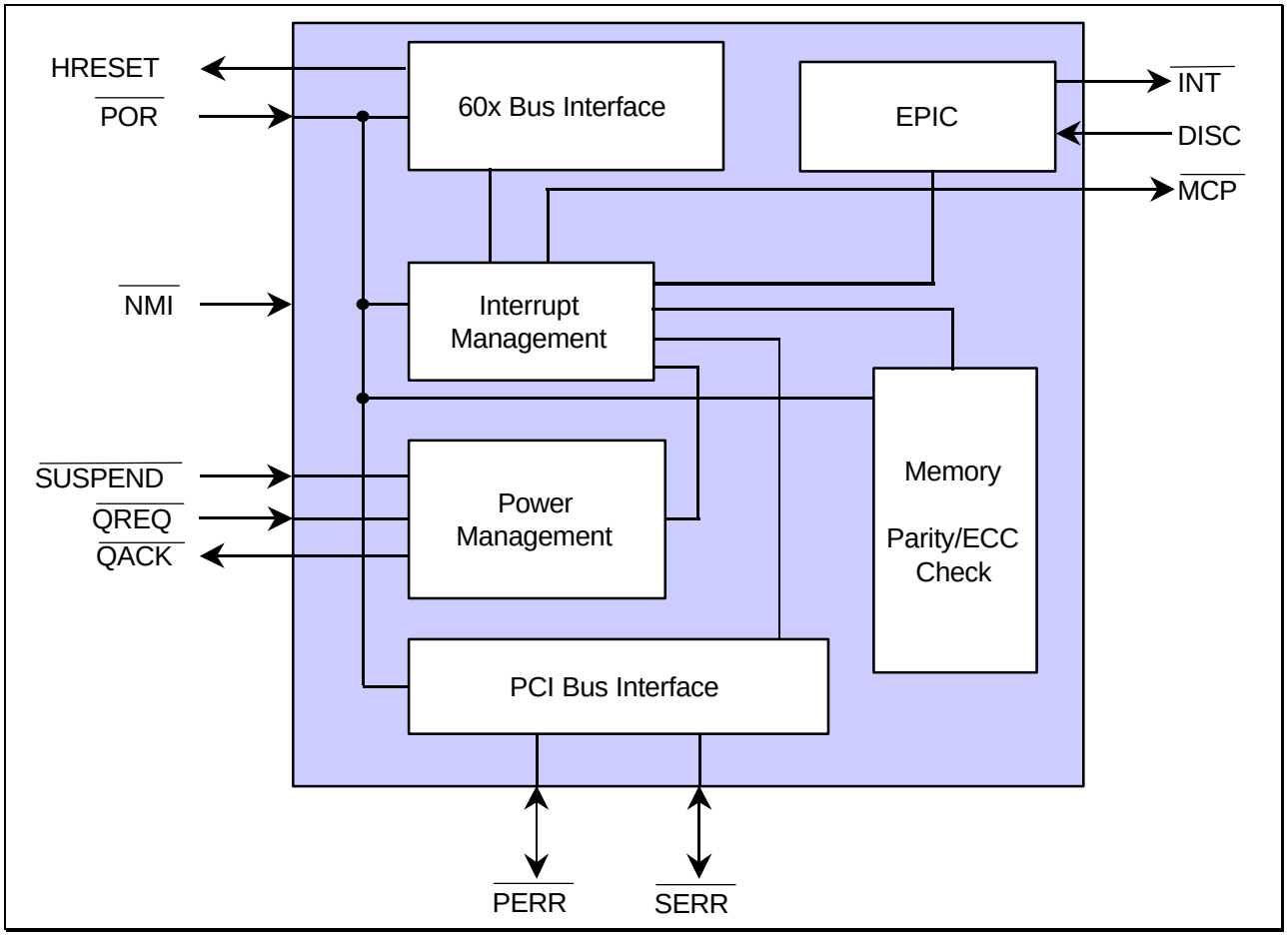


Figure 57: Power PCI Interrupt Signal Generation

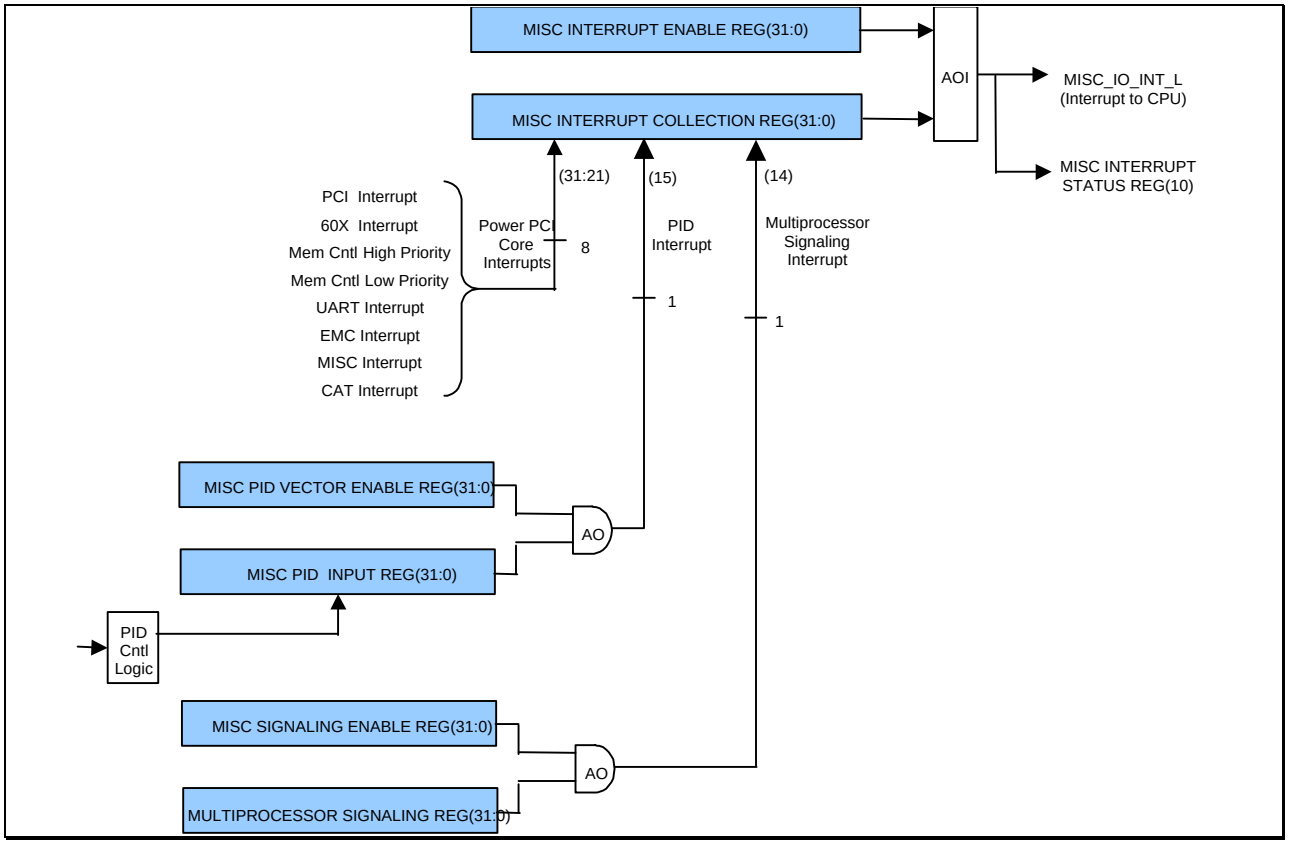


Figure 58: PowerPCI Interrupt Collection

At the top of the interrupt tree is the **Interrupt Collection** register and the **Interrupt Enable** register. The **Interrupt Collection** register provides status as to the source of an interrupt. Each bit in the **Interrupt Collection** register is reduced through an AND-OR function with each corresponding bit in the **Interrupt Enable** register to generate the single RAD750 interrupt. Each of the bits in the **Interrupt Collection** register represents free running status of the interrupting source. That is when the source of the interrupt is removed the corresponding bit in the **Interrupt Collection** register will clear.

