MTX Series Motherboard

Programmer's Reference Guide

MTXA/PG4

January 2001

Printed in the United States of America.

Safety Summary

The following general safety precautions must be observed during all phases of operation, service, and repair of this equipment. Failure to comply with these precautions or with specific warnings elsewhere in this manual could result in personal injury or damage to the equipment.

The safety precautions listed below represent warnings of certain dangers of which Motorola is aware. You, as the user of the product, should follow these warnings and all other safety precautions necessary for the safe operation of the equipment in your operating environment.

Ground the Instrument.

To minimize shock hazard, the equipment chassis and enclosure must be connected to an electrical ground. If the equipment is supplied with a three-conductor AC power cable, the power cable must be plugged into an approved three-contact electrical outlet, with the grounding wire (green/yellow) reliably connected to an electrical ground (safety ground) at the power outlet. The power jack and mating plug of the power cable meet International Electrotechnical Commission (IEC) safety standards and local electrical regulatory codes.

Do Not Operate in an Explosive Atmosphere.

Do not operate the equipment in any explosive atmosphere such as in the presence of flammable gases or fumes. Operation of any electrical equipment in such an environment could result in an explosion and cause injury or damage.

Keep Away From Live Circuits Inside the Equipment.

Operating personnel must not remove equipment covers. Only Factory Authorized Service Personnel or other qualified service personnel may remove equipment covers for internal subassembly or component replacement or any internal adjustment. Service personnel should not replace components with power cable connected. Under certain conditions, dangerous voltages may exist even with the power cable removed. To avoid injuries, such personnel should always disconnect power and discharge circuits before touching components.

Use Caution When Exposing or Handling a CRT.

Breakage of a Cathode-Ray Tube (CRT) causes a high-velocity scattering of glass fragments (implosion). To prevent CRT implosion, do not handle the CRT and avoid rough handling or jarring of the equipment. Handling of a CRT should be done only by qualified service personnel using approved safety mask and gloves.

Do Not Substitute Parts or Modify Equipment.

Do not install substitute parts or perform any unauthorized modification of the equipment. Contact your local Motorola representative for service and repair to ensure that all safety features are maintained.

Observe Warnings in Manual.

Warnings, such as the example below, precede potentially dangerous procedures throughout this manual. Instructions contained in the warnings must be followed. You should also employ all other safety precautions which you deem necessary for the operation of the equipment in your operating environment.

Warning

To prevent serious injury or death from dangerous voltages, use extreme caution when handling, testing, and adjusting this equipment and its components.

Flammability

All Motorola PWBs (printed wiring boards) are manufactured with a flammability rating of 94V-0 by UL-recognized manufacturers.

EMI Caution

Caution

This equipment generates, uses and can radiate electromagnetic energy. It may cause or be susceptible to electromagnetic interference (EMI) if not installed and used with adequate EMI protection.

Lithium Battery Caution

This product contains a lithium battery to power the clock and calendar circuitry.

Caution

Danger of explosion if battery is replaced incorrectly. Replace battery only with the same or equivalent type recommended by the equipment manufacturer. Dispose of used batteries according to the manufacturer's instructions.

Attention

Il y a danger d'explosion s'il y a remplacement incorrect de la batterie. Remplacer uniquement avec une batterie du même type ou d'un type équivalent recommandé par le constructeur. Mettre au rebut les batteries usagées conformément aux instructions du fabricant.

Vorsicht

Explosionsgefahr bei unsachgemäßem Austausch der Batterie. Ersatz nur durch denselben oder einen vom Hersteller empfohlenen Typ. Entsorgung gebrauchter Batterien nach Angaben des Herstellers.

CE Notice (European Community)

Motorola Computer Group products with the CE marking comply with the EMC Directive (89/336/EEC). Compliance with this directive implies conformity to the following European Norms:

> EN55022 “Limits and Methods of Measurement of Radio Interference Characteristics of Information Technology Equipment”; this product tested to Equipment Class B
>
> EN50082-1:1997 “Electromagnetic Compatibility—Generic Immunity Standard, Part 1. Residential, Commercial and Light Industry”

System products also fulfill EN60950 (product safety) which is essentially the requirement for the Low Voltage Directive (73/23/EEC).

Board products are tested in a representative system to show compliance with the above mentioned requirements. A proper installation in a CE-marked system will maintain the required EMC/safety performance.

In accordance with European Community directives, a “Declaration of Conformity” has been made and is on file within the European Union. The “Declaration of Conformity” is available on request. Please contact your sales representative.

Notice

Limited and Restricted Rights Legend

If the documentation contained herein is supplied, directly or indirectly, to the U.S. Government, the following notice shall apply unless otherwise agreed to in writing by Motorola, Inc.

Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (b)(3) of the Rights in Technical Data clause at DFARS 252.227-7013 (Nov. 1995) and of the Rights in Noncommercial Computer Software and Documentation clause at DFARS 252.227-7014 (Jun. 1995).

Motorola, Inc.
Computer Group
2900 South Diablo Way
Tempe, Arizona 85282

Contents

CHAPTER 2 Raven PCI Host Bridge & Multi-Processor Interrupt Controller

CHAPTER 3 Falcon ECC Memory Controller Chip Set

CHAPTER 4 Programming Details

APPENDIX A Related Documentation

List of Figures

List of Tables

About This Manual

This manual provides programming information for the MTX motherboard, equipped with a PowerPC Series microprocessor. Extensive programming information is provided for several Application-Specific Integrated Circuit (ASIC) devices used on the boards. Reference information is included in Appendix A for the Large Scale Integration (LSI) devices used on the boards and sources for additional information are listed.

The following table lists model numbers associated with this manual.

Model Number	Description
MTX603-001a	133 603, STANDARD I/O, 2 PMC
MTX603-002a	200 603, STD & OPT I/O, 2PMC
MTX603-003a	200 603, STD & OPT I/O, 3 PCI
MTX604-001a	300 MHz, 604, STANDARD I/O, 2 PMC
MTX604-002a	300 MHz, 604, STD & OPT I/O, 2 PMC
MTX604-003a	300 MHz, 604, STD & OPT I/O, 3 PCI
MTX604-010a	TWN 300, 604 STD & OPT I/O, 3PCI\

Summary of Changes

The following table shows any changes made to this manual since its last release.

Date	Changes
January 2001	Warning added to ensure user is aware of need for boot-up software to take steps to guarantee the DRAM memory component power-up refresh requirements are met. The Warnings can be found on pages 3-35 and 3-59.

Overview of Contents

Chapter 1, *Board Description and Memory Maps*, briefly describes the board level hardware features of the MTX series motherboard.

Chapter 2, *Raven PCI Host Bridge & Multi-Processor Interrupt Controller*, describes the architecture and usage of the Raven, a PowerPC to PCI Local Bus Bridge ASIC.

Chapter 3, *Falcon ECC Memory Controller Chip Set*, provides a functional description and programming model for the Falcon. Most of the information for using the device in a system, programming it in a system, and testing it is contained here.

Chapter 4, *Programming Details*, contains details of several programming functions that are not tied to any specific ASIC chip.

Appendix A, *Related Documentation*, contains all documentation related to this product.

Comments and Suggestions

Motorola welcomes and appreciates your comments on its documentation. We want to know what you think about our manuals and how we can make them better. Mail comments to:

Motorola Computer Group
Reader Comments DW164
2900 S. Diablo Way
Tempe, Arizona 85282

You can also submit comments to the following e-mail address:
reader-comments@mcg.mot.com

In all your correspondence, please list your name, position, and company. Be sure to include the title and part number of the manual and tell how you used it. Then tell us your feelings about its strengths and weaknesses and any recommendations for improvements.

Manual Terminology

Throughout this manual, a convention is used which precedes data and address parameters by a character identifying the numeric format as follows:

$	dollar	specifies a hexadecimal character
%	percent	specifies a binary number
&	ampersand	specifies a decimal number

For example, “12” is the decimal number twelve, and “$12” is the decimal number eighteen.

Unless otherwise specified, all address references are in hexadecimal.

An asterisk (*) following the signal name for signals which are *level significant* denotes that the signal is *true* or valid when the signal is low.

An asterisk (*) following the signal name for signals which are *edge significant* denotes that the actions initiated by that signal occur on high to low transition.

Note In some places in this document, an underscore (_) following the signal name is used to indicate an active low signal.

In this manual, *assertion* and *negation* are used to specify forcing a signal to a particular state. In particular, assertion and assert refer to a signal that is active or true; negation and negate indicate a signal that is inactive or false. These terms are used independently of the voltage level (high or low) that they represent.

Data and address sizes for MPC60*x* chips are defined as follows:

- A *byte* is eight bits, numbered 0 through 7, with bit 0 being the least significant.
- A *half-word* is 16 bits, numbered 0 through 15, with bit 0 being the least significant.

- A *word* or *single word* is 32 bits, numbered 0 through 31, with bit 0 being the least significant.
- A *double word* is 64 bits, numbered 0 through 63, with bit 0 being the least significant.

Refer to *Endian Issues* on page 4-11, which covers which parts of the MTX series use *big-endian* byte ordering, and which use *little-endian* byte ordering.

The terms *control bit* and *status bit* are used extensively in this document. The term control bit is used to describe a bit in a register that can be set and cleared under software control. The term *true* is used to indicate that a bit is in the state that enables the function it controls. The term *false* is used to indicate that the bit is in the state that disables the function it controls. In all tables, the terms 0 and 1 are used to describe the actual value that should be written to the bit, or the value that it yields when read. The term *status bit* is used to describe a bit in a register that reflects a specific condition. The status bit can be read by software to determine operational or exception conditions.

Conventions Used in This Manual

The following typographical conventions are used in this document:

bold

is used for user input that you type just as it appears; it is also used for commands, options and arguments to commands, and names of programs, directories and files.

italic

is used for names of variables to which you assign values. Italic is also used for comments in screen displays and examples, and to introduce new terms.

`courier`

is used for system output (for example, screen displays, reports), examples, and system prompts.

<**Enter**>, <**Return**> or <**CR**>

<**CR**> represents the carriage return or Enter key.

CTRL

represents the Control key. Execute control characters by pressing the Ctrl key and the letter simultaneously, for example, **Ctrl-d**.

Board Description and Memory Maps

1

Introduction

This chapter briefly describes the board level hardware features of the MTX series motherboard. The chapter begins with a board level overview and features list. Memory maps are next, and are the major feature of this chapter.

Programmable registers in the MTX series that reside in ASICs are covered in the chapters on those ASICs. Chapter 2, *Raven PCI Host Bridge & Multi-Processor Interrupt Controller*, covers the Raven chip. Chapter 3, *Falcon ECC Memory Controller Chip Set*, covers the Falcon chip set, and Chapter 4, *Programming Details*, covers certain programming features, such as interrupts and exceptions. Appendix A, *Related Documentation*, lists all related documentation.

Feature Summary

There are many models based on the MTX series architecture. The following table summarizes the major features of the MTX series:

Table 1-1. MTX Series Features Summary

Feature	Description
Processors	Single/Dual Processor (604ev) Supports BGA processors only: MPC603, MPC604. Bus Clock Frequencies up to 66 MHz
L2 Cache	Build-option for 256KB Look-aside L2 Cache (Glance)
Flash	4MB (16-bit wide) plus sockets for 1MB (16-bit)
DRAM	256MB to 2GB on board (DIMMs) Two-way Interleaved, ECC protected
NVRAM	8KB (MK48T59)

Table 1-1. MTX Series Features Summary (Continued)

Feature	Description
RTC and Watchdog Timer	MK48T59 Device
Memory Controller	Falcon Chipset
PCI Host Bridge	Raven ASIC
Interrupt Controller	RavenMPIC (Raven ASIC)
PCI Interface	32/64-bit Data, up to 33 MHz operation
Form Factor	ATX
Peripheral Support	Two 16550-compatible async serial ports Two sync/async serial ports One Host-mode Parallel Port One Peripheral-mode Parallel Port 8-bit or 16-bit single-ended SCSI interface AUI or 10BaseT/100BaseTX Ethernet interface One PS/2 Keyboard and one PS/2 Mouse One PS/2 Floppy Port Two EIDE ports
PCI/PMC Expansion	Build option for two 32/64-bit PMC slots with back-panel I/O Build option for one 32/64-bit PCI Connector for horizontal PCI card via special riser Build option for three 32-bit vertical PCI card slots with back panel I/O
Miscellaneous	RESET/ABORT Switch Status LEDs

System Block Diagram

The MTX series provides the 256KB look-aside external cache option. The Falcon chip set controls the boot Flash and the ECC DRAM. The Raven ASIC functions as the 64-bit PCI host bridge and the MPIC interrupt controller. PCI devices include: SCSI, Ethernet, and one PMC

slot. Standard I/O functions are provided by the Super I/O device which resides on the ISA bus. The NVRAM/RTC and the optional synchronous serial ports also reside on the ISA bus. The general system block diagram for MTX series is shown below:

Functional Description

Overview

The MTX is a motherboard. It consists of the MPC603e/604e processor, the Raven PCI Bridge and Interrupt Controller, the ECC Memory Controller Falcon chipset, 5M of Boot FLASH, ECC-protected DRAM, and a large set of I/O peripherals. The MTX will support single 603ev, single 604e processors, and dual 604e processors. In the dual processor configuration, the internal operating frequencies of the 604e's are independently configurable.

I/O peripheral devices on the PCI bus are: SCSI interface, Ethernet interface, two 64-bit PMC and one 64-bit PCI slot, or three 32-bit PCI slots. Functions provided from the ISA bus are: two EIDE ports, a host mode P1284 parallel port, a peripheral mode P1284 parallel port, two async serial ports, two sync/async serial ports, a real time clock, and counters/timers.

Rear panel connectors on the MTX motherboard include: a 6-pin circular DIN connector for the keyboard interface, a 6-pin circular DIN connector for the mouse interface, a 25-pin host mode parallel port connector, an RJ45 connector for 10/100BaseT connections, and a 15 pin connector for the AUI interface.

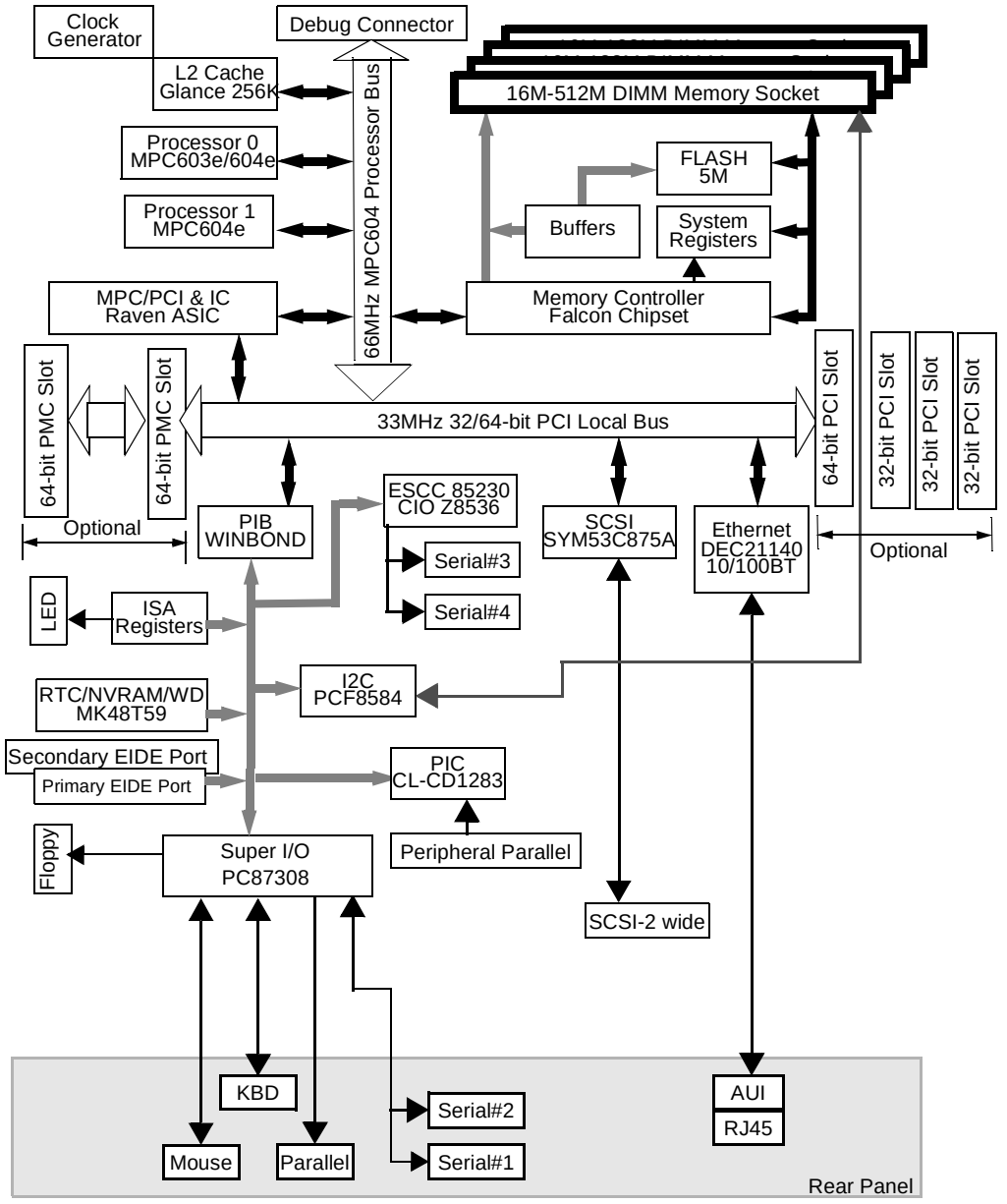


Figure 1-1. MTX Series System Block Diagram

The MTX series can be populated with two IEEE1386.1 PCI Mezzanine Card (PMC) slots plus a 64-bit PCI slot, or with three 32-bit PCI slots. The 32-bit PCI slots support ATX standard I/O spacing. All slots use rear panel I/O. The 64-bit PCI slot supports a horizontal PCI card via a custom riser card.

DRAM memory is added via DIMM sockets, and the serial presence detect (SPD) feature of the DIMM DRAMs is supported via the I^2C bus controller.

Programming Model

Memory Maps

The following sections describe the memory maps for the MTX series.

Processor Memory Maps

The Processor memory map is controlled by the Raven ASIC and the Falcon chipset. The Raven ASIC and the Falcon chipset have flexible programming Map Decoder registers to customize the system to fit many different applications.

Default Processor Memory Map

After a reset, the Raven ASIC and the Falcon chipset provide the default processor memory map as shown in the following table.

Table 1-2. Default Processor Memory Map

Processor Address		Size	Definition	Notes
Start	End			
0000 0000	7FFF FFFF	2G	Not mapped	
8000 0000	8001 FFFF	128K	PCI/ISA I/O Space	1
8002 0000	FEF7 FFFF	2G - 16M - 640K	Not mapped	
FEF8 0000	FEF8 FFFF	64K	Falcon Registers	

Table 1-2. Default Processor Memory Map (Continued)

Processor Address		Size	Definition	Notes
Start	End			
FEF9 0000	FEFE FFFF	384K	Not mapped	
FEFF 0000	FEFF FFFF	64K	Raven Registers	
FF00 0000	FFEF FFFF	15M	Not mapped	
FFF0 0000	FFFF FFFF	1M	ROM/FLASH Bank A or Bank B	2

Notes

1. This default map for PCI/ISA I/O space allows software to determine if the system is MPC105-based or Falcon/Raven-based by examining either the PHB Device ID or the CPU Type Register.
2. The first 1MB of ROM/FLASH Bank A appears at this range after a reset if the **rom_b_rv** control bit is cleared. If the **rom_b_rv** control bit is set then this address range maps to ROM/FLASH Bank B.

Processor CHRP Memory Map

The following table shows a recommended CHRP memory map from the point of view of the processor.

Table 1-3. CHRP Memory Map Example

Processor Address		Size	Definition	Notes
Start	End			
0000 0000	top_dram	dram_size	System Memory (onboard DRAM)	1, 2
4000 0000	FCFF FFFF	3G - 48M	PCI Memory Space: 4000 0000 to FCFF FFFF	3,4
FD00 0000	FDFF FFFF	16M	Zero-Based PCI/ISA Memory Space (mapped to 00000000 to 00FFFFFF)	3
FE00 0000	FE7F FFFF	8M	Zero-Based PCI/ISA I/O Space (mapped to 00000000 to 007FFFFF)	3,5

Table 1-3. CHRP Memory Map Example (Continued)

Processor Address		Size	Definition	Notes
Start	End			
FE80 0000	FEF7 FFFF	7.5M	Reserved	
FEF8 0000	FEF8 FFFF	64K	Falcon Registers	
FEF9 0000	FEFE FFFF	384K	Reserved	
FEFF 0000	FEFF FFFF	64K	Raven Registers	7
FF00 0000	FF7F FFFF	4M	ROM/FLASH Bank A	1,6
FF80 0000	FF8F FFFF	1M	ROM/FLASH Bank B	1,6
FF50 0000	FFEF FFFF	6M	Reserved	
FFF0 0000	FFFF FFFF	1M	ROM/FLASH Bank A or Bank B	6

Notes

1. Programmable via Falcon chipset.
2. To enable the *Processor-hole* area, program the Falcon chipset to ignore 0x000A0000 - 0x000BFFFF address range and program the Raven to map this address range to PCI memory space.
3. Programmable via Raven ASIC.
4. CHRP requires the starting address for the PCI memory space to be 256MB-aligned.
5. Programmable via Raven ASIC for either contiguous or spread-I/O mode.
6. The first 1MB of ROM/FLASH Bank A appears at this range after a reset if the **rom_b_rv** control bit is cleared. If the **rom_b_rv** control bit is set then this address range maps to ROM/FLASH Bank B.
7. The only method to generate a PCI Interrupt Acknowledge cycle (8259 IACK) is to perform a read access to the Raven's PIACK register at 0xFEFF0030.

The following table shows the programmed values for the associated Raven MPC registers for the processor CHRP memory map.

Table 1-4. Raven MPC Register Values for CHRP Memory Map

Address	Register Name	Register Value
FEFF 0040	MSADD0	4000 FCFF
FEFF 0044	MSOFF0 & MSATT0	0000 00C2
FEFF 0048	MSADD1	FD00 FDFF
FEFF 004C	MSOFF1 & MSATT1	0300 00C2
FEFF 0050	MSADD2	0000 0000
FEFF 0054	MSOFF2 & MSATT2	0000 0002
FEFF 0058	MSADD3	FE00 FE7F
FEFF 005C	MSOFF3 & MSATT3	0200 00C0

Processor PREP Memory Map

The Raven/Falcon chipset can be programmed for PREP-compatible memory map. The following table shows the PREP memory map of the MTX series from the point of view of the processor.

Table 1-5. PREP Memory Map Example

Processor Address		Size	Definition	Notes
Start	End			
0000 0000	top_dram	dram_size	System Memory (onboard DRAM)	1
8000 0000	BFFF FFFF	1G	Zero-Based PCI I/O Space: 0000 0000 - 3FFF FFFF	2
C000 0000	FCFF FFFF	1G - 48M	Zero-Based PCI/ISA Memory Space: 0000 0000 - 3CFFFFFF	2
FD00 0000	FEF7 FFFF	40.5M	Reserved	
FEF8 0000	FEF8 FFFF	64K	Falcon Registers	
FEF9 0000	FEFE FFFF	384K	Reserved	
FEFF 0000	FEFF FFFF	64K	Raven Registers	5

Table 1-5. PREP Memory Map Example (Continued)

Processor Address		Size	Definition	Notes
Start	End			
FF00 0000	FF7F FFFF	4M	ROM/FLASH Bank A	1, 3
FF80 0000	FF8F FFFF	1M	ROM/FLASH Bank B	1, 3
FF90 0000	FFEF FFFF	6M	Reserved	
FFF0 0000	FFFF FFFF	1M	ROM/FLASH Bank A or Bank B	4

Notes

1. Programmable via Falcon chipset.
2. Programmable via Raven ASIC.
3. The actual size of each ROM/FLASH bank may vary.
4. The first 1MB of ROM/FLASH Bank A appears at this range after a reset if the **rom_b_rv** control bit is cleared. If the **rom_b_rv** control bit is set then this address range maps to ROM/FLASH Bank B.
5. The only method to generate a PCI Interrupt Acknowledge cycle (8259 IACK) is to perform a read access to the Raven's PIACK register at 0xFEFF0030.

The following table shows the programmed values for the associated Raven MPC registers for the processor PREP memory map.

Table 1-6. Raven MPC Register Values for PREP Memory Map

Address	Register Name	Register Value
FEFF 0040	MSADD0	C000 FCFF
FEFF 0044	MSOFF0 & MSATT0	4000 00C2
FEFF 0048	MSADD1	0000 0000
FEFF 004C	MSOFF1 & MSATT1	0000 0002

Table 1-6. Raven MPC Register Values for PREP Memory Map (Continued)

Address	Register Name	Register Value
FEFF 0050	MSADD2	0000 0000
FEFF 0054	MSOFF2 & MSATT2	0000 0002
FEFF 0058	MSADD3	8000 BFFF
FEFF 005C	MSOFF3 & MSATT3	8000 00C0

PCI Configuration Access

PCI Configuration accesses are accomplished via the **CONADD** and **CONDA**T registers. These two registers are implemented by the Raven ASIC. In the CHRP memory map example, the CONADD and CONDAT registers are located at 0xFE000CF8 and 0xFE000CFC, respectively. With the PREP memory map, the CONADD register and the CONDAT register are located at 0x80000CF8 and 0x80000CFC, respectively.

PCI Memory Maps

The PCI memory map is controlled by the Raven ASIC. The Raven ASIC has flexible programming Map Decoder registers to customize the system to fit many different applications.

Default PCI Memory Map

After a reset, the Raven ASIC turns all the PCI slave map decoders off. Software must program the appropriate map decoders for a specific environment.

PCI CHRP Memory Map

The following table shows a PCI memory map of the MTX that is CHRP-compatible from the point of view of the PCI Local Bus.

Table 1-7. PCI CHRP Memory Map Example

Processor Address		Size	Definition	Notes
Start	End			
0000 0000	top_dram	dram_size	Onboard ECC DRAM	1
4000 0000	FBFF FFFF	3G - 64M	PCI Memory Space	1
FC00 0000	FC03 FFFF	256K	RavenMPIC	1
FC04 0000	FCFF FFFF	16M - 256K	PCI Memory Space	
FD00 0000	FDFF FFFF	16M	PCI Memory Space or System Memory Alias Space (mapped to 00000000 to 00FFFFFF)	1
FE00 0000	FFFF FFFF	32M	Reserved	

Notes

1. Programmable via the Raven's PCI Configuration Registers.

The following table shows the programmed values for the associated Raven PCI registers for the PCI CHRP memory map.

Table 1-8. Raven PCI Register Values for CHRP Memory Map

Configuration Address Offset	Configuration Register Name	Register Value (Aliasing OFF)	Register Value (Aliasing ON)
$14	RavenMPIC MBASE	FC00 0000	FC00 0000
$80	PSADD0	0000 3FFF	0100 3FFF
$84	PSOFF0 & PSATT0	0000 00FX	0000 00FX
$88	PSADD1	0000 0000	FD00 FDFF
$8C	PSOFF1 & PSATT1	0000 0000	0000 00FX
$90	PSADD2	0000 0000	0000 0000
$94	PSOFF2 & PSATT2	0000 0000	0000 0000
$98	PSADD3	0000 0000	0000 0000
$9C	PSOFF3 & PSATT3	0000 0000	0000 0000

PCI PREP Memory Map

The following table shows a PCI memory map of the MTX series that is PREP-compatible from the point of view of the PCI local bus.

Table 1-9. PCI PREP Memory Map

PCI Address		Size	Definition	Notes
Start	End			
0000 0000	00FF FFFF	16M	PCI/ISA Memory Space	
0200 0000	7FFF FFFF	2G -16M	PCI Memory Space	
8000 0000	FBFF FFFF	2G - 64M	Onboard ECC DRAM	1
FC00 0000	FC03 FFFF	256K	RavenMPIC	1
FC04 0000	FFFF FFFF	64M - 256K	PCI Memory Space	

Notes

1. Programmable via the Raven's PCI Configuration registers.

The following table shows the programmed values for the associated Raven PCI registers for the PREP-compatible memory map.

Table 1-10. Raven PCI Register Values for PREP Memory Map

Configuration Address Offset	Configuration Register Name	Register Value
$14	RavenMPIC MBASE	FC00 0000
$80	PSADD0	8000 FBFF
$84	PSOFF0 & PSATT0	8000 00FX
$88	PSADD1	0000 0000
$8C	PSOFF1 & PSATT1	0000 0000
$90	PSADD2	0000 0000
$94	PSOFF2 & PSATT2	0000 0000
$98	PSADD3	0000 0000
$9C	PSOFF3 & PSATT3	0000 0000

MPC System Bus

Memory maps for the MTX are described in the following sections.

Processors

The processors supported are: 603e, 603ev, 604, and 604e. Parity checking is not supported for the system address and data buses (that is, the Raven and Falcon ASICs do not generate parity and do not check for parity on the system address and data busses).

Processor Type Identification

The type of the processor can be determined by reading the Processor Version Register (PVR). The following table shows the PVR values for the supported processors:

Table 1-11. PVR Values

Processor	PVR Value
603e (Stretch)	0006 XXXXh
603ev (Valiant)	0007 XXXXh
604	0004 XXXXh
604e (Sirocco)	0009 XXXXh
604e (Mach5)	000A XXXXh

Processor PLL Configuration

The processor internal clock frequency (Core Frequency) is multiple of the system bus frequency. Each processor has four configuration pins, PLL_CFG, for hardware strapping of the processor core frequency.

Look-Aside Cache

The look-aside external cache, when present, is implemented with the Glance devices. Two Glance devices operate together to provide 256KB of look-aside cache. The Glance devices are controlled via the System External Cache Control Register (SXCCR).

Falcon FLASH Memory

The Falcon chipset supports up to two banks of FLASH memory. For the MTX each bank is 8-bits wide. Bank A FLASH size can be determined from the Memory Configuration Register. The maximum ROM/FLASH size is 64M per bank.

The FLASH type information for each bank is available via the Memory Configurator Register (MEMCR). The FLASH type information is helpful in determining the correct programming algorithm for the actual FLASH devices.

The reset vectors may be sourced by either Bank A or Bank B depending on the state of **rom_b_rv** control bit. When **rom_b_rv** bit is cleared, address range FFF00000-FFFFFFFF maps Bank A. When **rom_b_rv** bit is set, it maps to Bank B.

System Memory

The system memory is ECC-protected and is controlled by the Falcon chipset. Up to two blocks of system memory are supported. Each block of system memory can be up to 1GB in size. The design supports DRAM speed as slow as 70ns. Both fast-page mode and hyper-page (EDO) mode are supported. The best memory performance is achieved with 50ns EDO devices.

Software needs to obtain the DRAM configuration information for each block of memory by reading the DRAM serial presence detect (SPD) information via the two wire serial bus. The SPD provides the size, speed and type information for each DRAM DIMM device. The speed and refresh mode information are required to initialize the **ram_spd0**, **ram_spd1**, and **ram_fref** control bits in the Falcon's Revision ID/General Control Register.

The maximum size of each DRAM block is 1GB.

Table 1-12. Typical DIMM SPD Information

Byte#	Value (hex)	Entry Value	Description
0	0hxx	x	Number of SPD bytes
1	0h08	256	Total # bytes in SPD EEPROM
2	0h01 0h02	Fast Page EDO	Memory type
3	oh0C	12	# of row addresses
4	0h0B	11	# of column addresses
5	0h01	1	# of banks/DIMM
6	0h40 0h48	x64 x72	Module data width
7	0h00	0	Module data width (cont.)
8	0h01	LVTTL	Module interface levels
9	0h3C 0h46	60ns 70ns	RAS access time
10	0h0F 0h14	15ns 20ns	CAS access time
11	0h00 0h02	None ECC	Error detect/correction configuration
12	0h00	15.6 us Normal	Refresh rate/type
13	0h08	x8	Primary DRAM organization
14	0h00	Undefined	Secondary DRAM organization

Falcon Chipset

The Falcon chipset consists of two identical Falcon ASIC devices; The upper Falcon and the lower Falcon. The upper Falcon connects to the upper half of the system data bus, DH00 through DH31, while the lower Falcon connects to lower half of the system data bus, DL00 through DL31.

Falcon Registers

The base address of the Falcon chipset's Registers is hard coded to the address $FEF80000. Accesses to these registers are mapped differently depending on whether they are read or write operations. For reads, the data read on the upper half of the data bus comes from the upper Falcon, while the data read on the lower half of the data bus comes from the lower Falcon. For writes, internal register or test SRAM data written on the upperhalf of the data bus goes to the upper Falcon and is automatically copied by hardware to the lower Falcon. Internal register or test SRAM data written on the lower half of the data bus does not go to either Falcon in the pair, however the access is terminated normally with TA#.

Falcon-Controlled System Registers

The Falcon chipset latches the states of the DRAM data lines onto the PR_STAT1 and PR_STAT2 registers. The MTX series uses these status registers to provide the system configuration information. In addition, the Falcon chipset performs the decode and control for an external register port. This function is used by the MTX series to provide the system control registers.

Table 1-13. System Register Summary

BIT # ---->	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
FEF80400	System Configuration Register (Upper Falcon's PR_STAT1)																															
FEF80404	Memory Configuration Register (Lower Falcon's PR_STAT1)																															
FEF88000	System External Cache Control Register																															
FEF88100	Processor 0 External Cache Control Register																															
FEF88200	Processor 1 External Cache Control Register																															
FEF88300	CPU Control Register																															

The following sub-sections describe these system registers in detail.

System Configuration Register (SYSCR)

The states of the RD[0:31] DRAM data pins, which have weak internal pull-ups, are latched by the upper Falcon chip at a rising edge of the power-up reset and stored in this System Configuration Register to provide some information about the system. Configuration is accomplished with external pull-down resistors. This 32-bit read-only register is defined as follows:

Register	System Configuration Register - $FEF80400																													
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	SYSID								SYSCLK				SYSXC				P0STAT				P1STAT											
Operation	READ ONLY																															
Reset	$FB								X				X				X				X				$F				$F			

SYSID System Identification. This field specifies the type of the overall system configuration so that the software may appropriately handle any software visible differences. For the MTX series, this field returns a value of $FB.

SYSCLK System Clock Speed. This field relays the system clock speed and the PCI clock speed information as follows:

SYSCLK Value	System Clock Speed	PCI Clock Speed
0b0000 to 0b1100	Reserved	Reserved
0b1101	50 MHz	25 MHz
0b1110	60 MHz	30 MHz
0b1111	66.66 MHz	33.33 MHz

SYSXC System External Cache Size. This field reflects size of the look-aside cache on the system bus.

SYSXC Value	External Look-aside Cache Size
0b0000 to 0b1011	Reserved
0b1100	1MB
0b1101	512KB
0b1110	256KB
0b1111	None

P0/1STAT Processor 0/1 Status. This field is encoded as follows:

P0/1STAT Value	Processor 0/1 Present	External In-line Cache Size
0b0000 to 0b0011	Reserved	Reserved
0b0100	YES	1MB
0b0101	YES	512KB
0b0110	YES	256KB
0b0111	YES	None
0b1000 to 0b1111	NO	N/A

Memory Configuration Register (MEMCR)

The states of the RD[00:31] DRAM data pins, which have weak internal pull-ups, are latched by the lower Falcon chip at a rising edge of the power-up reset and stored in this Memory Configuration Register. In the MTX,

all RAM configuration information is contained in the SPD data; fields M_FREF and M_SPD [0:1} in the memory configuration register are not used. This 32-bit read-only Register is defined as follows:

Register	Memory Configuration Register - $FEF80404																															
Bit	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
Field				M_FREF			M_SPD0	M_SPD1		R_A_TYP0	R_A_TYP1	R_A_TYP2		R_B_TYP0	R_B_TYP1	R_B_TYP2											FLSHP0	FLSHP1	FLSHP2				
Operation																																
Reset	1	1	1	X	1	1	X	X	1	X	X	X	1	X	X	X	1	1	1	1	1	1	1	1	1	1	X	X	X	1	1	1

M_FREF Not used

M_SPD[0:1] Not used

R_A/B_TYP[0:1] ROM/Flash Type. This field is encoded as follows:

ROM_A/B_TYP[0:2]	ROM/FLASH Type
0b000 to 0b101	Reserved
0b110	Intel
0b111	Unknown type (that is, ROM/FLASH Sockets)

Note The device width is different from the width of the FLASH bank. If the bank width is 64-bit and the device width is 16-bit then the FLASH bank consists of four FLASH devices.

