

**MVME2600 Series  
Single Board Computer  
Programmer's  
Reference Guide**

V2600A/PG1

## **Notice**

While reasonable efforts have been made to assure the accuracy of this document, Motorola, Inc. assumes no liability resulting from any omissions in this document, or from the use of the information obtained therein. Motorola reserves the right to revise this document and to make changes from time to time in the content hereof without obligation of Motorola to notify any person of such revision or changes.

No part of this material may be reproduced or copied in any tangible medium, or stored in a retrieval system, or transmitted in any form, or by any means, radio, electronic, mechanical, photocopying, recording or facsimile, or otherwise, without the prior written permission of Motorola, Inc.

It is possible that this publication may contain reference to, or information about Motorola products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that Motorola intends to announce such Motorola products, programming, or services in your country.

## **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 (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013.

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

## Preface

The *MVME2600 Series Single Board Computer Programmer's Reference Guide* provides brief board level information, complete memory maps, and detailed ASIC chip information including register bit descriptions for the MVME2600 series Single Board Computers (also called MVME2603 and MVME2604 in this manual). The information contained in this manual applies to the single board computers built from some of the plug-together components listed in the following table.

MVME2603-1121	MVME2603-2121	RAM200-042
MVME2603-1131	MVME2603-2131	RAM200-043
MVME2603-1141	MVME2603-2141	RAM200-044
MVME2603-1191	MVME2603-2191	RAM200-049

MVME2604-1021	MVME2604-1121	MVME2604-2021	MVME2604-2121
MVME2604-1031	MVME2604-1131	MVME2604-2031	MVME2604-2131
MVME2604-1041	MVME2604-1141	MVME2604-2041	MVME2604-2141
MVME2604-1091	MVME2604-1191	MVME2604-2091	MVME2604-2191

This manual is intended for anyone who wants to program these boards in order to design OEM systems, supply additional capability to an existing compatible system, or work in a lab environment for experimental purposes.

A basic knowledge of computers and digital logic is assumed.

To use this manual, you should be familiar with the publications listed in *Appendix A, Related Documentation*.

The following conventions are used in this document:

### **bold**

is used for user input that you type just as it appears. Bold 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.

### courier

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

### <RETURN> or <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, e.g., CTRL-d.

The computer programs stored in the Read Only Memory of this device contain material copyrighted by Motorola Inc., first published 1990, and may be used only under a license such as the License for Computer Programs (Article 14) contained in Motorola's Terms and Conditions of Sale, Rev. 1/79.

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



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

Motorola® and the Motorola symbol are registered trademarks of Motorola, Inc.

PowerStack™, VMEmodule™, and VMEsystem™ are trademarks of Motorola, Inc.

PowerPC™, PowerPC 603™, and PowerPC 604™ are trademarks of IBM Corp, and are used by Motorola, Inc. under license from IBM Corp.

AIX™ is a trademark of IBM Corp.

Timekeeper™ and Zeropower™ are trademarks of Thompson Components.

All other products mentioned in this document are trademarks or registered trademarks of their respective holders.

©Copyright Motorola 1996

All Rights Reserved

Printed in the United States of America

November 1996

## **Safety Summary**

### **Safety Depends On You**

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 violates safety standards of design, manufacture, and intended use of the equipment. Motorola, Inc. assumes no liability for the customer's failure to comply with these requirements.

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. The equipment is supplied with a three-conductor ac power cable. The power cable must be plugged into an approved three-contact electrical outlet. The power jack and mating plug of the power cable meet International Electrotechnical Commission (IEC) safety standards.

#### **Do Not Operate in an Explosive Atmosphere.**

Do not operate the equipment in the presence of flammable gases or fumes. Operation of any electrical equipment in such an environment constitutes a definite safety hazard.

#### **Keep Away From Live Circuits.**

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

#### **Do Not Service or Adjust Alone.**

Do not attempt internal service or adjustment unless another person capable of rendering first aid and resuscitation is present.

#### **Use Caution When Exposing or Handling the CRT.**

Breakage of the Cathode-Ray Tube (CRT) causes a high-velocity scattering of glass fragments (implosion). To prevent CRT implosion, avoid rough handling or jarring of the equipment. Handling of the CRT should be done only by qualified maintenance personnel using approved safety mask and gloves.

#### **Do Not Substitute Parts or Modify Equipment.**

Because of the danger of introducing additional hazards, 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 safety features are maintained.

#### **Dangerous Procedure Warnings.**

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.