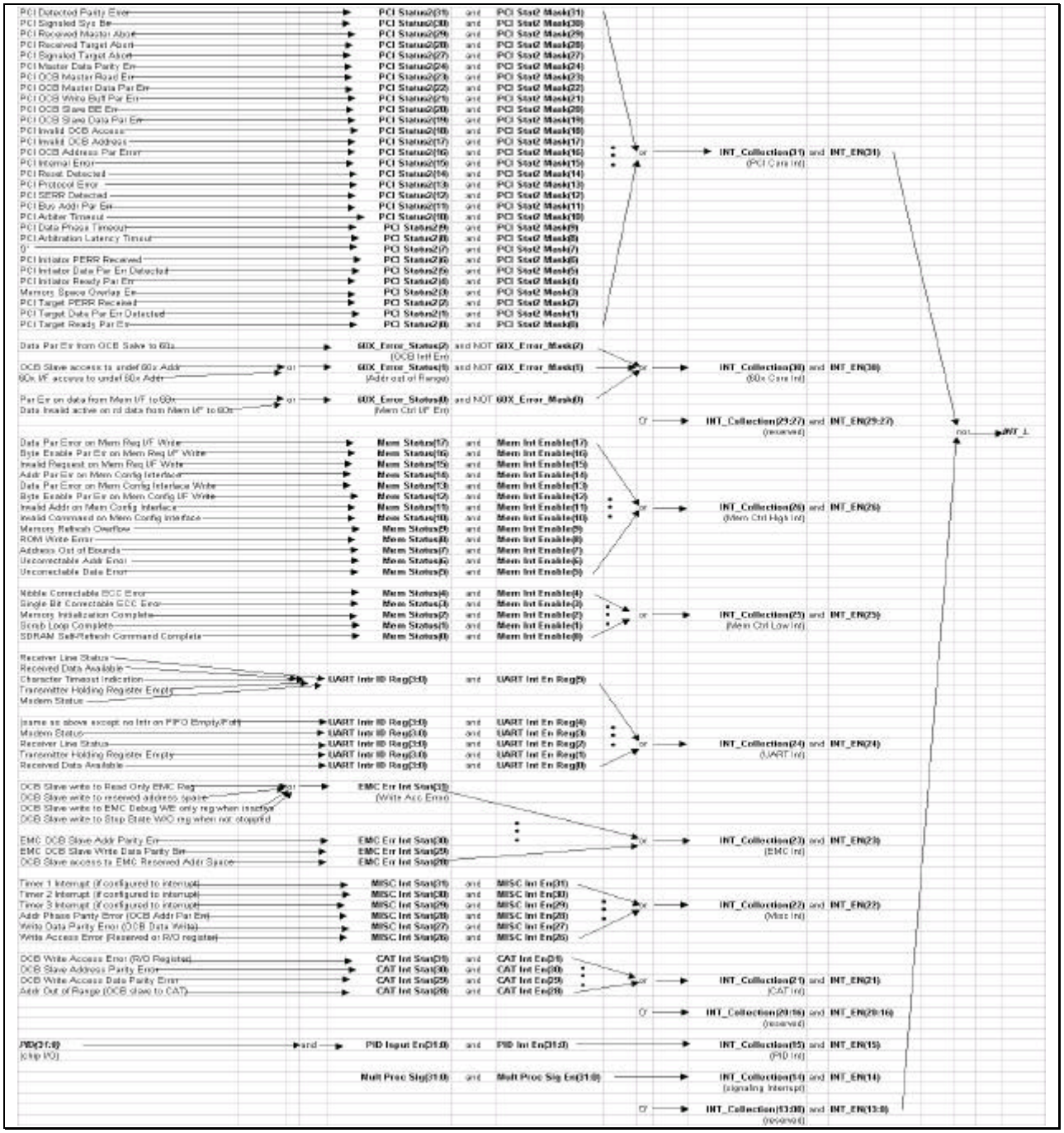


Figure 59: Power PCI INT_L Interrupt Tree

The **Interrupt Collection** register indicates that an interrupt is present at the Power PCI inputs, **MISC_INTERNAL_INT (15:0)**. **MISC_INTERNAL_INT(15:0)** is connected to the internal chip interrupt sources as shown in the following table:

Table 21 - Interrupt Register Definition

Bit	Source	Description
15	PCI Bus	PCI Function Interrupt
14	60x Bus	60x Function Interrupt
13:11	Reserved	0b‘0’
10	Memory Bus	Memory High Priority Interrupt
09	Memory Bus	Memory Low Priority Interrupt
08	UART	UART Interrupt
07	EMC	EMC Interrupt
06	MISC	Miscellaneous Interrupt
05	CAT	Clock & Test Interrupt
04:00	Reserved	

The **Interrupt Collection** register also provides a bit to indicate that at least one of the Programmable I/O Discretes (PID), configured as an input interrupt, is active and enabled. The source of this interrupt can be found in the **PID Input** register. Each of the PID input interrupts is enabled via the **PID Interrupt Enable** register.

The **Interrupt Collection** register also provides a bit to indicate that at least one of the Multiprocessing Signaling interrupts is active and enabled. The source of this interrupt can be found in the **Multiprocessor Signaling** register. Each of the Multiprocessor Signaling interrupts is enabled via the **Signaling Enable** register.

4.8.2 Programmable I/O Discretes (PIDs)

The 32 Programmable Interrupt/Discrete (PID) lines can be individually configurable via software to be enabled as inputs or outputs or to be disabled. The Power PCI allows software to use PIDs configured as inputs to provide controls for the internal chip timers, discrete inputs, and normal interrupts (Vector interrupts are handled by the internal EMC). Table 22 shows the configurations for each pin that are supported by the Power PCI. As an input, when a PID is configured to set the **PID Input (0x24)** Register, it will be configurable so that the **PID Input (0x24)** Register bit will be set to a one either when a level of 0b’1’ is seen on the pin or on a falling edge transition. The values in the **PID Input (0x24)** Register are configurable via software to generate interrupts or to be simply used as Discretes.

Table 22 - PID Definition Table

Signal Number	Input Functions		Output Functions	
PID 31	PID In Reg 31	Vector Interrupt 5	PID Out Reg 31	
PID 30	PID In Reg 30	Vector Interrupt 4	PID Out Reg 30	
PID 29	PID In Reg 29	Vector Interrupt 3	PID Out Reg 29	
PID 28	PID In Reg 28	Vector Interrupt 2	PID Out Reg 28	
PID 27	PID In Reg 27	Vector Interrupt 1	PID Out Reg 27	
PID 26	PID In Reg 26	Vector Interrupt 0	PID Out Reg 26	
PID 25	PID In Reg 25		PID Out Reg 25	
PID 24	PID In Reg 24		PID Out Reg 24	
PID 23	PID In Reg 23		PID Out Reg 23	
PID 22	PID In Reg 22		PID Out Reg 22	
PID 21	PID In Reg 21		PID Out Reg 21	

Signal Number	Input Functions		Output Functions	
PID 20	PID In Reg 20		PID Out Reg 20	
PID 19	PID In Reg 19		PID Out Reg 19	
PID 18	PID In Reg 18		PID Out Reg 18	
PID 17	PID In Reg 17		PID Out Reg 17	
PID 16	PID In Reg 16		PID Out Reg 16	
PID 15	PID In Reg 15		PID Out Reg 15	
PID 14	PID In Reg 14		PID Out Reg 14	
PID 13	PID In Reg 13	PTIM3 Clear	PID Out Reg 13	
PID 12	PID In Reg 12	PTIM3 Snapshot	PID Out Reg 12	
PID 11	PID In Reg 11	PTIM3 Clock	PID Out Reg 11	
PID 10	PID In Reg 10	PTIM2 Clear	PID Out Reg 10	
PID 09	PID In Reg 09	PTIM2 Snapshot	PID Out Reg 09	
PID 08	PID In Reg 08	PTIM2 Clock	PID Out Reg 08	
PID 07	PID In Reg 07	PTIM1 Clear	PID Out Reg 07	
PID 06	PID In Reg 06	PTIM1 Snapshot	PID Out Reg 06	
PID 05	PID In Reg 05	PTIM 1 Clock	PID Out Reg 05	
PID 04	PID In Reg 04		PID Out Reg 04	WD Timer Heartbeat
PID 03	PID In Reg 03		PID Out Reg 03	WD Timer Expired
PID 02	PID In Reg 02		PID Out Reg 02	PTIM 3 Output
PID 01	PID In Reg 01		PID Out Reg 01	PTIM 2 Output
PID 00	PID In Reg 00		PID Out Reg 00	PTIM 1 Output

The PIDs subfunction provides the capability to program each of the **MISC_IO_PID(31:0)** signals as an input discrete, input interrupt, input timer control, output discrete, or timer output. Several configuration registers are required to control the PIDs:

PID Output Enable

PID Output Select

PID Output

PID Input Enable

PID Input Select

PID Interrupt Enable

PID Vector Enable

PID Input

Each PID is configurable as an output via the **PID Output Enable** register. The **PID Output Select** register determines if the output is to be sourced from the **PID Output** register or from the Timer subfunction.

Each PID is configurable as an input via the **PID Input Enable** register. Each input PID is considered asynchronous and is synchronized to the System Clock by double latching. Each input PID is further configurable as negative edge sensitive or level sensitive via the **PID Input Select** register. For negative edge sensitive inputs, the corresponding bit in the **PID Input** register is set when a logical 0b'1' is sampled followed by a logical 0b'0' on the subsequent clock cycle. The **PID Input** register bit is cleared by writing a 0b'1'. For level sensitive inputs, the corresponding bit in the **PID Input** register follows the inverted state of the input.

The **PID Interrupt Enable** register enables each bit of the **PID Input** register to generate the PID Interrupt bit of the **Interrupt Collection** register. The **PID Vector Enable** register enables bits 31 to 26 of the **PID Input** register to generate **MISC_EMC_VECTOR_INT(5:0)** respectively.

4.8.3 Timers

The Power PCI contains 3 programmable timers and a watchdog timer.

4.8.3.1 Programmable Timers

The Power PCI contains three identical programmable timers. The general timer operation will only be described once. A reference to *N* indicates a resource that is common to the three timers and will be instantiated three times using the integer values 1, 2, and 3.

The timer is a programmable 32-bit up/down counter. The direction is determined by the Count Direction bit of the **PTIM *N* Configuration** register. The Enable bit of the **PTIM *N* Configuration** register enables/suspend the counter operation. The timer is accessed via the **PTIM *N* Timer** register.

The timer is capable of being clocked internally or externally, as determined by the Clock Source bit of the **PTIM *N* Configuration** register. The frequency for internally clocked timers is determined by the Frequency Select bit of the **PTIM *N* Configuration** register. This allows for selection between divide by 4 or 8 from the Real Time Clock. The timer uses the System Clock to oversample the divided Real Time Clock in updating the counter. Thus the System Clock must be at least twice the frequency of the divided Real Time Clock. Externally clocked timer configuration utilizes one of the PIDs as a clock source. The PID must be configured as an input. The negative edge transition is used to update the counter. The System Clock is used to oversample the PID, thus the System Clock must be a least twice the frequency of the PID.

Terminal count for an up count configured timer is 0x'FFFF FFFF'. Terminal count for a down count configured timer is X'0000 0000'. Upon reaching terminal count, the timer either reloads from the **PTIM *N* Reload** register and continues counting or stops counting as determined by the Terminal Event bit of the **PTIM *N* Configuration** register. Upon reaching terminal count the timer also sets the Timer *N* Interrupt bit in the **Misc Interrupt Status** register or **Vector Interrupt Status** register as selected by the Terminal Signal bit of the **PTIM *N* Configuration** register. Upon reaching terminal count the timer also pulses or toggles the output PID (provided the PID has been configured as a timer output) as determined by the Pulse/Toggle bit of the **PTIM *N* Configuration** register. An output pulse is active low for one timer clock period. Toggle mode causes the output PID state to invert upon reaching terminal count.