FLSHP[0:2] Bank A Flash memory size. This field is encoded as follows:

Flash Size	FLSHP0_	FLSHP1_	FLSHP2_
1MB	0	0	0
2MB	0	0	1
4MB	0	1	0
8MB	0	1	1
16MB	1	0	0
32MB	1	0	1
64MB	1	1	0
No Flash	1	1	1

System External Cache Control Register (SXCCR)

The System Cache Control Register is accessed via the RD[32:39] data lines of the upper Falcon device. This 8-bit register is defined as follows:

Register	System External Cache Control Register - $FEF88000							
Bit	0	1	2	3	4	5	6	7
Field	SXC_DIS_	SXC_RST_	SXC_MI_	SXC_FLSH_				
Operation	R/W							
Reset	1	1	1	1	X	X	X	X

SXC_DIS_ System External Cache Enable. When this bit is cleared, it disables this cache from responding to any bus cycles.

SXC_FLSH_ System External Cache Flush. When this bit is pulsed true for at least 8 clock periods, it causes the system external cache to write back dirty cache lines out to system memory and clears all the tag valid bits.

Note This operation causes the Glance pair to request and hold the MPC bus until it has completed the flush operation (approximately 4100 clock cycles). This may be an issue if other devices cannot wait that long to acquire MPC bus mastership.

SXC_RST_ System External Cache Reset. When this bit is cleared, it invalidates all tags and holds the cache in a reset condition.

Note Current Glance devices do not hold the cache in a reset condition; however, the tag invalidate function still operates.

SXC_MI_ System External Cache Miss Inhibit. When this bit is cleared, it prevents line fills on cache misses.

Warning Software should never clear more than one of these bits at the same time. If more than one is cleared at the same time, the Glance pair behaves indeterminately.

Processor 0 External Cache Control Register (P0XCCR)

The Processor 0 External Cache Control Register is accessed via the RD[32:39] data lines of the upper Falcon device. This register is not implemented for systems without In-line Cache. This 8-bit register is defined as follows:

Register	Processor 0 External Cache Control Register - FEF88100h							
Bit	0	1	2	3	4	5	6	7
Field	P0XC_CFG	P0XC_DIS_						
Operation	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	X	X	X	X

P0XC_CFG Processor 0 External Cache Configuration Access Mode. When this bit is set, it maps the Processor 0's external cache in Configuration Access Mode. Refer to the IBM15-C700A SLC User's Manual for details.

P0XC_DIS_ Processor 0 External Cache Disable. When this bit is cleared, it disables this cache from responding to any bus cycles.

Processor 1 External Cache Control Register (P1CCR)

The Processor 1 External Cache Control Register is accessed via the RD[32:39] data lines of the upper Falcon device. This register is not implemented for systems without In-line Cache. This 8-bit register is defined as follows:

Register	Processor 1External Cache Control Register - FEF88200h							
Bit	0	1	2	3	4	5	6	7
Field	P1XC_CFG	P1XC_DIS_						
Operation	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	X	X	X	X

P1XC_CFG Processor 1External Cache Configuration Access Mode. When this bit is set, it maps the Processor 1's external cache in Configuration Access Mode. Refer to the IBM15-C700A SLC User's Manual for details.

P1XC_DIS_ Processor 1External Cache Disable. When this bit is cleared, it disables this cache from responding to any bus cycles.

CPU Control Register

The CPU Control Register is accessed via the RD[32:39] data lines of the upper Falcon device. This 8-bit register is defined as follows:

Register	CPU Control Register - $FEF88300							
Bit	0	1	2	3	4	5	6	7
Field	GETPCI	LEMODE	P1_TBEN	P0_TBEN				
Operation	R	R	R/W	R/W	R	R	R	R
Reset	0	0	1	1	X	X	X	X

GETPCI Get PCI Bus. This bit is logically ORed with Raven's PCI bus request signal. It can be used to obtain the PCI bus for the Processor.

LEMODE Little-endian Mode. This bit must be set in conjunction with the LEND bit in the Raven for little-endian mode.

P0/1_TBEN Processor 0/1 Time Base Enable. When this bit is cleared, the TBEN pin of Processor 0/1 will be driven low.

ISA Local Resource Bus

W83C553 PIB Registers

The PIB contains ISA Bridge I/O registers for various functions. These registers are accessible from the PCI bus. Refer to the W83C553 Data Book for details.

Primary and Secondary EIDE Ports

The PIB also contains the EIDE controller. Refer to the 83C553 Data Book for details.

PC87308VUL Super I/O Strapping

The PC87308VUL Super I/O provides the following functions to the MTX series: a keyboard interface, a PS/2 mouse interface, a PS/2 floppy port, two async serial ports and a parallel port. Refer to the PC87308VUL Data Sheet for additional details and programming information.

The following table shows the hardware strapping for the Super I/O device:

Table 1-14. Strap Pins Configuration for the PC87308VUL

Pins	Reset Configuration
CFG0	0 - FDC, KBC and RTC wake up inactive.
CFG1	1 - Xbus Data Buffer (XDB) enabled.
CFG3, CFG2	00 - Clock source is 24MHz fed via X1 pin.
BADDR1, BADDR2	11 - PnP Motherboard, Wake in Config State. Index $002Eh.
SELCS	1 - CS0# on CS0# pin.

NVRAM/RTC & Watchdog Timer Registers

The MK48T59/559 provides the MTX series with 8KB of non-volatile SRAM, a time-of-day clock, and a watchdog timer. Accesses to the MK48T59/559 are accomplished via three registers: The NVRAM/RTC Address Strobe 0 Register, the NVRAM/RTC Address Strobe 1 Register, and the NVRAM/RTC Data Port Register. The NVRAM/RTC Address Strobe 0 Register latches the lower 8 bits of the address and the NVRAM/RTC Address Strobe 1 Register latches the upper 5 bits of the address.

Table 1-15. MK48T59/559 Access Registers

PCI I/O Address	Function
0000 0074	NVRAM/RTC Address Strobe 0 (A7 - A0)
0000 0075	NVRAM/RTC Address Strobe 1 (A15 - A8)
0000 0077	NVRAM/RTC Data Register

The NVRAM and RTC is accessed through the above three registers. When accessing a NVRAM/RTC location, follow the following procedure:

1. Write the low address (A7-A0) of the NVRAM to the NVRAM/RTC STB0 register,
2. Write the high address (A15-A8) of the NVRAM to the NVRAM/RTC STB1 register, and
3. Then read or write the NVRAM/RTC Data Port.

Refer to the MK48T59 Data Sheet, listed in Appendix A, *Related Documentation*, for additional details and programming information.

Module Configuration and Status Registers

Three registers provide the configuration and status information about the board. These registers are listed in the following table:

Table 1-16. Module Configuration and Status Registers

PCI I/O Address	Function
0000 0800	CPU Configuration Register
0000 0802	Base Module Feature Register
0000 0803	Base Module Status Register
0000 08C0 - 0000 08C1	Seven-Segment Display Register

The following sub-sections describe these registers in detail.

CPU Configuration Register

The CPU Configuration Register is an 8-bit register located at ISA I/O address x0800. This register is defined for the MTX to provide compatibility with existing firmware and AIX revisions. The Base Module Status Register should be used to identify the base module type and the System Configuration Register should be used to obtain information about the overall system for future releases.

Register	Old CPU Configuration Register - FE000800h							
Bit	SD7	SD6	SD5	SD4	SD3	SD2	SD1	SD0
Field	CPUTYPE							
Operation	R				R			
Reset	Eh				Fh			

CPUTYPE CPU Type. This field will always read as Eh for the MTX. The System Configuration Register should be used for additional information.

Base Module Feature Register

The Base Module Feature Register is an 8-bit register providing the configuration information about the MTX motherboard. This read-only register is located at ISA I/O address x0802.

Register	Base Module Feature Register - Offset $0802							
Bit	SD7	SD6	SD5	SD4	SD3	SD2	SD1	SD0
Field	EIDE_	SCCP_	PCI2P_	PCI1P_	PCI3P_	GFXP_	LANP_	SCSIP_
Operation	R	R	R	R	R	R	R	R
Reset	X	X	X	X	X	X	X	X

EIDE_ EIDE ports present. If set, there are no on-board EIDE ports. If cleared, there are on-board EIDE ports.

SCCP_ Z85230 ESCC Present. If set, there is no on-board sync serial support. If cleared, there is on-board support for sync serial interface.

PCI1P_ PMC or 32-bit PCI Slot 1 Present. If set, there is no PMC/PCI device installed in the PMC/PCI Slot 1. If cleared, the PMC/PCI Slot 1 contains a PMC or a PCI device.

PCI2P_ PMC or 32-bit Slot 2Present. If set, there is no PMC/PCI device installed in the PMC/PCI Slot 2. If cleared, the PMC/PCI Slot 2contains a PMC or PCI device.

PCI3P_ 64-bit or 32-bit PCI Slot 3 Present. If set, there is no PCI device install in PCI Slot 3. If cleared, the PCI Slot 3 contains a PCI device.

GFXP_ Graphics Present. If set, there is no on-board Graphics interface (default).

LANP_ Ethernet Present. If set, there is no Ethernet transceiver interface. If cleared, there is on-board Ethernet support.

SCSIP_ SCSI Present. If set, there is no on-board SCSI interface. If cleared, on-board SCSI is supported.

Base Module Status Register (BMSR)

The Base Module Status Register is an 8-bit read-only register located at ISA I/O address x0803.

Register	Base Module Status Register - Offset $0803							
Bit	SD7	SD6	SD5	SD4	SD3	SD2	SD1	SD0
Field	BASE_TYPE							
Operation								
Reset	N/A							

BASE_TYPE Base Module Type. This eight bit field is used to provide the category of the base module and is defined as follows:

BASE_TYPE Value	**Base Module Type**
$0 to $F7	Reserved
$F7	MTX without Peripheral Parallel Port
$F8	MTX with Peripheral Parallel Port
$F9 to FF	Reserved

Extended Status Register

The Extended Status Register is an 8-bit read-only register located at ISA I/O address x0951.

Register	Extended Status Register - Offset $0951							
Bit	SD7	SD6	SD5	SD4	SD3	SD2	SD1	SD0
Field							SCSITS	SCSIP
Operation	R	R	R	R	R	R	R	R
Reset	X	X	X	X	X	X	X	X

SCSITS On board SCSI terminator status. If set, the on board terminator is enabled. If cleared, the on board terminator is disabled.

SCSITP SCSI terminator power status. If set, SCSI terminator power is available. If cleared, there is no SCSI terminator power.

SCSI Terminator Select

The SCSI terminator select register is a one-bit (SD15) write only register located at ISA I/O address x0950. If a one is written to this register, the SCSI terminator is disabled. If a zero is written to this register, the terminator is enabled (assuming the on-board jumper described below is not installed). This register is cleared by a hard reset (terminators are enabled). This register may be overriden by installation of an on-board jumper at J36. The jumper disables the on-board SCSI terminator, regardless of the state of the terminator select register.

Seven-Segment Display Register

This 16-bit register allows data to be sent to the 4-digit hexadecimal diagnostic display. The register also allows the data to be read back.

Register	7-Segment Display Register - Offset $08C0															
Bit	SD15	SD14	SD13	SD12	SD11	SD10	SD9	SD8	SD7	SD6	SD5	SD4	SD3	SD2	SD1	SD0
Field	DIG3[3:0]				DIG2[3:0]				DIG1[3:0]				DIG0[3:0]			
Operation	R/W															
Reset	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

DIG3[3:0] Hexadecimal value of the most significant digit.

DIG2[3:0] Hexadecimal value of the third significant digit.

DIG1[3:0] Hexadecimal value of the second significant digit.

DIG0[3:0] Hexadecimal value of the least significant digit.

Z85230 ESCC and Z8536 CIO Registers and Port Pins

The Z85230 ESCC is used to provide the two sync/async serial ports on some MTX series models. The PCLK which can be used to derived the baud rates, is 5 MHz. Refer to the SCC User's Manual for programming information on the Z85230 ESCC device.

The Z8536 CIO is used to provide the modem control lines not provided by the Z85230 ESCC. Refer to the Z8536 Data Sheet, listed in Appendix A, *Related Documentation*, for programming information.

Z8536/Z85230 Registers

Accesses to the Z8536 CIO and the Z85230 ESCC are accomplished via Port Control and Port Data Registers. The PCLK to the Z8536 is 5 MHz. Also, a Pseudo IACK Register is also defined to retrieve interrupt vectors from these devices. The Z8536 CIO has higher priority than the Z85230 ESCC in the interrupt daisy chain. The following list the registers associated with accessing these two devices:

Table 1-17. Z8536/Z85230 Access Registers

PCI I/O Address	Function
0000 0840	Z85230: Port B (Serial Port 4) Control
0000 0841	Z85230: Port B (Serial Port 4) Data
0000 0842	Z85230: Port A (Serial Port 3) Control
0000 0843	Z85230: Port A (Serial Port 3) Data
0000 0844	Z8536 CIO: Port C's Data Register
0000 0845	Z8536 CIO: Port B's Data Register
0000 0846	Z8536 CIO: Port A's Data Register
0000 0847	Z8536 CIO: Control Register
0000 084F	Z85230/Z8536 Pseudo IACK

Z8536 CIO Port Pins

The assignment for the Port pins of the Z8536 CIO is as follows:

Table 1-18. Z8536 CIO Port Pins Assignment

Port Pin	Signal Name	Direction	Descriptions
PA0	TM3_	Input	Port 3 Test Mode.
PA1	DSR3_	Input	Port 3 Data Set Ready
PA2	RI3_	Input	Port 3 Ring Indicator
PA3	LLB3_	Output	Port 3 Local Loopback
PA4	RLB3_	Output	Port 3 Remote Loopback
PA5	DTR3_	Output	Port 3 Data Terminal Ready
PA6	BRDFAIL	Output	Board Fail: When set will cause FAIL LED to be lit.
PA7	Reserved	I/O	Reserved
PB0	TM4_	Input	Port 4 Test Mode
PB1	DSR4_	Input	Port 4 Data Set Ready
PB2	RI4_	Input	Port 4 Ring Indicator
PB3	LLB4_	Output	Port 4 Local Loopback
PB4	RLB4_	Output	Port 4 Remote Loopback

Table 1-18. Z8536 CIO Port Pins Assignment (Continued)

Port Pin	Signal Name	Direction	Descriptions
PB5	DTR4_	Output	Port 4 Data Terminal Ready
PB6	Reserved	I/O	Reserved
PB7	ABORT_	Input	Status of ABORT# signal
PC0	Reserved	I/O	Reserved
PC1	Reserved	I/O	Reserved
PC2	Reserved	I/O	Reserved
PC3	Reserved	I/O	

Note The direction and the polarity of the Z8536's port pins are software programmable.

Two Wire Serial (I^2C) Bus Controller

The I^2C bus controller controls communication with the DRAM DIMM on-module EEPROM devices. The EEPROM devices contain the Serial Presence Detect information (see section on System Memory for more information). Refer to the PCF8584 data sheet for additional programming information. The DIMM EEPROM addresses on the two wire serial bus are shown in Table 1-20.

Table 1-19. I^2C Controller Access Registers

PCI I/O Address	Function
0000 0980	Operation Registers
0000 0981	Control/Status Register

Table 1-20. Two Wire Serial (I^2C) Bus Addresses

I^2C Bus Address	Function
1010000	Bank A lower DIMM
1010001	Bank A upper DIMM
1010010	Bank B lower DIMM
1010011	Bank B upper DIMM

ISA DMA Channels

There are seven ISA DMA channels in the PIB. Channels 0 through 3 support only 8-bit DMA devices while Channels 5 through 7 support only 16-bit DMA devices. These DMA channels are assigned as follows:

Table 1-21. PIB DMA Channel Assignments

PIB Priority	PIB Label	Controller	DMA Assignment
Highest	Channel 0	DMA1	Serial Port 3 Receiver (Z85230 Port A Rx)
	Channel 1		Serial Port 3 Transmitter (Z85230 Port A Tx)
	Channel 2		Floppy Drive Controller
	Channel 3		Host Parallel Port
	Channel 4	DMA2	Not available - Cascaded from DMA1
	Channel 5		Serial Port 4 Receiver (Z85230 Port B Rx)
	Channel 6		Serial Port 4 Transmitter (Z85230 Port B Tx)
Lowest	Channel 7		Peripheral Parallel Port

Note Because the Z85230 is an 8-bit device and Channels 5 and 6 are 16-bit DMA Channels, only every other byte (the even bytes) from memory is valid.

Raven PCI Host Bridge & Multi-Processor Interrupt Controller

2

Introduction

Overview

This chapter describes the architecture and usage of the Raven, a PowerPC to PCI Local Bus Bridge ASIC. The Raven is intended to provide PowerPC microprocessor compliant devices access to devices residing on the PCI Local Bus in a very efficient manner. In the remainder of this chapter, the PowerPC bus will be referred to as the PPC bus and the PCI Local Bus as PCI.

No manufacturer currently has plans to support the PPC bus directly. Therefore, some alternative I/O bus will be necessary in any PowerPC product. This I/O bus must be robust and efficient enough to handle the high bandwidth, burst oriented traffic required for Ethernet, SCSI, graphics, and VMEbus interfaces.

PCI is a high performance 32-bit or 64-bit, burst mode, synchronous bus capable of transfer rates of 132 MB/sec in 32-bit mode or 264 MB/sec in 64-bit mode using a 33 MHz clock.

Requirements

The Raven must provide a high throughput interface between multiple PPC processors and 32/64-bit PCI local bus. It must be capable of supporting up to two PPC processors and contain a multiprocessing interrupt structure to efficiently distribute interrupts dynamically between these processors.

Features

- PPC Bus Interface
  - Direct interface to PowerPC processors.

  - 64-bit data bus, 32-bit address bus.
  - Optional bus arbitration logic supporting up to three bus masters.
  - Four independent software programmable slave map decoders.
  - Multi-level write post FIFO for writes to PCI.
  - Support for PPC bus clock speeds up to 66 MHz.
  - Selectable big or little-endian operation.

- PCI Interface
  - Fully PCI Rev. 2.0 compliant.
  - 32-bit or 64-bit address/data bus.
  - Support for accesses to all four PCI address spaces.
  - Single-level write posting buffers for writes to the PPC bus.
  - Read-ahead buffer for reads from the PPC bus.
  - Four independent software programmable slave map decoders.

- Interrupt Controller
  - MPIC compliant.
  - Support for 16 external interrupt sources and two processors.
  - Multiprocessor interrupt control allowing any interrupt source to be directed to either processor.
  - Multilevel cross processor interrupt control for multiprocessor synchronization.
  - Four 31 bit tick timers.

- Two 64-bit general purpose registers for cross-processor messaging.

Block Diagram

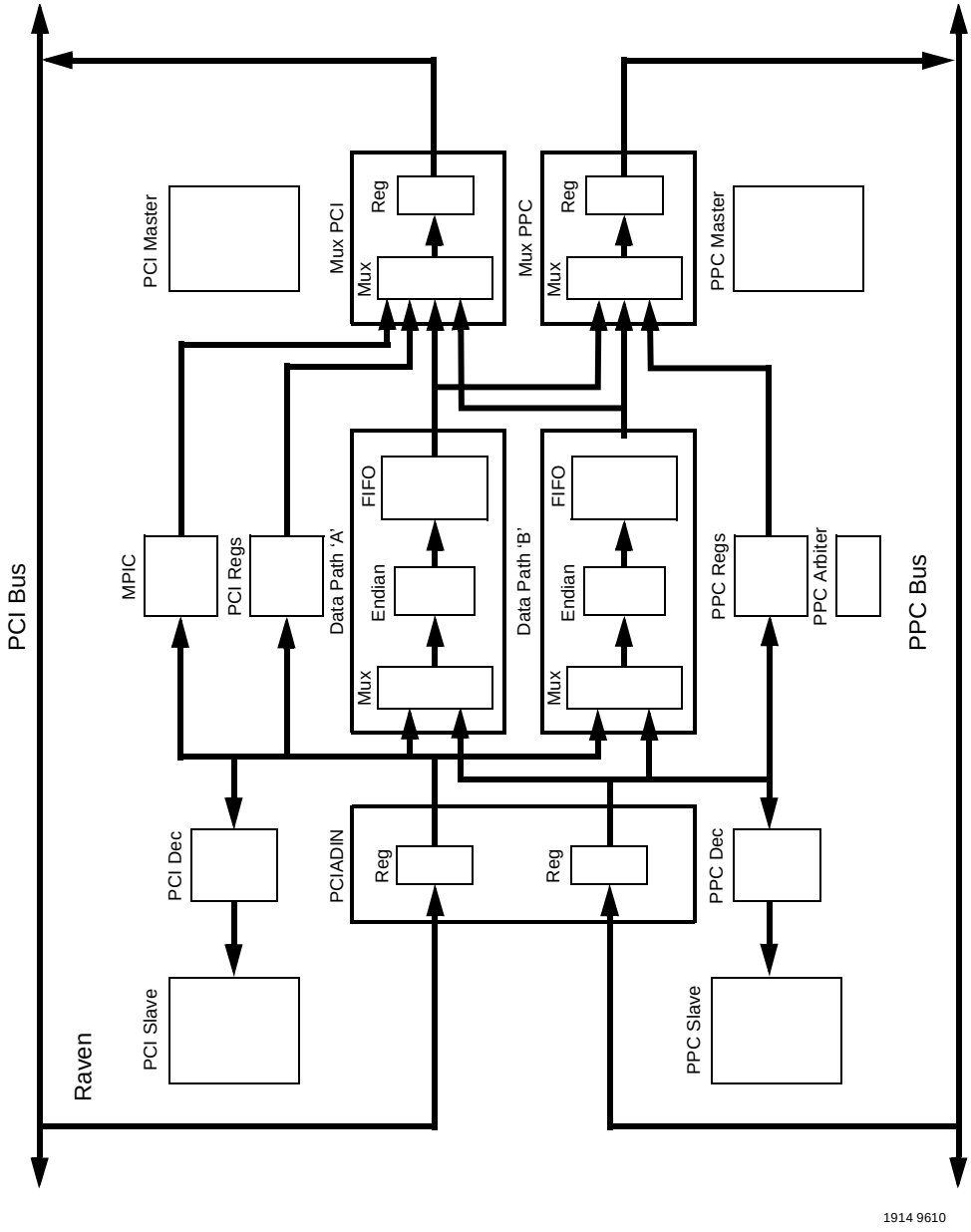


Figure 2-1. Raven Block Diagram

2

Functional Description

PPC Bus Interface

The PPC Bus Interface is designed to be coupled directly to up to two PPC microprocessors as well as a memory/cache subsystem. It uses a subset of the capabilities of the PPC bus protocol.

PPC Map Decoders

The Raven address decoders have been designed to be as flexible as possible to provide a wide range of addressing possibilities. There are five address map decoders in the Raven which determine the PPC bus addresses to which the Raven will respond: the PPC Register File Decoder, and four programmable decoders. Table 2-1 shows a typical CHRP compliant memory map. (Another similar map is shown in Table 1-3.)

Table 2-1. CHRP Compliant Memory Map

PPC Address	Function
$00000000-$7FFFFFFF	System Memory (2G)
$80000000-$FCFFFFFF	PCI Memory (2G - 48M)
$FD000000-$FDFFFFFFF	ISA Memory (16M)
$FE000000-$FE7FFFFF	Discontiguous PCI IO (8M)
$FE800000-$FEBFFFFF	Contiguous PCI IO (4M)
$FEC00000-$FEF7FFFF	reserved (3.5M)
$FEF80000-$FEF8FFFF	Falcon 0 Registers (64K)
$FEF90000-$FEF9FFFF	Falcon 1 Registers (64K)
$FEFA0000-$FEFAFFFF	Falcon 2 Registers (64K)
$FEFB0000-$FEFBFFFF	Falcon 3 Registers (64K)
$FEFC0000-$FEFEFFFF	reserved (192K)
$FEFF0000-$FEFFFFFF	Raven Registers (64K) (EXT00 => 0)
$FF000000-$FFFFFFFF	System ROM/Flash (16MB)

The PPC Register File decoder determines the address location of the Raven's PPC registers from the PPC bus. These registers may be accessed using only 1-, 2-, 3-, 4-, or 8-byte operations. The location of the PPC register file is fixed beginning at PPC address $FEFE0000 or $FEFF0000, depending on the state of the EXT01 bit at the time RST* is released. If the EXT01 pin is sampled in the low state, the PPC register file will start at address $FEFE0000. If the EXT01 pin is sampled in the high state, the PPC register file will start at address $FEFF0000. All references to the PPC register file within this specification will assume a base address of FEFF0000. All Raven registers are described in detail later in this chapter.

The Raven includes four programmable decoders which control accesses from the PPC bus to the PCI bus. These decoders provide a window into the PCI bus from the PPC bus. The most significant 16 bits of the PPC address are compared with the address range of each map decoder, and if the address falls within the specified range, the access is passed on to PCI. For each map, there is an associated set of attributes. These attributes are used to enable read accesses, enable write accesses, enable write posting, and define the PCI transfer characteristics. Each map decoder also includes a programmable 16-bit address offset. The offset is added to the 16 most significant bits of the PPC address, and the result is used as the PCI address. This offset allows PCI devices to reside at any PCI address, independent of the PPC address map.

Care should be taken to assure that all programmable decoders decode unique address ranges. Overlapping address ranges will lead to undefined operation.

PPC Write Posting

The PPC write FIFO stores up to eight data beats in any combination of single- and burst transactions. If write posting is enabled, Raven stores the data necessary to complete an PPC write transfer to the PCI bus and immediately acknowledges the transaction on the PPC bus. This frees the PPC bus from waiting for the potentially long PCI arbitration and transfer. The PPC bus may be used for more useful work while the Raven manages the completion of the write posted transaction on PCI.

If the write post FIFO is full, any other accesses to the Raven are delayed (AACK* will not be asserted) until there is room in the FIFO to store the complete transaction.

All write posted transfers will be completed before a non-write posted read or write is begun to assure that all transfers are completed in the order issued. All write posted transfers will also be completed before any access to the Raven's registers is begun.

PPC Master

Wherever possible, the PPC master will attempt to consolidate data movement into a pair of burst transfers called couplets. If there is not enough data movement to perform a couplet, the PPC master will attempt singular burst transfers. The PPC master will perform single beat transfers as required during all non-cache aligned writes and some non-cache aligned reads. A 64-bit by 16 entry FIFO is used to hold data between the PCI slave and the PPC master to ensure that optimum data throughput is maintained. While the PCI slave is filling the FIFO with one cache line worth of data, the PPC master can be moving another cache line worth onto the PPC bus. This will allow the PCI slave to receive long block transfers without stalling.

The PPC master has an optional read ahead mode controlled by the RAEN bit in the PCISATTx registers that allows the PPC master to prefetch data in bursts and store it in the FIFO. The contents of the FIFO will then be used to satisfy the data requirements for the remainder of the PCI read transaction. A second control bit within the PCISATTx registers called TDIS (Threshold Disable) determines how the PPC master will continue to fill the FIFO during prefetched reads. The following table shows the read ahead options.

TDIS	RAEN	PCI Command	Initial Read Size	Continuation	Subsequent Read Size
x	0	Read	1 cache line	PCI received data and FRAME* asserted	1 cache line
		Read Line			
0	1	Read	4 cache lines	FIFO <= 1cache line	2 cache lines
		Read Line			
	x	Read Multiple			
1	1	Read	4 cache lines	PCI received data and FRAME* asserted	2 cache lines
		Read Line			
	x	Read Multiple			

Upon completion of a prefetched read transaction, any residual read data left within the FIFO will be invalidated (discarded). The Raven does not have a mechanism for snooping the PPC bus for transactions associated with the prefetched read data within it's FIFO, therefore caution should be exercised when using the prefetch option within coherent memory space.

The user should select a PCI command type, a transaction size and a TDIS bit setting that minimizes wasted PPC bus bandwidth due to unnecessary prefetching. For example, there will be no unnecessary prefetching if the user always performs a burst read of 4 cache lines and the TDIS option is enabled. If the user wishes to perform very long burst transactions, then the TDIS option should be disabled since the benefits of a long uninterrupted transaction far exceed the penalty of a few unused prefetch cycles.

The PPC master will never perform prefetch reads beyond the address range mapped within the PCI slave map decoders. As an example, assume Raven has been programmed to respond to PCI address range $10000000 through $1001FFFF with an offset of $2000. The PPC master will perform its last read on the PPC bus at cache line address $3001FFFC or word address $3001FFF8.

The PPC bus transfer types generated by the PPC master depend on the PCI command code and the INV/GBL bits in the PCISATTx registers. The GBL bit determines whether or not the GBL* signal is asserted for all portions of a transaction and is fully independent of the PCI command code and INV bit. The following table shows the relationship between PCI command codes and the INV bit.

Table 2-2. PPC Transfer Types

PCI Command Code	INV	PPC Transfer Type	PPC Transfer Size	TT0-TT4
Memory Read, Memory Read Multiple, Memory Read Line	0	Read	Burst/Single Beat	01010
Memory Read, Memory Read Multiple, Memory Read Line	1	Read With Intent to Modify	Burst/Single Beat	01110
Memory Write, Memory Write and Invalidate	x	Write with Kill	Burst	00110
Memory Write, Memory Write and Invalidate	x	Write with Flush	Single Beat	00010

The PPC master incorporates an optional operating mode called Bus Hog. When Bus Hog is enabled, the PPC master will continually request the PPC bus for the entire duration of each PCI transfer. When Bus Hog is not enabled, the PPC master will structure its bus request actions around its desire to perform couplets. This means the bus request will be deasserted between couplets. Caution should be exercised when using this mode since the over-generosity of bus ownership to the PPC master can be detrimental to the host CPU's performance. The Bus Hog mode can be controlled by the BHOG bit within the GCSR. The default state for BHOG is disabled.

PPC Bus Timer

The PPC bus timer allows the current bus master to recover from a lock-up condition caused when no slave responds to the transfer request.

The time-out length of the bus timer is determined by the MBT field in the Global Control/Status Register.

The bus timer starts ticking at the beginning of an address transfer (TS* asserted), and if the address transfer is not terminated (AACK* asserted) before the time-out period has passed, the Raven will assert the MATO bit in the PPC Error Status Register, latch the PPC address in the PPC Error Address Register, and then immediately assert AACK*.

The MATO bit may be configured to generate an interrupt or a machine check through the MEREN register.

The timer is disabled if the transfer is intended for PCI. PCI bound transfers will be timed by the PCI master.

PCI Interface

The Raven PCI Interface is designed to connect directly to a PCI Local Bus compliant I/O bus.

The PCI interface may operate at any clock speed up to 33 MHz. The PCLK input must be externally synchronized with the MCLK input, and the frequency of the PCLK input must be exactly half the frequency of the MCLK input.

PCI Map Decoders

The Raven contains four programmable decoders which provide windows into the PPC bus from the PCI bus. The most significant 16 bits of the PCI address is compared with the address range of each map decoder, and if the address falls within the specified range, the access is passed on to the PPC bus. For each map, there is an independent set of attributes. These attributes are used to enable read accesses, enable write accesses, enable write posting, and define the PPC bus transfer characteristics. Each map decoder also includes a programmable 16-bit address offset. The offset is

added to the 16 most significant bits of the PCI address, and the result is used as the PPC address. This offset allows devices to reside at any PPC address, independent of the PCI address map.

All Raven address decoders are prioritized so that programming multiple decoders to respond to the same address will not be a problem. When the PCI address falls into the range of more than one decoder, only the highest priority one will respond. The decoders are prioritized as shown below.

Decoder	Priority
PCI Slave 0	highest
PCI Slave 1	
PCI Slave 2	
PCI Slave 3	lowest

PCI Configuration Space

The Raven does not have an IDSEL pin. An internal connection is made within the Raven that logically associates the assertion of IDSEL with the assertion of either AD30 or AD31. The exact association depends on the state of the EXT02 pin when the RST* pin is released. If EXT02 is sampled low, the Raven will associate AD30 with IDSEL. If EXT02 is sampled high, the Raven will associate AD31 with IDSEL.

PCI Write Posting

If write posting is enabled, the Raven stores the target address, attributes, and up to 128 bytes of data from one PCI write transaction and immediately acknowledges the transaction on the PCI bus. This allows the slower PCI to continue to transfer data at its maximum bandwidth, and the faster PPC bus to accept data in high performance cache-line burst transfers.

Only one PCI transaction may be write posted at any given time. If the Raven is busy processing a previous write posted transaction when a new PCI transaction begins, the next PCI transaction will be delayed (TRDY* will not be asserted) until the previous transaction has completed. If during a transaction the write post buffer gets full, subsequent PCI data transfers

will be delayed (TRDY* will not be asserted) until the Raven has removed some data from the FIFO. Under normal conditions, the Raven should be able to empty the FIFO faster than the PCI bus can fill it.

PCI Configuration cycles intended for internal Raven registers will also be delayed if Raven is busy so that control bits which may affect write posting do not change until all write posted transactions have completed.

PCI Master

The PCI master, in conjunction with the capabilities of the PPC slave, will attempt to move data in either single beat or burst transactions. All single beat transactions will be subdivided into one or two 32-bit transfers, depending on the alignment and size of the transaction. The PCI master will attempt to transfer all burst transactions in 64-bit mode. If at any time during the transaction the PCI target indicates it can not support 64-bit mode, the PCI master will continue to transfer the remaining data in 32-bit mode.

The PCI Command Codes generated by the PCI master depend on the PPC transfer type, TBST*, and the MEM field in the MSATT*x* registers.

Table 2-3. PCI Command Codes

PPC Transfer Type	TBST*	MEM	PCI Command
Write w/ Flush Write w/ Flush Atomic Write w/ Kill Graphics Write	x	1	0111 (Memory Write)
	x	0	0011 (I/O Write)
Read Read w/ ITM Read w/ ITM Atomic Graphics Read	0	1	1110 (Memory Read Line)
	1	1	0110 (Memory Read)
	x	0	0010 (I/O Read)

Generating PCI Memory and I/O Cycles

Each programmable slave may be configured to generate PCI I/O or memory accesses through the MEM and IOM fields in its Attribute register as shown below.

MEM	IOM	PCI Cycle Type
1	x	Memory
0	0	Contiguous I/O
0	1	Spread I/O

The IBM CHRP specification describes two approaches for handling PCI I/O addressing: contiguous or spread address modes. When the MEM bit is cleared, the IOM bit is used to select between these two modes whenever a PCI I/O cycle is to be performed.

When MEM is clear or IOM is clear, the Raven will take the PPC address, apply the offset specified in the MSOFFx register, and map the result directly to PCI.

When MEM is clear and IOM is set, the Raven will take the PPC address, apply the offset specified in the MSOFF*x* register, and map the result to PCI as shown in the following figure.