**Dangerous voltages, capable of causing death, are present in this equipment. Use extreme caution when handling, testing, and adjusting.**



# Contents

---

Introduction	1-1
Manual Terminology	1-1
Overview	1-3
Feature Summary	1-3
System Block Diagram	1-4
Functional Description	1-6
Overview	1-6
Programming Model	1-7
Memory Maps	1-7
Processor Memory Maps	1-7
PCI Memory Maps	1-13
VMEbus Mapping	1-20
Falcon-Controlled System Registers	1-26
System Configuration Register (SYSCR)	1-27
Memory Configuration Register (MEMCR)	1-29
System External Cache Control Register (SXCCR)	1-30
CPU Control Register	1-32
ISA Local Resource Bus	1-33
W83C553 PIB Registers	1-33
PC87308VUL Super I/O (ISASIO) Strapping	1-33
NVRAM/RTC & Watchdog Timer Registers	1-33
Module Configuration and Status Registers	1-34
CPU Configuration Register	1-35
Base Module Feature Register	1-36
Base Module Status Register (BMSR)	1-37
Seven-Segment Display Register	1-38
VME Registers	1-38
LM/SIG Control Register	1-39
LM/SIG Status Register	1-40
Location Monitor Upper Base Address Register	1-41
Location Monitor Lower Base Address Register	1-42
Semaphore Register 1	1-42
Semaphore Register 2	1-43
VME Geographical Address Register (VGAR)	1-43
Z85230 ESCC and Z8536 CIO Registers and Port Pins	1-44
Z8536/Z85230 Registers	1-44
Z8536 CIO Port Pins	1-45

---

---

ISA DMA Channels	1-48
Introduction	2-1
Overview	2-1
Requirements	2-2
Features	2-2
Block Diagram	2-4
Functional Description	2-5
MPC Bus Interface	2-5
MPC Arbiter	2-5
MPC Map Decoders	2-7
MPC Write Posting	2-8
MPC Master	2-9
MPC Bus Timer	2-10
PCI Interface	2-11
PCI Map Decoders	2-11
PCI Configuration Space	2-12
PCI Write Posting	2-12
PCI Master	2-13
Generating PCI Memory and I/O Cycles	2-14
Generating PCI Configuration Cycles	2-15
Generating PCI Special Cycles	2-16
Generating PCI Interrupt Acknowledge Cycles	2-16
Endian Conversion	2-17
When MPC Devices are Big-Endian	2-17
When MPC Devices are Little Endian	2-18
Cycles Originating From PCI	2-19
Error Handling	2-19
PCI/MPC Contention Handling	2-21
Registers	2-23
MPC Registers	2-23
Vendor ID/Device ID Registers	2-25
Revision ID Register	2-25
General Control-Status/Feature Registers	2-26
MPC Arbiter Control Register	2-29
Prescaler Adjust Register	2-30
MPC Error Enable Register	2-31
MPC Error Status Register	2-34
MPC Error Address Register	2-36
MPC Error Attribute Register - MERAT	2-36
PCI Interrupt Acknowledge Register	2-38
MPC Slave Address (0,1 and 2) Registers	2-39

---



---

- MPC Slave Address (3) Register 2-40
- MPC Slave Offset/ Attribute (0,1 and 2) Registers 2-41
- MPC Slave Offset/ Attribute (3) Registers 2-42
- General Purpose Registers 2-43
- PCI Registers 2-43
  - Vendor ID/ Device ID Registers 2-45
  - PCI Command/ Status Registers 2-45
  - Revision ID/ Class Code Registers 2-47
  - I/O Base Register 2-48
  - Memory Base Register 2-49
  - PCI Slave Address (0,1,2 and 3) Registers 2-50
  - PCI Slave Attribute/ Offset (0,1,2 and 3) Registers 2-51
  - CONFIG\_ADDRESS 2-52
  - PCI I/O CONFIG\_ADDRESS Register 2-52
  - PCI I/O CONFIG\_DATA Register 2-54
- Raven Interrupt Controller Implementation 2-55
  - Introduction 2-55
    - The Raven Interrupt Controller (RavenMPIC) Features 2-55
    - Architecture 2-55
    - CSR's Readability 2-56
    - Interrupt Source Priority 2-56
    - Processor's Current Task Priority 2-56
    - Nesting of Interrupt Events 2-56
    - Spurious Vector Generation 2-57
    - Interprocessor Interrupts (IPI) 2-57
    - 8259 Compatibility 2-57
    - Raven-Detected Errors 2-58
    - Timers 2-58
    - Interrupt Delivery Modes 2-58
  - Block Diagram Description 2-60
    - Program Visible Registers 2-61
    - Interrupt Pending Register (IPR) 2-61
    - Interrupt Selector (IS) 2-61
    - Interrupt Request Register (IRR) 2-62
    - In-Service Register (ISR) 2-62
    - Interrupt Router 2-62
  - MPIC Registers 2-64
    - RavenMPIC Registers 2-64
    - Feature Reporting Register 2-69
    - Global Configuration Register 2-70
    - Vendor Identification Register 2-71
    - Processor Init Register 2-71