The timer is capable of storing a snapshot of the current count into the **PTIM *N* Reload** register upon detecting a negative edge on the PID configured for snapshot trigger. This capability is enabled via the Snapshot bit of the **PTIM *N* Configuration** register.

The timer is capable of being cleared via an active low PID. This capability is enabled via the External Clear bit of the **PTIM *N* Configuration** register.

Simultaneous write access to the **PTIM *N* Timer** register has the following priority:

- Clearing via an enabled external clear.
- Write access from the RAD750.
- Count update.

Simultaneous write access to the **PTIM *N* Reload** register has the following priority:

- Write access from the RAD750.
- Snapshot update.

All PIDs used as inputs to the timer function must be configured as inputs via the **PID Input Enable** register. All PIDs used as outputs must be configured as timers outputs via the **PID Output Enable** register and **PID Output Select** register. The PID mapping for the timer functions is as shown in Table 23.

Table 23 - PID Mapping to Timer Function

PID	Mapped Function
MISC_IO_PID(0)	PTIM 1 Output
MISC_IO_PID(1)	PTIM 2 Output
MISC_IO_PID(2)	PTIM 3 Output
MISC_IO_PID(5)	PTIM 1 External Clock
MISC_IO_PID(6)	PTIM 1 Snapshot Trigger
MISC_IO_PID(7)	PTIM 1 External Clear
MISC_IO_PID(8)	PTIM 2 External Clock
MISC_IO_PID(9)	PTIM 2 Snapshot Trigger
MISC_IO_PID(10)	PTIM 2 External Clear
MISC_IO_PID(11)	PTIM 3 External Clock
MISC_IO_PID(12)	PTIM 3 Snapshot Trigger
MISC_IO_PID(13)	PTIM 3 External Clear

4.8.3.2 Watchdog Timer

The watchdog timer is a 32-bit down counter. The timer is accessed via the **Watchdog Timer** register.

The timer is clocked at a frequency of the Real Time Clock divided by 8. The timer uses the System Clock to oversample the divided Real Time Clock in updating the counter. Thus, the System Clock must be at least twice the frequency of the divided Real Time Clock.

Upon reaching a terminal count of 0x'0000 0000', the timer stops counting, sets the WDT Expired bit of the **Vector Interrupt Status** register, and drives **MISC_IO_PID(3)** active low, provided the PID is configured as a timer output. The WDT Expired bit of the **Vector Interrupt Status** register causes the Power PCI to activate **MISC_EMC_VECTOR_INT(7),** provided the corresponding bit in the **Vector Interrupt Enable** register is set.

Every write access to the **Watchdog Timer** register from the RAD750 causes the Power PCI to drive an active low single timer cycle heartbeat pulse out **MISC_IO_PID(4)** provided the PID is configured as a timer output.

4.8.4 Multiprocessor Support

The Power PCI has two facilities to support multiple processor configurations. These consist of 32 hardware semaphores and 32 processor signaling interrupts.

The 32 semaphores are stored in the 32-bit **Semaphore** register. Each semaphore bit is mapped to the lsb of each of 32 contiguous bytes of the address space called the **SemTake/SemGive** registers.

A semaphore take operation is attempted when the byte mapped to the particular semaphore in the **SemTake/SemGive** address space is read via the RAD750. If the lsb of the data returned is a 0b'0', then the semaphore take was successful and the Power PCI sets the semaphore bit. If the lsb of the data returned is a 0b'1', then the semaphore take was not successful and the device should try again later.

A semaphore give operation occurs when the byte mapped to the particular semaphore in the **SemTake/SemGive** address space is written with the lsb of the data byte being 0b'0'. The Power PCI clears the semaphore.

The 32 multiprocessor signaling bits are stored in the **Multiprocessor Signaling** register. This register is mapped to two 32-bit addresses. The first mapping is provided read with bit write set (a write of 0b'1' sets the corresponding bit position). The second mapping is provided read with bit write clear (a write of 0b'1' clears the corresponding bit position). Each bit of the **Multiprocessor Signaling** register is and-or reduced with the corresponding bits of the **Signaling Enable** register to generate the Multiprocessor Signal bit of the **Interrupt Collection** register.

4.8.5 CPU Support

CPU support includes the handling of reset, machine check, and checkstop with the CPU.

4.8.5.1 CPU Reset

During flush reset, the Power PCI drives **MISC_IO_HRESET_L** active. After flush reset, the state of **MISC_IO_RESET_L** is a function of the HRESET bit of the **CPU Discretes (0x28)** register and **IO_MISC_EMC_ENABLE** as shown in Table 24.

Table 24 - HRESET Source

IO_MISC_EMC_ENABLE	MISC_IO_HRESET_L
0	Inactive
1	Determined by HRESET bit of **CPU Discretes (0x28)** register

4.8.5.2 Machine Check Interrupt

The Power PCI Bridge chip provides the **MCP Collection** register which indicates status as to the source of a Machine Check interrupt. Each bit in the **MCP Collection** register is reduced through an and-or function with each corresponding bit in the **MCP Enable** register to generate the single RAD750 Machine Check interrupt. Each of the bits in the **MCP Collection** register represents free running status of the interrupting source. That is when the source of the interrupt is removed the corresponding bit in the **MCP Collection** register will clear. The Machine Check interrupt sources is the signals **MISC_INTERNAL_MCP(1:0)** and **IO_MISC_NMI**.

The Machine Check Interrupt is further enabled via the **MISC_MCP_ENABLE** input. This bit is intended to be connected (in a MPC106 compliant bridge chip) to the MPC106 defined register PICR1[MCP_EN]. This signal only affects the assertion of the Machine Check Interrupt, not the setting of bits within the **MCP Collection** register

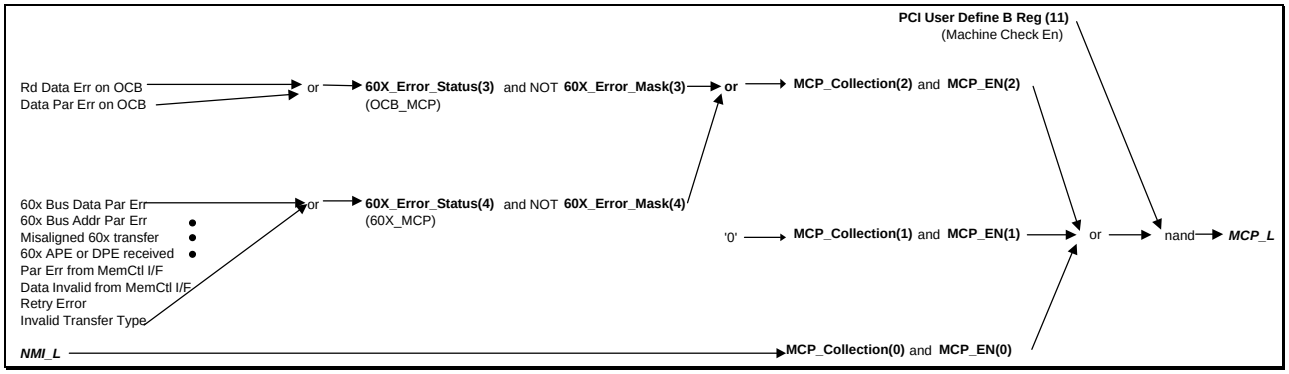


Figure 60: Power PCI MCP Generation Tree

The 60x interface function can send an **MISC_INTERNAL_MCP** signal to the collection register. The table below shows the main conditions that will generate the MCP signal. Please note that internal to the Power PCI, there are two MCP error bits. One if the error was detected on the 60x bus side of the transfer and another if the error was detected on the Power PCI internal bus side of the transfer.

Table 25 - Error Mechanisms

Error	Detection	Isolation	Reporting	Removal	Recovery
RD Data Error signal received by Power PCI internal bus Master Subcore	Signal active	Pass Read Data with Parity	Set "Power PCI internal bus MCP" bit in 60x Error Status Register	Write of 0b'1' to "Power PCI internal bus MCP" bit in 60x Error Status Register.	External
Data Parity Error in data read by 60x Subcore from Power PCI internal bus Master Subcore	Check parity across data and data error from Power PCI internal bus	Regenerate Parity on Data	Set "Power PCI internal bus MCP" bit in 60x Error Status Register	Write of 0b'1' to "Power PCI internal bus MCP" bit in 60x Error Status Register.	External
Received Parity Error on Data from 60x Bus Interface	Check Parity on data from 60x Interface	To Memory: Pass Data with incorrect parity; To Power PCI internal bus: activate **WR_D_ERR** signal; To internal 60x: discard data	Set "60x MCP" bit in 60x Error Status Register	Write of 0b'1' to "60x MCP" bit in 60x Error Status Register.	External
Received Parity Error on Address from 60x Bus Interface	Check Parity on address from 60x Interface	Terminate address transfer by asserting **P60X_IO_ARTRY_L**	Set "60x MCP" bit in 60x Error Status Register and store address information	Write of 0b'1' to "60x MCP" bit in 60x Error Status Register.	External
Received Illegal Misaligned transfer that crosses 32-bit boundary from 60x Bus Interface	Check for illegal misaligned transfers from 60x Interface	Terminate address transfer by asserting **P60X_IO_ARTRY_L**	Set "60x MCP" bit in 60x Error Status Register and store address information	Write of 0b'1' to "60x MCP" bit in 60x Error Status Register.	External
Received **IO_P60X_APE_L** or **IO_P60X_DPE**_L is active	Monitor signals	Operation continues	Set "60x MCP" bit in 60x Error Status Register	Write of 0b'1' to "60x MCP" bit in 60x Error Status Register.	External
Parity Error on Data from Memory Controller Interface	Check Parity on data from Memory Controller Interface	To Power PCI internal bus: activate RD_D_ERR signal; To 60x: send correct parity on 60x bus	Set "Memory Controller Interface Error" bit in 60x Error Status Register; To 60x: Also Set "60x MCP" bit in 60x Error Status Register	Write of 0b'1' to "Memory Controller Interface Error" bit in 60x Error Status Register. Write of 0b'1' to "60x MCP" bit in 60x Error Status Register.	External
Data Invalid active on read data from Memory Controller Interface	Monitor Data Invalid line from Memory Controller Interface	To Power PCI internal bus: activate RD_D_ERR signal; To 60x: send correct parity on 60x bus	Set "Memory Controller Interface Error" bit in 60x Error Status Register; To 60x: Also Set "60x MCP" bit in 60x Error Status Register	Write of 0b'1' to "Memory Controller Interface Error" bit in 60x Error Status Register. Write of 0b'1' to "60x MCP" bit in 60x Error Status Register.	External
Retry Error	Address on a retried read to the Power PCI internal bus does not match the original	Operation Continues	Set "60x MCP" bit in 60x Error Status Register	Write of 0b'1' to "60x MCP" bit in 60x Error Status Register.	External