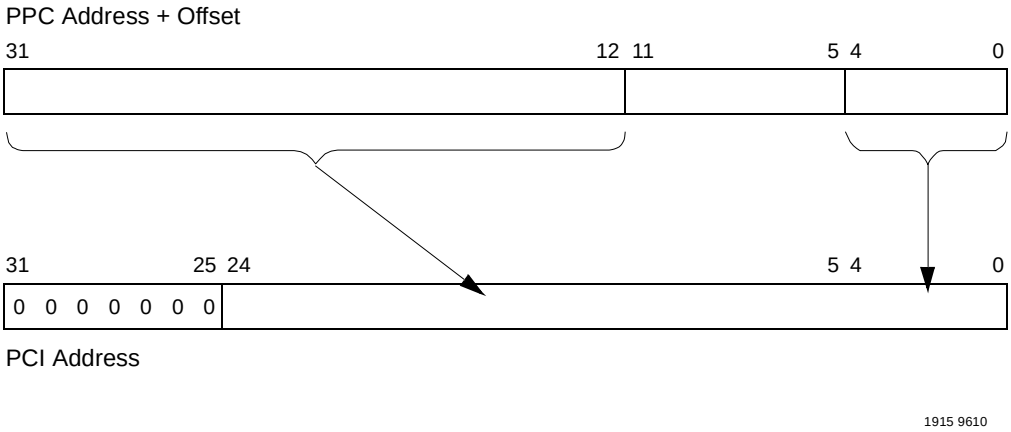


Figure 2-2. PCI Spread I/O Cycle Mapping

This CHRP compliant spread I/O mode allows each PCI device's I/O registers to reside on a different PPC memory page, so device drivers can be protected from each other using memory page protection.

All I/O accesses must be performed within natural word boundaries. Any I/O access that is not contained within a natural word boundary will result in unpredictable operation. For example, an I/O transfer of 4 bytes starting at address $80000010 is considered a valid transfer. An I/O transfer of 4 bytes starting at address $80000011 is considered an invalid transfer since it crosses the natural word boundary at address $80000013/$80000014.

Generating PCI Configuration Cycles

Mechanism one (as just described above) is utilized to generate configuration cycles. Two 32 bit PCI I/O ports at $CF8 and $CFC are used to access PCI configuration space. One of the four PPC Slave Address Registers is used to gain access to $CF8 and $CFC.

Note MSADD3 is initialized at reset to access PCI I/O space with an PPC address of $80000000.

The resource at $CF8 is a 32 bit configuration address port and is referred to as the CONFIG_ADDRESS register. The resource at $CFC is a 32 bit configuration data port and is referred to as the CONFIG_DATA register.

Accessing a PCI functions's configuration port is a two step process;

- Write the bus number, physical device number, function number and register number to the CONFIG_ADDRESS register.
- Perform an I/O read from or a write to the CONFIG_DATA register.

Generating PCI Special Cycles

To prime Raven to generate a special cycle, the host processor must write a 32-bit value to the CONFIG_ADDRESS register. The contents of the write are defined later in this chapter under the CONFIG_ADDRESS register definition. After the write to $CF8 has been accomplished, the next write to the CONFIG_DATA register causes the Raven to generate a special cycle on the PCI bus. The write data is driven onto AD[31:0] during the special cycle's data phase.

Generating PCI Interrupt Acknowledge Cycles

Performing a read from the PIACK register will initiate a single PCI Interrupt Acknowledge cycle. Any single byte or combination of bytes may be read from, and the actual byte enable pattern used during the read will be passed on to the PCI bus. Upon completion of the PCI interrupt acknowledge cycle, the Raven will present the resulting vector information obtained from the PCI bus as read data.

Endian Conversion

The Raven supports both big- and little-endian data formats. Since PCI is inherently little-endian, conversion is necessary if all PPC devices are configured for big-endian operation. The Raven may be programmed to perform the Endian conversion described below.

When PPC Devices are Big-Endian

When all PPC devices are operating in big-endian mode, all data to/from PCI must be swapped such that PCI looks big-endian from the PPC bus's perspective. This is shown in the following figure.

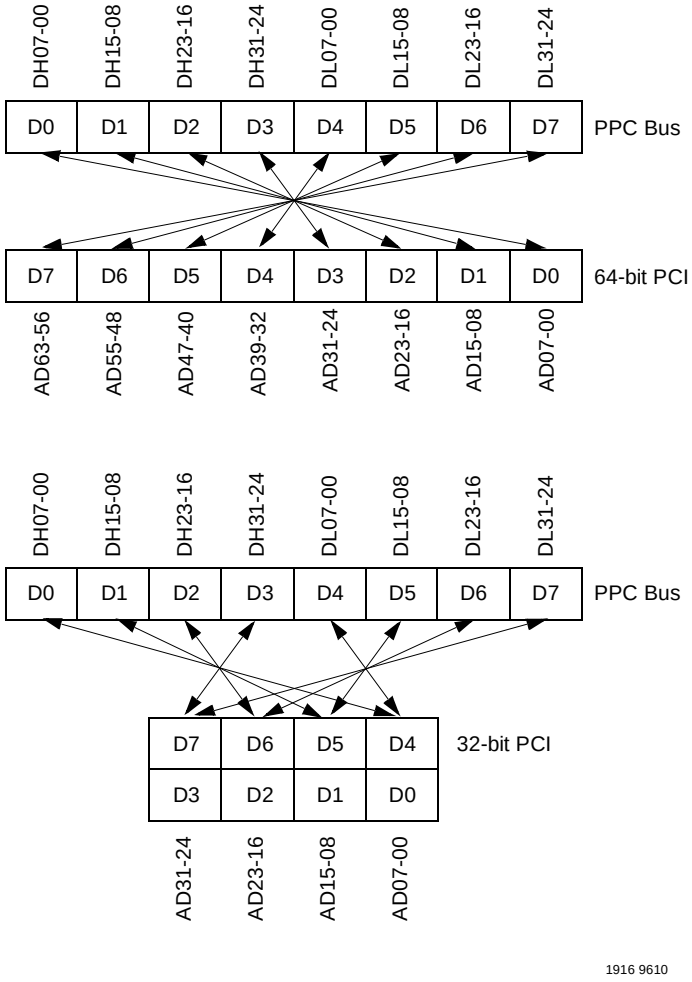


Figure 2-3. Big- to Little-Endian Data Swap

When PPC Devices are Little-Endian

When all PPC devices are operating in little-endian mode, the PPC address must be modified to remove the exclusive-ORing applied by PPC60*x* processors before being passed on to PCI. The three low order processor bus address bits are exclusive-ORed with a three-bit value that depends on the length of the operand, as shown in Table 2-4.

Table 2-4. Address Modification for Little-Endian Transfers

Data Length (bytes)	Address Modification
1	XOR with 111
2	XOR with 110
4	XOR with 100
8	no change

Note The only legal data lengths supported in little-endian mode are 1, 2, 4, or 8-byte aligned transfers.

Cycles Originating From PCI

For bus cycles initiated by PCI masters, the PCI address will be modified the same way the MCP60*x* processor does in little-endian mode. The modification will be the same as that described in the section above. Since this method has some difficulties dealing with unaligned transfers, the Raven will break up all unaligned PCI transfers into multiple aligned transfers on the PPC bus.

Error Handling

The Raven will be capable of detecting and reporting the following errors to one or more PPC masters:

- PPC address bus time-out
- PCI master signalled master abort
- PCI master received target abort

- PCI parity error
- PCI system error

Each of these error conditions will cause an error status bit to be set in the PPC Error Status Register. If a second error is detected while any of the error bits is set, the OVFL bit is asserted, but none of the error bits are changed. Each bit in the PPC Error Status Register may be cleared by writing a 1 to it; writing a 0 to it has no effect. New error bits may be set only when all previous error bits have been cleared.

When any bit in the PPC Error Status register is set, the Raven will attempt to latch as much information as possible about the error in the PPC Error Address and Attribute Registers. Information is saved as follows:

Error Status	Error Address and Attributes
MATO	From PPC bus
SMA	From PCI bus
RTA	From PCI bus
PERR	Invalid
SERR	Invalid

Each MERST error bit may be programmed to generate a machine check and/or a standard interrupt. The error response is programmed through the PPC Error Enable Register on a source by source basis. When a machine check is enabled, either the MID field in the PPC Error Attribute Register or the DFLT bit in the MEREN Register determine the master to which the machine check is directed. For errors in which the master who originated the transaction can be determined, the MID field is used, provided the MID is %00 (processor 0), %01 (processor 1), or %10 (processor 2). For errors not associated with a particular PPC master, or associated with masters other than processor 0, 1, or 2, the DFLT bit is used. One example of an error condition which cannot be associated with a particular PPC master would be a PCI system error.

PCI/PPC Contention Handling

The Raven has a stall detection mechanism that detects when there is a possible resource contention problem (that is, deadlock) as a result of overlapping PPC and PCI initiated transactions. The PPC Slave and the PCI Slave functions contain the logic needed to implement this feature.

The PCI Slave function contributes to the stall detection mechanism by issuing a stall signal to the PPC Slave function whenever it is currently processing a transaction that must have control of the PPC bus before the transaction can be completed. The events that activate this signal are:

- PCI read cycle
- PCI non-posted write cycle
- PCI posted write cycle with flush-before-read (FLBRD) enabled
- PCI posted write cycle and internal FIFO full

The PPC Slave function determines its future actions based on the stall signal and the current PPC bus activity. If the PPC Slave function determines there will be contention between a cycle completing on the PPC bus and an incoming PCI cycle, the PPC Slave will issue a retry for the current PPC transaction. This retry will free up the PPC bus and allow the PCI initiated transaction to complete. An idle PPC bus obviously gives immediate access to the pending PCI initiated transaction.

If the PPC bus is currently supporting a read cycle of any type, the cycle will be terminated with a retry.

Note If the read cycle is across a mod-4 address boundary (that is, from address 0x...02, 3 bytes), it is possible that a portion of the read could have been completed before the stall condition was detected. The previously read data will be discarded and the current transaction will be retried.

If the PPC bus is currently supporting a posted write transaction, the transaction will be allowed to complete since this type of transaction is guaranteed completion. If the PPC bus is currently supporting a non-

posted write transaction, the transaction will be terminated with a retry. Note that a mod-4 non-posted write transaction could be interrupted between write cycles, and thereby result in a partially completed write cycle. It is recommended that write cycles to write-sensitive non-posted locations be performed on mod-4 address boundaries.

The Raven has a programmable option to guarantee all PCI write posted transactions are completed before an PPC initiated read transaction may be allowed to complete. This option is controlled by the FLBRD bit in the GSCR register. If this bit is set, all PPC read transactions will be retried until all posted PCI write transactions have completed. It is recommended that this option be disabled, and the FLBRD bit be left in the default (disabled) state.

Transaction Ordering

The PCI Local Bus Specification 2.0, listed in Appendix A, *Related Documentation*, states that posted write buffers in both directions must be flushed before completing a read in either direction. Raven supports this by providing two optional FIFO flushing options. The PPCFBR (PPC Flush Before Read) bit within the GCSR register controls the flushing of PCI write posted data when performing PPC originated read transactions. The PCIFBR bit within the GCSR register controls the flushing of PPC write posted data when performing PCI originated read transactions. The PCIFBR and PPCFBR functions are completely independent of each other, however both functions must be enabled to guarantee full compliance with PCI Local Bus Specification 2.0, Appendix A, *Related Documentation*.

When the PPCFBR bit is set, Raven will handle read transactions originating from the PPC bus in the following manner:

- Write posted transactions originating from the processor bus are flushed by the nature of the FIFO architecture. The Raven will hold the processor with wait states until the PCI bound FIFO is empty.
- Write posted transactions originated from the PCI bus are flushed in the following manner. The PCI slave sets a signal called ‘pcis_fbrabt’ anytime it has committed to performing a posted write

transaction. This signal will remain asserted until the PPC bound FIFO count has reached zero.

The PPC slave address decode logic settles out several clocks after the assertion of TS*, at which time the PPC slave can determine the transaction type. If it is a read and PPCFBR is enabled, the PPC slave will look at the **pcis_fbrabt** signal. If this signal is active, the PPC slave will retry the processor.

When the PCIFBR bit is set, Raven will handle read transactions originating from the PCI bus in the following manner:

- Write posted transactions originating from the PCI bus are flushed by the nature of the FIFO architecture. The Raven will hold the PCI master with wait states until the PPC bound FIFO is empty.
- Write posted transactions originated from the PPC bus are flushed in the following manner. The PPC slave will set a signal called **ppcs_fbrabt** anytime it has committed to performing a posted write transaction. This signal will remain asserted until the PCI bound FIFO count has reached zero.

  The PCI slave decode logic settles out several clocks after the assertion of FRAME*, at which time the PCI slave can determine the transaction type. If it is a read and PCIFBR is enabled, the PCI slave will look at the **ppcs_fbrabt** signal. If this signal is active, the PCI slave will retry the PCI master.

Registers

This chapter provides a detailed description of all Raven registers. The Raven ASIC functions as the PPC to PCI Host Bridge. It also contains an interrupt controller, the Raven MPIC. The Raven ASIC has three sets of registers: The PPC Register set, the PCI Configuration Register set, and the Raven MPIC Register set. The PPC Registers are described first; the PCI Configuration Registers next, and the MPIC Register set last.

The following conventions are used in the Raven register charts:

- R Read Only field.

- R/W Read/Write field.
- S Writing a ONE to this field sets this field.
- C Writing a ONE to this field clears this field.

PPC Registers

The Raven PPC register map is shown in the following table.

Table 2-5. Raven PPC Register Map

Bit --->	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$FEFF0000	VENID																DEVID															
$FEFF0004									REVID																							
$FEFF0008	GCSR																FEAT															
$FEFF000C																																
$FEFF0010																									PADJ							
$FEFF0014																																
$FEFF0018																																
$FEFF001C																																
$FEFF0020	ERRTST																ERREN															
$FEFF0024																									ERRST							
$FEFF0028	ERRAD																															
$FEFF002C																	ERRAT															
$FEFF0030	PIACK																															
$FEFF0034																																
$FEFF0038																																
$FEFF003C																																
$FEFF0040	PPCSADD0																															
$FEFF0044	PPCSOFF0																								PPCSATT0							
$FEFF0048	PPCSADD1																															
$FEFF004C	PPCSOFF1																								PPCSATT1							

Table 2-5. Raven PPC Register Map (Continued)

Bit --->	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$FEFF0050	PPCSADD2																															
$FEFF0054	PPCSOFF2																								PPCSATT2							
$FEFF0058	PPCSADD3																															
$FEFF005C	PPCSOFF3																								PPCSATT3							
$FEFF0060	WDT1CNTL																															
$FEFF0064																									WDT1STAT							
$FFEF0068	WDT2CNTL																															
$FEFF006C																									WDT2STAT							
$FEFF0070	GPREG0(Upper)																															
$FEFF0074	GPREG0(Lower)																															
$FEFF0078	GPREG1(Upper)																															
$FEFF007C	GPREG1(Lower)																															

Vendor ID/Device ID Registers

Address	$FEFF0000																															
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Name	VENID																DEVID															
Operation	R																R															
Reset	$1057																$4801															

VENID Vendor ID. This register identifies the manufacturer of the device. This identifier is allocated by the PCI SIG to ensure uniqueness. $1057 has been assigned to Motorola. This register is duplicated in the PCI Configuration Registers.

DEVID Device ID. This register identifies this particular device. The Raven will always return $4801. This register is duplicated in the PCI Configuration Registers.

Revision ID Register

Address	$FEFF0004																															
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Name											REVID																					
Operation	R								R								R								R							
Reset	$00								$03								$00								$00							

REVID Revision ID. This register identifies the Raven revision level. This register is duplicated in the PCI Configuration Registers.

General Control-Status/Feature Registers

Address	$FEFF0008																															
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Name	GCSR																FEAT															
Operation	LEND			PCIFBR	BHOG	PPCFBR	PBT1	PBT0	P64		OPIC				MID1	MID0	EXT15	EXT14	EXT13	EXT12	EXT11	EXT10	EXT09	EXT08	EXT07	EXT06	EXT05	EXT04	EXT03	EXT02	EXT01	EXT00
Reset	R/W	R	R	R/W	R/W	R/W	R/W	R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Address	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1

LEND Endian Select. If set, the PPC bus is operating in little-endian mode. The PPC address will be modified as described in the section *When PPC Devices are Little-Endian* on page 2-16. When LEND is clear, the PPC bus is operating in big-endian mode, and all data to/from PCI is swapped as described in the section *When PPC Devices are Big-Endian* on page 2-15.

PCIFBR PCI Flush Before Read. If set, the Raven will guarantee that all PPC initiated read transactions will be completed before any PPC initiated read transactions will be allowed to complete. When PCIFBR is clear, there will be no correlation between these transaction types and their order of completion. Please refer to the section on *PCI/PPC Contention Handling* for more information.

BHOG Bus Hog. If set, the Raven PPC master will operate in the Bus Hog mode. Bus Hog mode means the PPC master will continually request the PPC bus for the entire duration of each PCI transfer. If Bus Hog is not enabled, the PPC master will request the bus in a normal manner. Please refer to the section on *PPC Master* for more information.

PPCFBR PPC Flush Before Read. If set, the Raven will guarantee that all PCI initiated posted write transactions will be completed before any PPC initiated read transactions will be allowed to complete. When PPCFBR is clear, there will be no correlation between these transaction types and their order of completion. Please refer to the section on *PCI/PPC Contention Handling* for more information.

PBT*x* PPC Bus Time-out. This field specifies the PPC bus time-out length. The time-out length is encoded as follows:

MBT	Time Out Length
00	256 µsec
01	64 µsec
10	8 µsec
11	disabled

P64 64-bit PCI Mode Enable. If set, the Raven is connected to a 64-bit PCI bus.

OPIC OpenPIC Interrupt Controller Enable. If set, the Raven internal OpenPIC interrupt controller is enabled. If cleared, Raven detected errors will be passed on to processor 0 INT pin.

MIDx Master ID. This field is encoded as shown below to indicate who is currently the PPC bus master. This information is obtained by sampling the CPUID pins. In a multiprocessor environment, these bits allow software to determine on which processor it is currently running. The internal PPC arbiter encodes this field as follows:

MID	Current PPC Data Bus Master
00	device on ABG0*
01	device on ABG1*
10	device on ABG2
11	Raven

FEAT Feature Register. Each bit in this register reflects the state of one of the external interrupt input pins on the rising edge of RESET*. This register may be used to report hardware configuration parameters to system software.

Prescaler Adjust Register

Address	$FEFF0010																															
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Name																									PADJ							
Operation	R								R								R							R/W							
Reset	$00								$00								$00								$BE							

PADJ Prescaler Adjust. This register is used to specify a scale factor for the prescaler to ensure that the time base for the bus timer is 1 MHz. The scale factor is calculated as follows:

PADJ = 256 - Clk,

where Clk is the frequency of the CLK input in MHz. The following table shows the scale factors for some common CLK frequencies.

Frequency	PADJ
66	$BE
50	$CE
40	$D8
33	$DF
25	$E7

PPC Error Enable Register

Address	$FEFF0020																													
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Name	ERRTST																ERREN															
	DPE0	DPE1	DPE2	DPE3	DPE4	DPE5	DPE6	DPE7										DFLT	PATOM	PDPEM	PERRM	SERRM	SMAM	RTAM			PATOII	PDPEI	PERRI	SERRI	SMAI	RTAI
Reset	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R								R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	R	R/W	R/W	R/W	R/W	R/W	R/W
Address	0	0	0	0	0	0	0	0	$00								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DPE*x* Data Parity Error Enable. These bits are used for test reasons to purposely inject data parity errors whenever Raven is sourcing PPC data. A data parity error will be created on the corresponding PPC data parity bus if a bit is set. For example, setting DPE0 will cause DP0 to be generated incorrectly. If the bit is cleared, Raven will generate correct data parity. These bits only have meaning if PPC data parity mode is enabled

DFLT Default PPC Master ID. This bit determines which MCHK* pin will be asserted for error conditions in which the PPC master ID cannot be determined or the Raven was the PPC master. For example, in event of a PCI parity error for a transaction in which the Raven's PCI master was not involved, the PPC master ID cannot be determined. When DFLT is set, MCHK1* is used. When DFLT is clear, MCHK0* will be used.

PATOM PPC Address Bus Time-out Machine Check Enable. When this bit is set, the PATO bit in the ERRST register will be used to assert the MCHK output to the current address bus master. When this bit is clear, MCHK will not be asserted.

PDPEM PPCData Parity Error Machine Check Enable. When this bit is set, the PDPE bit in the ERRST register will be used to assert the MCHK output to the current address bus master. When this bit is clear, MCHK will not be asserted.

PERRM PCI Parity Error Machine Check Enable. When this bit is set, the PERR bit in the ERRST register will be used to assert the MCHK output to bus master 0. When this bit is clear, MCHK will not be asserted.

SERRM PCI System Error Machine Check Enable. When this bit is set, the SERR bit in the ERRST register will be used to assert the MCHK output to bus master 0. When this bit is clear, MCHK will not be asserted.

SMAM PCI Signalled Master Abort Machine Check Enable. When this bit is set, the SMA bit in the ERRST register will be used to assert the MCHK output to the bus master which initiated the transaction. When this bit is clear, MCHK will not be asserted.

RTAM PCI Master Received Target Abort Machine Check Enable. When this bit is set, the RTA bit in the ERRST register will be used to assert the MCHK output to the bus master which initiated the transaction. When this bit is clear, MCHK will not be asserted.

PATOI PPC Address Bus Time-out Interrupt Enable. When this bit is set, the PATO bit in the ERRST register will be used to assert an interrupt through the OpenPIC interrupt controller. When this bit is clear, no interrupt will be asserted.

PDPEI PPC Data Parity Error Interrupt Enable. When this bit is set, the PDPE bit in the ERRST register will be used to assert an interrupt through the OpenPIC interrupt controller. When this bit is clear, no interrupt will be asserted.

PERRI PCI Parity Error Interrupt Enable. When this bit is set, the PERR bit in the ERRST register will be used to assert an interrupt through the OpenPIC interrupt controller. When this bit is clear, no interrupt will be asserted.

SERRI PCI System Error Interrupt Enable. When this bit is set, the PERR bit in the ERRST register will be used to assert an interrupt through the OpenPIC interrupt controller. When this bit is clear, no interrupt will be asserted.

SMAI PCI Master Signalled Master Abort Interrupt Enable. When this bit is set, the SMA bit in the ERRST register will be used to assert an interrupt through the OpenPIC interrupt controller. When this bit is clear, no interrupt will be asserted.

RTAI PCI Master Received Target Abort Interrupt Enable. When this bit is set, the RTA bit in the ERRST register will be used to assert an interrupt through the OpenPIC interrupt controller. When this bit is clear, no interrupt will be asserted.

PPC Error Status Register

Address	$FEFF0024																													
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Name																									ERRST							
																									OVF		PATO	PDPE	PERR	SERR	SMA	RTA
Operation	R								R								R							R/C	R	R/C	R/W	R/C	R/C	R/C	R/C
Reset	$00								$00								$00								0	0	0	0	0	0	0	0

OVF Error Status Overflow. This bit is set when any error is detected and any of the error status bits are already set. It may be cleared by writing a 1 to it; writing a 0 to it has no effect.

PATO PPC Address Bus Time-out. This bit is set when the PPC address bus timer times out. It may be cleared by writing it to a 1; writing it to a 0 has no effect. When the PATOM bit in the ERREN register is set, the assertion of this bit will assert MCHK to the master designated by the MID field in the ERRAT register. When the PATOI bit in the ERREN register is set, the assertion of this bit will assert an interrupt through the MPIC interrupt controller.

PDPE PPC Data Parity Error. This bit is set when Raven detects a data bus parity error. This bit is only valid if PPC data bus parity mode is enabled. It may be cleared by writing it to a 1: writing it to a 0 has no effect. When the PDPEM bit in the ERREN register is set, the assertion of this bit will assert MCHK to the master designated by the MID field in the ERRAT reigster. When the PDPEI bit in the ERREN register is set, the assertion of this bit will assert an interrupt through the OpenPIC interrupt controller.

PERR PCI Parity Error. This bit is set when the PCI PERR* pin is asserted. It may be cleared by writing it to a 1; writing it to a 0 has no effect. When the PERRM bit in the ERREN register is set, the assertion of this bit will assert MCHK to the master designated by the DFLT bit in the ERRAT register. When the PERRI bit in the ERREN register is set, the assertion of this bit will assert an interrupt through the OpenPIC interrupt controller.

SERR PCI System Error. This bit is set when the PCI SERR* pin is asserted. It may be cleared by writing it to a 1; writing it to a 0 has no effect. When the SERRM bit in the ERREN register is set, the assertion of this bit will assert MCHK to the master designated by the DFLT bit in the ERRAT register. When the SERRI bit in the ERREN register is set, the assertion of this bit will assert an interrupt through the OpenPIC interrupt controller.

SMA PCI Master Signalled Master Abort. This bit is set when the PCI master signals master abort to terminate a PCI transaction. It may be cleared by writing it to a 1; writing it to a 0 has no effect. When the SMAM bit in the ERREN register is set, the assertion of this bit will assert MCHK to the master designated by the MID field in the ERRAT register. When the SMAI bit in the ERREN register is set, the assertion of this bit will assert an interrupt through the OpenPIC interrupt controller.

RTA PCI Master Received Target Abort. This bit is set when the PCI master receives target abort to terminate a PCI transaction. It may be cleared by writing it to a 1; writing it to a 0 has no effect. When the RTAM bit in the ERREN register is set, the assertion of this bit will assert MCHK to the master designated by the MID field in the ERRAT register. When the RTAI bit in the ERREN register is set, the assertion of this bit will assert an interrupt through the OpenPIC interrupt controller.

PPC Error Address Register

Address	$FEFF0028																															
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Name	EERAD																															
Operation	R																															
Reset	$00000000																															

EERAD PPC Error Address. This register captures the PPC address when the PATO bit is set in the EERST register. It captures the PCI address when the SMA or RTA bits are set in the EERST register. Its contents are not defined when the PDPE, PERR or SERR bits are set in the EERST register.

PPC Error Attribute Register - MERAT

If the PDPE, PERR or SERR bits are set in the MERST register, the contents of the MERAT register are zero. If the MATO bit is set the register is defined by the following figure:

Address	$FEFF002C																															
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Name																	MERAT															
																			MID1	MID0			TBST	TSIZ0	TSIZ1	TSIZ2	TT0	TT1	TT2	TT3	TT4
Operation	R								R								R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Reset	$00								$00								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MIDx PPC Master ID. This field contains the ID of the PPC master which originated the transfer in which the error occurred. The encoding scheme is identical to that used in the GCSR register.

TBST* Transfer Burst. This bit is set when the transfer in which the error occurred was a burst transfer.

TSIZx* Transfer Size. This field contains the transfer size of the PPC transfer in which the error occurred.

TTx* Transfer Type. This field contains the transfer type of the PPC transfer in which the error occurred.

* Refer to PowerPC documents listed in Table A-2 on page A-2.

If the SMA or RTA bit are set the register is defined by the following table.

Address	$FEFF002C																															
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Name																	MERAT															
																	WP		MID1	MID0	COMM3	COMM2	COMM1	COMM0	BYTE7	BYTE6	BYTE5	BYTE4	BYTE3	BYTE2	BYTE1	BYTE0
Operation	R								R								R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Reset	$00								$00								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

WP Write Post Completion. This bit is set when the PCI master detects an error while completing a write post transfer.

MID*x* PPC Master ID. This field contains the ID of the PPC master which originated the transfer in which the error occurred. The encoding scheme is identical to that used in the GCSR register

COMM*x* PCI Command. This field contains the PCI command of the PCI transfer in which the error occurred.

BYTE*x* PCI Byte Enable. This field contains the PCI byte enables of the PCI transfer in which the error occurred. A set bit designates a selected byte.

PCI Interrupt Acknowledge Register

Address	$FEFF0030
Bit	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Name	PIACK
Operation	R
Reset	$00000000

PIACK PCI Interrupt Acknowledge. Performing a read from this register will initiate a single PCI Interrupt Acknowledge cycle. Any single byte or combination of bytes may be read from, and the actual byte enable pattern used during the read will be passed on to the PCI bus. Upon completion of the PCI interrupt acknowledge cycle, the Raven will present the resulting vector information obtained from the PCI bus as read data.

PPC Slave Address (0,1 and 2) Registers

Address	MSADD0 - $FEFF0040 MSADD1 - $FEFF0048 MSADD2 - $FEFF0050	
Bit	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31	
Name	MSADDx	
	START	END
Operation	R/W	R/W
Reset	$0000	$0000

To initiate a PCI cycle from the PPC bus, the PPC address must be greater than or equal to the START field and less than or equal to the END field.

START Start Address. This field determines the start address of a particular memory area on the PPC bus which will be used to access PCI bus resources. The value of this field will be compared with the upper 16 bits of the incoming PPC address.

END End Address. This field determines the end address of a particular memory area on the PPC bus which will be used to access PCI bus resources. The value of this field will be compared with the upper 16 bits of the incoming PPC address.

PPC Slave Address (3) Register

Address	MSADD3 - $FEFF0058	
Bit	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31	
Name	MSADD3	
	START	END
Operation	R/W	R/W
Reset	$8000	$8080

MSADD3, MSOFF3, and MSATT3 represent the only register group which can be used to initiate access to the PCI Configuration Address ($80000CF8) and Configuration Data ($80000CFC) registers. Note that this implies that MS*xxx*3 also represents the generation of PCI Special Cycles. The power up default values of MSADD3, MSOFF3, and MSATT3 are set to allow access to PCI configuration space without PPC register initialization. Please see the description of the following registers for additional information.

To initiate a PCI cycle from the PPC bus, the PPC address must be greater than or equal to the START field and less than or equal to the END field.

START Start Address. This field determines the start address of a particular memory area on the PPC bus which will be used to access PCI bus resources. The value of this field will be compared with the upper 16 bits of the incoming PPC address.

END End Address. This field determines the end address of a particular memory area on the PPC bus which will be used to access PCI bus resources. The value of this field will be compared with the upper 16 bits of the incoming PPC address.

PPC Slave Offset/Attribute (0,1 and 2) Registers

Address	MSOFF0/MSATT0 - $FEFF0044 MSOFF1/MSATT1 - $FEFF004C MSOFF2/MSATT2 - $FEFF0054																													
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Name	MSOFFx																								MSATTx							
																									REN	WEN		WPEN			MEM	IOM
Operation	R/W																R								R/W	R/W	R	R/W	R	R	R/W	R/W
Reset	$0000																$00								0	0	0	0	0	0	0	0

MSOFF*x* PPC Slave Offset. This register contains a 16-bit offset that is added to the upper 16 bits of the PPC address to determine the PCI address used for transfers from the PPC bus to PCI. This offset allows PCI resources to reside at addresses that would not normally be visible from the PPC bus.

REN Read Enable. If set, the corresponding PPC slave is enabled for read transactions.

WEN Write Enable. If set, the corresponding PPC slave is enabled for write transactions.

WPEN Write Post Enable. If set, write posting is enabled for the corresponding PPC slave.

MEM PCI Memory Cycle. If set, the corresponding PPC slave will generate transfers to or from PCI memory space. When clear, the corresponding PPC slave will generate transfers to or from PCI I/O space using the addressing mode defined by the IOM field.

IOM PCI I/O Mode. If set, the corresponding PPC slave will generate PCI I/O cycles using spread addressing as defined in the section on *Generating PCI Memory and I/O Cycles*. When clear, the corresponding PPC slave will generate PCI I/O cycles using contiguous addressing. This field only has meaning when the MEM bit is clear.

PPC Slave Offset/Attribute (3) Registers

Address	MSOFF3/MSATT3 - $FEFF005C																													
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Name	MSOFF3																								MSATT3							
																									REN	WEN		WPEN				IOM
Operation	R/W																R								R/W	R/W	R	R/W	R	R	R	R/W
Reset	$8000																$00								1	1	0	0	0	0	0	0

MSOFF3 PPC Slave Offset. This register contains a 16-bit offset that is added to the upper 16 bits of the PPC address to determine the PCI address used for transfers from the PPC bus to PCI. This offset allows PCI resources to reside at addresses that would not normally be visible from the PPC bus. It is initialized to $8000 to facilitate a zero-based access to PCI space.

REN Read Enable. If set, the corresponding PPC slave is enabled for read transactions.

WEN Write Enable. If set, the corresponding PPC slave is enabled for write transactions.

WPEN Write Post Enable. If set, write posting is enabled for the corresponding PPC slave.

IOM PCI I/O Mode. If set, the corresponding PPC slave will generate PCI I/O cycles using spread addressing as defined in the section on *Generating PCI Memory and I/O Cycles*. When clear, the corresponding PPC slave will generate PCI I/O cycles using contiguous addressing.

General Purpose Registers

Address	GPREG0 (Upper) - $FEFF0070 GPREG0 (Lower) - $FEFF0074 GPREG1 (Upper) - $FEFF0078 GPREG1 (Lower) - $FEFF007C
Bit	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Name	GPREGx
Operation	R/W
Reset	$00000000

These general purpose read/write registers are provided for inter-process message passing or general purpose storage. They do not control any hardware.

PCI Registers

The PCI Configuration Registers are compliant with the configuration register set described in the PCI Local Bus Specification, listed in Appendix A, *Related Documentation*. The CONFIG_ADDRESS and CONFIG_DATA registers described in this section are accessed within PCI I/O space.

All write operations to reserved registers will be treated as no-ops. That is, the access will be completed normally on the bus and the data will be discarded. Read accesses to reserved or unimplemented registers will be completed normally and a data value of 0 returned.

The Raven PCI Configuration Register map is shown in Table 2-6. The Raven PCI I/O Register map is shown in Table 2-7.

Table 2-6. Raven PCI Configuration Register Map

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	<--- Bit
DEVID																VENID																**$00**
PSTAT																PCOMM																**$04**
CLASS																								REVID								**$08**
								HEADER																								**$0C**
IOBASE																																**$10**
MEMBASE																																**$14**
																																$18 - $7F
PSADD0																																**$80**
PSOFF0																									PSATT0								**$84**
PSADD1																																**$88**
PSOFF1																									PSATT1								**$8C**
PSADD2																																**$90**
PSOFF2																									PSATT2								**$94**
PSADD3																																**$98**
PSOFF3																									PSATT3								**$9C**

Table 2-7. Raven PCI I/O Register Map

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	<--- Bit
CONFIG_ADDRESS																																**$CF8**
CONFIG_DATA																																**$CFC**

Vendor ID/ Device ID Registers

Offset	$00																															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	DEVID																VENID															
Operation	R																R															
Reset	$4801																$1057															

VENID Vendor ID. This register identifies the manufacturer of the device. This identifier is allocated by the PCI SIG to ensure uniqueness. $1057 has been assigned to Motorola. This register is duplicated in the PPC Registers.

DEVID Device ID. This register identifies the particular device. The Raven will always return $4801. This register is duplicated in the PPC Registers.

PCI Command/ Status Registers

Offset	$04																													
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	PSTAT																PCOMM															
	RCVPE	SIGSE	RCVMA	RCVTA	SIGTA	SELTIM1	SELTIM0	DPAR	FAST														SERR		PERR				MSTR	MEMSP	IOSP
Operation	R/C	R/C	R/C	R/C	R/C	R	R	R/C	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R/W	R	R/W	R	R	R	R/W	R/W	R/W
Reset	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

IOSP IO Space Enable. If set, the Raven will respond to PCI I/O accesses when appropriate. If cleared, the Raven will not respond to PCI I/O space accesses.

MEMSP Memory Space Enable. If set, the Raven will respond to PCI memory space accesses when appropriate. If cleared, the Raven will not respond to PCI memory space accesses.

MSTR Bus Master Enable. If set, the Raven may act as a master on PCI. If cleared, the Raven may not act as a PCI master.

PERR Parity Error Response. If set, the Raven will check parity on all PCI transfers. If cleared, the Raven will ignore any parity errors that it detects and continue normal operation.

SERR System Error Enable. This bit enables the SERR* output pin. If clear, the Raven will never drive SERR*. If set, the Raven will drive SERR* active when a system error is detected.

FAST Fast Back-to-Back Capable. This bit indicates that the Raven is capable of accepting fast back-to-back transactions with different targets.

DPAR Data Parity Detected. This bit is set when three conditions are met: 1) the Raven asserted PERR* itself or observed PERR* asserted; 2) the Raven was the PCI master for the transfer in which the error occurred; 3) the PERR bit in the PCI Command Register is set. This bit is cleared by writing it to 1; writing a 0 has no effect.

SELTIM DEVSEL Timing. This field indicates that the Raven will always assert DEVSEL* as a 'medium' responder.