---

---

- IPI Vector/Priority Registers 2-72
- Spurious Vector Register 2-73
- Timer Frequency Register 2-73
- Timer Current Count Registers 2-74
- Timer Basecount Registers 2-74
- Timer Vector/Priority Registers 2-75
- Timer Destination Registers 2-76
- External Source Vector/Priority Registers 2-77
- External Source Destination Registers 2-78
- Raven-Detected Errors Vector/Priority Register 2-79
- Raven-Detected Errors Destination Register 2-80
- Interprocessor Interrupt Dispatch Registers 2-80
- Interrupt Task Priority Registers 2-81
- Interrupt Acknowledge Registers 2-82
- End-of-Interrupt Registers 2-82
- Programming Notes 2-83
  - External Interrupt Service 2-83
  - Reset State 2-84
- Operation 2-85
  - Interprocessor Interrupts 2-85
  - Dynamically Changing I/O Interrupt Configuration 2-85
  - EOI Register 2-86
  - Interrupt Acknowledge Register 2-86
  - 8259 Mode 2-86
  - Current Task Priority Level 2-86
- Architectural Notes 2-87
- Introduction 3-1
  - Overview 3-1
  - Bit Ordering Convention 3-1
  - Features 3-1
- Block Diagrams 3-2
- Functional Description 3-6
  - Performance 3-6
    - Four-beat Reads/Writes 3-6
    - Single-beat Reads/Writes 3-7
    - DRAM Speeds 3-7
    - ROM/Flash Speeds 3-11
  - PowerPC 60x Bus Interface 3-12
    - Responding to Address Transfers 3-12
    - Completing Data Transfers 3-12
    - Cache Coherency 3-12

---

---

Cache Coherency Restrictions	3-13
L2 Cache Support	3-13
ECC	3-13
Cycle Types	3-13
Error Reporting	3-14
Error Logging	3-15
DRAM Tester	3-15
ROM/Flash Interface	3-16
Refresh/Scrub	3-20
Blocks A and/or B Present, Blocks C and D Not Present	3-20
Blocks A and/or B Present, Blocks C and/or D present	3-21
DRAM Arbitration	3-22
Chip Defaults	3-22
External Register Set	3-23
CSR Accesses	3-23
Programming Model	3-24
CSR Architecture	3-24
Register Summary	3-29
Detailed Register Bit Descriptions	3-32
Vendor/Device Register	3-33
Revision ID/ General Control Register	3-33
DRAM Attributes Register	3-35
DRAM Base Register	3-37
CLK Frequency Register	3-37
ECC Control Register	3-38
Error Logger Register	3-41
Error_Address Register	3-43
Scrub/Refresh Register	3-43
Refresh/Scrub Address Register	3-44
ROM A Base/Size Register	3-45
ROM B Base/Size Register	3-48
DRAM Tester Control Registers	3-51
32-Bit Counter	3-51
Test SRAM	3-52
Power-Up Reset Status Register 1	3-53
Power-Up Reset Status Register 2	3-53
External Register Set	3-54
Software Considerations	3-55
Parity Checking on the PowerPC Bus	3-55
Programming ROM/Flash Devices	3-55
Writing to the Control Registers	3-55

---

---

- Sizing DRAM 3-56
- ECC Codes 3-59
- Data Paths 3-61
- General Information 4-1
  - Introduction 4-1
  - Product Overview - Features 4-1
- Functional Description 4-2
  - Architectural Overview 4-2
    - VMEbus Interface 4-4
    - PCI Bus Interface 4-5
    - Interrupter and Interrupt Handler 4-6
    - DMA Controller 4-7
- Registers - Universe Control and Status Registers (UCSR) 4-8
  - Universe Register Map 4-9
- Universe Chip Problems after a PCI Reset 4-14
  - Problem Description 4-14
  - Examples 4-16
    - Example 1: MVME2600 Series Board Exhibits Problem 4-16
    - Example 2: MVME3600 Series Board Acts Differently 4-17
    - Example 3: Universe Chip is Checked at Tundra 4-19
- Introduction 5-1
- PCI Arbitration 5-1
- Interrupt Handling 5-2
  - RavenMPIC 5-3
  - 8259 Interrupts 5-4
- ISA DMA Channels 5-7
- Exceptions 5-8
  - Sources of Reset 5-8
  - Soft Reset 5-9
  - Universe Chip Problems after a PCI Reset 5-9
  - Error Notification and Handling 5-10
- Endian Issues 5-11
  - Processor/Memory Domain 5-14
  - Raven's Involvement 5-14
  - PCI Domain 5-14
    - PCI-SCSI 5-14
    - PCI-Ethernet 5-15
    - PCI-Graphics 5-15
  - Universe's Involvement 5-15
  - VMEbus Domain 5-15
- ROM/Flash Initialization 5-16

---

---

Motorola Computer Group Documents A-1  
Manufacturers' Documents A-2  
Related Specifications A-7  
Abbreviations, Acronyms, and Terms to Know GL-1