Error	Detection	Isolation	Reporting	Removal	Recovery
	address				
Invalid Transfer Type from the 60x interface	Check for invalid Transfer Type	Terminate Address transfer by asserting **P60X_IO_ARTRY_L**	Set "60x MCP" bit in 60x Error Status Register	Write of 0b'1' to "60x MCP" bit in 60x Error Status Register.	External

4.8.5.3 CPU Checkstop

The Power PCI statuses the state of the primary input **IO_MISC_CKSTP_L** in the Checkstop bit of the **Vector Interrupt Status** register. While the Checkstop bit is active and the corresponding enable in the **Vector Interrupt Enable** register is active then the Power PCI activates **MISC_EMC_VECTOR_INT(5)**.

The Power PCI activates the primary output **MISC_IO_CKSTP_L** when a Critical Error bit is active in the **Vector Interrupt Status** register, the corresponding bit in the **Vector Interrupt Enable** register is active, and the checkstop Enable bit of the **RAD750 Discretes** register is set. Figure 61 shows the actual logic tree used in the Power PCI for this function. The Power PCI latches the **MISC_IO_CKSTP_L** signal.

Sources of critical errors include internal chip errors such as SEU errors causing hot-bit state machines to be in two states at once, internal bus parity errors, etc. These errors are classified as critical because they will generally require a reset from the EMC to recover from. The Power PCI chip sends the Checkstop signal to the RAD750 which will then stop all operation and take place its I/O in high impedance state (safe mode).

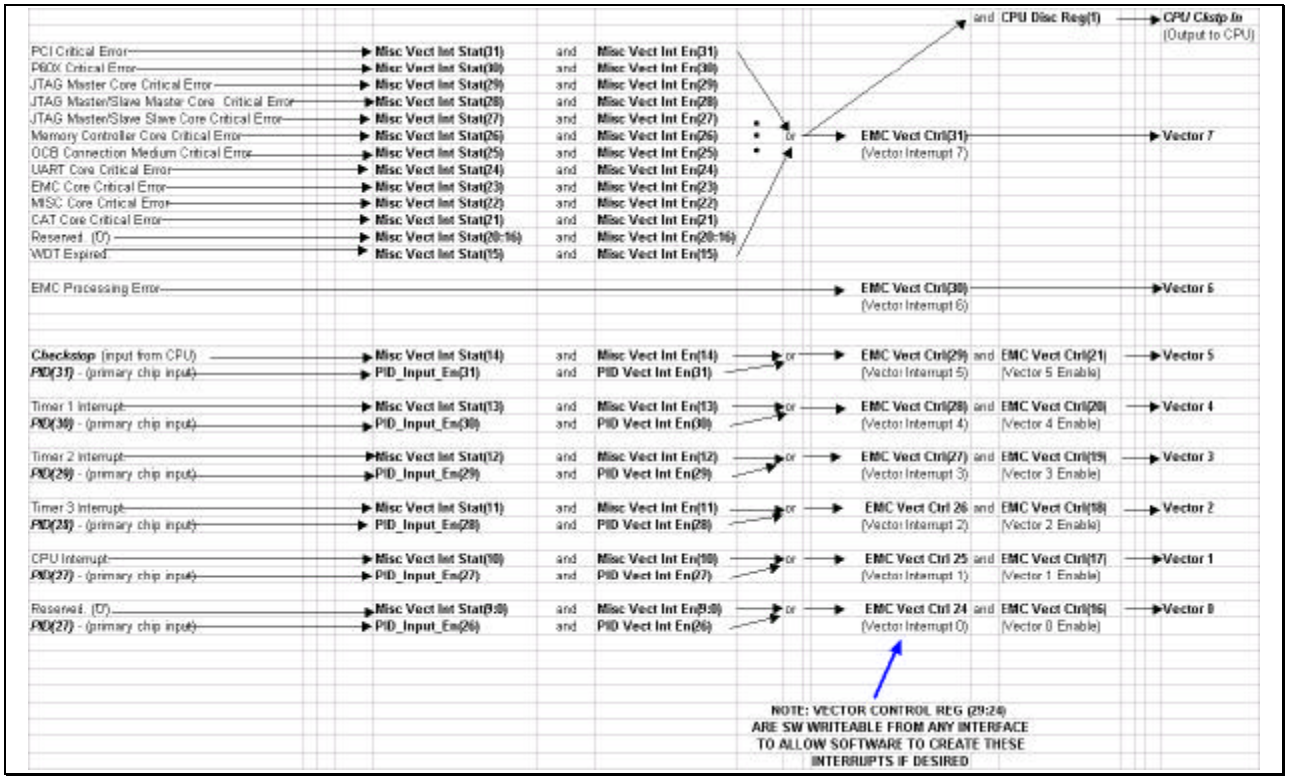


Figure 61: Power PCI Checkstop Generation Tree and Vector Interrupt Tree

4.8.6 Critical Error Support

Internal to the Power PCI is a function that collects critical error signals caused by SEUs. These signals set Critical Error bits in the **Vector Interrupt Status** register. The Critical Error and WDT Expired bits of

the **Vector Interrupt Status** register are and-or reduced with the corresponding bits of the **Vector Interrupt Enable** register to generate **MISC_EMC_VECTOR_INT(7)**.

The **Vector Interrupt Status** register also contains the Checkstop, Timer *N* Interrupt, and CPU Interrupt bits. These bits are ANDed with the corresponding bits of the **Vector Interrupt Enable** register to generate **MISC_EMC_VECTOR_INT(5:1)** respectively.

4.8.7 Register Matrix

From the RAD750 perspective, the Base Address for the Miscellaneous Function Registers is 0x'BF88 0000'. From the PCI Bus perspective, the Base Address is 0x'BA28 0000'. The Miscellaneous function contains the registers as described in Table 26.

Table 26 - Power PCI Register/Resource Map

Register / Array / Command	Address	Access	Reset State	Notes	Page
Interrupt Collection	0b'0000'	R	0b'0000 0000'		143
Interrupt Enable	0b'0004'	R/W	0b'0000 0000'		143
PID Output Enable	0b'0008'	R/W	0b'0000 0000'		144
PID Output Select	0b'000C'	R/W	0b'0000 0000'		144
PID Output	0b'0010'	R/W	0b'0000 0000'		144
PID Input Enable	0b'0014'	R/W	0b'0000 0000'		144
PID Input Select	0b'0018'	R/W	0b'0000 0000'		145
PID Interrupt Enable	0b'001C'	R/W	0b'0000 0000'		145
PID Vector Enable	0b'0020'	R/W	0b'0000 0000'		145
PID Input	0b'0024'	R/W	0b'0000 0000'		145
CPU Discretes	0b'0028'	R/W	0b'0000 0000'		145
MCP Collection	0b'002C'	R	0b'0000 0000'		152
MCP Enable	0b'0030'	R/W	0b'0000 0000'		152
PTIM 1 Configuration	0b'0034'	R/W	0b'0000 0000'		146
PTIM 1 Timer	0b'0038'	R/W	0b'0000 0000'		147
PTIM 1 Reload	0b'003C'	R/W	0b'0000 0000'		148
PTIM 2 Configuration	0b'0040'	R/W	0b'0000 0000'		148
PTIM 2 Timer	0b'0044'	R/W	0b'0000 0000'		149
PTIM 2 Reload	0b'0048'	R/W	0b'0000 0000'		149
PTIM 3 Configuration	0b'004C'	R/W	0b'0000 0000'		149
PTIM 3 Timer	0b'0050'	R/W	0b'0000 0000'		150
PTIM 3 Reload	0b'0054'	R/W	0b'0000 0000'		151
Watchdog Timer	0b'0058'	R/W	0b'FFFF FFFF'		151
Semaphore	0b'005C'	R	0b'0000 0000'		151
SemTake/SemGive	0b'0060 – 007C'	R/W	0b'0000 0000'		151
Multiprocessor Signaling	0b'0080 – 0084'	R/W	0b'0000 0000'	0b'0080' bit write set 0b'0084' bit	151

Register / Array / Command	Address	Access	Reset State	Notes	Page
				write clear	
Signaling Enable	0b'0088'	R/W	0b'0000 0000'		152
Vector Interrupt Status	0b'008C'	R/W	0b'0000 0000'		152
Vector Interrupt Enable	0b'0090'	R/W	0b'0000 0000'		154
Misc Interrupt Status	0b'0094'	R/W	0b'0000 0000'		154
Misc Interrupt Enable	0b'0098'	R/W	0b'0000 0000'		155
Revision ID	0b'009C'	R	hardwired		

4.8.8 Register Descriptions

4.8.8.1 Interrupt Collection (0x00)

This register contains status as to the source of a CPU interrupt.