SIGTA Signalled Target Abort. This bit is set by the PCI slave whenever it terminates a transaction with a target-abort. It is cleared by writing it to 1; writing a 0 has no effect.

RCVTA Received Target Abort. This bit is set by the PCI master whenever its transaction is terminated by a target-abort. It is cleared by writing it to 1; writing a 0 has no effect.

RCVMA Received Master Abort. This bit is set by the PCI master whenever its transaction (except for Special Cycles) is terminated by a master-abort. It is cleared by writing it to 1; writing a 0 has no effect.

SIGSE Signaled System Error. This bit is set whenever the Raven asserts SERR*. It is cleared by writing it to 1; writing a 0 has no effect.

RCVPE Detected Parity Error. This bit is set whenever the Raven detects a parity error, even if parity error checking is disabled (see bit PERR in the PCI Command Register). It is cleared by writing it to 1; writing a 0 has no effect.

Revision ID/ Class Code Registers

Offset	\$08																													
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CLASS																								REVID							
Operation	R																								R							
Reset	\$060000																								\$03							

REVID Revision ID. This register identifies the Raven revision level. This register is duplicated in the PPC Registers.

CLASS Class Code. This register identifies Raven as the following:

Base Class Code	$06	PCI Bridge Device
Subclass Code	$00	PCI Host Bridge
Program Class Code	$00	Not Used

Header Type Register

Offset	$0C			
Bit	31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
Name		HEADER		
Operation	R	R	R	R
Reset	$00	$00	$00	$00

HEADER Header Type. This register identifies Raven as the following:

Header Type	$00	Single Function Configuration Header

I/O Base Register

Offset	$10			
Bit	31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
Name	IOBASE			
	IOBA			RES / IO/MEM
Operation	R/W	R		R / R
Reset	$0000	$0000		0 / 1

This register controls the mapping of the MPIC control registers in PCI I/O space.

IO/MEM IO Space Indicator. This bit is hard-wired to a logic one to indicate PCI I/O space.

RES Reserved. This bit is hard-wired to zero.

IOBA I/O Base Address. These bits define the I/O space base address of the MPIC control registers. The IOBASE decoder is disabled when the IOBASE value is zero.

Memory Base Register

Offset	\$14																															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	MEMBASE																															
	MEMBA																												PRE	MTYP1	MTYP0	IO/MEM
Operation	R/W																						R						R	R	R	R
Reset	\$0000																						\$0000						0	0	0	0

This register controls the mapping of the MPIC control registers in PCI memory space.

IO/MEMIO Space Indicator. This bit is hard-wired to a logic zero to indicate PCI memory space.

MTYP*x* Memory Type. These bits are hard-wired to zero to indicate that the MPIC registers can be located anywhere in the 32-bit address space

PRE Prefetch. This bit is hard-wired to zero to indicate that the MPIC registers are not prefetchable.

MEMBA Memory Base Address. These bits define the memory space base address of the MPIC control registers. The MBASE decoder is disabled when the MBASE value is zero.

PCI Slave Address (0,1,2 and 3) Registers

Offset	PSADD0 - $80 PSADD1 - $88 PSADD2 - $90 PSADD3 - $98																															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	PSADD*x*																															
	START																END															
Operation	R/W																R/W															
Reset	$0000																$0000															

To initiate an PPC cycle from the PCI bus the PCI address must be greater than or equal to the START field and less than or equal to the END field.

START Start Address. This field determines the start address of a particular memory area on the PCI bus which will be used to access PPC bus resources. The value of this field will be compared with the upper 16 bits of the incoming PCI address.

END End Address. This field determines the end address of a particular memory area on the PCI bus which will be used to access PPC bus resources. The value of this field will be compared with the upper 16 bits of the incoming PCI address.

2

PCI Slave Attribute/ Offset (0,1,2 and 3) Registers

Offset	PSATT0/PCISOFF0 - $84 PSATT1/PCISOFF1 - $8C PSATT2/PCISOFF2 - $94 PSATT3/PCISOFF3 - $9C																															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	PCISOFFx																								PSATT*x*							
																									REN	WEN	WPEN	RAEN	TDIS		GBL	INV
Operation	R/W																R								R/w	R/W	R/W	R/W	RW	R	R/W	R/W
Reset	$0000																$00								0	0	0	0	0	0	0	0

INV Invalidate Enable. If set, the PPC master will issue a transfer type code which specifies the current transaction should cause an invalidate for each PC transaction originated by the corresponding PCI slave. The transfer type codes generated are shown in Table 2-3.

GBL Global Enable. If set, the PPC master will assert the GBL* pin for each PPC transaction originated by the corresponding PCI slave.

TDIS Threshold Disable. If set, the PPC master FIFO threshold mechanism is disabled. See the section titled *PPC Master* for more information on the relationship between TDIS and RAEN.

RAEN Read Ahead Enable. If set, read ahead is enabled for the corresponding PCI slave.

WPEN Write Post Enable. If set, write posting is enabled for the corresponding PCI slave.

WEN Write Enable. If set, the corresponding PCI slave is enabled for write transactions.

REN Read Enable. If set, the corresponding PCI slave is enabled for read transactions.

PCIOFFx PCI Slave Offset. This register contains a 16-bit offset that is added to the upper 16 bits of the PCI address to determine the PPC address used for transfers from PCI to the PPC bus. This offset allows PPC resources to reside at addresses that would not normally be visible from PCI.

CONFIG_ADDRESS

The routing of the PPC data bus to and from the CONFIG_ADDRESS register depends on the endian bit setting. Refer to the sections on *Endian Conversion*, and the LEND bit in the GCSR. The following register diagrams have two additional rows of information. These rows indicate the source bit positions on the PPC data bus when data is written to this register. A read from the CONFIG_ADDRESS register will return contents to these locations. One row defines the source for big-endian operation and the second is for little-endian operation. LEND is the little-endian control bit in the GCSR.

The MSADD3, MSOFF3, and MSATT3 are initialized at reset so software can access PCI configuration address and data space without changing Raven registers.

PCI I/O CONFIG_ADDRESS Register

Offset	$CF8																															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CONFIG_ADDRESS																															
	EN								BUS								DEV					FUN				REG						
Operation	R/W	R							R/W								R/W					R/W								R	R	
Reset	1	$00							$00								$00					$0				$00					0	0
LEND = 0	PPC[39:32]								PPC[47:40]								PPC[55:48]									PPC[63:56]						
LEND = 1	PPC[31:24]								PPC[23:16]								PPC[15:8]									PPC[7:0]						

REG Register Number. For PCI Configuration cycles, bits 7 through 2 identify the target double word within the target function's configuration space. Bits 1 and 0 must always be zero for a type 0 configuration cycle. These bits are copied to the PCI AD bus during the address phase on a Configuration cycle.This field must be all zeros for Special cycles.

FUN Function Number. For PCI Configuration cycles, bits 10 through 8 identify the function number within the target physical PCI device. These bits are copied to the PCI AD bus during the address phase on a Configuration cycle. This field must be all ones for Special cycles.

DEV Device Number. For PCI Configuration cycles, bits 15 through 11 identify the target physical PCI device number. Raven does a decode of the Device Number field to assert the appropriate IDSEL line. Values of $01 through $0a and $1f are illegal entries for the device number. The Raven will drive all 0's in bit position AD11 through AD31 if a illegal device id is initialized into the configuration address register. A value of $0B sets PCI AD bit 11 (IDSEL 11) during the address phase of a Configuration cycle. A value of $0C sets AD bit 12. As the device number increments the AD bit increments until a the value of $1E sets AD bit 30. A value of $00 in device number field will select AD bit 31. The device number field must be all ones for Special cycles.

BUS Bus Number. For PCI Configuration cycles or Special cycles, bits 23 through 16 identify the bus number. Raven is always connected to PCI bus number zero. Bits 23 through 16 must be zero for a Configuration cycle or Special cycle. Raven will execute a type 1 translation cycle if the bus number is set to a value not equal to zero.

EN Enable. Bit 31 must be set to a one, enabling the translation of a subsequent host bus I/O access to the CONFIG_DATA register into a configuration access on the PCI bus. If bit 31 is zero and the processor initiates an I/O read from or write to the CONFIG_DATA register, the transaction is passed through to the PCI bus as a PCI I/O transaction.

PCI I/O CONFIG_DATA Register

Offset	$CFC																													
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CONFIG_DATA																															
Operation	R/W																															
Reset	$00000000																															

If the CONFIG_ADDRESS register is initialized for a PCI Configuration cycle, an access to the CONFIG_DATA register will write to or read from a PCI configuration space location. If the CONFIG_ADDRESS register is initialized for a PCI Special cycle, a word write to the CONFIG_DATA register will generate a special cycle on the PCI bus.

Raven Interrupt Controller Implementation

Introduction

The Raven Interrupt Controller (Raven MPIC) Features

- MPIC programming model
- Support for two processors
- Support for 16 external interrupts
- Support for 15 programmable Interrupt & Processor Task priority levels
- Support for the connection of an external 8259 for ISA/AT compatibility
- Distributed interrupt delivery for external I/O interrupts
- Direct/Multicast interrupt delivery for Interprocessor and timer interrupts

- Four Interprocessor Interrupt sources
- Four timers
- Processor initialization control

Architecture

The Raven PCI Slave implements two address decoders for placing the Raven MPIC registers in PCI IO or PCI Memory space. Access to these registers require MPC and PCI bus mastership. These accesses include interrupt and timer initialization and interrupt vector reads.

The Raven MPIC receives interrupt inputs from 16 external sources, four interprocessor sources, four timer sources, and one Raven internal error detection source. The externally sourced interrupts 1 through 15 have two modes of activation: low level or active high positive edge. External interrupt 0 can be either level or edge activated with either polarity. The Interprocessor and timers interrupts are event activated.

CSR's Readability

Unless explicitly specified, all registers are readable and return the last value written. The exceptions are the IPI dispatch registers and the EOI registers which return zeros on reads, the interrupt source ACT bit which returns current interrupt source status, the interrupt acknowledge register which returns the vector of the highest priority interrupt which is currently pending, and reserved bits which returns zeros. The interrupt acknowledge register is also the only register which exhibits any read side-effects.

Interrupt Source Priority

Each interrupt source is assigned a priority value in the range from 0 to 15 where 15 is the highest. In order for delivery of an interrupt to take place the priority of the source must be greater than that of the destination processor. Therefore setting a source priority to zero inhibits that interrupt.

Processor's Current Task Priority

Each processor has a task priority register which is set by system software to indicate the relative importance of the task running on that processor. The processor will not receive interrupts with a priority level equal to or lower than its current task priority. Therefore setting the current task priority to 15 prohibits the delivery of all interrupts to the associated processor.

Nesting of Interrupt Events

A processor is guaranteed never to have an in-service interrupt preempted by an equal or lower priority source. An interrupt is considered to be in service from the time its vector is returned during an interrupt acknowledge cycle until an EOI is received for that interrupt. The EOI cycle indicates the end of processing for the highest priority in-service interrupt.

Spurious Vector Generation

Under certain circumstances the Raven MPIC will not have a valid vector to return to the processor during an interrupt acknowledge cycle. In these cases the spurious vector from the spurious vector register will be returned. The following cases would cause a spurious vector fetch.

- INT is asserted in response to an externally sourced interrupt which is activated with level sensitive logic and the asserted level is negated before the interrupt is acknowledged.
- INT is asserted for an interrupt source which is masked using the mask bit in the Vector-Priority register before the interrupt is acknowledged.

Interprocessor Interrupts (IPI)

Processor 0 and 1 can generate interrupts which are targeted for the other processor or both processors. There are four Interprocessor Interrupts (IPI) channels. The interrupts are initiated by writing a bit in the IPI dispatch

registers. If subsequent IPIs are initiated before the first is acknowledged, only one IPI will be generated. The IPI channels deliver interrupts in the Direct Mode and can be directed to more than one processor.

8259 Compatibility

The Raven MPIC provides a mechanism to support PC-AT compatible chip sets using the 8259 interrupt controller architecture. After power-on reset, the Raven MPIC defaults to 8259 pass-through mode. In this mode, interrupts from external source number 0 (the interrupt signal from the 8259 is connected to this external interrupt source on the Raven MPIC) are passed directly to processor 0. If the pass-through mode is disabled, the 8259 interrupts are delivered using the priority and distribution mechanisms of the Raven MPIC.

The Raven MPIC does not interact with the vector fetch from the 8259 interrupt controller.

Raven-Detected Errors

Raven-detected errors are grouped together and sent to the interrupt logic as a singular interrupt source. The interrupt delivery mode for this interrupt is distributed.

For system implementations where the Raven MPIC controller is not used, the Raven-Detected Error condition will be made available by a signal which is external to the Raven ASIC. Presumably this signal would be connected to an externally sourced interrupt input of a MPIC controller in a different device. Since the MPIC specification defines external I/O interrupts to operate in the distributed mode, the delivery mode of this error interrupt should be consistent.

Timers

There is a divide by eight pre-scaler which is synchronized to the Raven clock (MPC processor clock). The output of the prescaler enables the decrement of the four timers. The timers may be used for system timing or to generate periodic interrupts. Each timer has four registers which are used for configuration and control. They are:

- Current Count Register

- Base Count Register
- Vector-Priority Register
- Destination Register

Interrupt Delivery Modes

The direct and distributed interrupt delivery modes are supported.

Note The direct deliver mode has sub modes of multicast or non-multicast. The Interprocessor Interrupts (IPIs) and Timer interrupts operate in the direct delivery mode. The externally sourced or I/O interrupts operate in the distributed mode.

In the direct delivery mode, the interrupt is directed to one or both processors. If it is directed to two processors (that is, multicast), it will be delivered to two processors. The interrupt is delivered to the processor when the priority of the interrupt is greater than the priority contained in the task register for that processor, and when the priority of the interrupt is greater than any interrupt which is in-service for that processor. An interrupt is considered to be in service from the time its vector is returned during an interrupt acknowledge cycle until an EOI is received for that interrupt. The EOI cycle indicates the end of processing for the highest priority in- service interrupt.

In the distributed delivery mode, the interrupt is pointed to one or more processors but it will be delivered to only one processor. Therefore, for externally sourced or I/O interrupts, multicast delivery is not supported.The interrupt is delivered to a processor when the priority of the interrupt is greater than the priority contained in the task register for that processor, and when the priority of the interrupt is greater than any interrupt which is in-service for that processor, and when the priority of that interrupt is the highest of all interrupts pending for that processor, and when that interrupt is not in-service for the other processor. If both destination bits are set for each processor, the interrupt will be delivered to the processor that has a lower task register priority.

Note Because a deadlock condition can occur when the task register priorities for each processor are the same and both processors are targeted for interrupt delivery, the interrupt will be delivered to processor 0 or processor 1 as determined by the TIE mode.

Block Diagram Description

The description of the block diagram focuses on the theory of operation for the interrupt delivery logic. If the preceding section is a satisfactory description of the interrupt delivery modes and the reader is not interested the logic implementation, this section can be skipped.

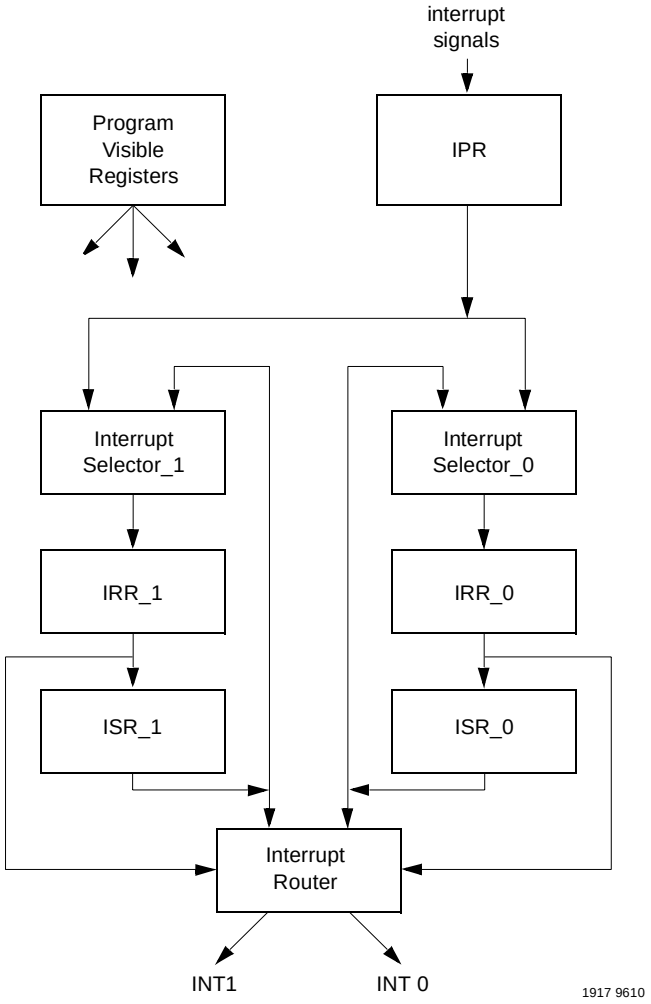


Figure 2-4. Raven MPIC Block Diagram

Program Visible Registers

These are the registers which software can access. They are described in detail in the Register section.

Interrupt Pending Register (IPR)

The interrupt signals to Raven MPIC are qualified and synchronized to the clock by the IPR. If the interrupt source is internal to the Raven ASIC or external with their Sense bit = 0 (edge sensitive), a bit is set in the IPR. That bit is cleared when the interrupt associated with that bit is acknowledge. If the interrupt source is external and level activated, the output from the IPR is not negated until the level into the IPR is negated.

Externally sourced interrupts are qualified based upon their Sense and/or Pol bits in the Vector-Priority register. IPI and Timer Interrupts are generated internally to the Raven ASIC and are qualified by their Destination bit. Since the internally generated interrupts use direct delivery mode with multicast capability, there are two bits in the IPR, one for each processor, associated with each IPI and Timer interrupt source.

The MASK bits from the Vector-Priority registers are used to qualify the output of the IPR. Therefore, if an interrupt condition is detected when the MASK bit is set, that interrupt will be requested when the MASK bit is lowered.

Interrupt Selector (IS)

There is a Interrupt Selector (IS) for each processor. The IS receives interrupt requests from the IPR. If the interrupt request are from an external source, they are qualified by the destination bit for that interrupt and processor. If they are from an internal source, they have been qualified. The output of the IS will be the highest priority interrupt that has been qualified. This output is the priority of the selected interrupt and its source identification. The IS will resolve an interrupt request in two Raven clock ticks.

The IS also receives a second set of inputs from the ISR. During the End Of Interrupt cycle, these inputs are used to select which bits are to be cleared in the ISR.

Interrupt Request Register (IRR)

There is a Interrupt Request Register (IRR) for each processor. The IRR always passes the output of the IS except during Interrupt Acknowledge cycles. This guarantees that the vector which is read from the Interrupt Acknowledge Register is not changing due to the arrival of a higher priority interrupt. The IRR also serves as a pipeline register for the two tick propagation time through the IS.

In-Service Register (ISR)

There is a In-Service Register (ISR) for each processor. The contents of the ISR is the priority and source of all interrupts which are in-service. The ISR receives a bit-set command during Interrupt Acknowledge cycles and a bit-clear command during End Of Interrupt cycles.

The ISR is implemented as a 40 bit register with individual bit set and clear functions. Fifteen bits are used to store the priority level of each interrupt which is in-service. Twenty-five bits are used to store the source identification of each interrupt which is in service. Therefore there is one bit for each possible interrupt priority and one bit for each possible interrupt source.

Interrupt Router

The Interrupt Router monitors the outputs from the ISRs, Current Task Priority Registers, Destination Registers, and the IRRs to determine when to assert a processor's INT pin.

When considering the following rule sets, it is important to remember that there are two types of inputs to the Interrupt Selectors. If the interrupt is a distributed class interrupt, there is a single bit in the IPR associated with this interrupt and it is delivered to both Interrupt Selectors. This IPR bit is qualified by the destination register contents for that interrupt before the Interrupt Selector compares its priority to the priority of all other requesting interrupts for that processor. If the interrupt is programmed to be edge sensitive, the IPR bit is cleared when the vector for that interrupt is returned when the Interrupt Acknowledge register is examined. On the other hand, if the interrupt is a direct/multicast class interrupt, there are two bits in the IPR associated with this interrupt. One bit for each processor.

Then one of these bits are delivered to each Interrupt Selector. Since this interrupt source can be multicast, each of these IPR bits must be cleared separately when the vector is returned for that interrupt to a particular processor.

If one of the following sets of conditions are true, the interrupt pin for processor 0 is driven active.

- Set1

  The source ID in IRR_0 is from an external source.

  The destination bit for processor 1 is a 0 for this interrupt.

  The priority from IRR_0 is greater than the highest priority in ISR_0.

  The priority from IRR_0 is greater than the contents of task register_0.

- Set2

  The source ID in IRR_0 is from an external source.

  The destination bit for processor 1 is a 1 for this interrupt.

  The source ID in IRR_0 is not present is ISR_1.

  The priority from IRR_0 is greater than the highest priority in ISR_0.

  The priority from IRR_0 is greater than the Task Register_0 contents.

  The contents of Task Register_0 is less than the contents of Task Register_1.

- Set3

  The source ID in IRR_0 is from an internal source.

  The priority from IRR_0 is greater than the highest priority in ISR_0.

  The priority from IRR_0 is greater than the Task Register_0 contents.

There is a possibility for a priority tie between the two processors when resolving external interrupts. In that case the interrupt will be delivered to processor 0 or processor 1 as determined by the TIE mode bit. This case is not defined in the above rule set.

MPIC Registers

The following conventions are used in the Raven register charts:

- R Read Only field.
- R/W Read/Write field.
- S Writing a ONE to this field sets this field.
- C Writing a ONE to this field clears this field.

Raven MPIC Registers

The Raven MPIC register map is shown in the following table. The Off field is the address offset from the base address of the Raven MPIC registers in the MPC-IO or MPC-MEMORY space.

Note This map does not depict linear addressing. The Raven PCI-SLAVE has two decoders for generating the Raven MPIC select. These decoders will generate a select and acknowledge all accesses which are in a reserved 256KB range. If the index into that 256KB block does not decode a valid Raven MPIC register address, the logic will return $00000000.

The registers are 8, 16, or 32 bits accessible.

Table 2-8. Raven MPIC Register Map

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Off
FEATURE REPORTING REGISTER 0																																**$01000**
GLOBAL CONFIGURATION REGISTER 0																																**$01020**
MPIC VENDOR IDENTIFICATION REGISTER																																**$01080**
PROCESSOR INIT REGISTER																																**$01090**
IPI0 VECTOR-PRIORITY REGISTER																																**$010a0**
IPI1 VECTOR-PRIORITY REGISTER																																**$010b0**
IPI2 VECTOR-PRIORITY REGISTER																																**$010c0**
IPI3 VECTOR-PRIORITY REGISTER																																**$010d0**
																								SP REGISTER								**$010e0**
TIMER FREQUENCY REPORTING REGISTER																																**$010f0**
TIMER 0 CURRENT COUNT REGISTER																																**$01100**
TIMER 0 BASE COUNT REGISTER																																**$01110**
TIMER 0 VECTOR-PRIORITY REGISTER																																**$01120**
TIMER 0 DESTINATION REGISTER																																**$01130**
TIMER 1 CURRENT COUNT REGISTER																																**$01140**
TIMER 1 BASE COUNT REGISTER																																**$01150**
TIMER 1VECTOR-PRIORITY REGISTER																																**$01160**
TIMER 1DESTINATION REGISTER																																**$01170**
TIMER 2 CURRENT COUNT REGISTER																																**$01180**
TIMER 2 BASE COUNT REGISTER																																**$01190**
TIMER 2 VECTOR-PRIORITY REGISTER																																**$011a0**
TIMER 2 DESTINATION REGISTER																																**$011b0**
TIMER 3 CURRENT COUNT REGISTER																																**$011c0**
TIMER 3 BASE COUNT REGISTER																																**$011d0**
TIMER 3 VECTOR-PRIORITY REGISTER																																**$011e0**

Table 2-8. Raven MPIC Register Map (Continued)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Off
TIMER 3 DESTINATION REGISTER																																**$011f0**
INT. SRC. 0 VECTOR-PRIORITY REGISTER																																**$10000**
INT. SRC. 0 DESTINATION REGISTER																																**$10010**
INT. SRC. 1 VECTOR-PRIORITY REGISTER																																**$10020**
INT. SRC. 1 DESTINATION REGISTER																																**$10030**
INT. SRC. 2 VECTOR-PRIORITY REGISTER																																**$10040**
INT. SRC. 2 DESTINATION REGISTER																																**$10050**
INT. SRC. 3 VECTOR-PRIORITY REGISTER																																**$10060**
INT. SRC. 3 DESTINATION REGISTER																																**$10070**
INT. SRC. 4 VECTOR-PRIORITY REGISTER																																**$10080**
INT. SRC. 4 DESTINATION REGISTER																																**$10090**
INT. SRC. 5 VECTOR-PRIORITY REGISTER																																**$100a0**
INT. SRC. 5 DESTINATION REGISTER																																**$100b0**
INT. SRC. 6 VECTOR-PRIORITY REGISTER																																**$100c0**
INT. SRC. 6 DESTINATION REGISTER																																**$100d0**
INT. SRC. 7 VECTOR-PRIORITY REGISTER																																**$100e0**
INT. SRC. 7 DESTINATION REGISTER																																**$100f0**
INT. SRC. 8 VECTOR-PRIORITY REGISTER																																**$10100**
INT. SRC. 8 DESTINATION REGISTER																																**$10110**
INT. SRC. 9 VECTOR-PRIORITY REGISTER																																**$10120**
INT. SRC. 9 DESTINATION REGISTER																																**$10130**
INT. SRC. 10 VECTOR-PRIORITY REGISTER																																**$10140**
INT. SRC. 10 DESTINATION REGISTER																																**$10150**
INT. SRC. 11 VECTOR-PRIORITY REGISTER																																**$10160**
INT. SRC. 11 DESTINATION REGISTER																																**$10170**
INT. SRC. 12 VECTOR-PRIORITY REGISTER																																**$10180**
INT. SRC. 12 DESTINATION REGISTER																																**$10190**
INT. SRC. 13 VECTOR-PRIORITY REGISTER																																**$101a0**

Table 2-8. Raven MPIC Register Map (Continued)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Off
INT. SRC. 13 DESTINATION REGISTER																																**$101b0**
INT. SRC. 14 VECTOR-PRIORITY REGISTER																																**$101c0**
INT. SRC. 14 DESTINATION REGISTER																																**$101d0**
INT. SRC. 15 VECTOR-PRIORITY REGISTER																																**$101e0**
INT. SRC. 15 DESTINATION REGISTER																																**$101f0**
RAVEN DETECTED ERRORS VECTOR-PRIORITY REGISTER																																**$10200**
RAVEN DETECTED ERRORS DESTINATION REGISTER																																**$10210**
IPI 0 DISPATCH REGISTER PROC. 0																																**$20040**
IPI 1 DISPATCH REGISTER PROC. 0																																**$20050**
IPI 2 DISPATCH REGISTER PROC. 0																																**$20060**
IPI 3 DISPATCH REGISTER PROC. 0																																**$20070**
CURRENT TASK PRIORITY REGISTER PROC. 0																																**$20080**
																								IACK REGISTER P0								**$200a0**
																								EOI REGISTER P0								**$200b0**
IPI 0 DISPATCH REGISTER PROC. 1																																**$21040**
IPI 1 DISPATCH REGISTER PROC. 1																																**$21050**
IPI 2 DISPATCH REGISTER PROC. 1																																**$21060**
IPI 3 DISPATCH REGISTER PROC. 1																																**$21070**
CURRENT TASK PRIORITY REGISTER PROC. 1																																**$21080**
																								IACK REGISTER P1								**$210a0**
																								EOI REGISTER P1								**$210b0**

Feature Reporting Register

Offset	$01000																															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	FEATURE REPORTING																															
			NIRQ															NCPU					VID									
Operation	R		R														R		R			R									
Reset	$0		$00F														$0		$01			$03									

NIRQ NUMBER OF IRQs. The number of the highest external IRQ source supported. The IPI, Timer, and Raven Detected Error interrupts are excluded from this count.

NCPU NUMBER OF CPUs. The number of the highest physical CPU supported. There are two CPUs supported by this design. CPU #0 and CPU #1.

VID VERSION ID. Version ID for this interrupt controller. This value reports what level of the specification is supported by this implementation. Version level of 03 is used for this release of the MPIC specification.

Global Configuration Register

Offset	$01020																															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	GLOBAL CONFIGURATION																															
	RESET		M	T																												
Operation	C	R	R/W	R/W	R				R				R				R				R											
Reset	0	0	0	$0	$00				$00				$00				$00				$00											

R RESET CONTROLLER. Writing a one to this bit forces the controller logic to be reset. This bit is cleared automatically when the reset sequence is complete. While this bit is set, the values of all other register are undefined.

M CASCADE MODE. Allows cascading of an external 8259 pair connected to the first interrupt source input pin (0). In the pass through mode, interrupt source 0 is passed directly through to the processor 0 INT pin. Raven MPIC is essentially disabled. In the mixed mode, 8259 interrupts are delivered using the priority and distribution mechanism of Raven MPIC. The Vector/Priority and Destination registers for interrupt source 0 are used to control the delivery mode for all 8259 generated interrupt sources.

M	MODE
0	Pass Through
1	Mixed

T Tie Mode. Writing a one to this register bit will cause a tie in external interrupt processing to, swap back and forth between processor 0 and 1. The first tie in external interrupt processing always goes to Processor 0 after a reset. When this register bit is set to 0, a tie in external interrupt processing will always go to processor 0 (Mode used on Version $02 of MPIC).

T	MODE
0	Processor 0 always selected
1	Swap between Processors

Vendor Identification Register

Offset	$01080			
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0			
Name	VENDOR IDENTIFICATION			
		STP		
Operation	R	R	R	R
Reset	$00	$02	$00	$00

There are two fields in the Vendor Identification Register which are not defined for the Raven MPIC implementation but are defined in the MPIC specification. They are the vendor identification and device ID fields.

STP STEPPING. The stepping or silicon revision number is initially 0.

Processor Init Register

Offset	$01090					
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0					
Name	PROCESSOR INIT					
					P1	P0
Operation	R	R	R	R	R/W	R/W
Reset	$00	$00	$00	$00	0	0

P1 PROCESSOR 1. Writing a 1 to P1 will assert the Soft Reset input of processor 1. Writing a 0 to it will negate the SRESET signal.

P0 PROCESSOR 0. Writing a 1 to P0 will assert the Soft Reset input of processor 0. Writing a 0 to it will negate the SRESET signal.

The Soft Reset input to the 604 is negative edge-sensitive.

IPI Vector/Priority Registers

Offset	IPI 0 - $010A0 IPI 1 - $010B0 IPI 2 - $010C0 IPI 3 - $010D0																															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	IPI VECTOR/PRIORITY																															
	MASK	ACT											PRIOR																			VECTOR
Operation	R/W	R	R										R/W										R									R/W
Reset	1	0	$000										$0										$00									$00

MASK MASK. Setting this bit disables any further interrupts from this source. If the mask bit is cleared while the bit associated with this interrupt is set in the IPR, the interrupt request will be generated.

ACT ACTIVITY. The activity bit indicates that an interrupt has been requested or that it is in-service. The ACT bit is set to a one when its associated bit in the Interrupt Pending Register or In-Service Register is set.

PRIOR Interrupt priority 0 is the lowest and 15 is the highest. Note that a priority level of 0 will not enable interrupts.

VECTOR This vector is returned when the Interrupt Acknowledge register is examined during a request for the interrupt associated with this vector.

Spurious Vector Register

Offset	$010E0																															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name																							VECTOR									
Operation	R								R								R							R/W								
Reset	$00								$00								$00								$FF							

VECTOR This vector is returned when the Interrupt Acknowledge register is read during a spurious vector fetch.

Timer Frequency Register

Offset	$010F0																															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	TIMER FREQUENCY																															
Operation	R/W																															
Reset	$00000000																															

This register is used to report the frequency (in Hz) of the clock source for the global timers. Following reset, this register contains zero. system initialization code must initialize this register to one-eighth the MPIC clock frequency. For the Raven implementation of MPIC, a typical value would be $7de290 which is 66/8 MHz or 8.25 MHz.

Timer Current Count Registers

Offset	Timer 0 - $01100 Timer 1 - $01140 Timer 2 - $01180 Timer 3 - $011C0																															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	TIMER CURRENT COUNT																															
	T	CC																														
Operation	R	R																														
Reset	0	$00000000																														

T TOGGLE. This bit toggles when ever the current count decrements to zero.

CC CURRENT COUNT. The current count field decrements while the Count Inhibit bit is the Base Count Register is zero. When the timer counts down to zero, the Current Count register is reloaded from the Base Count register and the timer's interrupt becomes pending in the MPIC processing.

Timer Basecount Registers

Offset	Timer 0 - $01110 Timer 1 - $01150 Timer 2 - $01190 Timer 3 - $011D0																															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	TIMER BASECOUNT																															
	CI	BC																														
Operation	R/W	R/W																														
Reset	1	$00000000																														

CI COUNT INHIBIT. Setting this bit to one inhibits counting for this timer. Setting this bit to zero allows counting to proceed.

BC BASE COUNT. This field contains the 31 bit count for this timer. When a value is written into this register and the CI bit transitions from a 1 to a 0, it is copied into the corresponding Current Count register and the toggle bit in the Current Count register is cleared. When the timer counts down to zero, the Current Count register is reloaded from the Base Count register and the interrupt becomes pending in MPIC processing.

Timer Vector/Priority Registers

Offset	Timer 0 - $01120 Timer 1 - $01160 Timer 2 - $011A0 Timer 3 - $011E0																															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	TIMER VECTOR/PRIORITY																															
	MASK	ACT											PRIOR								VECTOR											
Operation	R/W	R	R										R/W				R				R/W											
Reset	1	0	$000										$0				$00				$00											

MASK MASK. Setting this bit disables any further interrupts from this source. If the mask bit is cleared while the bit associated with this interrupt is set in the IPR, the interrupt request will be generated.

ACT ACTIVITY. The activity bit indicates that an interrupt has been requested or that it is in-service. The ACT bit is set to a one when its associated bit in the Interrupt Pending Register or In-Service Register is set.

PRIOR Interrupt priority 0 is the lowest and 15 is the highest. Note that a priority level of 0 will not enable interrupts.

VECTOR This vector is returned when the Interrupt Acknowledge register is examined upon acknowledgement of the interrupt associated with this vector.

Timer Destination Registers

Offset	Timer 0 - $01130 Timer 1 - $01170 Timer 2 - $011B0 Timer 3 - $011F0																													
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	TIMER DESTINATION																															
																														P1	P0
Operation	R								R				R				R						R							R/W	R/W
Reset	$00								$00				$00				$00						$00							0	0

This register indicates the destinations for this timer's interrupts. Timer interrupts, operate in the Directed delivery interrupt mode. This register may specify multiple destinations (multicast delivery).

P1 PROCESSOR 1. The interrupt is directed to processor 1.

P0 PROCESSOR 0. The interrupt is directed to processor 0.

External Source Vector/Priority Registers

Offset	Int Src 0 - $10000 Int Src 2 -> Int Src15 - $10020 -> $101E0																															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	EXTERNAL SOURCE VECTOR/PRIORITY																															
	MASK	ACT							POL	SENSE			PRIOR				VECTOR															
Operation	R/W	R	R						R/W	R/W	R	R	R/W	R			R/W															
Reset	1	0	$000						0	0	0	0	$0	$00			$00															

MASK MASK. Setting this bit disables any further interrupts from this source. If the mask bit is cleared while the bit associated with this interrupt is set in the IPR, the interrupt request will be generated.

ACT ACTIVITY. The activity bit indicates that an interrupt has been requested or that it is in-service. The ACT bit is set to a one when its associated bit in the Interrupt Pending Register or In-Service Register is set.