---

# List of Figures

---

MVME2600 Series System Block Diagram	1-5
VMEbus Master Mapping	1-21
VMEbus Slave Mapping	1-23
Raven Block Diagram	2-4
PCI Spread I/O Cycle Mapping	2-15
Big to Little Endian Data Swap	2-18
RavenMPIC Block Diagram	2-60
Falcon Pair Used with DRAM in a System	3-3
Falcon Internal Data Paths (Simplified)	3-4
Overall DRAM Connections	3-5
Data Path for Reads from the Falcon Internal CSRs	3-24
Data Path for Writes to the Falcon Internal CSRs	3-25
Memory Map for Byte Reads to the CSR	3-26
Memory Map for Byte Writes to the Internal Register Set and Test SRAM	3-27
Memory Map for 4-Byte Reads to the CSR	3-28
Memory Map for 4-Byte Writes to the Internal Register Set and Test SRAM	3-28
PowerPC Data to DRAM Data Correspondence	3-62
Architectural Diagram for the Universe	4-3
UCSR Access Mechanisms	4-8
MVME2600 Series Interrupt Architecture	5-2
PIB Interrupt Handler Block Diagram	5-5
Big-Endian Mode	5-12
Little-Endian Mode	5-13

---

# List of Tables

---

MVME2600 Series Features Summary	1-3
Default Processor Memory Map	1-8
CHRP Memory Map Example	1-9
Raven MPC Register Values for CHRP Memory Map	1-10
PREP Memory Map Example	1-11
Raven MPC Register Values for PREP Memory Map	1-12
PCI CHRP Memory Map	1-13
Raven PCI Register Values for CHRP Memory Map	1-15
Universe PCI Register Values for CHRP Memory Map	1-15
PCI PREP Memory Map	1-17
Raven PCI Register Values for PREP Memory Map	1-18
Universe PCI Register Values for PREP Memory Map	1-19
Universe PCI Register Values for VMEbus Slave Map Example	1-24
VMEbus Slave Map Example	1-25
System Register Summary	1-26
Strap Pins Configuration for the PC87308VUL	1-33
MK48T59/559 Access Registers	1-34
Module Configuration and Status Registers	1-35
VME Registers	1-39
Z8536/Z85230 Access Registers	1-44
Z8536 CIO Port Pins Assignment	1-45
Interpretation of MID3-MID0	1-47
PIB DMA Channel Assignments	1-48
CHRP Compliant Memory Map	2-7
MPC Transfer Types	2-10
PCI Command Codes	2-13
Address Modification for Little Endian Transfers	2-19
Raven MPC Register Map	2-23
Raven PCI Configuration Register Map	2-44
Raven PCI I/O Register Map	2-44
RavenMPIC Register Map	2-65
PowerPC 60x Bus to DRAM Access Timing When Configured for 70ns Page Devices	3-8
PowerPC 60x Bus to DRAM Access Timing When Configured for 60ns Page Devices.	3-9

---

PowerPC 60x Bus to DRAM Access Timing When Configured for 50ns Hyper Devices 3-10

PowerPC 60x Bus to ROM/Flash Access Timing When Configured for 32/64-bit Devices 3-11

PowerPC 60x Bus to ROM/Flash Access Timing When Configured for 8-bit Devices 3-11

Error Reporting 3-14

PowerPC 60x to ROM/Flash Address Mapping with Two 8-bit Devices 3-18

PowerPC 60x Address to ROM/Flash Address Mapping with Two 32-bit or One 64-bit Device(s) 3-19

Register Summary 3-30

ram spd1, ram spd0 and DRAM Type 3-34

Block\_A/B/C/D Configurations 3-36

rtest encodings 3-44

ROM Block A Size Encoding 3-46

rom\_a\_rv and rom\_b\_rv encoding 3-47

Read/Write to ROM/Flash 3-48

ROM Block B Size Encoding 3-50

Sizing Addresses 3-57

PowerPC 60x Address to DRAM Address Mappings 3-58

Syndrome Codes Ordered by Bit in Error 3-59

Single-Bit Errors Ordered by Syndrome Code 3-60

PowerPC Data to DRAM Data Mapping 3-63

Universe Register Map 4-9

PCI Arbitration Assignments 5-1

RavenMPIC Interrupt Assignments 5-3

PIB PCI/ISA Interrupt Assignments 5-6

Reset Sources and Devices Affected 5-9

Error Notification and Handling 5-10

ROM/FLASH Bank Default 5-16



# Board Description and Memory Maps

---

# 1

## Introduction

This manual provides programming information for the MVME2603 and MVME2604 Single Board Computers (SBCs), that are based on the MVME260x-xxxx base board and the RAM200 mezzanine modules. 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.

This chapter briefly describes the board level hardware features of the MVME2600 series Single Board Computers. 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 MVME2600 series that reside in ASICs are covered in the chapters on those ASICs. Chapter 2 covers the Raven chip, Chapter 3 covers the Falcon chip set, Chapter 4 covers the Universe chip, and Chapter 5 covers certain programming features, such as interrupts and exceptions. Appendix A lists all related documentation.