Bits 31:16	***Internal Interrupt(15:0):*** These bits represent free running status of the **MISC_INTERNAL_INT(15:0)** inputs. 0b'0' – No internal interrupt 0b'1' – Internal interrupt The reset value of these bits is 0x'0000'.
Bit 15	***PID Interrupt:*** This free running status bit indicates that there is an active bit in the **PID Input** register for which the corresponding bit in the **PID Interrupt Enable** register is also set. 0b'0' – No PID interrupt 0b'1' – PID interrupt The reset value of this bit is 0b'0'.
Bit 14	***Signaling Interrupt:*** This free running status bit indicates that there is an active bit in the **Multiprocessor Signaling** register for which the corresponding bit in the **Signaling Enable** register is also set. 0b'0' – No Signaling interrupt 0b'1' – Signaling interrupt The reset value of this bit is 0b'0'.
Bits 13:0	***Reserved***: These bits are reserved, and return 0b'0' when read.

4.8.8.2 Interrupt Enable (0x04)

This register enables the **Interrupt Collection** register to generate a CPU interrupt.

Bits 31:16	***Internal Interrupt Enable(15:0):*** 0b'0' – Disabled 0b'1' – Enabled The reset value of these bits is 0x'0000'.
Bit 15	***PID Interrupt Enable:*** 0b'0' – Disabled

	0b'1' – Enabled The reset value of this bit is 0b'0'.
Bit 14	***Signaling Interrupt Enable:*** 0b'0' – Disabled 0b'1' – Enabled The reset value of this bit is 0b'0'.
Bits 13:0	***Reserved***: These bits are reserved, and return 0b'0' when read.

4.8.8.3 PID Output Enable (0x08)

This register allows each of the **MISC_IO_PID(31:0)** to be configured as a primary output.

Bits 31:0	***PID Output Enable(31:0):*** Each bit enables the corresponding **MISC_IO_PID(31:0)** to be driven as a primary output. 0b'0' – Disabled 0b'1' – Enabled The reset value of these bits is 0x'0000 0000'.

4.8.8.4 PID Output Select (0x0C)

This register determines the source for an output PID.

Bits 31:5	***Reserved***: These bits are reserved, and return 0b'0' when read.
Bit 4:0	***PID Output Select(4:0):*** Each bit selects the source for the corresponding **MISC_IO_PID(4:0)** outputs. 0b'0' – **PID Output** register 0b'1' – Timer subfunction The reset value of these bits is 0b'00000'.

4.8.8.5 PID Output (0x10)

This register contains the data value to be driven onto a properly configured output PID.

Bits 31:0	***PID Output(31:0):*** Each bit is driven onto the corresponding **MISC_IO_PID(31:0)** outputs provided the PID has been configured as an output and selected to be sourced from the **PID Output** register. The reset value of these bits is 0x'0000 0000'.

4.8.8.6 PID Input Enable (0x14)

This register allows each of the **MISC_IO_PID(31:0)** to be a primary input.

Bits 31:0	***PID Input Enable(31:0):*** Each bit configures the corresponding **MISC_IO_PID(31:0)** as an input PID. 0b'0' – Disabled 0b'1' – Enabled The reset value of these bits is 0x'0000 0000'.

4.8.8.7 PID Input Select (0x18)

This register determines edge or level sensitivity of an input PID.

Bits 31:0	***PID Input Select(31:0):*** Each bit configures the corresponding **MISC_IO_PID(31:0)** input sensitivity. 0b'0' – Active low level sensitive 0b'1' – Negative edge sensitive The reset value of these bits is 0x'0000 0000'.

4.8.8.8 PID Interrupt Enable (0x1C)

This register enables an input PID to generate a PID interrupt.

Bits 31:0	***PID Interrupt Enable(31:0):*** Each bit enables the corresponding bit in the **PID Input** register to generate the PID Interrupt bit of the **Interrupt Collection** register. 0b'0' – Disabled 0b'1' – Enabled The reset value of these bits is 0x'0000 0000'.

4.8.8.9 PID Vector Enable (0x20)

This register enables an input PID to generate **Vector Interrupt(5:0).**

Bits 31:26	***PID Vector Enable(31:26):*** Each bit enables the corresponding bit in the **PID Input** register to generate **MISC_EMC_VECTOR_INT(5:0),** respectively. 0b'0' – Disabled 0b'1' – Enabled The reset value of these bits is 0b'00000'.
Bits 25:0	***Reserved***: These bits are reserved, and return 0b'0' when read.

4.8.8.10 PID Input (0x24)

This register contains the data received for those PIDs configured as inputs.

Bits 31:0	***PID Input(31:0):*** Each bit represents the active state of the corresponding **MISC_IO_PID(31:0)** configured as input PIDs. Negative edge sensitive: Each bit is set on the transition from 0b'1' to 0b'0' of the corresponding input PID. Each bit is cleared by a write of a 0b'1'. Active low sensitive: Each bit represents the inverted state of the corresponding input PID. Write access has no affect. 0b'0' – Inactive PID 0b'1' – Active PID The reset value of these bits is 0x'0000 0000'.

4.8.8.11 CPU Discretes (0x28)

This register contains discrete control information for the RAD750.

Bits 31:4	***Reserved***: These bits are reserved, and return 0b'0' when read.
Bit 3	***Critical Error Disable:*** This bit provides the inverted source for **MISC_CRITICAL_ERROR_ENABLE**. 0b'0' – Enabled 0b'1' – Disabled The reset value of this bit is 0b'0'.
Bit 2	***Critical Error Injection:*** This bit when set causes the Power PCI to activate **MISC_CRITICAL_ERROR,** provided that **MISC_CE_INJECT_EN** is set. 0b'0' – No critical error injection 0b'1' – Critical error injection The reset value of this bit is 0b'0'.
Bit 1	***Checkstop Enable:*** This bit enables the driving of **MISC_IO_CKSTP_L** active whenever an enabled Critical Error is active in the **Vector Interrupt Status** register. 0b'0' – Disabled 0b'1' – Enabled The reset value of this bit is 0b'0'.
Bit 0	***HRESET:*** This bit determines the state of **MISC_IO_HRESET_L,** provided **IO_MISC_EMC_ENABLE** is active. 0b'0' – Inactive (logic 1) 0b'1' – Active (logic 0) The reset value of this bit is 0b'1'.

4.8.8.12 PTIM 1 Configuration (0x34)

This register contains configuration information for the PTIM 1 Timer.

Bit 31	***Enable:*** This bit enables the **PTIM 1 Timer** to count and to source terminal count events. 0b'0' – Suspend 0b'1' – Enable The reset value of this bit is 0b'0'.
Bit 30	***Count Direction:*** This bit determines the count direction. 0b'0' – Count Down 0b'1' – Count Up The reset value of this bit is 0b'0'.
Bit 29	***Clock Source:*** This bit determines the source of the clock. 0b'0' – Internal

	0b'1' – **MISC_IO_PID(5)** The reset value of this bit is 0b'0'.
Bit 28	***Frequency Select:*** This bit determines the timer frequency for internal clock sources. 0b'0' – Real Time Clock divided by 4 0b'1' – Real Time Clock divided by 8 The reset value of this bit is 0b'0'.
Bit 27	***Terminal Event:*** This bit determines if the timer is to reload or stop upon reaching terminal count. 0b'0' – Stop 0b'1' – Reload The reset value of this bit is 0b'0'.
Bit 26	***Terminal Signal:*** This bit selects the destination interface for posting a terminal count interrupt. 0b'0' – **Misc Interrupt Status** register 0b'1' – **Vector Interrupt Status** register The reset value of this bit is 0b'0'.
Bit 25	***Pulse/Toggle:*** This bit determines signaling mode for the timer output PID. 0b'0' – Pulse mode 0b'1' – Toggle mode The reset value of this bit is 0b'0'.
Bit 24	***Snapshot:*** This bit enables the timer to store a snapshot upon receiving a snapshot trigger. 0b'0' – Disabled 0b'1' – Enabled The reset value of this bit is 0b'0'.
Bit 23	***External Clear:*** This bit enables the properly configured input PID to act as an external clear. 0b'0' – Disabled 0b'1' – Enabled The reset value of this bit is 0b'0'.
Bits 22:0	***Reserved***: These bits are reserved, and return 0b'0' when read.

4.8.8.13 PTIM 1 Timer (0x38)

This register is the counter for PTIM 1.

Bits 31:0	***Timer Value:*** The reset value of these bits is 0x'0000 0000'.

4.8.8.14 PTIM 1 Reload (0x3C)

This register contains the reload value or snapshot value for PTIM 1.

Bits 31:0	***Reload Value:*** The reset value of these bits is 0x'0000 0000'.

4.8.8.15 PTIM 2 Configuration (0x40)

This register contains configuration information for the PTIM 2 Timer.

Bit 31	***Enable:*** This bit enables the **PTIM 2 Timer** to count and to source terminal count events. 0b'0' – Suspend 0b'1' – Enable The reset value of this bit is 0b'0'.
Bit 30	***Count Direction:*** This bit determines the count direction. 0b'0' – Count Down 0b'1' – Count Up The reset value of this bit is 0b'0'.
Bit 29	***Clock Source:*** This bit determines the source of the clock. 0b'0' – Internal 0b'1' – **MISC_IO_PID(8)** The reset value of this bit is 0b'0'.
Bit 28	***Frequency Select:*** This bit determines the timer frequency for internal clock sources. 0b'0' – Real Time Clock divided by 4 0b'1' – Real Time Clock divided by 8 The reset value of this bit is 0b'0'.
Bit 27	***Terminal Event:*** This bit determines if the timer is to reload or stop upon reaching terminal count. 0b'0' – Stop 0b'1' – Reload The reset value of this bit is 0b'0'.
Bit 26	***Terminal Signal:*** This bit selects the destination interface for posting a terminal count interrupt. 0b'0' – **Misc Interrupt Status** register 0b'1' – **Vector Interrupt Status** register The reset value of this bit is 0b'0'.
Bit 25	***Pulse/Toggle:*** This bit determines signaling mode for the timer output PID.