POL POLARITY. This bit sets the polarity for external interrupts. Setting this bit to a zero enables active low or negative edge. Setting this bit to a one enables active high or positive edge. Only External Interrupt Source 0 uses this bit in this register.

SENSE SENSE. This bit sets the sense for external interrupts. Setting this bit to a zero enables edge sensitive interrupts. Setting this bit to a one enables level sensitive interrupts. For external interrupt sources 1 through 15, setting this bit to a zero enables positive edge triggered interrupts. Setting this bit to a one enables active low level triggered interrupts.

PRIOR Interrupt priority 0 is the lowest and 15 is the highest. Note that a priority level of 0 will not enable interrupts.

VECTOR This vector is returned when the Interrupt Acknowledge register is examined upon acknowledgement of the interrupt associated with this vector.

External Source Destination Registers

Offset	Int Src 0 - $10010 Int Src 2 -> Int Src 15 - $10030 -> $101F0																													
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	EXTERNAL SOURCE DESTINATION																															
																														P1	P0
Operation	R								R				R				R				R									R/W	R/W
Reset	$00								$00				$00				$00				$00									0	0

This register indicates the possible destinations for the external interrupt sources. These interrupts operate in the Distributed interrupt delivery mode.

P1 PROCESSOR 1. The interrupt is pointed to processor 1.

P0 PROCESSOR 0. The interrupt is pointed to processor 0.

Raven-Detected Errors Vector/Priority Register

Offset	$10200																															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	RAVEN DETECTED ERRORS VECTOR/PRIORITY																															
	MASK	ACT							SENSE				PRIOR								VECTOR											
Operation	R/W	R	R						R	R	R	R	R/W	R							R/W											
Reset	1	0	$000						0	1	0	0	$0	$00							$00											

MASK MASK. Setting this bit disables any further interrupts from this source. If the mask bit is cleared while the bit associated with this interrupt is set in the IPR, the interrupt request will be generated.

ACT ACTIVITY. The activity bit indicates that an interrupt has been requested or that it is in-service. The ACT bit is set to a one when its associated bit in the Interrupt Pending Register or In-Service Register is set.

SENSE SENSE. This bit sets the sense for the internal Raven Detected Errors interrupt. This bit is hardwired to 1 to enable active low level sensitive interrupts.

PRIOR Interrupt priority 0 is the lowest and 15 is the highest. Note that a priority level of 0 will not enable interrupts.

VECTOR This vector is returned when the Interrupt Acknowledge register is examined upon acknowledgedment of the interrupt associated with this vector.

Raven-Detected Errors Destination Register

Offset	$10210																															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	RAVEN DETECTED ERROR DESTINATION																														P1	P0
Operation	R								R				R				R														R/W	R/W
Reset	$00								$00				$00				$00														0	0

This register indicates the possible destinations for the Raven detected error interrupt source. These interrupts operate in the Distributed interrupt delivery mode.

P1 PROCESSOR 1. The interrupt is pointed to processor 1.

P0 PROCESSOR 0. The interrupt is pointed to processor 0.

Interprocessor Interrupt Dispatch Registers

Offset	Processor 0 $20040, $20050, $20060, $20070 Processor 1 $21040, $21050,$21060, $21070																															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	IPI DISPATCH																														P1	P0
Operation	R								R				R				R														R/W	R/W
Reset	$00								$00				$00				$00														0	0

There are four Interprocessor Interrupt Dispatch Registers. Writing to an IPI Dispatch Register with the P0 and/or P1 bit set causes an interprocessor interrupt request to be sent to one or more processors.

Note Each IPI Dispatch Register has two addresses. These registers are considered to be per processor registers and there is one address per processor. Reading these registers returns zeros.

P1 PROCESSOR 1. The interrupt is directed to processor 1.

P0 PROCESSOR 0. The interrupt is directed to processor 0.

Interrupt Task Priority Registers

Offset	Processor 0 $20080 Processor 1 $21080																															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	INTERRUPT TASK PRIORITY																															
																															TP	
Operation	R								R								R					R								R/W	
Reset	$00								$00								$00						$0								$F	

There is one Task Priority Register per processor. Priority levels from 0 (lowest) to 15 (highest) are supported. Setting the Task Priority Register to 15 masks all interrupts to this processor. Hardware will set the task register to $F when it is reset or when the Init bit associated with this processor is written to a one.

Interrupt Acknowledge Registers

Offset	Processor 0 $200A0 Processor 1 $210A0																															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name																									VECTOR							
Operation	R								R								R							R								
Reset	$00								$00								$00								$FF							

On PowerPC-based systems, Interrupt Acknowledge is implemented as a read request to a memory-mapped Interrupt Acknowledge register. Reading the Interrupt Acknowledge register returns the interrupt vector corresponding to the highest priority pending interrupt. Reading this register also has the following side effects.

- The associated bit in the Interrupt Pending Register is cleared.
- Reading this register will update the In-Service register.

Reading this register without a pending interrupt will return a value of $FF hex.

End-of-Interrupt Registers

Offset	Processor 0 $200B0 Processor 1 $210B0																															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name																													EOI			
Operation	R								R								R							R					W			
Reset	$00								$00								$00								$0				$0			

EOI END OF INTERRUPT. There is one EOI register per processor. EOI Code values other than 0 are currently undefined. Data values written to this register are ignored; zero is assumed. Writing to this register signals the end of processing for the highest priority interrupt currently in service by the associated processor. The write operation will update the In-Service register by retiring the highest priority interrupt. Reading this register returns zeros.

Programming Notes

External Interrupt Service

The following summarizes how an external interrupt is serviced:

1. An external interrupt occurs.

2. The processor state is saved in the machine status save/restore registers. A new value is loaded into the Machine State Register (MSR). The External Interrupt Enable bit in the new MSR (MSRee) is set to zero. Control is transferred to the O/S external interrupt handler.

3. The external interrupt handler calculates the address of the Interrupt Acknowledge register for this processor (MPIC Base Address + 0x200A00 + (processor ID shifted left 12 bits)).

4. The external interrupt handler issues an Interrupt Acknowledge request to read the interrupt vector from the MPIC. If the interrupt vector indicates the interrupt source is the 8259, the interrupt handler issues a second Interrupt Acknowledge request to read the interrupt vector from the 8259. The Raven MPIC does not interact with the vector fetch from the 8259.

5. The interrupt handler saves the processor state and other interrupt-specific information in system memory and re-enables for external interrupts (the MSRee bit is set to 1). Raven MPIC blocks interrupts from sources with equal or lower priority until an End-of-Interrupt is received for that interrupt source. Interrupts from higher priority interrupt sources continue to be enabled. If the interrupt source was the 8259, the interrupt handler issues an EOI request to the MPIC. This resets the In-Service bit for the 8259 with in the Raven MPIC and allows it to recognize higher priority interrupt requests, if any, from the 8259. If none of the nested interrupt modes of the 8259 are enabled, the interrupt handler issues an EOI request to the 8259.

   a. The device driver interrupt service routine associated with this interrupt vector is invoked.

   b. If the interrupt source was not the 8259, the interrupt handler issues an EOI request for this interrupt vector to the MPIC. If the interrupt source was the 8259 and any of the nested interrupt modes of the 8259 are enabled, the interrupt handler issues an EOI request to the 8259.

   Normally, interrupts from ISA devices are connected to the 8259 interrupt controller. ISA devices typically rely on the 8259 Interrupt Acknowledge to flush buffers between the ISA device and system

memory. If interrupts from ISA devices are directly connected to the Raven MPIC (bypassing the 8259), the device driver interrupt service routine must read status from the ISA device to ensure buffers between the device and system memory are flushed.

Reset State

After a power-on reset the Raven MPIC state is:

- Current task priority for all CPUs set to 15.
- All interrupt source priorities set to zero.
- All interrupt source mask bits set to a one.
- All interrupt source activity bits cleared.
- Processor Init Register is cleared.
- All counters stopped and interrupts disabled.
- Controller mode set to 8259 pass-through.

Operation

Interprocessor Interrupts

Four interprocessor interrupt (IPI) channels are provided for use by all processors. During system initialization the IPI vector/priority registers for each channel should be programmed to set the priority and vector returned for each IPI event. During system operation a processor may generate an IPI by writing a destination mask to one of the IPI dispatch registers.

Note that each IPI dispatch register is shared by both processors. Each IPI dispatch register has two addresses but they are shared by both processors. That is, there is a total of four IPI dispatch registers in the Raven MPIC.

The IPI mechanism may be used for self interrupts by programming the dispatch register with the bit mask for the originating processor.

Dynamically Changing I/O Interrupt Configuration

The interrupt controller provides a mechanism for safely changing the vector, priority, or destination of I/O interrupt sources. This is provided to support systems which allow dynamic configuration of I/O devices. In order to change the vector, priority, or destination of an active interrupt source, the following sequence should be performed:

1. Mask the source using the MASK bit in the vector/priority register.
2. Wait for the activity bit (ACT) for that source to be cleared.
3. Make the desired changes.
4. Unmask the source.

This sequence ensures that the vector, priority, destination, and mask information remain valid until all processing of pending interrupts is complete.

EOI Register

Each processor has a private EOI register which is used to signal the end of processing for a particular interrupt event. If multiple nested interrupts are in service, the EOI command terminates the interrupt service of the highest priority source. Once an interrupt is acknowledged, only sources of higher priority will be allowed to interrupt the processor until the EOI command is received. This register should always be written with a value of zero which is the nonspecific EOI command.

Interrupt Acknowledge Register

Upon receipt of an interrupt signal, the processor may read this register to retrieve the vector of the interrupt source which caused the interrupt.

8259 Mode

The 8259 mode bits control the use of an external 8259 pair for PC-AT compatibility. Following a reset, this mode is set for pass-through, which essentially disables the advanced controller and passes an 8259 input on

external interrupt source 0 directly through to processor zero. During interrupt controller initialization this channel should be programmed for mixed mode in order to take advantage of the interrupt delivery modes.

Current Task Priority Level

Each processor has a separate Current Task Priority Level register. The system software uses this register to indicate the relative priority of the task running on the corresponding processor. The interrupt controller will not deliver an interrupt to a processor unless it has a priority level which is greater than the current task priority level of that processor. This value is also used in determining the destination for interrupts which are delivered using the distributed deliver mode.

Architectural Notes

The hardware and software overhead required to update the task priority register synchronously with instruction execution may far outweigh the anticipated benefits of the task priority register. To minimize this overhead, the interrupt controller architecture should allow the task priority register to be updated asynchronously with respect to instruction execution. Lower priority interrupts may continue to occur for an indeterminate number of cycles after the processor has updated the task priority register. If this is not acceptable, the interrupt controller architecture should recommend that, if the task priority register is not implemented with the processor, the task priority register should be updated only when the processor enter or exits an idle state.

Only when the task priority register is integrated within the processor, (such that it can be accessed as quickly as the MSRee bit, for example), should the architecture require the task priority register to be updated synchronously with instruction execution.

Falcon ECC Memory Controller Chip Set

3

Introduction

The Falcon DRAM controller ASIC is designed for the PowerPC families of boards. It is used in sets of two to provide the interface between the PowerPC 60*x* bus (also called MPC60*x* bus or MPC bus) and a 144-bit ECC-DRAM memory system. It also provides an interface to ROM/Flash.

Overview

This chapter provides a functional description and programming model for the Falcon. Most of the information for using the device in a system, programming it in a system, and testing it is contained here.

Bit Ordering Convention

All Falcon bused signals are named using big-endian bit ordering (bit 0 is the most significant bit).

Features

- DRAM Interface
  - Double-bit error detect/Single-bit error correct on 72-bit basis.
  - Up to four blocks.
  - Programmable base address for each block.
  - Two-way interleave factor.
  - Built-in Refresh/Scrub.
- Error Notification for DRAM
  - Software programmable Interrupt on Single/Double-Bit Error.
  - Error address and Syndrome Log Registers for Error Logging.

  - Does not provide TEA_ on Double-Bit Error. (Chip has no TEA_ pin.)

- ROM/Flash Interface
  - Two blocks with each block being 16 bits wide (8 bits per Falcon), or 64 bits wide (32 bits per Falcon).
  - Software programmable access time for each block.

Block Diagrams

Figure 3-1 depicts a Falcon pair as it would be connected in a system. Figure 3-2 shows the Falcon's internal data paths. Figure 3-3 shows the overall DRAM connections.

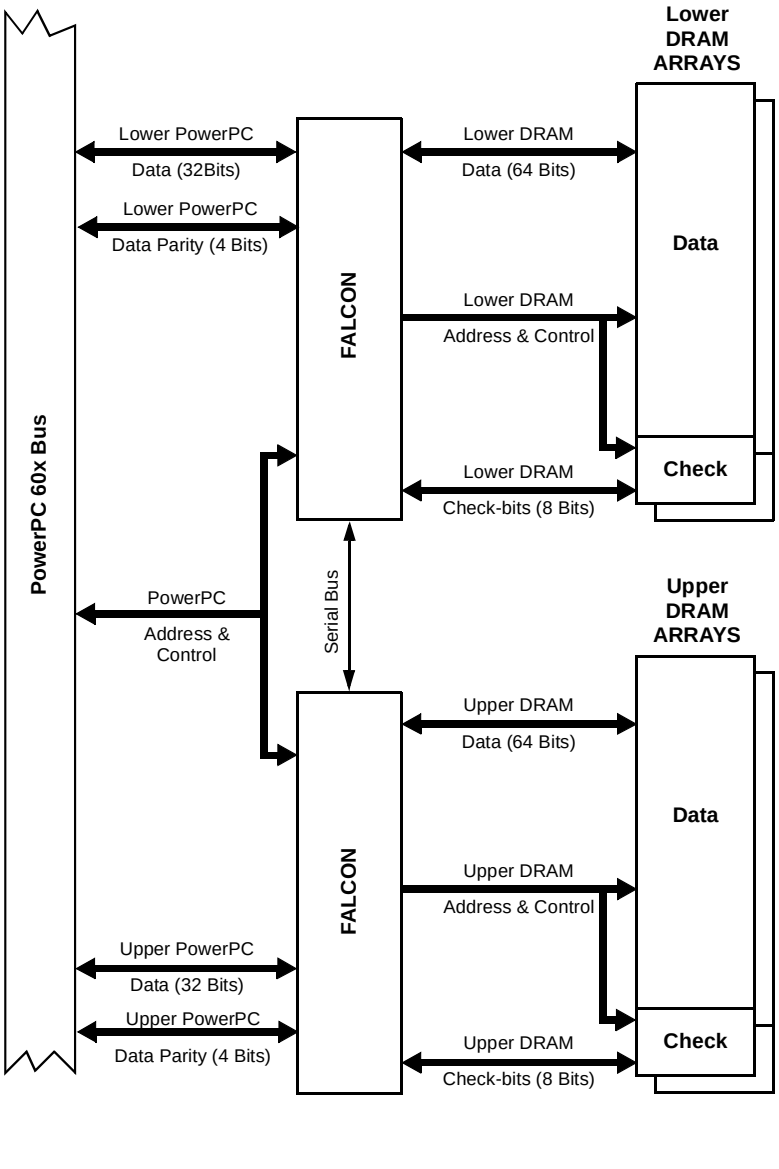


Figure 3-1. Falcon Pair Used with DRAM in a System

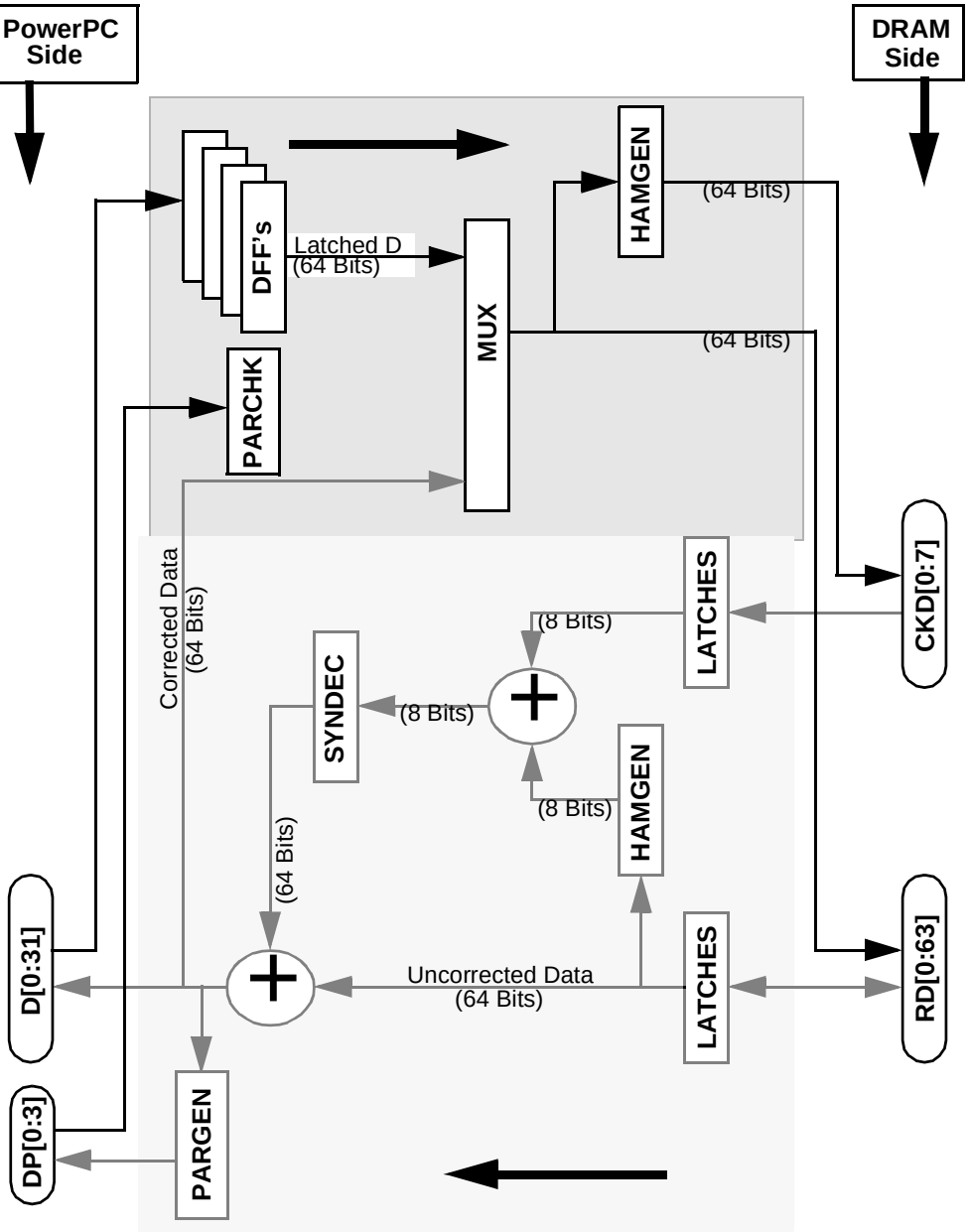


Figure 3-2. Falcon Internal Data Paths (Simplified)

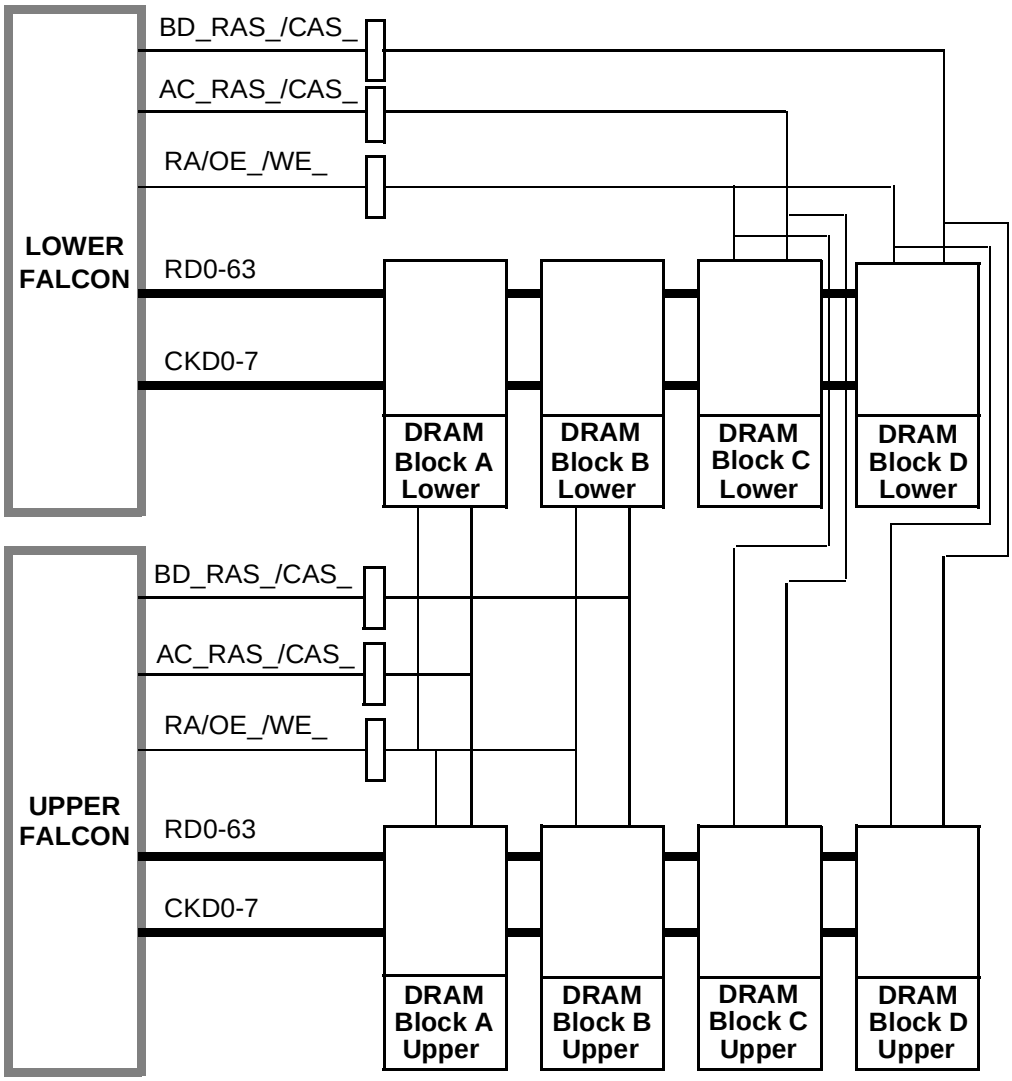


Figure 3-3. Overall DRAM Connections

Functional Description

The following sections describe the logical function of the ASIC. The Falcon is designed to be used as a set of two chips. A pair of Falcons works with *x1* or wider DRAM memory devices to form a memory system for the PowerPC 60*x* bus. A pair of Falcons that is connected to implement a memory control function is referred to in this document as a *Falcon pair*.

Performance

Four-beat Reads/Writes

The Falcon pair is specifically designed to provide maximum performance for cache line (four-beat) cycles to and from the PowerPC 60*x* bus at 66 MHz. This is done by providing a two-way interleave between the 64-bit PowerPC 60*x* data bus and the 128-bit (144 with check-bits) DRAM bus. When a PowerPC 60*x* bus master begins a quad-aligned, four-beat read to DRAM, the Falcon pair accesses the full 144-bit width of DRAM at once so that when the DRAM access time is reached, not only is the first 64-bit double-word of data ready to be transferred to the PowerPC 60*x* bus master, but so is the next. While the Falcon pair is presenting the first two double-words to the PowerPC 60*x* bus, it cycles CAS without cycling RAS to obtain the next two double-words. The Falcon pair transfers the next two double-words to the PowerPC 60*x* bus after 0 or more idle clocks.

The Falcon pair also takes advantage of the fact that PowerPC 60*x* processors can do address pipelining. Many times while a data cycle is finishing, the PowerPC 60*x* processor begins a new address cycle. The Falcon pair can begin the next DRAM access earlier when this happens, thus shortening the access time. Further savings come when the new address cycle is to an address close enough to the previous one that it falls within the same row in the DRAM array. When this happens, the Falcon pair can transfer the data for the next cycle by cycling CAS without cycling RAS.

Single-beat Reads/Writes

Single-beat cycles to and from the PowerPC 60*x* bus do not achieve data rates as high as do four-beat cycles. The Falcon pair does take advantage of the PowerPC 60*x* address pipelining as much as possible for single-beat accesses.

Single-beat writes are the slowest kind of accesses because they require that the Falcon pair perform a read cycle then a write cycle to the DRAM in order to complete. Fortunately, in most 60*x* systems, single-beat accesses can be held to a minimum especially with data cache and copyback modes in place.

DRAM Speeds

The Falcon pair can be configured for 3 different speeds of DRAM: 50ns, 60ns, and 70ns. When the Falcon pair is configured for 50ns DRAMs, it assumes that the devices are Extended Data Out (EDO) parts. When the Falcon pair is configured for 70ns DRAMs it assumes that the devices are fast page mode parts. When the pair is configured for 60ns DRAMs, it allows the devices to be either fast page or EDO parts. Performance summaries using the different devices are shown in Table 3-1, Table 3-2, and Table 3-3.

Table 3-1. PowerPC 60x Bus to DRAM Access Timing when Configured for 70ns Fast Page Devices

ACCESS TYPE	CLOCK PERIODS REQUIRED FOR:				Total Clocks
	1st Beat	2nd Beat	3rd Beat	4th Beat	
4-Beat Read after Idle (Quad-word aligned)	10	1	3	1	15
4-Beat Read after Idle (Quad-word misaligned)	10	4	1	1	16
4-Beat Read after 4-Beat Read (Quad-word aligned)	9/3 [1]	1	3	1	14/8
4-Beat Read after 4-Beat Read (misaligned)	7/2 [1]	4	1	1	13/8
4-Beat Write after Idle	4	1	1	1	7

Table 3-1. PowerPC 60x Bus to DRAM Access Timing when Configured for 70ns Fast Page Devices (Continued)

ACCESS TYPE	CLOCK PERIODS REQUIRED FOR:				Total Clocks
	1st Beat	2nd Beat	3rd Beat	4th Beat	
4-Beat Write after 4-Beat Write (Quad-word aligned)	10/6 [1]	1	1	1	13/9
1-Beat Read after Idle	10	-	-	-	10
1-Beat Read after 1-Beat Read	11/7 [1]	-	-	-	11/7
1-Beat Write after Idle	4	-	-	-	4
1-Beat Write after 1-Beat Write	15/11 [1]	-	-	-	15/11

Notes

1. These numbers assume that the PowerPC 60*x* bus master is doing address pipelining with TS_ occurring at the minimum time after AACK_ is asserted. Also the two numbers shown in the 1st beat column are for page miss/page hit.
2. In some cases, the numbers shown are averages and specific instances may be longer or shorter..

Table 3-2. PowerPC 60x Bus to DRAM Access Timing when Configured for 60ns Fast Page Devices

ACCESS TYPE	CLOCK PERIODS REQUIRED FOR:				Total Clocks
	1st Beat	2nd Beat	3rd Beat	4th Beat	
4-Beat Read after Idle (Quad-word aligned)	9	1	2	1	13
4-Beat Read after Idle (Quad-word misaligned)	9	3	1	1	14
4-Beat Read after 4-Beat Read (Quad-word aligned)	7/3 [1]	1	2	1	11/7
4-Beat Read after 4-Beat Read (misaligned)	6/2 [1]	3	1	1	11/7

Table 3-2. PowerPC 60x Bus to DRAM Access Timing when Configured for 60ns Fast Page Devices (Continued)

ACCESS TYPE	CLOCK PERIODS REQUIRED FOR:				Total Clocks
	1st Beat	2nd Beat	3rd Beat	4th Beat	
4-Beat Write after Idle	4	1	1	1	7
4-Beat Write after 4-Beat Write (Quad-word aligned)	7/3 [1]	1	1	1	10/6
1-Beat Read after Idle	9	-	-	-	9
1-Beat Read after 1-Beat Read	9/6 [1]	-	-	-	9/6
1-Beat Write after Idle	4	-	-	-	4
1-Beat Write after 1-Beat Write	13/10 [1]	-	-	-	13/10

Notes

1. These numbers assume that the PowerPC 60*x* bus master is doing address pipelining with TS_ occurring at the minimum time after AACK_ is asserted. Also the two numbers shown in the 1st beat column are for page miss/page hit.

2. In some cases, the numbers shown are averages and specific instances may be longer or shorter.

Table 3-3. PowerPC 60x Bus to DRAM Access Timing when Configured for 50ns EDO Devices

ACCESS TYPE	CLOCK PERIODS REQUIRED FOR:				Total Clocks
	1st Beat	2nd Beat	3rd Beat	4th Beat	
4-Beat Read after Idle (Quad-word aligned)	8	1	1	1	11
4-Beat Read after Idle (Quad-word misaligned)	8	2	1	1	12
4-Beat Read after 4-Beat Read (Quad-word aligned)	5/2 [1]	1	1	1	8/5
4-Beat Read after 4-Beat Read (misaligned)	4/2 [1]	2	1	1	8/6
4-Beat Write after Idle	4	1	1	1	7
4-Beat Write after 4-Beat Write (Quad-word aligned)	4/3 [1]	1	1	1	7/6
1-Beat Read after Idle	8	-	-	-	8
1-Beat Read after 1-Beat Read	7/5 [1]	-	-	-	7/5
1-Beat Write after Idle	4	-	-	-	4
1-Beat Write after 1-Beat Write	9/7 [1]	-	-	-	9/7

Notes

1. These numbers assume that the PowerPC 60*x* bus master is doing address pipelining with TS_ occurring at the minimum time after AACK_ is asserted. Also the two numbers shown in 1st beat column are for page miss/page hit.

2. In some cases, the numbers shown are averages and specific instances may be longer or shorter.

ROM/Flash Speeds

The Falcon pair provides the interface for two blocks of ROM/Flash. Access times to ROM/Flash are programmable for each block. Access times are also affected by block width. The access times for ROM/Flash are shown in Table 3-4, Table 3-5, Table 3-6, and Table 3-7.

Table 3-4. PowerPC 60x Bus to ROM/Flash Access Timing when configured for 180ns Devices

ACCESS TYPE	CLOCK PERIODS REQUIRED FOR:								Total Clocks	
	1st Beat		2nd Beat		3rd Beat		4th Beat			
	16 Bits	64 Bits	16 Bits	64 Bits	16 Bits	64 Bits	16 Bits	64 Bits	16 Bits	64 Bits
4-Beat Read	68	20	64	16	64	16	64	16	260	68
4-Beat Write	N/A								N/A	
1-Beat Read (1 byte)	20	20	-	-	-	-	-	-	20	20
1-Beat Read (2 to 8 bytes)	68	20	-	-	-	-	-	-	68	20
1-Beat Write	19	19	-	-	-	-	-	-	19	19

Table 3-5. PowerPC 60x Bus to ROM/Flash Access Timing when configured for 120ns Devices

ACCESS TYPE	CLOCK PERIODS REQUIRED FOR:								Total Clocks	
	1st Beat		2nd Beat		3rd Beat		4th Beat			
	16 Bits	64 Bits	16 Bits	64 Bits	16 Bits	64 Bits	16 Bits	64 Bits	16 Bits	64 Bits
4-Beat Read	52	16	48	12	48	12	48	12	196	52
4-Beat Write	N/A								N/A	
1-Beat Read (1 byte)	16	16	-	-	-	-	-	-	16	16
1-Beat Read (2 to 8 bytes)	52	16	-	-	-	-	-	-	52	16
1-Beat Write	19	19	-	-	-	-	-	-	19	19

Table 3-6. PowerPC 60x Bus to ROM/Flash Access Timing when configured for 75ns Devices

ACCESS TYPE	CLOCK PERIODS REQUIRED FOR:								Total Clocks	
	1st Beat		2nd Beat		3rd Beat		4th Beat			
	16 Bits	64 Bits	16 Bits	64 Bits	16 Bits	64 Bits	16 Bits	64 Bits	16 Bits	64 Bits
4-Beat Read	40	13	36	9	36	9	36	9	148	40
4-Beat Write	N/A								N/A	
1-Beat Read (1 byte)	13	13	-	-	-	-	-	-	13	13
1-Beat Read (2 to 8 bytes)	40	13	-	-	-	-	-	-	40	13
1-Beat Write	19	19	-	-	-	-	-	-	19	19

Table 3-7. PowerPC 60x Bus to ROM/Flash Access Timing when configured for 45ns Devices

ACCESS TYPE	CLOCK PERIODS REQUIRED FOR:								Total Clocks	
	1st Beat		2nd Beat		3rd Beat		4th Beat			
	16 Bits	64 Bits	16 Bits	64 Bits	16 Bits	64 Bits	16 Bits	64 Bits	16 Bits	64 Bits
4-Beat Read	32	11	28	7	28	7	28	7	116	32
4-Beat Write	N/A								N/A	
1-Beat Read (1 byte)	11	11	-	-	-	-	-	-	11	11
1-Beat Read (2 to 8 bytes)	32	11	-	-	-	-	-	-	32	11
1-Beat Write	19	19	-	-	-	-	-	-	19	19

PowerPC 60x Bus Interface

The Falcon pair has a PowerPC slave interface only. It has no PowerPC master interface. The slave interface is the mechanism for all accesses to DRAM, ROM/Flash, and internal and external register sets.

Responding to Address Transfers

When the Falcon pair detects an address transfer that it is to respond to, it asserts AACK_ immediately if there is no uncompleted PowerPC 60*x* bus data transfer in process. If there is one in process, then the Falcon pair waits and asserts AACK_ coincident with the uncompleted data transfer's last data beat if the Falcon pair is the slave for the previous data. If it is not, it holds off AACK_ until the CLOCK after the previous data transfer's last data beat.

Completing Data Transfers

If an address transfer to the Falcon pair will have an associated data transfer, the Falcon pair begins a read or write cycle to the accessed entity (DRAM/ROM/Flash/Internal and External Register) as soon as the entity is free. If the data transfer will be a read, the Falcon pair begins providing data to the PowerPC 60*x* bus as soon as the entity has data ready and the PowerPC 60*x* data bus is granted. If the data transfer will be a write, the Falcon pair begins latching data from the PowerPC data bus as soon as any previously latched data is no longer needed and the PowerPC 60*x* data bus is available.

Data Parity

The Falcon pair has 8 DP pins (4 per Falcon) for generating and checking 60*x* data bus parity. In addition, each Falcon has a DPERR_ pin to provide real-time data parity error notification for its half of the 60*x* data bus.

During read cycles that access the Falcon pair, the pair generates the correct value on DP0-DP7 so that each data byte lane along with its corresponding DP signal has odd parity. This can be changed on a lane basis to even parity by software bits that can force the generation of wrong (even) parity.

During write cycles to the Falcon pair, each Falcon in the pair checks each of its four 60*x* data byte lanes and the corresponding DP signal for odd parity. If any of the four lanes has even parity, that Falcon logs the error in the CSR and can generate a machine check if so enabled. In addition to

logging the error, that Falcon also pulses its DPERR_ signal true for the duration of one clock period, two clock periods after the TA_ during which the error data is captured.

While normal (default) operation is for the Falcon to check data parity only on writes to itself, it can also be programmed to check data parity on all reads or writes to any device on the 60*x* bus

Cache Coherency

The Falcon supports cache coherency to DRAM only. It does this by monitoring the ARTRY_ control signal on the PowerPC 60*x* bus and behaving appropriately when it is asserted. When ARTRY_ is asserted, if the access is a DRAM read, the Falcon does not source the data for that access. If the access is a DRAM write, the Falcon does not write the data for that access. Depending upon when the retry occurs, the Falcon may cycle the DRAM even though the data transfer does not happen.

Cache Coherency Restrictions

The PowerPC 60*x* GBL_ signal must not be asserted in the CSR areas.

L2 Cache Support

The Falcon pair provides support for a look-aside L2 cache by implementing a hold-off input, L2CLM_. On cycles that select the Falcon pair, the Falcon pair samples L2CLM_ on the second rising edge of CLOCK after the assertion of TS_. If L2CLM_ is high, the Falcon pair responds normally to the cycle. If it is low, the Falcon pair ignores the cycle.