## 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 MPC60x 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 Chapter 5 for *Endian Issues*, which covers which parts of the MVME2600 series use *big-endian* byte ordering, and which use *small-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.

## Overview

The MVME2600 series SBC family, hereafter sometimes referred to simply as the MVME2600 or the V2600 series, provides many standard features required by a computer system: SCSI, Ethernet interface, keyboard interface, mouse interface, sync and async serial ports, parallel port, boot Flash, and up to 256MB of ECC DRAM.

## Feature Summary

There are many models based on the MVME2600 series architecture. The following table summarizes the major features of the MVME2600 series:

**Table 1-1. MVME2600 Series Features Summary**

Feature	Description
Processors	Single Supports BGA processors only: MPC603, MPC604. Bus Clock Frequencies up to 66MHz
L2 Cache	Build-option for 256KB Look-aside L2 Cache
Flash	4MB or 8MB (64-bit wide), with socketed 1MB (16-bit wide)
DRAM	16MB to 256MB, ECC Protected (Single-bit Correction, Double-bit Detection) Two-way Interleaved
NVRAM	8KB

**Table 1-1. MVME2600 Series Features Summary (Continued)**

Feature	Description
RTC	MK48T59/559 Device
Peripheral Support	Two async serial ports Two sync/async serial ports One (IEEE1284, or printer) Parallel Port 8-bit or 16-bit single-ended SCSI interface AUI or 10Base-T/100Base-TX Ethernet interface NO Graphics Interface on MVME2600 series One PS/2 Keyboard and one PS/2 Mouse One PS/2 Floppy Port
VME Interface	32-bit Address/64-bit Data PCI A32/A24/A16, D64 (MBLT)/D32/D16/D08 Master and Slave Programmable Interrupter & Interrupt Handler Full System Controller Functions Programmable DMA Controller with link list support Location Monitor
PMC Slots	One 32/64-bit Slot
Miscellaneous	RESET/ABORT Switch Status LEDs

## System Block Diagram

The MVME2600 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, VME, 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 MVME2600 series is shown below:

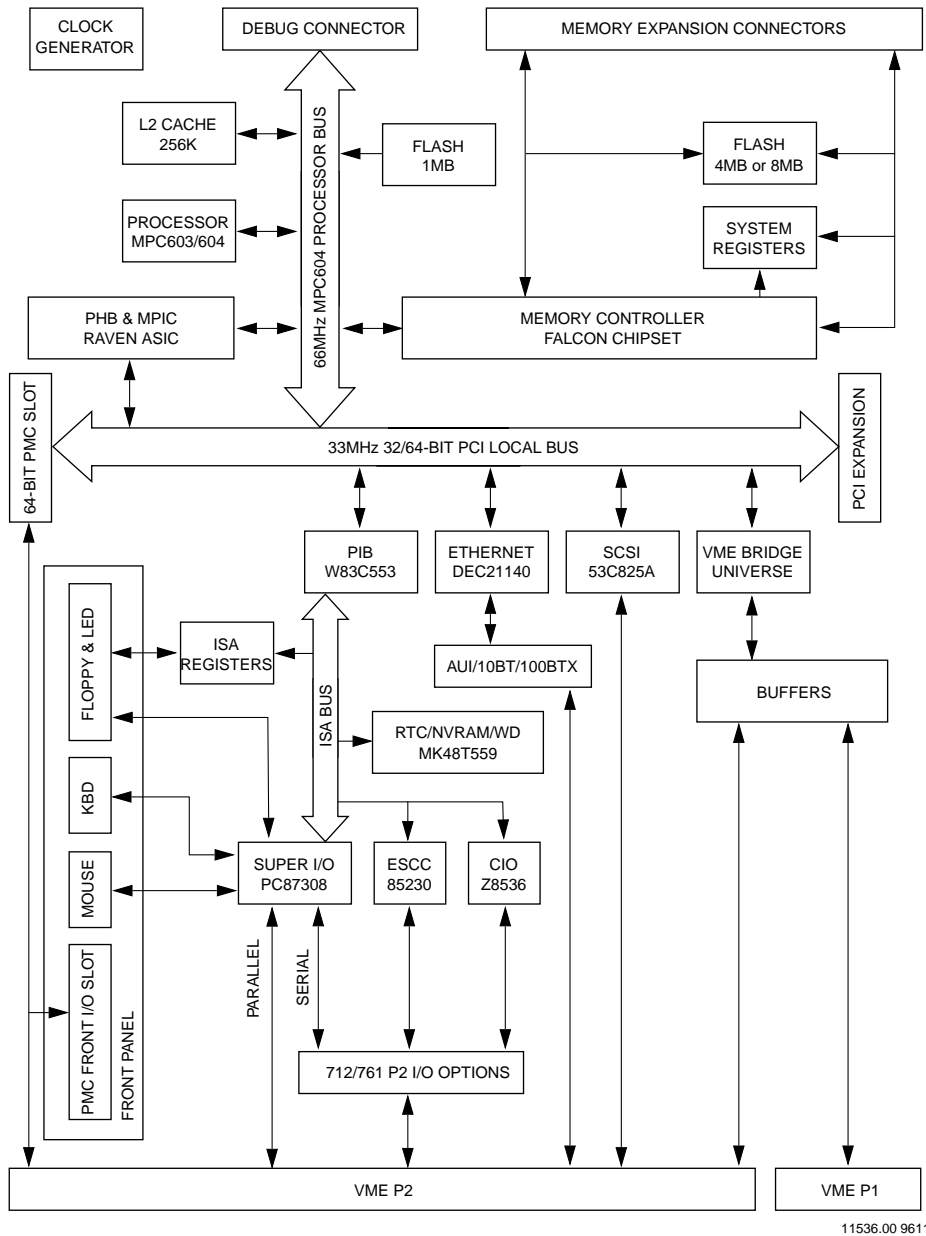


Figure 1-1. MVME2600 Series System Block Diagram

11536.00 9611

# Functional Description

## Overview

The MVME2600 series is a family of single-slot SBCs. It consists of the MPC603/604 processor, the Raven PCI Bridge & Interrupt Controller, the ECC Memory Controller Falcon chipset, 5MB or 9MB of Flash memory, 16MB to 256MB of ECC-protected DRAM, and a rich set of features of I/O peripherals.

I/O peripheral devices on the PCI bus are: SCSI chip, Ethernet chip, Universe VMEbus interface ASIC, and one PMC slot. Functions provided from the ISA bus are: a P1284/Parallel port, two asynchronous serial ports, two sync/asyn serial ports, a real-time clock, and counters/timers.

The MVME2600 series board interfaces to the VMEbus via the P1 and P2 connectors, which use the new 5-row 160-pin connectors as specified in the proposed VME64 Extension Standard. It also draws +5V, +12V, and -12V power from the VMEbus backplane through these two connectors. 3.3V supply is regulated onboard from the +5V power.

Front panel connectors on the MVME2600 series board include: a 6-pin circular DIN connector for the keyboard interface, a 6-pin circular DIN connector for the mouse interface, and a 50-pin connector for the floppy and status LEDs. All signals for the serial ports, the P1284/Printer port, the SCSI interface, and the Ethernet interface are routed to P2.

There are two P2 I/O options supported by the MVME2600 series: the MVME712M mode and the MVME761 mode. The MVME761 mode provides the enhanced P1284 parallel port interface and the full synchronous support for Serial Ports 3 and 4. The MVME712M version provides backward compatibility with previous SBCs. In either mode, 16-bit SCSI capability can only be used by systems with 5-row DIN support because the additional 8 bits of SCSI data lines reside on row Z of P2.

The MVME2600 series contains one IEEE1386.1 PCI Mezzanine Card (PMC) slot. This PMC slot is 64-bit capable and supports both front and rear I/O. Pins 1 through 30 of the PMC connector J14 are routed to pins D1 through D30 of the 5-row DIN P2 connector. J14 pin 31 is connected to P2 pin Z29, and J14 pin 32 is connected to P2 pin Z31.

Additional PCI expansion is supported with a 114-pin Mictor connector. This connection allows stacking of a carrier board to increase the I/O capability, such as a dual-PMC carrier board.

## Programming Model

### Memory Maps

The following sections describe the memory maps for the MVME2600 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	
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 one Mbyte 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	3,4,8
FD00 0000	FDFE FFFF	16M	Zero-Based PCI/ISA Memory Space (mapped to 00000000 to 00FFFFFF)	3,8
FE00 0000	FE7F FFFF	8M	Zero-Based PCI/ISA I/O Space (mapped to 00000000 to 007FFFFFF)	3,5,8
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	9
FF00 0000	FF7F FFFF	8M	ROM/FLASH Bank A	1,7
FF80 0000	FF8F FFFF	1M	ROM/FLASH Bank B	1,7
FF50 0000	FFEF FFFF	6M	Reserved	
FFF0 0000	FFFF FFFF	1M	ROM/FLASH Bank A or Bank B	7

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 actual size of each ROM/FLASH bank may vary.
7. The first one Mbyte 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.
8. This range can be mapped to the VMEbus by programming the Universe ASIC accordingly. The map shown is the recommended setting which uses the Special PCI Slave Image and two of the four programmable PCI Slave Images.
9. 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 MVME2600 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 - 3FFFF FFFF	2
C000 0000	FCFF FFFF	1G - 48M	Zero-Based PCI/ISA Memory Space: 0000 0000 - 3CFFFFFF	2, 5
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	6
FF00 0000	FF7F FFFF	8M	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 1 Mbyte 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. This range can be mapped to the VMEbus by programming the Universe ASIC accordingly.
6. 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
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 CONFIG\_ADD and CONFIG\_DAT registers. These two registers are implemented by the Raven ASIC. In the CHRP memory map example, the CONFIG\_ADD and CONFIG\_DAT registers are located at 0xFE000CF8 and 0xFE000CFC, respectively. With the PREP memory map, the CONFIG\_ADD register and the CONFIG\_DAT register are located at 0x80000CF8 and 0x80000CFC, respectively.