	0b'0' – Pulse mode 0b'1' – Toggle mode The reset value of this bit is 0b'0'.
Bit 24	***Snapshot:*** This bit enables the timer to store a snapshot upon receiving a snapshot trigger. 0b'0' – Disabled 0b'1' – Enabled The reset value of this bit is 0b'0'.
Bit 23	***External Clear:*** This bit enables the properly configured input PID to act as an external clear. 0b'0' – Disabled 0b'1' – Enabled The reset value of this bit is 0b'0'.
Bits 22:0	***Reserved***: These bits are reserved, and return 0b'0' when read.

4.8.8.16 PTIM 2 Timer (0x44)

This register is the counter for PTIM 2.

Bits 31:0	***Timer Value:*** The reset value of these bits is 0x'0000 0000'.

4.8.8.17 PTIM 2 Reload (0x48)

This register contains the reload value or snapshot value for PTIM 2.

Bits 31:0	***Reload Value:*** The reset value of these bits is 0x'0000 0000'.

4.8.8.18 PTIM 3 Configuration (0x4C)

This register contains configuration information for the PTIM 3 Timer.

Bit 31	***Enable:*** This bit enables the **PTIM 3 Timer** to count and to source terminal count events. 0b'0' – Suspend 0b'1' – Enable The reset value of this bit is 0b'0'.
Bit 30	***Count Direction:*** This bit determines the count direction. 0b'0' – Count Down 0b'1' – Count Up The reset value of this bit is 0b'0'.
Bit 29	***Clock Source:*** This bit determines the source of the clock. 0b'0' – Internal

	0b'1' – **MISC_IO_PID(11)** The reset value of this bit is 0b'0'.
Bit 28	***Frequency Select:*** This bit determines the timer frequency for internal clock sources. 0b'0' – Real Time Clock divided by 4 0b'1' – Real Time Clock divided by 8 The reset value of this bit is 0b'0'.
Bit 27	***Terminal Event:*** This bit determines if the timer is to reload or stop upon reaching terminal count. 0b'0' – Stop 0b'1' – Reload The reset value of this bit is 0b'0'.
Bit 26	***Terminal Signal:*** This bit selects the destination interface for posting a terminal count interrupt. 0b'0' – **Misc Interrupt Status** register 0b'1' – **Vector Interrupt Status** register The reset value of this bit is 0b'0'.
Bit 25	***Pulse/Toggle:*** This bit determines signaling mode for the timer output PID. 0b'0' – Pulse mode 0b'1' – Toggle mode The reset value of this bit is 0b'0'.
Bit 24	***Snapshot:*** This bit enables the timer to store a snapshot upon receiving a snapshot trigger. 0b'0' – Disabled 0b'1' – Enabled The reset value of this bit is 0b'0'.
Bit 23	***External Clear:*** This bit enables the properly configured input PID to act as an external clear. 0b'0' – Disabled 0b'1' – Enabled The reset value of this bit is 0b'0'.
Bits 22:0	***Reserved***: These bits are reserved, and return 0b'0' when read.

4.8.8.19 PTIM 3 Timer (0x50)

This register is the counter for PTIM 3.

Bits 31:0	***Timer Value:*** The reset value of these bits is 0x'0000 0000'.

4.8.8.20 PTIM 3 Reload (0x54)

This register contains the reload value or snapshot value for PTIM 3.

Bits 31:0	***Reload Value:*** The reset value of these bits is 0x'0000 0000'.

4.8.8.21 Watchdog Timer (0x58)

This register is the counter for the Watchdog Timer.

Bits 31:0	***Timer Value:*** The reset value of these bits is 0x'FFFF FFFF'.

4.8.8.22 Semaphore (0x5C)

This register provides a single nondestructive read access to all 32 semaphores.

Bits 31:0	***Semaphore(31:0):*** The reset value of these bits is 0x'0000 0000'.

4.8.8.23 SemTake/SemGive (0x60,¼, 0x7C)

These registers occupy 32 contiguous bytes. Each byte contains a single semaphore bit. Semaphore bit 0 is located at byte address 0x'0060', semaphore bit 1 at 0x'0061', …semaphore bit 31 at 0x'007F'.

If a semaphore bit is clear (available) then a read access to the byte returns a zero and causes the semaphore to be set (taken). A read access that returns a 0b'1' indicates that the semaphore is taken.

Direct write access is provided to the semaphore bits. Therefore, a semaphore give operation is just a write of zero to the semaphore.

Bits 7:1	***Reserved***: These bits are reserved, and return 0b'0' when read.
Bit 0	***Semaphore(n):*** n is determined by the 5 lsb's of the byte address. 0b'0' – Available 0b'1' - Taken The reset value of this bit is 0b'0'.

4.8.8.24 Multiprocessor Signaling (0x80, ¼, 0x84)

This register is mapped twice to the address space. The first mapping provides the ability to indicate a signal via bit write set. The second mapping provides the ability to clear a signal via bit write clear.

Bits 31:0	***Signal(31:0):*** Each bit indicates that the corresponding signal is active. Write access via 0x'0080': A write of a 0b'1' causes the corresponding bit position to be set. Write access via 0x'0084': A write of a 0b'1' causes the corresponding bit position to be cleared. 0b'0' – Signal active 0b'1' – Signal inactive The reset value of these bits is 0x'0000 0000'.

4.8.8.25 Signaling Enable (0x88)

This register enables the **Multiprocessor Signaling** register to generate the Multiprocessor Signal bit of the **Interrupt Collection** register.

Bits 31:0	***Signal Enable(31:0):*** 0b'0' – Disabled 0b'1' - Enabled The reset value of these bits is 0x'0000 0000'.

4.8.8.26 MCP Collection (0x2C)

This register contains status as to the source of a Machine Check interrupt.

Bits 31:3	***Reserved***: These bits are reserved, and return 0b'0' when read.
Bits 2:1	***Internal MCP(1:0):*** These bits represent free running status of the corresponding **MISC_INTERNAL_MCP(1:0)** inputs. 0b'0' – No internal MCP 0b'1' – Internal MCP The reset value of these bits is 0b'00'.
Bit 0	***NMI:*** This bit represents free running status of the **IO_MISC_NMI** primary input. 0b'0' – No NMI 0b'1' – NMI The reset value of this bit is 0b'0'.

4.8.8.27 MCP Enable (0x30)

This register enables the **MCP Collection** register to source a Machine Check interrupt.

Bits 31:3	***Reserved***: These bits are reserved, and return 0b'0' when read.
Bits 2:1	***Internal MCP Enable(1:0):*** Each bit enables the corresponding bit of the **MCP Collection** register to generate a Machine Check interrupt. 0b'0' – Disabled 0b'1' – Enabled The reset value of these bits is 0b'00'.
Bit 0	***NMI:*** This bit enables the NMI bit of the **MCP Collection** register to generate a Machine Check interrupt. 0b'0' – Disabled 0b'1' – Enabled The reset value of this bit is 0b'0'.

4.8.8.28 Vector Interrupt Status (0x8C)

This register contains status as to the source of a Vector Interrupt. This register is output onto **MISC_JTS_CRITICAL_ERROR_REG(31:0)** for direct read from the JTAG slave.

Bits 31:16	***Critical Error(15:0):*** These bits is set by an active level on the corresponding **MISC_CRITICAL_ERROR_IN(15:0)** inputs. These bits are cleared by a write with a 0b'1' in the corresponding bit position. If enabled these bits generate Vector Interrupt(7). 0b'0' – No critical error 0b'1' – Critical error The reset value of these bits is 0x'0000'.
Bit 15	***WDT Expired:*** This bit indicates that the Watchdog Timer has expired. It is cleared by writing a 0b'1' to this bit position. If enabled this bit generates Vector Interrupt(7). 0b'0' – No Watchdog Timer expiration 0b'1' – Watchdog Timer expired The reset value of this bit is 0b'0'.
Bit 14	***Checkstop:*** This bit is set by an active level on **IO_MISC_CKSTP_L**. This bit is cleared by a write of a 0b'1' to this bit position. If enabled this bit generates a Vector Interrupt(5). 0b'0' – No Checkstop 0b'1' – Checkstop The reset value of this bit is 0b'0'.
Bit 13	***Timer 1 Interrupt:*** This bit indicates that PTIM 1 has reached terminal count and is configured to generate a Vector Interrupt. This bit is cleared by a write of a 0b'1' to this bit position. If enabled this bit generates a Vector Interrupt(4). 0b'0' – No timer interrupt 0b'1' – Timer interrupt The reset value of this bit is 0b'0'.
Bit 12	***Timer 2 Interrupt:*** This bit indicates that PTIM 2 has reached terminal count and is configured to generate a Vector Interrupt. This bit is cleared by a write of a 0b'1' to this bit position. If enabled this bit generates a Vector Interrupt(3). 0b'0' – No timer interrupt 0b'1' – Timer interrupt The reset value of this bit is 0b'0'.
Bit 11	***Timer 3 Interrupt:*** This bit indicates that PTIM 3 has reached terminal count and is configured to generate a Vector Interrupt. This bit is cleared by a write of a 0b'1' to this bit position. If enabled this bit generates a Vector Interrupt(2). 0b'0' – No timer interrupt 0b'1' – Timer interrupt

	The reset value of this bit is 0b'0'.
Bit 10	***CPU Interrupt:*** This bit represents free running status of the conditions that generate **MISC_IO_INT_L**. Writes have no affect on this bit. If enabled this bit generates a Vector Interrupt(1). 0b'0' – No CPU interrupt 0b'1' – CPU interrupt The reset value of this bit is 0b'0'.
Bits 9:0	***Reserved***: These bits are reserved, and return 0b'0' when read.