ECC

The Falcon pair performs single-bit error correction and double-bit error detection for DRAM. (No checking is provided for ROM /Flash.) The 64-bit wide PowerPC 60*x* data bus is divided into upper (DH0-DH31) and lower (DL0-DL31) halves. Each half is routed through a Falcon which

multiplexes it with half of the DRAM data bus. Each Falcon connects to 64 DRAM data-bits and to 8 DRAM check-bits. The total DRAM array width is 144 bits (2*[64+8]).

Cycle Types

To support ECC, the Falcon pair always deals with DRAM using full width (144-bit) accesses. When the PowerPC 60*x* bus master requests any size read of DRAM, the Falcon pair reads 144 bits at least once. When the PowerPC 60*x* bus master requests a four-beat write to DRAM, the Falcon pair writes all 144 bits twice. When the PowerPC 60*x* bus master requests a single-beat write to DRAM, the Falcon pair performs a 144-bit wide read cycle to DRAM, merges in the appropriate PowerPC 60*x* bus write data, and writes 144 bits back to DRAM.

Error Reporting

The Falcon pair checks data from the DRAM during single- and four-beat reads, during single-beat writes, and during scrubs. Table 3-8 shows the actions it takes for different errors during these accesses.

Note The Falcon pair does not assert TEA_ on double-bit errors. In fact, the Falcon pair does not have a TEA_ signal pin and it assumes that the system does not implement TEA_. The Falcon

can, however, assert machine check (MCP_) on double-bit error.

Table 3-8. Error Reporting

Error Type	Single-Beat /Four-Beat Read	Single-Beat Write	Four-Beat Write	Scrub
Single-Bit Error	Terminate the PowerPC 60*x* bus cycle normally. Provide corrected data to the PowerPC 60*x* bus master. Assert INT_ if so enabled.	Terminate the PowerPC 60*x* bus cycle normally. Correct the data read from DRAM, merge with the write data, and write the corrected, merged data to DRAM. Assert INT_ if so enabled.	N/A [1]	This cycle is not seen on the PowerPC 60*x* bus. Write corrected data back to DRAM if so enabled. Assert INT_ if so enabled.
Double-Bit Error	Terminate the PowerPC 60*x* bus cycle normally. Provide miss-corrected, raw DRAM data to the PowerPC 60*x* bus master. Assert INT_ if so enabled. Assert MCP_ if so enabled.	Terminate the PowerPC 60*x* bus cycle normally. May or may not perform the write portion of the read-modify-write cycle to DRAM.[2] . Assert INT_ if so enabled. Assert MCP_ if so enabled.	N/A [1]	This cycle is not seen on the PowerPC 60*x* bus. May or may not perform the write portion of the read-modify-write cycle to DRAM.[2] . Assert INT_ if so enabled.
Triple- (or greater) Bit Error	Some of these errors are detected correctly and are treated the same as double-bit errors. The rest could show up as “no error” or “single-bit error”, both of which are incorrect.			

Notes

1. No opportunity for error since no read of DRAM occurs during a four-beat write.
2. The recommended connection for Falcon is for WE* to connect from upper Falcon to both upper and lower DRAM's of banks A and B and for WE* of lower Falcon to connect to both upper and lower DRAM's of banks C and D. With this configuration, the write portion of single-bit-writes and of scrubs *does* happen if the double-bit error is in the lower portion of banks A or B or if it is in the upper portion of banks C or D

Error Logging

ECC error logging is facilitated by the Falcon because of its internal latches. When an error (single- or double-bit) occurs in the DRAMs to which a Falcon is connected, it records the address and syndrome bits associated with the data in error. Each Falcon performs this logging function independently of the other. Once a Falcon has logged an error, it does not log any more until the **elog** control /status bit has been cleared by software unless the currently logged error is single-bit and a new, double-bit error is encountered. The logging of errors that occur during scrub can be enabled/disabled in software. Refer to the *Error Logger Register* section for more information.

ROM/Flash Interface

The Falcon pair provides the interface for two blocks of ROM/Flash. Each block provides addressing and control for up to 64MB.

Note No ECC error checking is provided for the ROM/Flash.

The ROM/Flash interface allows each block to be individually configured by jumpers and/or by software as follows:

1. Access for each block is controlled by three software programmable control register bits: an overall enable, a write enable, and a reset

vector enable. The overall enable controls normal read acesses. The write enable is used to program Flash devices. The reset vector enable controls whether the block is also enabled at $FFF00000 - $FFFFFFFF. The overall enable and write enable bits are always cleared at reset. The reset vector enable bit is cleared or set at reset depending on external jumper configuration. This allows the board designer to use external jumpers to enable/disable Block A/B ROM/Flash as the source of reset vectors.

2. The base address for each block is software programmable. At reset, Block A's base address is $FF000000 and Block B's base address is $FF400000.

   As noted above, in addition to appearing at the programmed base address, the first 1Mbyte of Block A/B also appears at $FFF00000-$FFFFFFFF if the reset vector enable bit is set.

3. The assumed size for each block is software programmable. It is initialized to its smallest setting at reset.

4. The access time for each block is software programmable

5. The assumed width for Block A/B is determined by an external jumper at reset time. It also is available as a status bit and cannot be changed by software.

   When the width status bit is cleared, the block's ROM /Flash is considered to be 16 bits wide, where each Falcon interfaces to 8 bits. In this mode, the following rules are enforced:

   a. only single-byte writes are allowed (all other sizes are ignored), and

   b. all reads are allowed (multiple accesses are performed to the ROM/Flash devices when the read is for greater than one byte).

   When the width status bit is set, the block's ROM/Flash is considered to be 64 bits wide, where each Falcon interfaces with 32 bits. In this mode, the following rules are enforced:

   c. only aligned, 4-byte writes should be attempted (all other sizes are ignored), and

d. all reads are allowed (multiple accesses to the ROM/Flash device are performed for burst reads).

More information about ROM/Flash is found in the section on the *Programming Model*.

In order to place code correctly in the ROM/Flash devices, address mapping information is required. Table 3-9 shows how PowerPC 60*x* addresses map to the ROM/Flash addresses when ROM/Flash is 16 bits wide (8 bits per Falcon). Table 3-10 shows how they map when Flash is 64 bits wide (32 bits per Falcon).

Table 3-9. PowerPC 60x to ROM/Flash Address Mapping when ROM/Flash is 16 Bits Wide (8 Bits per Falcon)

PowerPC 60x A0-A31	ROM/Flash A22-A0	ROM/Flash Selected
$XX000000	$000000	Upper
$XX000001	$000001	Upper
$XX000002	$000002	Upper
$XX000003	$000003	Upper
$XX000004	$000000	Lower
$XX000005	$000001	Lower
$XX000006	$000002	Lower
$XX000007	$000003	Lower
$XX000008	$000004	Upper
$XX000009	$000005	Upper
$XX00000A	$000006	Upper
$XX00000B	$000007	Upper
$XX00000C	$000004	Lower
$XX00000D	$000005	Lower
$XX00000E	$000006	Lower
$XX00000F	$000007	Lower
$XXFFFFF8	$7FFFFC	Upper
$XXFFFFF9	$7FFFFD	Upper
$XXFFFFFA	$7FFFFE	Upper

Table 3-9. PowerPC 60x to ROM/Flash Address Mapping when ROM/Flash is 16 Bits Wide (8 Bits per Falcon) (Continued)

PowerPC 60x A0-A31	ROM/Flash A22-A0	ROM/Flash Selected
$XXFFFFFB	$7FFFF	Upper
$XXFFFFFC	$7FFFC	Lower
$XXFFFFFD	$7FFFD	Lower
$XXFFFFFE	$7FFFE	Lower
$XXFFFFFF	$7FFFF	Lower

Table 3-10. PowerPC 60x to ROM/Flash Address Mapping when ROM/Flash is 64 Bits Wide (32 Bits per Falcon)

PowerPC 60x A0-A31	ROM/Flash A22-A0	ROM/Flash Selected
$X0000000	$000000	Upper
$X0000001	$000000	Upper
$X0000002	$000000	Upper
$X0000003	$000000	Upper
$X0000004	$000000	Lower
$X0000005	$000000	Lower
$X0000006	$000000	Lower
$X0000007	$000000	Lower
$X0000008	$000001	Upper
$X0000009	$000001	Upper
$X000000A	$000001	Upper
$X000000B	$000001	Upper
$X000000C	$000001	Lower
$X000000D	$000001	Lower
$X000000E	$000001	Lower
$X000000F	$000001	Lower
$X3FFFFF0	$7FFFE	Upper
$X3FFFFF1	$7FFFE	Upper

Table 3-10. PowerPC 60x to ROM/Flash Address Mapping when ROM/Flash is 64 Bits Wide (32 Bits per Falcon) (Continued)

PowerPC 60x A0-A31	ROM/Flash A22-A0	ROM/Flash Selected
$X3FFFFF2	$7FFFFE	Upper
$X3FFFFF3	$7FFFFE	Upper
$X3FFFFF4	$7FFFFE	Lower
$X3FFFFF5	$7FFFFE	Lower
$X3FFFFF6	$7FFFFE	Lower
$X3FFFFF7	$7FFFFE	Lower
$X3FFFFF8	$7FFFFF	Upper
$X3FFFFF9	$7FFFFF	Upper
$X3FFFFFA	$7FFFFF	Upper
$X3FFFFFB	$7FFFFF	Upper
$X3FFFFFC	$7FFFFF	Lower
$X3FFFFFD	$7FFFFF	Lower
$X3FFFFFE	$7FFFFF	Lower
$X3FFFFFF	$7FFFFF	Lower

Refresh/Scrub

Refresh/Scrub is done differently based on which DRAM blocks are populated: (A and/or B) but not (C and D), or (A and/or B) and (C and/or D).

Blocks A and/or B Present, Blocks C and D Not Present

The Falcon pair performs refresh by doing a burst of four RAS_ cycles approximately once every 60µs. This increases to once every 30µs when certain DRAM devices are used. (Controlled by the ram_fref bit in the status registers.) RAS_ is asserted to both of Blocks A and B during each of the 4 cycles. Along with RAS_, the Falcon pair also asserts CAS_ with (OE_ then WE_) to one of the blocks during one of the four cycles. This forms a read-modify-write which is a scrub cycle to that location.

After each of the 4 cycles, the DRAM row address increments by one. When it reaches all 1's, it rolls over and starts over at 0. Each time the row address rolls over, the block that is scrubbed toggles between A and B. Every second time that the row address rolls over, which of the 4 cycles that is a scrub changes from 1st to 2nd, from 2nd to 3rd, from 3rd to 4th, or from 4th to 1st. Every eighth time that the row address rolls over, the column address increments by one. When the column address reaches all 1's, it rolls over and starts over at 0. Each time the column address rolls over, the SC1, SC0 bits in the scrub/refresh register increment by one.

Blocks A and/or B Present, Blocks C and/or D Present

The Falcon pair performs refresh by doing a burst of four RAS_ cycles approximately once every 30µs. This increases to once every 15µs when certain DRAM devices are used. (Controlled by the ram_fref bit in the status registers.) RAS_ is asserted to blocks A and B during the first cycle, to blocks C and D during the second cycle, back to blocks A and B during the third cycle and to blocks C and D during the fourth cycle. Along with RAS, the Falcon pair also asserts CAS_ with (OE_ then WE_) to one of the blocks during one of the four cycles. This forms a read-modify-write which is a scrub cycle to that location.

After the second and fourth cycles, the DRAM row address increments by one. When it reaches all 1's, it rolls over and starts over at 0. Each time the row address rolls over, the block that is scrubbed toggles between A/C and B/D. Every second time the row address rolls over, which of the 4 cycles that is a scrub changes from 1st to 2nd, from 2nd to 3rd, from 3rd to 4th, or from 4th to 1st. Every eighth time that the row address rolls over, the column address increments by one. When the column address reaches all 1's, it rolls over and starts over at 0. Each time the column address rolls over, the SC1, SC0 bits in the scrub/refresh register increment by one.

An entire refresh of DRAM is achieved every time the row address rolls over, and an entire scrub of DRAM is achieved every time the column address rolls over.

During scrub cycles, if the SWEN bit is cleared, the Falcon pair *does not* perform the write portion of the read-modify write cycle. If the SWEN bit is set, the Falcon pair *does* perform the write unless it encounters a double-bit error during the read.

If so enabled, single- and double-bit scrub errors are logged, and the PowerPC 60*x* bus master is notified via interrupt.

Chip Defaults

Some jumper option kinds of parameters need to be configured by software in the Falcon pair. These parameters include DRAM and ROM/Flash attributes. In order to set up these parameters correctly, software needs some way of knowing about the devices that are being used with the Falcon pair. One way of providing this information is by using the power-up status registers in the Falcon pair. At power-up reset, each Falcon latches the level on its RD0-RD63 signal pins into its power-up status registers. Since the RD signal pins are high impedance during reset, their power-up reset level can be controlled by pullup/pulldown resistors. (They are pulled-up internally.)

External Register Set

Each chip in the Falcon pair has an external register chip select pin which enables it to talk to an external set of registers. This interface is like the ROM/Flash interface but with less flexibility. It is intended for the system designer to be able to implement general-purpose status/control signals with this external set. Refer to the *Programming Model* for a description of this register set.

CSR Accesses

An important part of the operation of a Falcon pair is that the value written to the internal control registers and SRAM in each of the two chips must be the same at all times. In order to facilitate this, writes to the pair itself are restricted to the upper Falcon only. When software writes to the upper Falcon, hardware in the two chips shifts this same value into the lower Falcon before the cycle completion is acknowledged. The shifting is done in holding registers such that the actual update of the control register happens on the same CLOCK cycle in both chips. Writes to the upper Falcon can be single-byte or 4-byte. Writes to the lower Falcon are ignored.

This duplicating of writes from upper to lower applies to the Falcon's internal registers only. No duplication is performed for writes to DRAM, ROM/Flash, or the External Register set.

Programming Model

CSR Architecture

The CSR (control and status register set) consists of the chip's internal register set and its external register set. The base address of the CSR is hard coded to the address $FEF80000 (or $FEF90000 if the SIO pin is low at reset).

Accesses to the CSR are mapped differently depending on whether they are reads or writes. For reads, CSR data read on the upper half of the data bus comes from the upper Falcon while CSR data read on the lower half of the data bus comes from the lower Falcon. See the figure below.

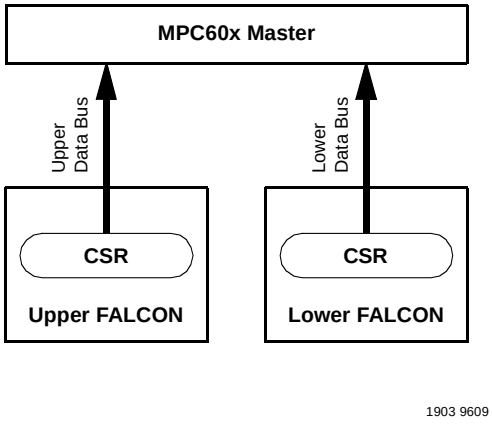


Figure 3-4. Data Path for Reads from the Falcon Internal CSRs

For writes, internal register data written on the upper half of the data bus goes to the upper Falcon and is automatically copied by hardware to the lower Falcon. Internal register data written on the lower half of the data bus does not go to either Falcon in the pair, but the access is terminated normally with TA_. See the figure below.

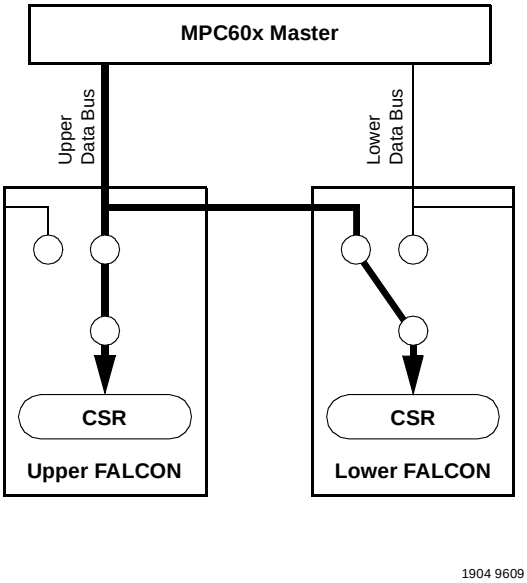


Figure 3-5. Data Path for Writes to the Falcon Internal CSRs

External register data that is written on the upper data bus goes through the upper Falcon, while data that is written on the lower data bus goes through the lower Falcon. Unlike the internal register set, *t*here is no automatic copying of upper data to lower data for the external register set.

CSR read accesses can have a size of 1, 2, 4, or 8 bytes with any alignment. CSR write accesses are restricted to a size of 1 or 4 bytes and they must be aligned. Figure 3-6, Figure 3-7, Figure 3-8, and Figure 3-9 show the memory maps for the different kinds of access.

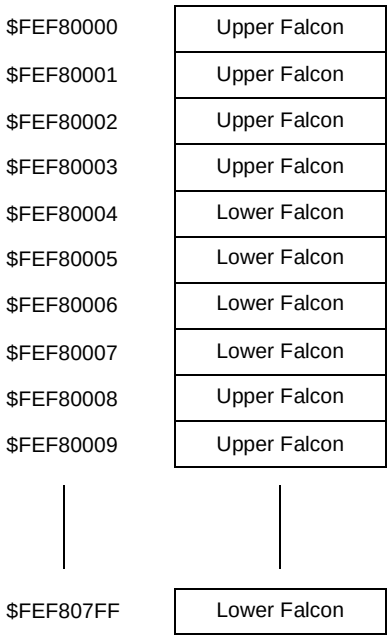


Figure 3-6. Memory Map for Byte Reads to the CSR

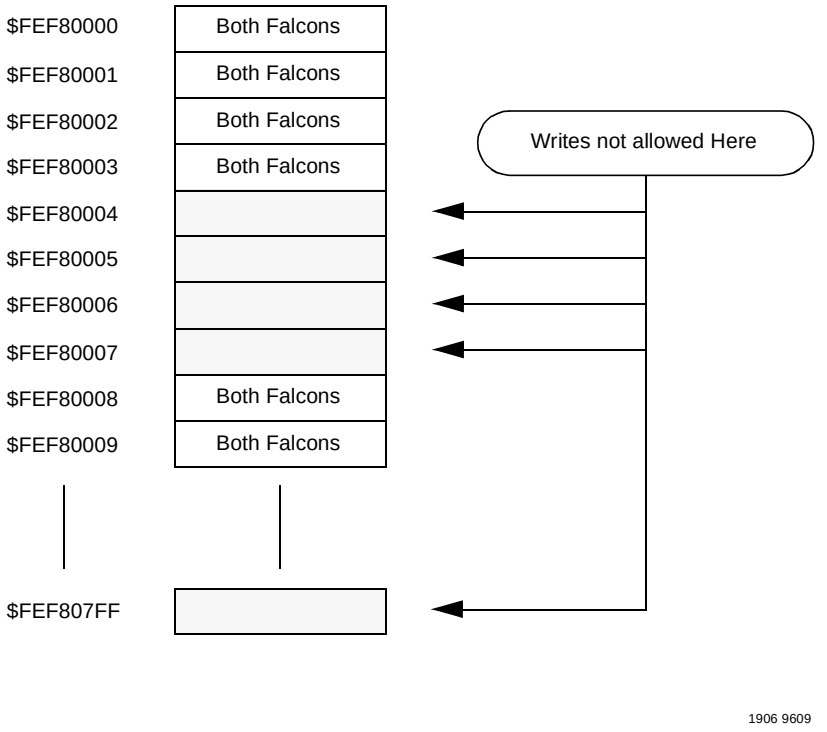


Figure 3-7. Memory Map for Byte Writes to the Internal Register Set

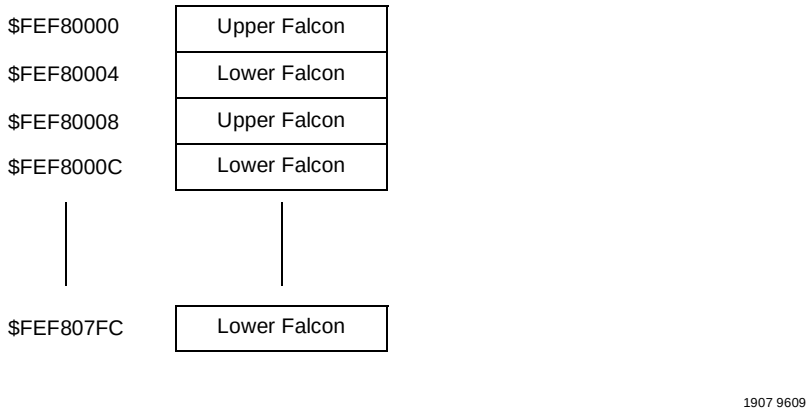


Figure 3-8. Memory Map for 4-Byte Reads to the CSR

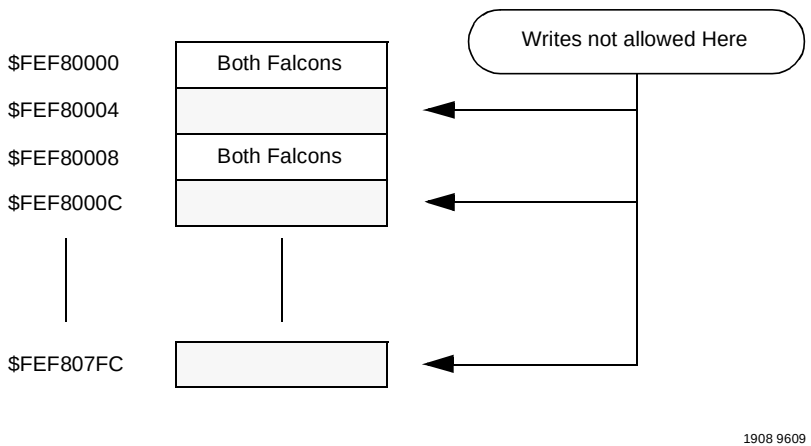


Figure 3-9. Memory Map for 4-Byte Writes to the Internal Register Set

Register Summary

Table 3-11 shows a summary of the CSR.

3

Note The table only shows addresses for accesses to the upper Falcon. To get the addresses for accesses to the lower Falcon, add 4 to the address shown. Since the only way to write to the lower Falcon's internal register set is to duplicate what is written to the upper Falcon, only the addresses shown in the table should be used for writes to them. Writes to the external register set are not duplicated from upper to lower, so writes to them can be via the upper or lower Falcon.

Table 3-11. Register Summary

BIT # ---->	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
FEF80000	VENDID																DEVID															
FEF80008									REVID														aonly_en	isa_hole				adis	ram_fref	ram_spd0	ram_spd1	chipu
FEF80010	ram_a_en					RAM A SIZ			ram_b_en					RAM B SIZ			ram_c_en					RAM C SIZ		ram_d_en					RAM D SIZ		
FEF80018	RAM A BASE								RAM B BASE								RAM C BASE								RAM D BASE							
FEF80020	CLK FREQUENCY																															por
FEF80028						refdis	rwcb	derc					scien	dpien	sien	mien																mcken
FEF80030	elog				escb	esen	embt	esbt	ERROR_SYNDROME									esblk0	esblk1				scof		SBE COUNT							
FEF80038	ERROR_ADDRESS																															
FEF80040	scb0	scb1						swen						rtest0	rtest1	rtest2																

Table 3-11. Register Summary (Continued)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FEF80048		ROW ADDRESS																														
FEF80050	rom a we	rom a en	rom_a_rv														ROM A BASE															
FEF80058	rom b we	rom b en	rom_b_rv														ROM B BASE															
FEF80060	rom_b_spd11	rom_b_spd0						rom_a_spd1	rom_a_spd0																							
FEF80068	GWDP								dpe_me	dpe_ckall							DPE_DP								DPE_TT					dpelog
FEF80070	DPE_A																															
FEF80078	DPE_D																															
FEF80100	CTR32																															
FEF80400	PR_STAT1																															
FEF80500	PR_STAT2																															
FEF88000 FEF8FFF8	EXTERNAL REGISTER SET																															
BIT # ----->	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Notes

1. All shaded bit fields are reserved and read as zeros.
2. All status bits are shown in italics.
3. All control bits are shown with underline.
4. All control-and-status bits are shown with italics and underline.

Detailed Register Bit Descriptions

The following sections describe the registers and their bits in detail. The possible operations for each bit in the register set are as follows:

- R The bit is a read only status bit.
- R/W The bit is readable and writable.
- R/C The bit is cleared by writing a one to itself.

The possible states of the bits after local and power-up reset are as defined below.

- P The bit is affected by power-up reset (PURESET_).
- L The bit is affected by local reset (HRESET_).
- X The bit is not affected by reset.
- V The effect of reset on the bit is variable.

Vendor/Device Register

Address	$FEF80000																													
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Name	VENDID																DEVID															
Operation	READ ONLY																READ ONLY															
Reset	X																X															

VENDID This read-only register contains the value $1057. It is the vendor number assigned to Motorola Inc.

DEVID This read-only register contains the value $4802. It is the device number for the Falcon.

Revision ID/General Control Register

Address	$FEF80008																															
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Name									REVID								0	0	0	0	0	0	aonly_en	isa_hole	0	0	0	adis	ram fref	ram spd0	ram spd1	chipu
Operation	READ ZERO								READ ONLY								R	R	R	R	R	R	R	R/W	R	R	R	R/W	R/W	R/W	R/W	R
Reset	X								$03								X	X	X	X	X	X	V P	0 PL	X	X	X	0 P	1 P	0 P	0 P	V P

REVID The REVID bits are hard-wired to indicate the revision level of the Falcon. The value for the first revision is $01, for the second is $02, for the third it is $03.

aonly_en Normally, the Falcon pair responds to address-only cycles only if they fall within the address range of one of its enabled map decoders. When the **aonly_en** bit is set, the Falcon pair also responds to address-only cycles that fall outside of the range of its enabled map decoders provided they are not acknowledged by some other slave within 8 clock periods. **aonly_en** is read-only and reflects the level that was on the CKD4 pin at power-up reset.

isa_hole When it is set, **isa_hole** disables any of the DRAM or ROM/Flash blocks from responding to PowerPC accesses in the range from $000A0000 to $000BFFFF. This has the effect of creating a hole in the DRAM memory map for accesses to ISA. When **isa_hole** is cleared, there is no hole created in the memory map.

adis When **adis** is clear, fast page mode operation is used for back-to-back pipelined accesses to the same page within DRAM. When it is set, RAS is cycled between accesses. This bit should normally be cleared unless the Falcon has a problem operating that way.

ram fref Some DRAMs require that they be refreshed at the rate of 7.8µs per row rather than the standard 15.6µs per row. If any of the DRAM devices require the higher rate, then the **ram fref** bit should be left set, otherwise, it can be cleared.

ram spd0,ram spd1 Together **ram spd0,ram spd1** control DRAM timing used by the Falcon pair. They are encoded as shown:

Table 3-12. ram spd1,ram spd0 and DRAM Type

ram spd0, ram spd1	DRAM Speed	DRAM Type
%00	70ns	Fast Page
%01	60ns	Fast Page
%10	-	Reserved
%11	50ns	EDO

EDO refers to DRAMs that use an output latch on data. Sometimes these parts are referred to as Hyper-Page Mode DRAMs.

To ensure reliable operation, the system should always be configured so that these two bits are encoded to match the slowest devices that are used. Also, if any parts do not support EDO, then these bits must set for Page Mode. The only case in which it is permissible to set **ram spd0,ram spd1** for 50ns, EDO is when **all** parts are 50ns and **all** support EDO.

chipu indicates which of the two positions within the Falcon pair is occupied by this chip. When **chipu** is low, this chip is connected to the lower half of the PowerPC 60*x* data bus and it does not drive TA_ or AACK_. When **chipu** is high, this chip is connected to the upper half of the PowerPC 60*x* data bus, and it drives TA_ and AACK_. **chipu** reflects the level that was on the ERCS_ pin during power-up reset.

DRAM Attributes Register

Address	$FEF80010																													
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Name	ram a en	0	0	0	0	ram a siz0	ram a siz1	ram a siz2	ram b en	0	0	0	0	ram b siz0	ram b siz1	ram b siz2	ram c en	0	0	0	0	ram c siz0	ram c siz1	ram c siz2	ram d en	0	0	0	0	ram d siz0	ram d siz1	ram d siz2
Operation	R/W	R	R	R	R	R/W	R/W	R/W	R/W	R	R	R	R	R/W	R/W	R/W	R/W	R	R	R	R	R/W	R/W	R/W	R/W	R	R	R	R	R/W	R/W	R/W
Reset	0 PL	X	X	X	X	0 P	0 P	0 P	0 PL	X	X	X	X	0 P	0 P	0 P	0 PL	X	X	X	X	0 P	0 P	0 P	0 PL	X	X	X	X	0 P	0 P	0 P

Warning

To satisfy DRAM component requirements before the memory is used at start-up, software must always wait at least 500µs between the initial setting of a bank's size bits, to a non-zero value, and the first accessing of that bank. These settings are in the DRAM Attributes Register (offset $FEF80010). The delay is intended to make sure that the bank has been refreshed at least 8 times before it is used. The 500µs is sufficient as long as the CLK Frequency Register (offset $FEF80020) is within a factor of 2 of matching the actual 60*x* clock frequency

ram a/b/c/d en **ram a/b/c/d en** enables accesses to the corresponding block of DRAM when set, and disables them when cleared.

ram a/b/c/d siz0-2 These control bits define the size of their corresponding block of DRAM. Table 3-13 shows the block configuration assumed by the Falcon pair for each value of **ram siz0-ram siz2**.

Table 3-13. Block_A/B/C/D Configurations

ram a/b/c/d siz0-2	Block SIZE	Devices Used			Technology	Comments
%000	0MB	-	-	-	-	Block Not Present
%001	16MB	36	-	1Mx4's	4Mb	
		8	-	1Mx18's	16Mb	
		4	-	1Mx36's	4Mb/1Mb	SIMM/DIMM
%010	32MB	18	-	2Mx8's	16Mb	
%011	64MB	144	-	4Mx1's	4Mb	
		36	-	4Mx4's	16Mb	
		8	-	4Mx18's	64Mb	
		4	-	4Mx36's	16Mb/4Mb	SIMM/DIMM
%100	128MB	18	-	8Mx8's	64Mb	
%101	256MB	144	-	16Mx1's	16Mb	
		36	-	16Mx4's	64Mb	
		4	-	16Mx36's	64Mb/16Mb	SIMM/DIMM
%110	1024MB	144	-	64Mx1's	64Mb	
%111	0MB	-	-	-	-	Reserved

Note It is important that all of the **ram a/b/c/d siz0-2** bits be set to accurately match the actual size of their corresponding blocks. This includes clearing them to %000 if their corresponding blocks are not present. Failure to do so will cause problems with addressing and with scrub error logging.

DRAM Base Register

Address	$FEF80018			
Bit	0 1 2 3 4 5 6 7	8 9 10 11 12 13 14 15	16 17 18 19 20 21 22 23	24 25 26 27 28 29 30 31
Name	RAM A BASE	RAM B BASE	RAM C BASE	RAM D BASE
Operation	READ/WRITE	READ/WRITE	READ/WRITE	READ/WRITE
Reset	0 PL	0 PL	0 PL	0 PL

RAM A/B/C/D BASE These control bits define the base address for their block's DRAM. RAM A/B/C/D BASE bits 0-7/8-15/16-23/24-31 correspond to PowerPC 60*x* address bits 0 - 7. For larger DRAM sizes, the lower significant bits of A/B/C/D BASE are ignored. This means that the block's base address will always appear at an even multiple of its size.

Note Bit 0 is MSB.

Also note that the combination of **RAM_X_BASE** and **ram_x_siz** should never be programmed such that DRAM responds at the same address as the CSR, ROM/Flash, External Register Set, or any other slave on the PowerPC bus.

CLK Frequency Register

Address	$FEF80020																													
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Name	CLK FREQUENCY																								0	0	0	0	0	0	0	por
Operation	READ/WRITE								READ ZERO								READ ZERO								R	R	R	R	R	R	R	R/C
Reset	42 P								X								X								X	X	X	X	X	X	X	1 P

CLK FREQUENCY These bits should be programmed with the hexadecimal value of the operating CLOCK frequency in MHz (that is, $42 for 66 MHz). When these bits are programmed this way, the chip's

prescale counter produces a 1 MHz output. The output of the chip prescale counter is used by the refresher/scrubber and the 32-bit counter. After power-up, this register is initialized to $42 (for 66MHz).

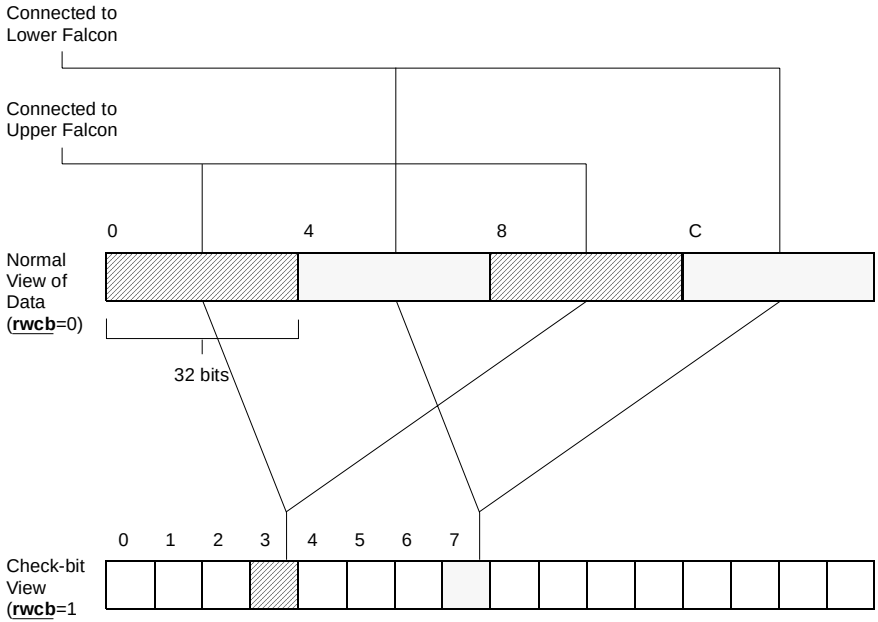
por **por** is set by the occurrence of power up reset. It is cleared by writing a one to it. Writing a 0 to it has no effect.

ECC Control Register

Address	$FEF80028																															
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Name	0	0	0	0	0	refdis	rwcb	derc	0	0	0	0	scien	dpien	sien	mien									0	0	0	0	0	0	0	mcken
Operation	R	R	R	R	R	R/W	R/W	R/W	R	R	R	R	R/W	R/W	R/W	R/W	READ ZERO								R	R	R	R	R	R	R	R/W
Reset	X	X	X	X	X	0 PL	0 PL	1 PL	X	X	X	X	0 PL	0 PL	0 PL	0 PL	X								X	X	X	X	X	X	X	0 PL

refdis When set, **refdis** causes the refresher and all of its associated counters and state machines to be cleared and maintained that way until **refdis** is removed (cleared). If a refresh cycle is in process when **refdis** is updated by a write to this register, the update does not take effect until the refresh cycle has completed. This prevents the generation of illegal cycles to the DRAM when **refdis** is updated.

rwcb, When set, causes reads and writes to DRAM from the PowerPC 60*x* bus to access check-bit data rather than normal data. The data path used for this mode is DH24-31 for check-bit data controlled by the upper Falcon, and DL24-31 for check-bit data controlled by the lower Falcon. Each 8-bit check-bit location services 64 bits of normal data. The 64 bits of data are all within the same Falcon. Each Falcon provides every other 32 bits of data in the normal mode. The figure below shows the relationship between normal data and check-bit data.



So, for example, the check-bits that correspond to the 64 bits of data found in normal mode (**rwcb**=0) at $00001000-$00001003 and $00001008-$0000100b are written and read in check-bit mode (**rwcb**=1) at location $00001003.