## PCI Memory Maps

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

### Default PCI Memory Map

After a reset, the Raven ASIC and the Universe ASIC turn 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 MVME2600 series that is CHRP-compatible from the point of view of the PCI local bus.

**Table 1-7. PCI CHRP Memory Map**

PCI Address		Size	Definition	Notes
Start	End			
0000 0000	top_dram	dram_size	Onboard ECC DRAM	1
4000 0000	FFFF FFFF	3G - 256M	VMEbus A32/D32 (Super/Program)	3
F000 0000	F7FF FFFF	128M	VMEbus A32/D16 (Super/Program)	3
F800 0000	F8FE FFFF	16M - 64K	VMEbus A24/D16 (Super/Program)	4
F8FF 0000	F8FF FFFF	64K	VMEbus A16/D16 (Super/Program)	4
F900 0000	F9FE FFFF	16M - 64K	VMEbus A24/D32 (Super/Data)	4
F9FF 0000	F9FF FFFF	64K	VMEbus A16/D32 (Super/Data)	4
FA00 0000	FAFE FFFF	16M - 64K	VMEbus A24/D16 (User/Program)	4
FAFF 0000	FAFF FFFF	64K	VMEbus A16/D16 (User/Program)	4
FB00 0000	FBFE FFFF	16M - 64K	VMEbus A24/D32 (User/Data)	4
FBFF 0000	FBFF FFFF	64K	VMEbus A16/D32 (User/Data)	4

**Table 1-7. PCI CHRP Memory Map (Continued)**

PCI Address		Size	Definition	Notes
Start	End			
FC00 0000	FC03 FFFF	256K	RavenMPIC	1
FC04 0000	FCFF FFFF	16M - 256K	PCI Memory Space	
FD00 0000	FDFE FFFF	16M	PCI Memory Space or System Memory Alias Space (mapped to 00000000 to 00FFFFFF)	1
FE00 0000	FFFF FFFF	48M	Reserved	

## Notes:

1. Programmable via the Raven's PCI Configuration registers.
2. To enabled the CHRP "io-hole", program the Raven to ignore the 0x000A0000 - 0x000FFFFFFF address range.
3. Programmable mapping via the four PCI Slave Images in the Universe ASIC.
4. Programmable mapping via the Special Slave Image (SLSI) in the Universe ASIC.

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 FDFE
\$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

The next table shows the programmed values for the associated Universe PCI registers for the PCI CHRP memory map.

**Table 1-9. Universe PCI Register Values for CHRP Memory Map**

Configuration Address Offset	Configuration Register Name	Register Value
\$100	LSI0_CTL	C082 5100
\$104	LSI0_BS	4000 0000
\$108	LSI0_BD	F000 0000
\$10C	LSI0_TO	XXXX 0000
\$114	LSI1_CTL	C042 5100
\$118	LSI1_BS	F000 0000

**Table 1-9. Universe PCI Register Values for CHRP Memory Map (Continued)**

Configuration Address Offset	Configuration Register Name	Register Value
\$11C	LSI1_BD	F800 0000
\$120	LSI1_TO	XXXX 0000
\$128	LSI2_CTL	0000 0000
\$12C	LSI2_BS	XXXX XXXX
\$130	LSI2_BD	XXXX XXXX
\$134	LSI2_TO	XXXX XXXX
\$13C	LSI3_CTL	0000 0000
\$140	LSI3_BS	XXXX XXXX
\$144	LSI3_BD	XXXX XXXX
\$148	LSI3_TO	XXXX XXXX
\$188	SLSI	C0A053F8



## PCI PREP Memory Map

The following table shows a PCI memory map of the MVME2600 series that is PREP-compatible from the point of view of the PCI local bus.