4.8.8.29 Vector Interrupt Enable (0x90)

This register enables the **Vector Interrupt Status** register to generate the respective Vector Interrupts.

Bits 31:10	***Vector Interrupt Enable(31:10):*** 0b'0' – Disabled 0b'1' – Enabled The reset value of these bits is 0x'0 0000'.
Bits 9:0	***Reserved***: These bits are reserved, and return 0b'0' when read.

4.8.8.30 Misc Interrupt Status (0x94)

This register contains status as to the source of a **MISC_INTERRUPT**. This interrupt is intended to be wrapped to one of the **MISC_INTERNAL_INT(15:0)** inputs at the Power PCI. Each bit of the **Misc Interrupt Status** register is and-or reduced with the corresponding bits of the **Misc Interrupt Enable** register to generate the **MISC_INTERRUPT**.

Bit 31	***Timer 1 Interrupt:*** This bit is set when PTIM 1 reaches terminal count and the timer has been configured to generate a Misc Interrupt. This bit is cleared by writing a 0b'1' to this bit position. 0b'0' – No Timer 1 interrupt 0b'1' – Timer 1 interrupt The reset value of this bit is 0b'0'.
Bit 30	***Timer 2 Interrupt:*** This bit is set when PTIM 2 reaches terminal count and the timer has been configured to generate a Misc Interrupt. This bit is cleared by writing a 0b'1' to this bit position. 0b'0' – No Timer 2 interrupt 0b'1' – Timer 2 interrupt The reset value of this bit is 0b'0'.
Bit 29	***Timer 3 Interrupt:*** This bit is set when PTIM 3 reaches terminal count and the timer has been configured to generate a Misc Interrupt. This bit is cleared by writing a 0b'1' to this bit position. 0b'0' – No Timer 3 interrupt 0b'1' – Timer 3 interrupt The reset value of this bit is 0b'0'.
Bit 28	***Address Phase Parity Error:*** This bit is set by a Power PCI Internal bus address parity error. This bit is cleared by writing

	a 0b'1' in this bit position. 0b'0' – No address phase parity error 0b'1' – Address phase parity error The reset value of this bit is 0b'0'.
Bit 27	***Write Data Parity Error:*** This bit is set by an Power PCI Internal bus access with bad parity. This bit is cleared by writing a 0b'1' to this bit position. 0b'0' – No write data parity error 0b'1' – Write data parity error The reset value of this bit is 0b'0'.
Bit 26	***Write Access Error:*** This bit is set by an Power PCI Internal bus access to a read only register or to a reserved address. This bit is cleared by writing a 0b'1' to this bit position. 0b'0' – No write access error 0b'1' – Write access error The reset value of this bit is 0b'0'.
Bits 25:0	***Reserved***: These bits are reserved, and return 0b'0' when read.

4.8.8.31 Misc Interrupt Enable (0x98)

This register enables the **Misc Interrupt Status** register to generate a Misc interrupt.

Bits 31:26 ***Misc Interrupt Enable(31:26):***

0b'0' – Disabled

0b'1' – Enabled

The reset value of this bit is 0b'00000'.

Bits 25:0 ***Reserved***:
These bits are reserved, and return 0b'0' when read.

4.8.8.32 Misc Revision Code (0x9C)

This register contains the hardwired Revision ID for the Miscellaneous core used in the Power PCI bridge ASIC.

Bits 31:16 ***Reserved***:
These bits are reserved, and return 0b'0' when read.

Bits 15:8 ***Misc Core ID:***

Hardwired as 0x'50'

Bits 7:0 ***Revision Number:***

Hardwired to 0b'01' indicating initial release

4.9 Endian Conventions

The Power PCI performs all required endian conversions between interfaces. The following sections describe how the various 'interfaces' will be required to perform endian translations.

4.9.1 60x Interface

As done in the MPC-106, a configuration bit can be set as either Big Endian or Little Endian. Memory accesses coming from the 60x bus must be setup to match this mode and the Power PCI performs a

direct pass of the data internal to the Power PCI. For the remaining interfaces, conversion will be made depending on the endian mode. If the mode is Big Endian, then the Power PCI performs byte swapping on the accesses. Otherwise, the Power PCI will unmunge the address. For Big Endian systems, the data is byte swapped. For Little Endian systems, the address is Munged.

4.9.2 Memory Interface

It uses memory addresses and stores data unchanged from what is passed to it. It also uses register addresses and stores data unchanged (Little Endian) from what is passed to it.

4.9.3 PCI Interface

The PCI interface is per specification Little Endian, as are its registers, so no byte ordering changes are be performed.

4.9.4 UART

The UART is a simple, byte-oriented interface no endian conversion are required. All accesses are 8 bits at a time.

4.9.5 Power PCI Flow

The following diagrams show how bytes are to be transferred between functions in each of the two endian configuration. Figure 62 below shows data transfers in Big Endian Mode, and Figure 63 shows data transfers in Little Endian Mode.

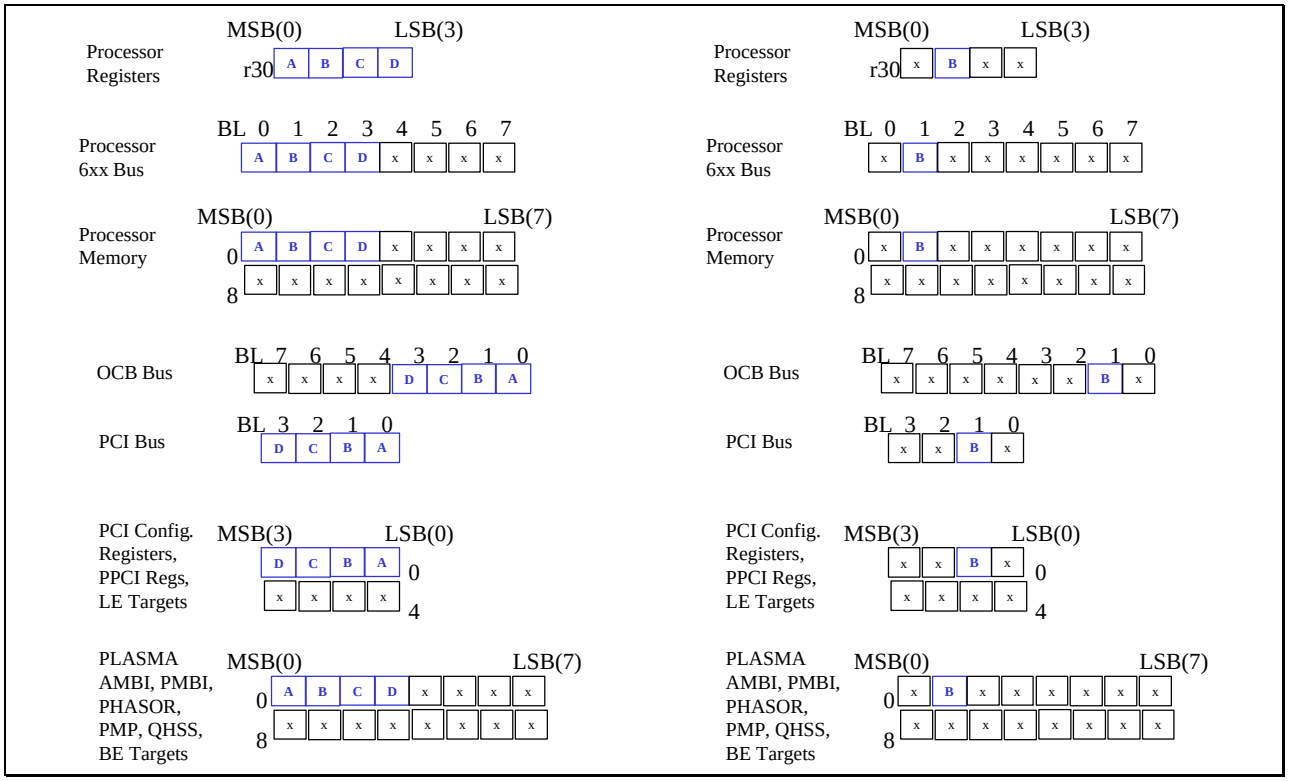


Figure 62: Data Transfers in Big Endian Mode.