Note If test software wishes to force a single-bit error to a location using the **rwcb** function, the scrubber may correct the location before the test software gets a chance to check for the single-bit error. This can be avoided by disabling scrub writes. Also note that writing bad check-bits can set the **elog** bit in the Error Logger Register. The writing of check-bits causes the Falcon to perform a read-modify-write to DRAM. If the location to which check-bits are being written has a single- or double-bit error, data in the location may be altered by the write check-bits operation. To avoid this, it is recommended that the **derc** bit also be set while the **rwcb** bit is set. A possible sequence for performing read-write check-bits is as follows:

1. Disable scrub writes by clearing the **swen** bit if it is set.
2. Make sure software is not using DRAM at this point, because while **rwcb** is set, DRAM will not function as normal memory.
3. Set the **derc** and **rwcb** bits in the Data Control register.
4. Perform the desired read and/or write check-bit operations.
5. Clear the **derc** and **rwcb** bits in the Data Control register.
6. Perform the desired testing related to the location/locations that have had their check-bits altered.
7. Enable scrub writes by setting the **swen** bit if it was set before.

derc Setting **derc** to one alters Falcon pair operation as follows:

1. During reads, data is presented to the PowerPC 60*x* data bus uncorected from the DRAM array.
2. During single-beat writes, data is written without correcting single-bit errors that may occur on the read portion of the read-modify-write. Check-bits are generated for the data being written.
3. During single-beat writes, the write portion of the read-modify-write happens regardless of whether there is a multiple-bit error during the read portion. No correction of data is attempted. Check-bits are generated for the data being written.
4. During refresh/scrub cycles, if **swen** is set, a read-write to DRAM happens with no attempt to correct data bits. Check-bits are generated for the data being written.

derc is useful for initializing DRAM after power-up and for testing DRAM, but it should be cleared during normal system operation.

scien When **scien** is set, the rolling over of the **SBE COUNT** register causes the INT_ signal pin to pulse true.

dpien When **dpien** is set, the logging of a 60*x* data parity error causes the INT_ signal pin to pulse true.

sien When **sien** is set, the logging of a single-bit error causes the INT_ signal pin to pulse true.

mien When **mien** is set, the logging of a non-correctable error causes the INT_ signal pin to pulse true.

mcken When **mcken** is set, the detection of a multiple-bit error during a PowerPC read or write causes the Falcon to pulse its machine check interrupt request pin (MCP_) true. When **mcken** is cleared, the Falcon does not ever assert its MCP_ pin.

The Falcon never asserts its MCP_ pin in response to a multiple-bit error detected during a scrub cycle.

Caution

Note that the INT_ and MCP_ pins are the only non-polled notification that a multiple-bit error has occurred. The Falcon pair does not assert TEA as a result of a multiple bit error. In fact, the Falcon pair does not have a TEA_ signal pin and it assumes that the system does not implement TEA_.

Error Logger Register

Address	$FEF80030																													
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Name	elog	0	0	0	escb	esen	embt	esbt	ERROR_SYNDROME								0	0	eblk0	eblk1	0	0	0	scof	SBE_COUNT							
Operation	R/C	R	R	R	R	R/W	R	R	READ ONLY								R	R	R	R	R	R	R	R/C	READ/WRITE							
Reset	0 P	X	X	X	X P	0 PL	X P	X P	X P								X	X	X	X	X	X	X	0 P	0 P							

The Error Logger and Error Address Registers behave the same as the other registers, in that data written to the upper Falcon is automatically duplicated in the lower Falcon. They also behave the same as the other registers, in that status read from the upper Falcon pertains to the upper Falcon, and status read from the lower Falcon pertains to the lower Falcon. Unlike most of the other registers, however, it is normal for this status to differ between the two. This is due to the fact that each Falcon is connected

to its own set of DRAMs. The upper Falcon can log an error during a cycle and the local Falcon not, or vice-versa. Or they can both log an error during the same cycle and have the attributes of the errors differ.

Because of the above, software needs to monitor both the upper and lower Falcon's Error Logger and Error Address Registers. This includes checking the **elog** bit from the upper Falcon and from the lower Falcon. When the upper Falcon logs an error, it updates its attribute bits (**escb**, **embt**, **esbt**, **ERROR_SYNDROME**, **eblk0**, **eblk1**, and **ERROR_ADDRESS**) to match the results of the read cycle for its portion of the DRAM array. When the lower Falcon logs an error, it updates its attribute bits to match the results of the read cycle for its portion of the DRAM array.

While the logging of errors by one Falcon in a pair does not affect the logging of errors by the other, writing to the Error Logger Register control bits affects both Falcons. This is of particular interest as regards the **elog** bit. Writing a one to the **elog** bit clears the **elog** bit for both the upper and lower Falcons. Because of this, software needs to check the status of both upper and lower Error Logger and Error Address Registers before it clears the **elog** bits. Otherwise, it could miss a logged error.

elog When set, **elog** indicates that a single- or a multiple-bit error has been logged by its Falcon. If **elog** is set by a multiple-bit error, then no more errors will be logged until software clears it. If **elog** is set by a single-bit error, then no more single-bit errors will be logged until software clears it; however if **elog** is set by a single-bit error and a multiple-bit error occurs, the multiple-bit error will be logged and the single-bit error information overwritten. **elog** can only be set by the logging of an error and cleared by the writing of a one to itself or by power-up reset.

escb indicates the entity that was accessing DRAM at the last logging of a single- or multiple-bit error by its Falcon. If **escb** is 1, it indicates that the scrubber was accessing DRAM. If **escb** is 0, it indicates that the PowerPC 60x bus master was accessing DRAM.

esen When set, **esen** allows errors that occur during scrubs to be logged. When cleared, **esen** does not allow errors that occur during scrubs to be logged.

embt is set by the logging of a multiple-bit error in its Falcon. It is cleared by the logging of a single-bit error in its Falcon. It is undefined after power-up reset. A Falcon's syndrome code is meaningless if its **embt** bit is set.

esbt is set by the logging of a single-bit error in its Falcon. It is cleared by the logging of a multiple-bit error in its Falcon. When a Falcon logs a single-bit error, its syndrome code indicates which bit was in error. See *ECC Codes* for more information.

ERROR_SYNDROME reflects the syndrome value at the last logging of an error by its Falcon. This eight-bit code indicates the position of the data error. When all the bits are zero, there was no error. Note that if the logged error was non-correctable, then these bits are meaningless. Refer to the section on *ECC Codes* for a decoding of the syndromes.

esblk0,esblk1 Together these two bits indicate which block of DRAM was being accessed when their Falcon logged a scrub error. **esblk0**,**esblk1** are 0,0 for Block A; 0,1 for Block B; 1,0 for Block C; and 1,1 for Block D.

scof is set by its **SBE COUNT** register rolling over from $FF to $00. It is cleared by software writing a 1 to it.

SBE COUNT This register keeps track of the number of single-bit errors that have occurred since it was last cleared. It counts up by one each time its half of the Falcon pair detects a single-bit error (independent of the state of the elog bit). It is cleared by power-up reset and by software writing all zeros to it. When **SBE COUNT** rolls over from $FF to $00, its Falcon sets the **scof** bit. It also pulses the INT_ signal low if the **scien** bit is set.

Error_Address Register

Address	$FEF80038																															
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Name	ERROR_ADDRESS																												0	0	0	0
Operation	READ ONLY																												R	R	R	R
Reset	X P																												X	X	X	X

ERROR_ADDRESS These bits reflect the value that corresponds to bits 0-27 of the PowerPC 60*x* address bus when their Falcon last logged an error during a PowerPC access to DRAM. They reflect the value of the DRAM row and column addresses if the error was logged during a scrub cycle. In this case, bits 2-14 correspond to row address signals 0-12 respectively and bits 15-27 correspond to column address signals 0-12 respectively. Refer to Table 3-20 on page 3-61. It shows how PowerPC addresses correspond to DRAM row and column addresses.

Scrub/Refresh Register

Address	$FEF80040																															
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Name	scb0	scb1	0	0	0	0	0	swen	0	0	0	0	0	rtest0	rtest1	rtest2																
Operation	R	R	R	R	R	R	R	R/W	R	R	R	R	R	R/W	R/W	R/W	READ ZERO								READ ZERO							
Reset	0 P	0 P	X	X	X	X	X	0 P	X	X	X	X	X	0 P	0 P	0 P	X								X							

scb0,scb1 These bits increment every time the scrubber completes a scrub of the entire DRAM. When these bits reach binary 11, they roll over to binary 00 and continue. These bitsare cleared by power-up reset.

swen When set, **swen** allows the scrubber to perform write cycles. When cleared, **swen** prevents scrubber writes.

rtest0,1,2 The **rtest** bits enable certain refresh counter test modes. Table 3-14 shows their encodings.

Note These test modes are not intended to be used once the chip is in a system.

Table 3-14. rtest encodings

rtest0,rtest1,rtest2	Test Mode selected
%000	Normal Counter Operation
%001	RA counts at 16x
%010	RA counts at 256x
%011	RA is always at roll value for CA
%100	CA counts at 16x
%101	CA counts at 256x
%110	reserved
%111	reserved

Refresh/Scrub Address Register

Address	$FEF80048																													
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Name	0	0	0	ROW ADDRESS													0	0	0	COL ADDRESS												
Operation	R	R	R	READ/WRITE													R	R	R	READ/WRITE												
Reset	X	X	X	0 P													X	X	X	0 P												

ROW ADDRESS These bits form the row address counter used by the refresher/scrubber for all blocks of DRAM. The row address counter increments by one after each refresh/scrub cycle. When it reaches all 1s, it rolls back over to all 0s and continues counting. **ROW ADDRESS** is readable and writable for test purposes.

Note Within each block, the most significant bits of **ROW ADDRESS** are used only when their DRAM devices are large enough to require them.

COL ADDRESS These bits form the column address counter used by the refresher/scrubber for all blocks of DRAM. The counter increments by one every eighth time the **ROW ADDRESS** rolls over. **COL ADDRESS** is readable and writable for test purposes.

Note Within each block, the most significant bits of **COL ADDRESS** are only used when their DRAM devices are large enough to require them.

ROM A Base/Size Register

Address	$FEF80050																													
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Name	ROM A BASE												rom_a_64	rom a siz0	rom a siz1	rom a siz2									0	0	0	0	0	rom_a_rv	rom a en	rom a we
Operation	READ/WRITE												R	R/W	R/W	R/W	READ ZERO								R	R	R	R	R	R/W	R/W	R/W
Reset	$FF0 PL												V P	0 PL	0 PL	0 PL	X								X	X	X	X	X	V P	0 PL	0 PL

ROM A BASE These control bits define the base address for ROM/Flash Block A. **ROM A BASE** bits 0-11 correspond to PowerPC 60*x* address bits 0 - 11 respectively. For larger ROM/Flash sizes, the lower significant bits of **ROM A BASE** are ignored. This means that the block's base address will always appear at an even multiple of its size. **ROM A BASE** is initialized to $FF0 at power-up or local bus reset.

Note In addition to the programmed address, the first 1Mbyte of Block A also appears at $FFF00000 - $FFFFFFFF if the **rom_a_rv** bit is set and the **rom_b_rv** bit is cleared.

Also note that the combination of **ROM_A_BASE** and **rom_a_siz** should never be programmed such that ROM/Flash Block A responds at the same address as the CSR, DRAM, External Register Set, or any other slave on the PowerPC bus.

rom_a_64 indicates the width of ROM/Flash device/devices being used for Block A. When **rom_a_64** is cleared, Block A is 16 bits wide, where each Falcon interfaces to 8 bits. When **rom_a_64** is set, Block A is 64 bits wide, where each Falcon interfaces to 32 bits. **rom_a_64** matches the value that was on the CKD2 pin at power-up reset. It cannot be changed by software.

rom a siz The **rom a siz** control bits are the size of ROM/Flash for Block A. They are encoded as shown in the following table.

Table 3-15. ROM/Flash Block A Size Encoding

rom a siz	BLOCK SIZE
%000	1MB
%001	2MB
%010	4MB
%011	8MB
%100	16MB
%101	32MB
%110	64MB
%111	Reserved

rom_a_rv and **rom_b_rv** determine which if either of Blocks A and B is the source of reset vectors or any other access in the range $FFF00000 - $FFFFFFFF as shown in the table below.

Table 3-16. rom_a_rv and rom_b_rv encoding

rom_a_rv	rom_b_rv	Result
0	0	Neither Block is the source of reset vectors.
0	1	Block B is the source of reset vectors.
1	0	Block A is the source of reset vectors.
1	1	Block B is the source of reset vectors.

rom_a_rv is initialized at power-up reset to match the value on the CKD0 pin.

rom a en When **rom a en** is set, accesses to Block A ROM/Flash in the address range selected by **ROM A BASE** are enabled. When **rom a en** is cleared, they are disabled.

rom a we When **rom a we** is set, writes to Block A ROM/Flash are enabled. When **rom a we** is cleared, they are disabled. Note that if **rom_a_64** is cleared, only 1-byte writes are allowed. If **rom_a_64** is set, only 4-byte writes are allowed. The Falcon ignores other writes. If a valid

write is attempted and **rom a we** is cleared, the write does not happen but the cycle is terminated normally. See Table 3-17 for details of ROM/Flash accesses.

Table 3-17. Read/Write to ROM/Flash

Cycle	Transfer Size	Alignment	rom_x_64	rom_x_we	Falcon Response
write	1-byte	X	0	0	Normal termination, but no write to ROM/Flash
write	1-byte	X	0	1	Normal termination, write occurs to ROM/Flash
write	1-byte	X	1	X	No Response
write	4-byte	Misaligned	X	X	No Response
write	4-byte	Aligned	0	X	No Response
write	4-byte	Aligned	1	0	Normal termination, but no write to ROM/Flash
write	4-byte	Aligned	1	1	Normal termination, write occurs to ROM/Flash
write	2,3,5,6,7, 8,32-byte	X	X	X	No Response
read	X	X	X	X	Normal Termination

ROM B Base/Size Register

Address	$FEF80058																													
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Name	ROM B BASE												rom_b_64	rom b siz0	rom b siz1	rom b siz2									0	0	0	0	0	rom_b_rv	rom b en	rom b we
Operation	READ/WRITE												R	R/W	R/W	R/W	READ ZERO								R	R	R	R	R	R/W	R/W	R/W
Reset	$FF4 PL												V P	0 PL	0 PL	0 PL	X								X	X	X	X	X	V P	0 PL	0 PL

ROM B BASE These control bits define the base address for ROM/Flash Block B. **ROM B BASE** bits 0-11 correspond to PowerPC 60*x* address bits 0 - 11 respectively. For larger ROM/Flash sizes, the lower significant bits of **ROM B BASE** are ignored. This means that the block's base address will always appear at an even multiple of its size. **ROM B BASE** is initialized to $FF4 at power-up or local bus reset.

Note In addition to the programmed address, the first 1MB of Block B also appears at $FFF00000 - $FFFFFFFF if the **rom_b_rv** bit is set.

Also note that the combination of **ROM_B_BASE** and **rom_b_siz** should never be programmed such that ROM/Flash Block B responds at the same address as the CSR, DRAM, External Register Set, or any other slave on the PowerPC bus.

rom_b_64 indicates the width of ROM/Flash device/devices being used for Block B. When **rom_b_64** is cleared, Block B is 16 bits wide, where each Falcon interfaces to 8 bits. When **rom_b_64** is set, Block B is 64 bits wide, where each Falcon interfaces to 32 bits. **rom_b_64** matches the inverse of the value that was on the CKD3 pin at power-up reset. It cannot be changed by software.

rom b siz The **rom b siz** control bits are the size of ROM/Flash for Block B. They are encoded as shown in the following table.

Table 3-18. ROM/Flash Block B Size Encoding

rom b siz	BLOCK SIZE
%000	1Mbytes
%001	2Mbytes
%010	4Mbytes
%011	8Mbytes
%100	16Mbytes
%101	32Mbytes
%110	64Mbytes
%111	Reserved

rom_b_rv and **rom_a_rv** determine which if either of Blocks A and B is the source of reset vectors or any other access in the range $FFF00000 - $FFFFFFFF as shown in Table 3-16.

rom_b_rv is initialized at power-up reset to match the inverse of the value on the CKD1 pin.

rom b en When **rom b en** is set, accesses to Block B ROM/Flash in the address range selected by **ROM B BASE** are enabled. When **rom b en** is cleared they are disabled.

rom b we When **rom b we** is set, writes to Block B ROM/Flash are enabled. When **rom b we** is cleared they are disabled. Refer back to Table 3-17 for more details.

ROM Speed Control Register

Address	$FEF80060																													
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Name																									0	0	rom a spd0	rom a spd1	0	0	rom b spd0	rom b spd1
Operation	READ ZERO								READ ZERO								READ ZERO								R	R	R/W	R/W	R	R	R/W	R/W
Reset	X								X								X								X	X	0 PL	0 PL	X	X	0 PL	0 PL

rom_a_spd0,1 determine the access timing used for ROM/Flash Block A. The encodings of these bits are as follows.

Table 3-19. Rom Speed Bit Encodings

rom_a/b_spd0,1	ROM Block A/B Access Time
%00	180ns
%01	120ns
%10	75ns
%11	45ns

rom_b_spd0,1 determine the access timing used for ROM/Flash Block B. See table above.

Data Parity Error Logger Register

Address	$FEF80068																															
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Name	dpelog	0	0	dpe_tt0	dpe_tt1	dpe_tt2	dpe_tt3	dpe_tt4	0	0	0	0	dpe_dp0	dpe_dp1	dpe_dp2	dpe_dp3	0	0	0	0	0	0	dpe_ckall	dpe_me	GWDP							
Operation	R/C	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R/W	R/W	READ/WRITE							
Reset	0 P		X	0 P	0 P	0 P	0 P	0 P	X	X	X	X	0 P	0 P	0 P	0 P	X	X	X	X	X	X	0 PL	0 PL	0 PL							

The Data Parity Error Logger and Data Parity Error Address and Data registers are much like the Error Logger and Error Address registers in that it is normal for status to differ between the upper and lower Falcons. This is due to the fact that each Falcon is connected to its own half of the 60*x* data bus and data parity bus. The upper Falcon can log parity error during a cycle and the lower Falcon not, or vice-versa. Or they can both log a parity error during the same cycle and have the attributes of the errors differ.

Because of the above, software needs to monitor both the upper and lower Falcon's Data Parity Error Logger, Data Parity Error Address and Data Parity Error Data Registers. This includes checking the **dpelog** bit from the upper Falcon and from the lower Falcon. When the upper Falcon logs an error, it updates its attribute bits (**dpe0-3**, **DATA PARITY ERROR ADDRESS,** and **DATA PARITY ERROR DATA**) to match the results of the read cycle for the upper 60*x* data bus. When the lower Falcon logs an error, it updates its attribute bits to match the results of the read cycle for the lower 60*x* data bus.

While the logging of data parity errors by one Falcon in a pair does not affect the logging of data parity errors by the other, writing to the Data Parity Error Logger control bits does affect both Falcons. This is of particular interest as regards the **dpelog** bit. Writing a one to the **dpelog** bit clears the **dpelog** bit for both the upper and lower Falcons. Because of this,

software needs to check the status of both upper and lower Data Parity Error Logger, Address, and Data registers before it clears the **dpelog** bits. Otherwise, it could miss a logged error.

dpelog is set when a parity error occurs on the Falcon's half of the 60*x* data bus during any 60*x* data cycle. It is cleared by writing a one to the upper Falcon's **dpelog** bit or by power-up reset.

dpe_tt is the value that was on the tt0-tt4 signals when the Falcon's **dpelog** bit was set.

dpe_dp is the value that was on the Falcon's dp0-dp3 signals when its **dpelog** bit was set.

When **dpe_ckall** is set, the Falcon checks data parity on all cycles in which ta_ is asserted. When **dpe_ckall** is cleared, the Falcon checks data parity on cycles when ta_ is asserted only during writes to the Falcon Pair.

Note The Falcon does not check parity during cycles in which there is a qualified artry_ at the same time as the ta_

When **dpe_me** is set, the transition of a Falcon's **dpelog** bit from false to true causes it to pulse its machine check interrupt request pin (MCP_) true. When **dpe_me** is cleared, the Falcon does not assert its MCP_ pin based on its **dpelog** bit.

The **GWDP0-7** bits are used to invert the value that is driven onto DP0-DP7 respectively during reads to the Falcon pair. This allows test software to generate wrong (even) parity on selected byte lanes. For example, to create a parity error on the 60*x* DL24-DL31 and DP7 signals during Falcon reads, software should set **GWDP7**.

Note While the value on **GWDP** is duplicated on both the upper and lower Falcons, **GWDP0-3** affect only the upper Falcon and **GWDP4-7** affect only the lower Falcon.

Data Parity Error Address Register

Address	$FEF80070																															
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Name	DPE_A																															
Operation	READ ONLY																															
Reset	0 P																															

DPE_A is the address of the last 60*x* data bus parity error that was logged by a Falcon. It is updated only when the Falcon's **dpelog** bit goes from 0 to 1.

Data Parity Error Data Register

Address	$FEF80078																															
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Name	DPE_D																															
Operation	READ ONLY																															
Reset	0 P																															

DPE_D is the value on the Falcon's half of the 60*x* data bus at the time it last logged a 60*x* data bus parity error. **DPE_D** updates only when the associated **dpelog** bit goes from 0 to 1.

32-Bit Counter

Address	$FEF80100
Bit	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Name	CTR32
Operation	READ/WRITE
Reset	0 PL

CTR32 is a 32-bit, free-running counter that increments once per microsecond if the CLK_FREQUENCY register has been programmed properly. Notice that **CTR32** is cleared by power-up and local reset. It does not exist in Revision 1 of Falcon.

Note When the system clock is a fractional frequency, such as 66.67 MHz, **CTR32** will count at a fractional amount faster or slower than 1 MHz, depending on the programming of the CLK_FREQUENCY Register.

Power-Up Reset Status Register 1

Address	$FEF80400
Bit	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Name	PR_STAT1
Operation	READ
Reset	V P

PR_STAT1 (power-up reset status) reflects the value that was on the RD0-RD31 signal pins at power-up reset. This register is read-only.

Note For descriptions of how this register is used in the PowerPC series boards, refer to the *Falcon-Controlled System Registers* on page 1-16, especially the *System Configuration Register (SYSCR)* on page 1-17 and the *Memory Configuration Register (MEMCR)* on page 1-18.

Power-Up Reset Status Register 2

Address	$FEF80500																													
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Name	PR_STAT2																															
Operation	READ																															
Reset	V P																															

PR_STAT2 (power-up reset status) reflects the value that was on the RD32-RD63 signal pins at power-up reset. This register is read-only.

External Register Set

Address	$FEF88000 - $FEF8FFF8																															
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Name	EXTERNAL REGISTER SET																															
Operation	READ/WRITE																															
Reset	X PL																															

The **EXTERNAL REGISTER SET** is user provided and is external to the Falcon pair. The Falcon pair provides a static RAM style interface for the external registers. The external registers can be SRAM, ROM, Flash or some other user device. There can be one device connected to either the upper or lower Falcon, or there can be two devices with one connected to each Falcon. The devices can be 8, 16, or 32 bits wide. The data path to the external devices is via the RD32-RD63 pins. Reads to the external devices can be any size except burst. Note that if the devices are less than 32 bits wide, reads to unused data lanes will yield undefined data. Note that writes are restricted to one or 4-byte length only. 4-byte writes can be used for any size device, data should be placed on the correct portion of the data bus so that valid data is written to the device. Data duplication is turned off for the **EXTERNAL REGISTER SET** so writes can be to either the upper Falcon, or to the lower Falcon.

Note For descriptions of how these registers are used in the PowerPC series boards, refer to the *Falcon-Controlled System Registers* on page 1-16, especially the *System External Cache Control Register (SXCCR)* on page 1-20 and the *CPU Control Register* on page 1-23.

Software Considerations

This section contains information that will be useful in programming a system that uses the Falcon pair.

Parity Checking on the PowerPC Bus

The Falcon does not generate parity on the PowerPC address bus. Because of this, the appropriate registers in the MPC60*x* should not be programmed to enable parity checking for the address bus.

The Falcon does generate parity on the PowerPC data bus. The appropriate registers in the MPC60*x* can be programmed to enable parity checking for the data bus.

Programming ROM/Flash Devices

Those who program devices to be controlled by the Falcon should make note of the address mapping that is shown in Table 3-9 and in Table 3-10. For example, when using 8-bit devices, the code will be split so that every other 4-byte segment goes in each device.

Writing to the Control Registers

Software should not change control register bits that affect DRAM operation while DRAM is being accessed. Because of pipelining, software should always make sure that the two accesses before and after the updating of critical bits are not DRAM accesses. A possible scenario for trouble would be to execute code out of DRAM while updating the critical

DRAM control register bits. The preferred method is to be executing code out of ROM/Flash and avoiding DRAM accesses while updating these bits.

Since software has no way of controlling refresh accesses to DRAM, the hardware is designed so that updating control bits coincidentally with refreshes is not a problem. An exception to this is the **ROW_ADDRESS** and **COL_ADDRESS** bits. It is not intended that software write to these bits anyway.

As with DRAM, software should not change control register bits that affect ROM/Flash while the affected Block is being accessed. This generally means that the ROM/Flash size, base address, enable, write enable, etc. are changed only while executing initially in the reset vector area ($FFF00000 - $FFFFFFFF).

To satisfy DRAM component requirements before the memory is used at start-up, software must always wait at least 500µs between the initial setting of a bank's size bits, to a non-zero value, and the first accessing of that bank. These settings are in the DRAM Attributes Register (offset $FEF80010). The delay is intended to make sure that the bank has been refreshed at least 8 times before it is used. The 500µs is sufficient as long as the CLK Frequency Register (offset $FEF80020) is within a factor of 2 of matching the actual 60*x* clock frequency.

Sizing DRAM

The following routine can be used to size DRAM for the Falcon.

Initialize the Falcon control register bits to a known state as follows:

1. Clear the **isa_hole** bit.
2. Make sure that **ram_fref** and **ram_spd0**,**ram_spd1** are correct.
3. Set CLK_FREQUENCY to match the operating frequency.
4. Clear the **refdis**, **rwcb** bits.
5. Set the **derc** bit.

6. Clear the **scien**, **dpien**, **sien**, and **mien** bits.
7. Clear the **mcken** bit.
8. Clear the **swen** and **rtest0**, **rtest1**, **rtest2** bits.
9. Make sure that ROM/Flash banks A and B are not enabled to respond in the range from $00000000 to $40000000.
10. Make sure that no other devices respond in the range from $00000000 to $40000000.

For each block:

1. Set the block's base address to $00000000.
2. Enable the block and make sure that the other three blocks are disabled.
3. Set the block's size control bits. Start with the largest possible (1024MB).
4. Write differing 64-bit data patterns to certain addresses within the block. The data patterns do not matter as long as each 64-bit data pattern is unique. The addresses to be written vary depending on the size that is currently being checked and are specified in Table 3-20. Table 3-20 shows how PowerPC addresses correspond to DRAM row/column addresses.
5. Read back all of the addresses that have been written.

   If all of the addresses still contain exactly what was written then the block's size has been found. It is the size for which the block is currently programmed.

   If any of the addresses do not match exactly then the amount of memory is less than that for which it is currently programmed. Sizing needs to continue for this block by programming its control bits to the next smaller size and repeating steps 4 and 5.
6. If no match is found for any size then the block is unpopulated and has a size of 0MB.

Each size that is checked has a specific set of locations that must be written and read. The following table shows the addresses that go with each size.

1024MB	256MB	128MB	64MB	32MB	16MB
$00000000 $20000000	$00000000, $02000000, $08000000, $0A000000	$00000000 $00002000 $02000000 $02002000 $04000000 $04002000 $06000000 $06002000	$00000000 $00002000 $02000000 $02002000	$00000000 $00001000 $00002000 $00003000 $01000000 $01001000 $01002000 $01003000	$00000000

Table 3-20. PowerPC 60x Address to DRAM Address Mappings

RA ---- ▸ Block Size | ▾		0	1	2	3	4	5	6	7	8	9	10	11	12
16MB	ROW		A19	A18	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17
	COL				A18	A19	A20	A21	A22	A23	A24	A25	A26	A27
32MB	ROW		A18	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17
	COL				A18	A19	A20	A21	A22	A23	A24	A25	A26	A27
64MB	ROW	A18	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17
	COL			A6	A18	A19	A20	A21	A22	A23	A24	A25	A26	A27
128MB	ROW	A6	A5	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17
	COL			A6	A18	A19	A20	A21	A22	A23	A24	A25	A26	A27
256MB	ROW	A4	A5	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17
	COL		A4	A6	A18	A19	A20	A21	A22	A23	A24	A25	A26	A27
1024MB	ROW	A3	A5	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17
	COL	A2	A4	A6	A18	A19	A20	A21	A22	A23	A24	A25	A26	A27

ECC Codes

When the Falcon reports a single-bit error, software can use the syndrome that was logged by the Falcon (upper or lower depending where the error occurred) to determine which bit was in error. Table shows the syndrome for each possible single bit error. Table 3-23 shows the same information ordered by syndrome. In order to relate this information to PowerPC addresses and bit numbers, the user needs to understand how the Falcon pair positions PowerPC data in DRAM. See the section on *Data Paths* on page 3-64 for an explanation of this.

Note Table and Table 3-23 are the same whether the Falcon is configured as upper or as lower.

Table 3-21. Syndrome Codes Ordered by Bit in Error

Bit	Syndrome	Bit	Syndrome	Bit	Syndrome	Bit	Syndrome	Bit	Syndrome
rd0	$4A	rd16	$92	rd32	$A4	rd48	$29	ckd0	$01
rd1	$4C	rd17	$13	rd33	$C4	rd49	$31	ckd1	$02
rd2	$2C	rd18	$0B	rd34	$C2	rd50	$B0	ckd2	$04
rd3	$2A	rd19	$8A	rd35	$A2	rd51	$A8	ckd3	$08
rd4	$E9	rd20	$7A	rd36	$9E	rd52	$A7	ckd4	$10
rd5	$1C	rd21	$07	rd37	$C1	rd53	$70	ckd5	$20
rd6	$1A	rd22	$86	rd38	$A1	rd54	$68	ckd6	$40
rd7	$19	rd23	$46	rd39	$91	rd55	$64	ckd7	$80

Table 3-21. Syndrome Codes Ordered by Bit in Error

Bit	Syndrome	Bit	Syndrome	Bit	Syndrome	Bit	Syndrome	Bit	Syndrome
rd8	$25	rd24	$49	rd40	$52	rd56	$94		
rd9	$26	rd25	$89	rd41	$62	rd57	$98		
rd10	$16	rd26	$85	rd42	$61	rd58	$58		
rd11	$15	rd27	$45	rd43	$51	rd59	$54		
rd12	$F4	rd28	$3D	rd44	$4F	rd60	$D3		
rd13	$0E	rd29	$83	rd45	$E0	rd61	$38		
rd14	$0D	rd30	$43	rd46	$D0	rd62	$34		
rd15	$8C	rd31	$23	rd47	$C8	rd63	$32		

Table 3-22. Single-Bit Errors Ordered by Syndrome Code

Syndrome	Bit	Syndrome	Bit	Syndrome	Bit	Syndrome	Bit	Syndrome	Bit	Syndrome	Bit	Syndrome	Bit	Syndrome	Bit
$00	-	$20	ckd5	$40	ckd6	$60	-	$80	ckd7	$A0	-	$C0	-	$E0	rd45
$01	ckd0	$21	-	$41	-	$61	rd42	$81	-	$A1	rd38	$C1	rd37	$E1	-
$02	ckd1	$22	-	$42	-	$62	rd41	$82	-	$A2	rd35	$C2	rd34	$E2	-
$03	-	$23	rd31	$43	rd30	$63	-	$83	rd29	$A3	-	$C3	-	$E3	-
$04	ckd2	$24	-	$44	-	$64	rd55	$84	-	$A4	rd32	$C4	rd33	$E4	-
$05	-	$25	rd8	$45	rd27	$65	-	$85	rd26	$A5	-	$C5	-	$E5	-
$06	-	$26	rd9	$46	rd23	$66	-	$86	rd22	$A6	-	$C6	-	$E6	-
$07	rd21	$27	-	$47	-	$67	-	$87	-	$A7	rd52	$C7	-	$E7	-
$08	ckd3	$28	-	$48	-	$68	rd54	$88	-	$A8	rd51	$C8	rd47	$E8	-
$09	-	$29	rd48	$49	rd24	$69	-	$89	rd25	$A9	-	$C9	-	$E9	rd4
$0A	-	$2A	rd3	$4A	rd0	$6A	-	$8A	rd19	$AA	-	$CA	-	$EA	-
$0B	rd18	$2B	-	$4B	-	$6B	-	$8B	-	$AB	-	$CB	-	$EB	-

Table 3-22. Single-Bit Errors Ordered by Syndrome Code (Continued)

Syndrome	Bit	Syndrome	Bit	Syndrome	Bit	Syndrome	Bit	Syndrome	Bit	Syndrome	Bit	Syndrome	Bit	Syndrome	Bit
$0C	-	$2C	rd2	$4C	rd1	$6C	-	$8C	rd15	$AC	-	$CC	-	$EC	-
$0D	rd14	$2D	-	$4D	-	$6D	-	$8D	-	$AD	-	$CD	-	$ED	-
$0E	rd13	$2E	-	$4E	-	$6E	-	$8E	-	$AE	-	$CE	-	$EE	-
$0F	-	$2F	-	$4F	rd44	$6F	-	$8F	-	$AF	-	$CF	-	$EF	-
$10	ckd4	$30	-	$50	-	$70	rd53	$90	-	$B0	rd50	$D0	rd46	$F0	-
$11	-	$31	rd49	$51	rd43	$71	-	$91	rd39	$B1	-	$D1	-	$F1	-
$12	-	$32	rd63	$52	rd40	$72	-	$92	rd16	$B2	-	$D2	-	$F2	-
$13	rd17	$33	-	$53	-	$73	-	$93	-	$B3	-	$D3	rd60	$F3	-
$14	-	$34	rd62	$54	rd59	$74	-	$94	rd56	$B4	-	$D4	-	$F4	rd12
$15	rd11	$35	-	$55	-	$75	-	$95	-	$B5	-	$D5	-	$F5	-
$16	rd10	$36	-	$56	-	$76	-	$96	-	$B6	-	$D6	-	$F6	-
$17	-	$37	-	$57	-	$77	-	$97	-	$B7	-	$D7	-	$F7	-
$18	-	$38	rd61	$58	rd58	$78	-	$98	rd57	$B8	-	$D8	-	$F8	-
$19	rd7	$39	-	$59	-	$79	-	$99	-	$B9	-	$D9	-	$F9	-
$1A	rd6	$3A	-	$5A	-	$7A	rd20	$9A	-	$BA	-	$DA	-	$FA	-
$1B	-	$3B	-	$5B	-	$7B	-	$9B	-	$BB	-	$DB	-	$FB	-
$1C	rd5	$3C	-	$5C	-	$7C	-	$9C	-	$BC	-	$DC	-	$FC	-
$1D	-	$3D	rd28	$5D	-	$7D	-	$9D	-	$BD	-	$DD	-	$FD	-
$1E	-	$3E	-	$5E	-	$7E	-	$9E	rd36	$BE	-	$DE	-	$FE	-
$1F	-	$3F	-	$5F	-	$7F	-	$9F	-	$BF	-	$DF	-	$FF	-

Data Paths

Because of the Falcon pair architecture, data paths can be confusing. Figure 3-10 attempts to show the placement of data that is written by a PowerPC master to DRAM. Table 3-23 shows the same information in tabular format.

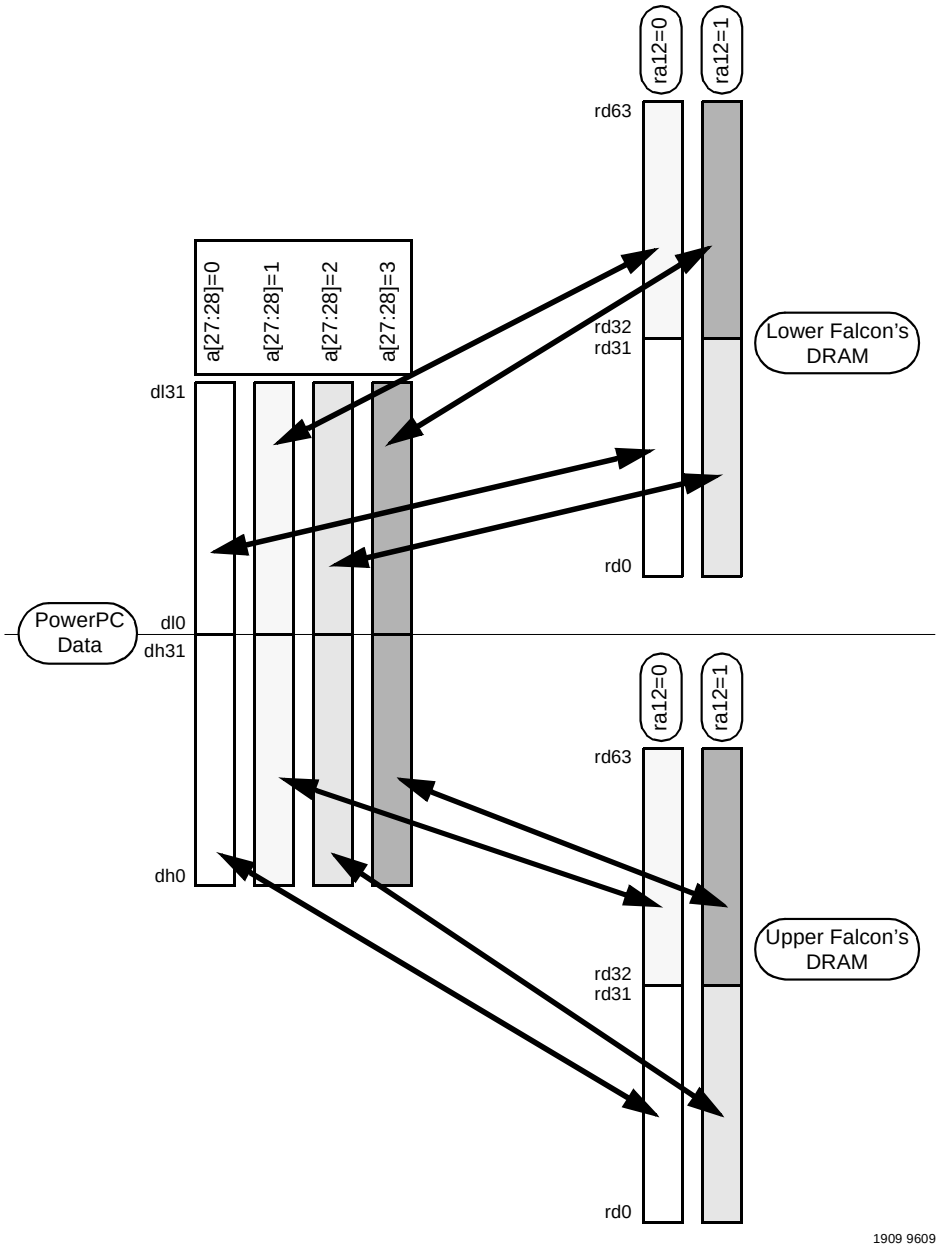


Figure 3-10. PowerPC Data to DRAM Data Correspondence

Table 3-23. PowerPC Data to DRAM Data Mapping

PowerPC			DRAM Array		
A[27]	**A[28]**	**Data Bits**	**RA[12]**	**Upper Falcon DRAM Data Bits**	**Lower Falcon DRAM Data Bits**
0	0	dh[00:07]	0	rd[00:07]	-
0	0	dh[08:15]	0	rd[08:15]	-
0	0	dh[16:23]	0	rd[16:23]	-
0	0	dh[24:31]	0	rd[24:31]	-
0	0	dl[00:07]	0	-	rd[00:07]
0	0	dl[08:15]	0	-	rd[08:15]
0	0	dl[16:23]	0	-	rd[16:23]
0	0	dl[24:31]	0	-	rd[24:31]
0	1	dh[00:07]	0	rd[32:39]	-
0	1	dh[08:15]	0	rd[40:47]	-
0	1	dh[16:23]	0	rd[48:55]	-
0	1	dh[24:31]	0	rd[56:63]	-
0	1	dl[00:07]	0	-	rd[32:39]
0	1	dl[08:15]	0	-	rd[40:47]
0	1	dl[16:23]	0	-	rd[48:55]
0	1	dl[24:31]	0	-	rd[56:63]
1	0	dh[00:07]	1	rd[00:07]	-
1	0	dh[08:15]	1	rd[08:15]	-
1	0	dh[16:23]	1	rd[16:23]	-
1	0	dh[24:31]	1	rd[24:31]	-
1	0	dl[00:07]	1	-	rd[00:07]
1	0	dl[08:15]	1	-	rd[08:15]
1	0	dl[16:23]	1	-	rd[16:23]
1	0	dl[24:31]	1	-	rd[24:31]
1	1	dh[00:07]	1	rd[32:39]	-
1	1	dh[08:15]	1	rd[40:47]	-

Table 3-23. PowerPC Data to DRAM Data Mapping (Continued)

PowerPC			DRAM Array		
1	1	dh[16:23]	1	rd[48:55]	-
1	1	dh[24:31]	1	rd[56:63]	-
1	1	dl[00:07]	1	-	rd[32:39]
1	1	dl[08:15]	1	-	rd[40:47]
1	1	dl[16:23]	1	-	rd[48:55]
1	1	dl[24:31]	1	-	rd[56:63]

Programming Details 4

Introduction

This chapter contains details of several programming functions that are not tied to any specific ASIC chip.

PCI Device Addressing

Each PCI device has an associated address line connected via a resistor to its IDSEL pin for Configuration Space accesses. The following table shows the IDSEL assignments for the PCI devices on the MTX.

Table 4-1. IDSEL Mapping for PCI Devices

Device Number Field	PCI Address Line	IDSEL Connection
0b0_0000	AD31	Raven PCI Host Bridge & MPIC ASIC
0b0_1011	AD11	PCI/ISA Bridge
0b0_1100	AD12	SCSI Device
0b0_1101	AD13	Unused
0b0_1110	AD14	Ethernet Device
0b0_1111	AD15	Unused
0b1_0000	AD16	PMC or 32-bit PCI Slot 1
0b1_0001	AD17	PMC or 32-bit PCI Slot 2
0b1_0010	AD18	32-bit or 64-bit PCI Slot 3

PCI Arbitration

PCI arbitration is performed by the PCI-to-ISA Bridge (PIB) which supports six PCI external PCI masters. The PIB is also a PCI master for ISA DMA functions. The arbitration assignments on the MTX series are as follows:

Table 4-2. PCI Arbitration Assignments

PCI BUS REQUEST	PCI Master(s)
PIB (internal)	PIB
CPU	Raven ASIC
Request 0	PMC/PCI Slot 2
Request 1	PMC/PCI Slot 1
Request 2	Ethernet
Request 3	SCSI
Request 4	PCI Slot 3

Upon power-up, the PIB defaults to a *round-robin* arbitration mode. The relative priority of each request/grant pair can be customized via the PCI Priority Control Register 1. Refer to the W83C553 Data Book, listed in Appendix A, *Related Documentation*, for additional details.

Interrupt Handling

The interrupt architecture of the MTX series SBC is shown in the following figure:

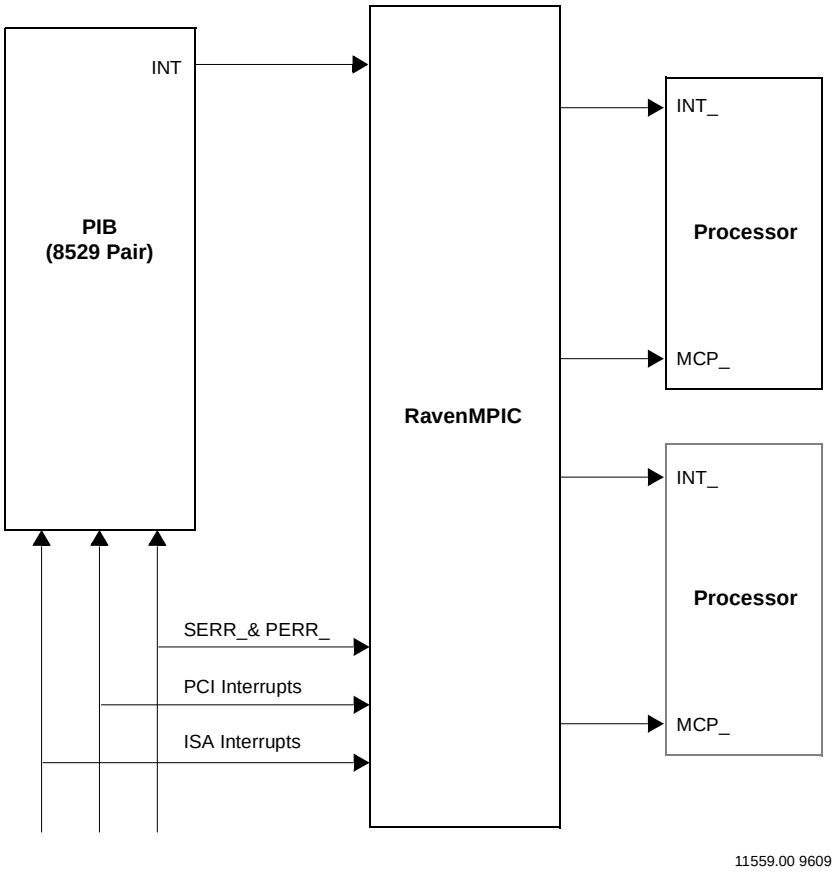


Figure 4-1. MTX Series Interrupt Architecture

Raven MPIC

The Raven ASIC has a built-in interrupt controller that meets the Multi-Processor Interrupt Controller (MPIC) Specification. This MPIC supports up to two processors and 16 external interrupt sources. There are also six other interrupt sources inside the MPIC: two cross-processor interrupts and four timer interrupts. All ISA interrupts go through the 8259 pair in the PIB. The output of the PIB then goes through the MPIC in the Raven. Refer to Chapter 2, *Raven PCI Host Bridge & Multi-Processor Interrupt Controller*, for details on the Raven MPIC. The following table shows the interrupt assignments for the Raven MPIC on the MTX series:

Table 4-3. Raven MPIC Interrupt Assignments

MPIC IRQ	Edge/ Level	Polarity	Interrupt Source	Notes
IRQ0	Level	High	PIB (8259)	1
IRQ1	Edge	Low	Falcon-ECC Error	2
IRQ2	Level	Low	PCI-Ethernet	3
IRQ3	Level	Low	PCI-SCSI	3
IRQ4	Level	Low	Reserved	
IRQ5	Level	Low	Reserved	
IRQ6	Level	Low	Reserved	
IRQ7	Level	Low	Reserved	
IRQ8	Level	Low	Reserved	
IRQ9	Level	Low	PCI/PMC1 INTA#, PCI/PMC2 INTD#,PCI3 INTC#	3
IRQ10	Level	Low	PCI/PMC1 INTB#, PCI/PMC2 INTA#, PCI3 INTD#	
IRQ11	Level	Low	PCI/PMC1 INTC#, PCI/PMC2 INTB, PCI3 INTA#	
IRQ12	Level	Low	PCI/PMC1 INTD#, PCI/PMC2 INTC#, PCI3 INTB#	
IRQ13	Level	Low	Not Used	
IRQ14	Level	Low	Not Used	
IRQ15	N/A	N/A	Not Used	

Notes

1. Interrupt from the PCI/ISA Bridge.
2. Interrupt from the Falcon chipset for a Single and/or Double bit memory error.
3. These interrupts also appear at the PIB for backward compatibility with older MVME1600 and PM603/4 modules.

8259 Interrupts

There are 15 interrupt requests supported by the PIB. These 15 interrupts are ISA-type interrupts that are functionally equivalent to two 82C59 interrupt controllers. Except for IRQ0, IRQ1, IRQ2, IRQ8_, and IRQ13, each of the interrupt lines can be configured for either edge-sensitive mode or level-sensitive mode by programming the appropriate ELCR registers in the PIB.

There is also support for four PCI interrupts, PIRQ3_-PIRQ0_. The PIB has four PIRQ Route Control Registers to allow each of the PCI interrupt lines to be routed to any of eleven ISA interrupt lines (IRQ0, IRQ1, IRQ2, IRQ8_, and IRQ13 are reserved for ISA system interrupts). Since PCI interrupts are defined as level-sensitive, software must program the selected IRQ(s) for level-sensitive mode. Note that more than one PCI interrupt can be routed to the same ISA IRQ line. The PIB can be programmed to handle the PCI interrupts if the Raven MPIC is either not present or not used.

The following figure shows the interrupt structure of the PIB.

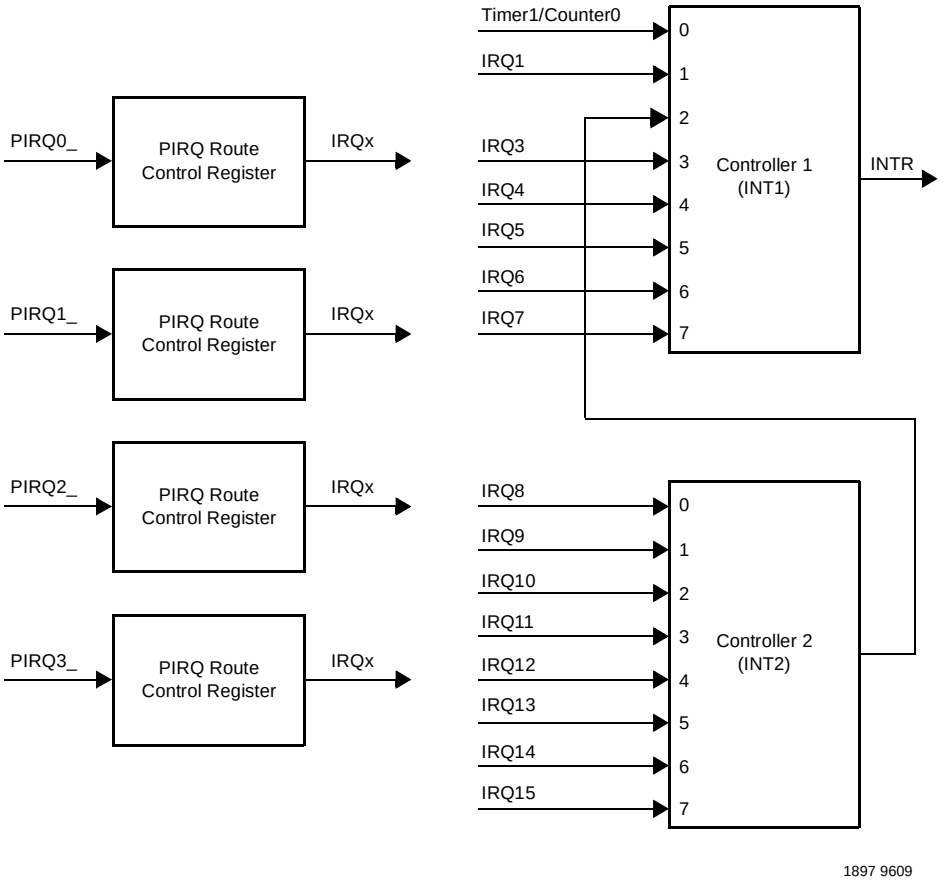


Figure 4-2. PIB Interrupt Handler Block Diagram

The assignments of the PCI and ISA interrupts supported by the PIB are as follows:

Table 4-4. PIB PCI/ISA Interrupt Assignments

PRI	ISA IRQ	PCI IRQ	Controller	Edge /Level	Polarity	Interrupt Source	Notes
1	IRQ0		INT1	Edge	High	Timer 1 / Counter 0	1
2	IRQ1			Edge	High	Keyboard	2
3-10	IRQ2			Edge	High	Cascade Interrupt from INT2	
3	IRQ8_		INT2	Edge	Low	ABORT Switch Interrupt	
4	IRQ9			Level	High	Z8536 CIO	3,4
						Z85230 ESCC	
5	IRQ10	PIRQ0_		Level	Low	PCI-Ethernet Interrupt	3,5,6
6	IRQ11			Level	Low	Not Used	6
7	IRQ12			Edge	High	Mouse	
8	IRQ13			Edge	High	Not Used	6
9	IRQ14	PIRQ2_		Level	Low	PCI-SCSI Interrupt	3,5,6
10	IRQ15	PIRQ3_		Level	Low	PMC Interrupt	3,5,6
11	IRQ3		INT1	Edge	High	COM2 (Async Serial Port 2)	
12	IRQ4			Edge	High	COM1 (Async Serial Port 1)	
13	IRQ5			Level	High	Peripheral Parallel Port Interrupt	
14	IRQ6			Edge	High	Floppy Interrupt	
15	IRQ7			Edge	High	Host Parallel Port Interrupt	

Notes

1. Internally generated by the PIB.
2. Bit 4 of ISA Clock Divisor Register in the PIB must be set to 0 to support external keyboard interrupt (from the ISASIO device).

3. After a reset, all ISA IRQ interrupt lines default to edge-sensitive mode.
4. Interrupts from Z8536 and Z85230 devices are externally wire-ORed. External logic will determine which device to acknowledge during a pseudo IACK cycle. The Z8536 CIO has higher priority than the Z85230 ESCC. This IRQ MUST be programmed for level-sensitive mode.
5. These PCI interrupts are routed to the ISA interrupts by programming the PRIQ Route Control Registers in the PIB. The PCI to ISA interrupt assignments in this table are suggested. Each ISA IRQ to which a PCI interrupt is routed to MUST be programmed for level-sensitive mode. Use this routing for PCI interrupts only when the Raven MPIC is either not present or not used.
6. The Raven MPIC, when present, should be used for these interrupts.

ISA DMA Channels

Refer to Chapter 1, *Board Description and Memory Maps*, for information on the ISA DMA channels.

Exceptions

Sources of Reset

There are five potential sources of reset on the MTX series. They are:

1. Power-On Reset
2. RESET Switch
3. Watchdog Timer Reset via the MK48T59 Timekeeper device
4. Port 92 Register via the PIB

5. I/O Reset via the Clock Divisor Register in the PIB

The following table shows which devices are affected by various reset sources:

Table 4-5. Reset Sources and Devices Affected

Device Affected	Processor (s)	Raven ASIC	Falcon Chipset	PCI Devices	ISA Devices
Power-On	3	3	3	3	3
Reset Switch	3	3	3	3	3
Watchdog (MK48T59)	3	3	3	3	3
Hot Reset (Port 92 Register)	3	3	3	3	3
PCI/ISA Reset (Clock Divisor Register)				3	3

Soft Reset

Software can assert the SRESET# pin of any processor by programming the Processor Init Register of the Raven MPIC appropriately.

Error Notification and Handling

The Raven and Falcon chipset can detect certain hardware errors and can be programmed to report these errors via the Raven MPIC interrupts or Machine Check Interrupt.

Note The TEA* signal is not used at all by the MTX series. The following table summarizes how the hardware errors are handled by the MTX series:

Table 4-6. Error Notification and Handling

Cause	Action
Single-bit ECC	*Store*: Write corrected data to memory *Load*: Present corrected data to the MPC master Generate interrupt via Raven MPIC if so enabled
Double-bit ECC	*Store*: Terminate the bus cycle normally without writing to DRAM *Load*: Present un-corrected data to the MPC master Generate interrupt via Raven MPIC if so enabled Generate Machine Check Interrupt to the Processor(s) if so enabled
MPC Bus Time Out	*Store*: Discard write data and terminate bus cycle normally *Load*: Present undefined data to the MPC master Generate interrupt via Raven MPIC if so enabled Generate Machine Check Interrupt to the Processor(s) if so enabled
PCI Target Abort	*Store*: Discard write data and terminate bus cycle normally *Load*: Return all 1's and terminate bus cycle normally Generate interrupt via Raven MPIC if so enabled Generate Machine Check Interrupt to the Processor(s) if so enabled
PCI Master Abort	*Store*: Discard write data and terminate bus cycle normally *Load*: Return all 1's and terminate bus cycle normally Generate interrupt via Raven MPIC if so enabled Generate Machine Check Interrupt to the Processor(s) if so enabled
PERR# Detected	Generate interrupt via Raven MPIC if so enabled Generate Machine Check Interrupt to the Processor(s) if so enabled
SERR# Detected	Generate interrupt via Raven MPIC if so enabled Generate Machine Check Interrupt to the Processor(s) if so enabled

Endian Issues

The MTX series suppors both little-endian software (for example, NT) and big-endian software (for example, AIX). Because the PowerPC processor is inherently big-endian, PCI is inherently little-endian, and the VMEbus is big-endian, things do get rather confusing. The following figures shows how the MTX series handles the endian issue in big-endian and little-endian modes:

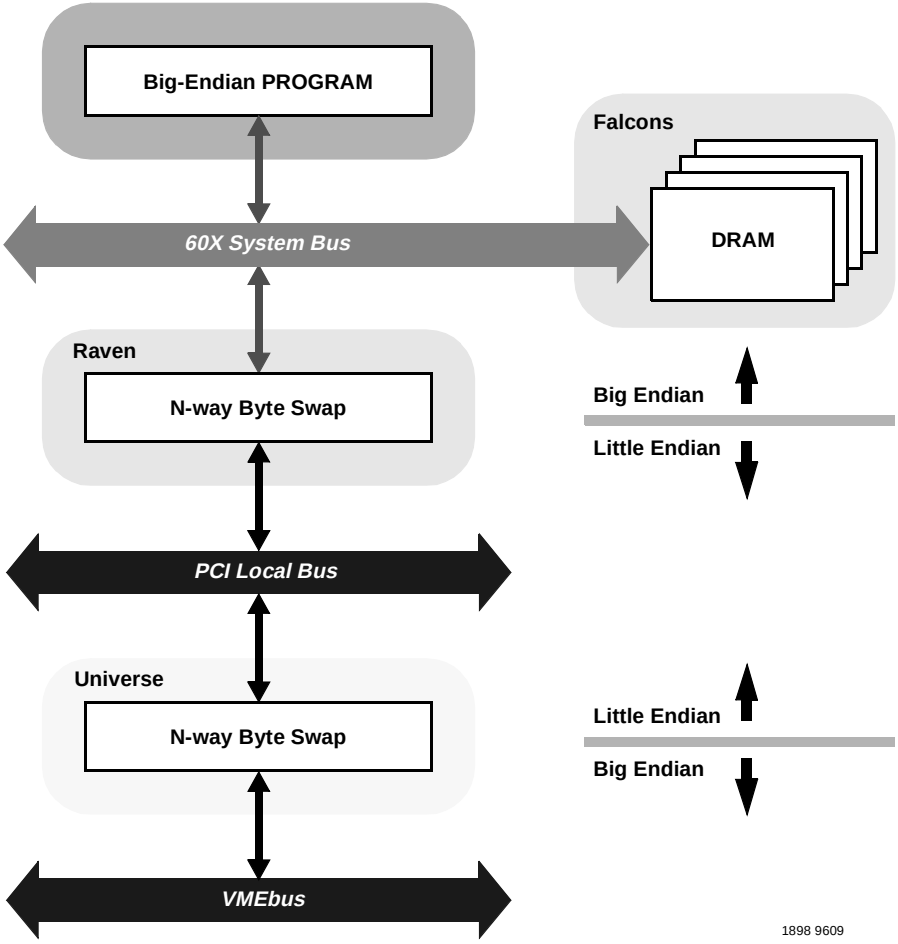


Figure 4-3. Big-Endian Mode

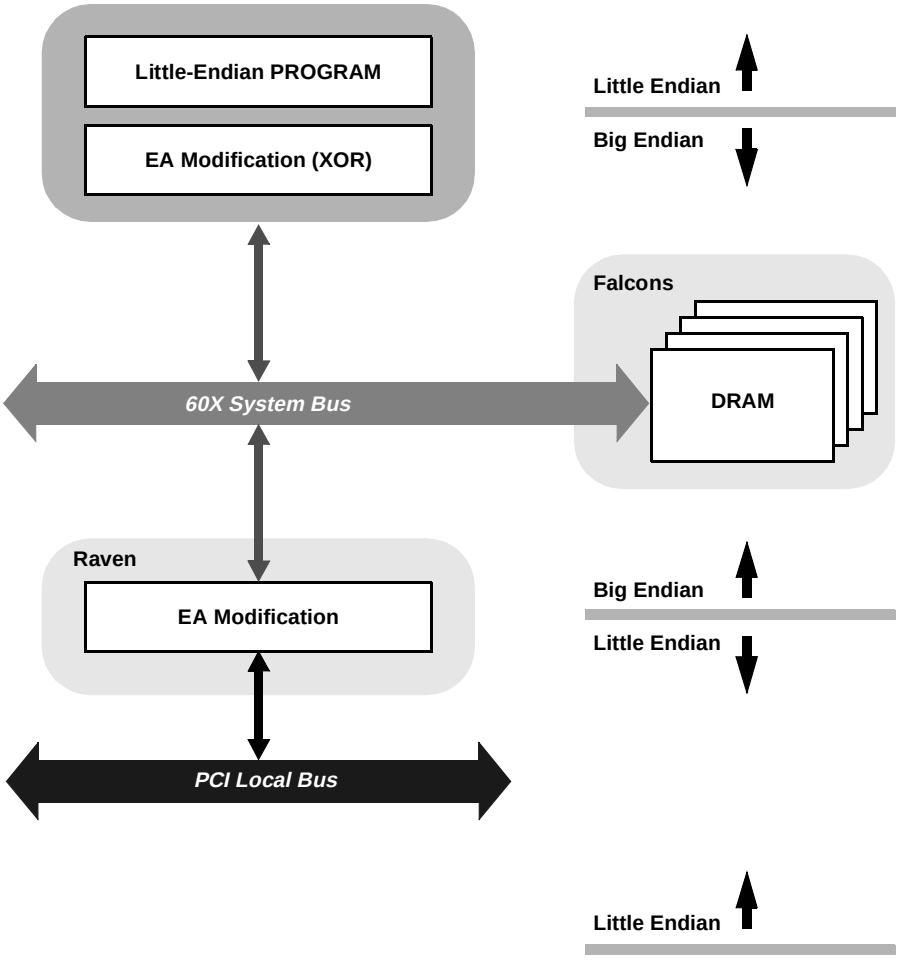


Figure 4-4. Little-Endian Mode

Processor/Memory Domain

The MPC604 processor can operate in both big-endian and little-endian mode. However, it always treats the external processor/memory bus as big-endian by performing *address rearrangement and reordering* when running in little-endian mode.

The MPC registers inside Raven, the registers inside the Falcon chipset, the DRAM, the ROM/FLASH and the system registers always appear as big-endian.

Raven's Involvement

Since PCI is little-endian, the Raven performs byte swapping in both directions (from PCI to memory and from the processor to PCI) to maintain address invariance when it is programmed to operate in big-endian mode with the processor and the memory sub-system.

In little-endian mode, it *reverse-rearranges* the address for PCI-bound accesses and *rearranges* the address for memory-bound accesses (from PCI). In this case, no byte swapping is done.

PCI Domain

The PCI bus is inherently little-endian and all devices connected directly to PCI will operate in little-endian mode, regardless of the mode of operation in the processor's domain.

PCI-SCSI

SCSI is byte stream oriented with the byte having the lowest address in memory being the first one to be transferred regardless of the endian mode. Since address invariance is maintained by the Raven in both little-endian and big-endian mode, there should be no endian issues for the SCSI data. Big-endian software must still however be aware of the byte-swapping effect when accessing the registers of the PCI-SCSI device.

PCI-Ethernet

Ethernet is byte stream oriented with the byte having the lowest address in memory being the first one to be transferred regardless of the endian mode. Since address invariance is maintained by the Raven in both little-endian and big-endian mode, there should be no endian issues for the Ethernet data. Big-endian software must still however be aware of the byte-swapping effect when accessing the registers of the PCI-Ethernet device.

ROM/Flash Initialization

There are two methods used to inject code into the Flash in Bank A: (1) In-circuit programming and (2) Loading it from the ROM/Flash Bank B. For the second method, the hardware must direct the Falcon chipset to map the FFF00000-FFFFFFFF address range to Bank B following a hard reset. Bank A then can be programmed by code from Bank B.

Software can determine the mapping of the FFF00000-FFFFFFFF address range by examining the **rom_b_rv** bit in the Falcon's Rom B Base/Size Register.

Table 4-7. ROM/FLASH Bank Default

rom_b_rv	Default Mapping for FFF00000-FFFFFFFF
0	ROM/FLASH Bank A
1	ROM/FLASH Bank B

Determining PHB Type

The initialization software can determine the PCI Host Bridge (PHB) type by reading its Device ID. To be backward compatible with the older Genesis products which used the MPC105 as the PHB, the Raven defaults the addresses of its **CONADD** register and its **CONDAT** register to 80000CF8 and 80000CFC, respectively.

The alternative method is to read the CPUTYPE from the Old CPU Configuration Register which is located at offset 800h from the PCI I/O Base Address.

Determining CPU Type

The SYID field in the System Configuration Register allows up to 256 types of CPU. This field is always FBh for the MTX.

Related Documentation A

Motorola Computer Group Documents

The Motorola publications listed below are referenced in this manual. You can obtain paper or electronic copies of Motorola Computer Group publications by:

- Contacting your local Motorola sales office
- Visiting Motorola Computer Group's World Wide Web literature site, http://www.motorola.com/computer/literature.

Table A-1. Motorola Computer Group Documents

Document Title	Publication Number
MTX Series Single Board Computer Installation and Use	MTXA/IH
PPCBug Firmware Package User's Manual (Parts 1 and 2)	PPCBUGA1/UM PPCBUGA2/UM
PPCBug Diagonostics User's Manual	PPCDIAA/UM

To obtain the most up-to-date product information in PDF or HTML format, visit http://www.motorola.com/computer/literature.

Manufacturers' Documents

For additional information, refer to the following table for manufacturers' data sheets or user's manuals. As an additional help, a source for the listed document is provided. Please note that, while these sources have been verified, the information is subject to change without notice.

Table A-2. Manufacturers' Documents

Document Title and Source	Publication Number
PowerPC 603™ RISC Microprocessor Technical Summary PowerPC 604™ RISC Microprocessor Technical Summary Literature Distribution Center for Motorola Telephone: 1-800- 441-2447 FAX: (602) 994-6430 or (303) 675-2150 WebSite: http://merchant.hibbertco.com/mtrlext/ E-mail: ldcformotorola@hibbertco.com	MPC603E/D MPC604E/D
PowerPC 603™ RISC Microprocessor User's Manual PowerPC 604™ RISC Microprocessor User's Manual Literature Distribution Center for Motorola Telephone: 1-800- 441-2447 FAX: (602) 994-6430 or (303) 675-2150 Web Site: http://merchant.hibbertco.com/mtrlext/ E-mail: ldcformotorola@hibbertco.com OR IBM Microelectronics PowerPC603/EM603e User Manual PowerPC604e User Manual Web Site: http://www.chips.ibm.com/techlib/products/powerpc/manuals	MPC603EUM/D MPC604EUM/AD G522-0297-00 G522-0330-00
PowerPC™ Microprocessor Family: The Programming Environment for 32-Bit Microprocessors Literature Distribution Center for Motorola Telephone: 1-800- 441-2447 FAX: (602) 994-6430 or (303) 675-2150 http://merchant.hibbertco.com/mtrlext/ E-mail: ldcformotorola@hibbertco.com OR IBM Microelectronics Programming Environment Manual Web Site: http://www.chips.ibm.com/techlib/products/powerpc/manuals	MPCFPE/AD G522-0290-01

Table A-2. Manufacturers' Documents (Continued)

Document Title and Source	Publication Number
MPC2604GA Integrated Secondary Cache for PowerPC Microprocessors (Glance) Data Sheets http://merchant.hibbertco.com/mtrlext/	MPC2604GA
DECchip 21140 PCI Fast Ethernet LAN Controller Hardware Reference Manual http://developer.intel.com/design/network/mature/21140a.htm	21140-AF Revision 1.0
PC87308VUL (Super I/O Enhanced Sidewinder Lite) Floppy Disk Controller, Keyboard Controller, Real-Time Clock, Dual UARTs, IEEE 1284 Parallel Port, and IDE Interface National Semiconductor Corporation Telephone: 1-800-272-9959 http://www.national.com/pf/PC/PC87308.html	PC87308.html
M48T59 CMOS 8K x 8 TIMEKEEPER™ SRAM Data Sheet STMicroelectronics; http://eu.st.com/stonline/index.shtml	M48T59
SYM 53CXX (was NCR 53C8XX) Family PCI-SCSI I/O Processor Data Manual LSI Logic Corporation http://www.lsilogic.com	SYM53C875/875E Data Manual
SCC (Serial Communications Controller) User's Manual (for Z85230 and other Zilog parts) http://www.zilog.com/pdfs/serial/scc_escc_iscc_manual/contents.html	SCC/ESCC User's Manual
Z8536 CIO Counter/Timer and Parallel I/O Unit Product Specification and User's Manual (in Z8000® Family of Products Data Book) http://www.zilog.com/products/zx80dev.html#um	DM10001176
W83C553 Enhanced System I/O Controller with PCI Arbiter (PIB) Winbond Electronics Corporation; http://www.winbond.com.tw/product/	W83C553F
CL-CD1283: IEEE 1284-Compatible Parallel Interface Data Book Cirrus Logic, Inc. (or nearest Sales Office) http://www.cirrus.com	

Table A-2. Manufacturers' Documents (Continued)

Document Title and Source	Publication Number
PCF8584 I^2C Bus Controller; Philips Semiconductor; http://www-us6.semiconductors.com/acrobat/datasheets/PCF8584_4.pdf	PCF8584_4.pdf Dated Oct 21 1997

Related Specifications

For additional information, refer to the following table for related specifications. As an additional help, a source for the listed document is provided. Please note that, while these sources have been verified, the information is subject to change without notice.

Table A-3. Related Specifications

Document Title and Source	Publication Number
ANSI Small Computer System Interface-2 (SCSI-2), Draft Document Global Engineering Documents http://global.ihs.com/index.cfm	X3.131.1990
IEEE - Common Mezzanine Card Specification (CMC) Institute of Electrical and Electronics Engineers, Inc. http://standards.ieee.org/catalog/	P1386 Draft 2.0
IEEE - PCI Mezzanine Card Specification (PMC) Institute of Electrical and Electronics Engineers, Inc. http://standards.ieee.org/catalog/	P1386.1 Draft 2.0
Bidirectional Parallel Port Interface Specification Institute of Electrical and Electronics Engineers, Inc. http://standards.ieee.org/catalog/	IEEE Standard 1284
Peripheral Component Interconnect (PCI) Local Bus Specification, Revision 2.0 PCI Special Interest Group http://www.pcisig.com/	PCI Local Bus Specification

Table A-3. Related Specifications (Continued)

Document Title and Source	Publication Number
PowerPC Microprocessor Common Hardware Reference Platform: A System Architecture (CHRP), Version 1.0 Literature Distribution Center for Motorola Telephone: 1-800- 441-2447 FAX: (602) 994-6430 or (303) 675-2150 http://merchant.hibbertco.com/mtrlext/ E-mail: ldcformotorola@hibbertco.com OR Morgan Kaufmann Publishers, Inc. Telephone: (415) 392-2665 Telephone: 1-800-745-7323 http://www.mkp.com/books_catalog/	ISBN 1-55860-394-8
PowerPC Reference Platform (PRP) Specification,Third Edition, Version 1.0, Volumes I and II; International Business Machines Corporation http://www.ibm.com	MPR-PPC-RPU-02

URLs

The following URLs (uniform resource locators) may provide helpful sources of additional information about this product, related services, and development tools. Please note that, while these URLs have been verified, they are subject to change without notice.

- Motorola Computer Group, http://www.motorola.com/computer
- Motorola Computer Group OEM Services, http://www.motorola.com/computer/support

Index

E

F

G

H

P

R

S

T

U

V

W

Z