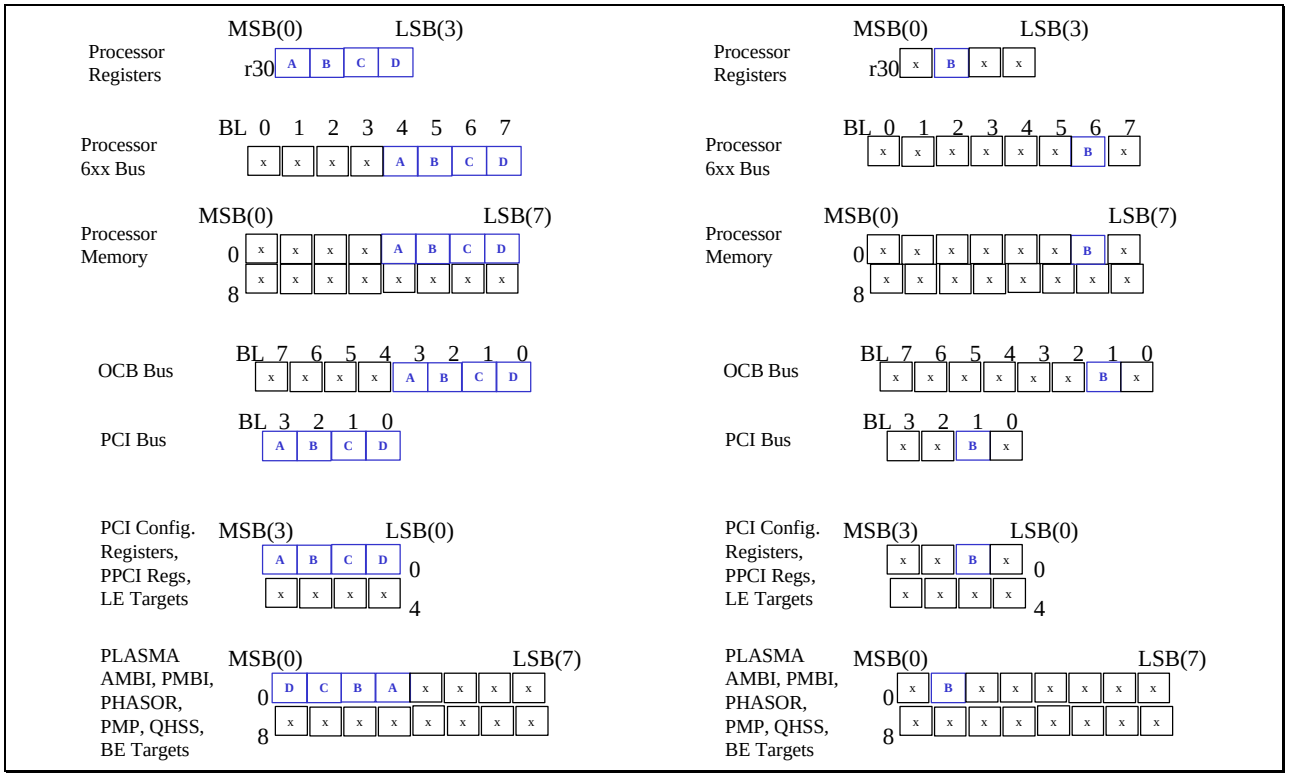


Figure 63: Data Transfers in Little Endian Mode

5 Documentation

BAE SYSTEMS has reference documentation available for the RAD750 family of products. Visit the RAD750 web site http://www.rad750.com/ or contact your BAE SYSTEMS sales representative to obtain copies.

- RAD750 Fact Sheet
- Power PCI Fact Sheet
- Power PCI Specification
- RAD750 Datasheet
- RAD750 Software Users Guide
- RAD750 Frequently Asked Questions (FAQ)

In addition to the documentation from BAE SYSTEMS, PowerPC documentation (i.e., Datasheets, User Manuals, Application Notes, etc.) is also available from

IBM at http://www.chips.ibm.com/products/powerpc/index.html, and

Motorola at http://www.mot.com/SPS/PowerPC/.

5.1 Suggested Reading

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

5.1.1 General PowerPC Information

The following documentation provides useful information about the PowerPC architecture and computer architecture in general:

- The following book is available from the Morgan-Kaufmann Publishers, 340 Pine Street, Sixth Floor, San Francisco, CA 94104; Tel. (800) 745-7323 (U.S.A.), (415) 392-2665 (International); internet address: mkp@mkp.com.
  1. — The PowerPC Architecture: A Specification for a New Family of RISC Processors, Second Edition, by International Business Machines, Inc.

  Updates to the architecture specification are accessible via the world-wide web at http://www.austin.ibm.com/tech/ppc-chg.html.

- *PowerPC Programming for Intel Programmers,* by Kip McClanahan; IDG Books Worldwide, Inc., 919 East Hillsdale Boulevard, Suite 400, Foster City, CA, 94404; Tel. (800) 434-3422 (U.S.A.), (415) 655-3022 (International).
- *PowerPC System Architecture,* by Tom Shanley; Mindshare, Inc., 2202 Buttercup Drive, Richardson, TX 75082; Tel. (214)231-2216 (U.S.A.), 021-706 6000 (United Kingdom), (800)420-2677 (International).

5.1.2 IBM PowerPC Documentation

The PowerPC documentation is available from the links indicated above; the document order numbers are included in parentheses for ease in ordering:

- *Embedded Market Solutions - Publications*: SC09-3032 - This is a CD-ROM containing documentation on all the PowerPC family of products from IBM. The CD-ROM is updated quarterly.
- Programming environments manuals - This book provides information about resources defined by the PowerPC architecture that are common to PowerPC processors.
  1. PowerPC Microprocessor Family: The Programming Environments G522-0290-00

- Implementation Variances Relative to Rev. 1 of The Programming Environments Manual is available via the world-wide web at http://www.chips.ibm.com/.

- Hardware specifications—Hardware specifications provide specific data regarding bus timing, signal behavior, and AC, DC, and thermal characteristics, as well as other design considerations for each PowerPC implementation. This include the following:
  1. *PowerPC 740Ô and PowerPC 750Ô Embedded RISC Microprocessor: Hardware Specifications* is available via the world-wide web at http://www.chips.ibm.com/.
  2. *PowerPC 750Ô SCM RISC Microprocessor: Hardware Specification* G522-0324-00
- Technical Summaries—Each PowerPC implementation has a technical summary that provides an overview of its features. This document is roughly the equivalent to the overview (Chapter 1) of an implementation's user's manual.
  3. *PowerPC 750 RISC Microprocessor Technical Summary* is available via the world-wide web at http://www.chips.ibm.com/.
  1. *PowerPC Microprocessor Family: 60x Bus Interface for 32-Bit Microprocessors*, G522-0291-00, provides a detailed functional description of the 60x bus interface, as implemented on the 601, 603, 604 and 740/750 family of PowerPC microprocessors. This document is intended to help system and chipset developers by providing a centralized reference source to identify the bus interface presented by the 60x family of PowerPC microprocessors.
- *PowerPC Microprocessor Family: The Programmer's Reference Guide*, MPRPPCPRG-01, is a concise reference that includes the register summary, memory control model, exception vectors, and the PowerPC instruction set.
- *PowerPC Microprocessor Family: The Programmer's Pocket Reference Guide*, SA14-2093-00 This foldout card provides an overview of the PowerPC registers, instructions, and exceptions for 32-bit implementations.
- Application notes—These short documents contain useful information about specific design issues useful to programmers and engineers working with PowerPC processors.

Additional literature on PowerPC implementations is being released as new processors become available. For a current list of PowerPC documentation, refer to the web sites listed at the beginning of this section.

5.2 Applicable Documents

The following documents are also relevant reading material for use with this board.

5.2.1 Specifications

IBM Manual 3530	LSSD-Based Design Rules for Testability - R92
PCI Local Bus Specification	*Peripheral Component Interface (PCI) Local Bus Specification*, Rev 2.2. December 18, 1998. PCI Special Interest Group, 2575 NE Kathryn St #17, Hillsboro, OR, 97124, Telephone: (800) 433-5177 (inside the U.S.), or (503) 693-6232 (outside the U.S.), FAX: (503) 693-8344, http://www.pcisig.com/
CompactPCI Specification	*CompactPCI Specification*, PICMG 2.0 R2.1, September 2, 1997, PCI Industrial Manufacturers Group (PICMG), 401 Edgewater Pl, Suite 500, WakeÞeld, MA 01880, Telephone: 781-246-9318, Fax: 781-224-1239, http://www.picmg.com/
On Chip Bus (OCB) Specification	*On-Chip Bus Specification*, July 8, 1999, Rev. 1.1

5.2.2 Standards

IEEE Standard 1149.1a	*Standard Test Access Port and Boundary Scan Architecture (JTAG),* October 21, 1993, Institute of Electrical and Electronics Engineers, Inc., Publication and Sales Department, 345 East 47th Street, New York, New York 10017-21633, Telephone: 1-800-678-4333, http://standards.ieee.org/

IEEE Standard 1101.1-1991	*Mechanical Core specifications for Microcomputers Using IEC 603-2 Connectors*, Institute of Electrical and Electronics Engineers, Inc., Publication and Sales Department, 345 East 47th Street, New York, New York 10017-21633, Telephone: 1-800-678-4333, http://standards.ieee.org/
JEDEC Standard No. 21-C	*SDRAM Architectural and Operational Features* (section 3.11.5), Release 7, http://www.jedec.org/menu.htm
ANSI/EIA/TIA-232-E, July 1991	*Interface Between Data Terminal Equipment and Data Circuit-Terminating Equipment Employing Serial Binary Data Interchange*, Electronic Industries Association, Engineering Department, 2001 Eye Street, N.W., Washington, D.C. 20006, http://www.eia.org/, or http://www.tiaonline.org/standards/search_n_order.html
ANSI/TIA/EIA-422-B-94, May, 1994	*Electrical Characteristics of Balanced Voltage Digital Interface Circuits*, Electronic Industries Association, Engineering Department, 2001 Eye Street, N.W., Washington, D.C. 20006, http://www.eia.org/, or http://www.tiaonline.org/standards/search_n_order.html

5.2.3 Design Descriptions and Design Guides

N/A	BIST User's Guide - Version 1.0 - 1/1/99
N/A	Code Design Methodology Guidelines
G522-0291-00	PowerPC Microprocessor Family: The Bus Interface for 32-Bit Microprocessors, 3/97, Rev. 0
N/A	VHDL Coding Standards; 10/05/98
N/A	WEC/DAP Design for Testability Guide - 10/22/94

For additional information, please contact us at: 1-800-RAD750S or mailto:rad750.manassas@baesystems.com

.

6 Comments

Document Title: ***RAD750 Board Hardware Reference Manual Appendix***	Revision Date:
Document number: 234A533	Originator:
Company:	Point of Contact:
Telephone:	e-mail Address:
Address: _____ _____ _____ _____	

Problem Description:

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

Please send this report to the BAE SYSTEMS address listed on the documentation page, or back cover.

If you require a written or electronic reply. please furnish your address in the form above.

BAE SYSTEMS
9300 Wellington Road
Manassas, Va
20110

The BAE SYSTEMS North America home page can be found at http://www.na.baesystems.com/

The BAE SYSTEMS Manassas home page can be found at http://www.baesystems-iews.com/space/