**Table 1-10. PCI PREP Memory Map**

PCI Address		Size	Definition	Notes
Start	End			
0000 0000	00FF FFFF	16M	PCI/ISA Memory Space	
0100 0000	2FFF FFFF	752M	VMEbus A32/D32 (Super/Program)	3
3000 0000	37FF FFFF	128M	VMEbus A32/D16 (Super/Program)	3
3800 0000	38FE FFFF	16M - 64K	VMEbus A24/D16 (Super/Program)	4
38FF 0000	38FF FFFF	64K	VMEbus A16/D16 (Super/Program)	4
3900 0000	39FE FFFF	16M - 64K	VMEbus A24/D32 (Super/Data)	4
39FF 0000	39FF FFFF	64K	VMEbus A16/D32 (Super/Data)	4
3A00 0000	3AFE FFFF	16M - 64K	VMEbus A24/D16 (User/Program)	4
3AFF 0000	3AFF FFFF	64K	VMEbus A16/D26 (User/Program)	4
3B00 0000	3BFE FFFF	16M - 64K	VMEbus A24/D32 (User/Data)	4
3BFF 0000	3BFF FFFF	64K	VMEbus A16/D32 (User/Data)	4
3C00 0000	7FFF FFFF	1G + 64M	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.
2. To enabled the CHRP "io-hole", program the Raven to ignore the 0x000A0000 - 0x000FFFFFFF address range.

3. Programmable mapping via the four PCI Slave Images in the Universe ASIC.
4. Programmable mapping via the Special Slave Image (SLSI) in the Universe ASIC.

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

**Table 1-11. 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

The next table shows the programmed values for the associated Universe PCI registers for the PCI PREP memory map.

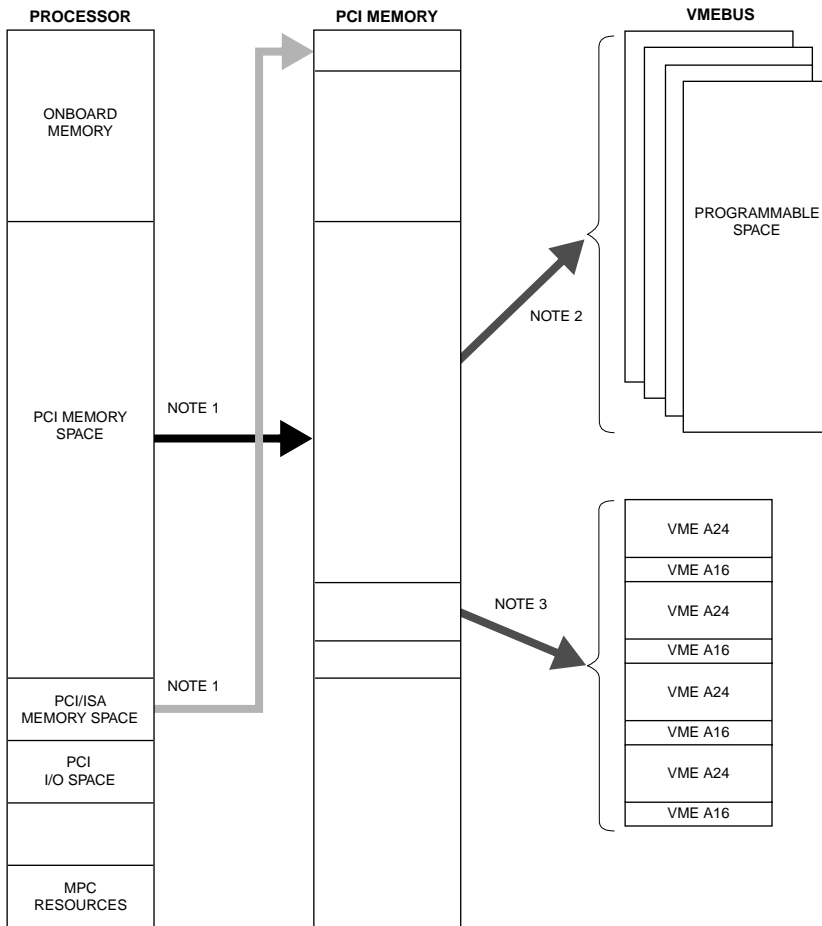
**Table 1-12. Universe PCI Register Values for PREP Memory Map**

Configuration Address Offset	Configuration Register Name	Register Value
\$100	LSI0_CTL	C082 5100
\$104	LSI0_BS	0100 0000
\$108	LSI0_BD	3000 0000
\$10C	LSI0_TO	XXXX 0000
\$114	LSI1_CTL	C042 5100
\$118	LSI1_BS	3000 0000
\$11C	LSI1_BD	3800 0000
\$120	LSI1_TO	XXXX 0000
\$128	LSI2_CTL	0000 0000
\$12C	LSI2_BS	XXXX XXXX
\$130	LSI2_BD	XXXX XXXX
\$134	LSI2_TO	XXXX XXXX
\$13C	LSI3_CTL	0000 0000
\$140	LSI3_BS	XXXX XXXX
\$144	LSI3_BD	XXXX XXXX
\$148	LSI3_TO	XXXX XXXX
\$188	SLSI	C0A05338

## VMEbus Mapping

### VMEbus Master Map

The processor can access any address range in the VMEbus with the help from the address translation capabilities of the Universe ASIC. The recommended mapping is shown in the *Processor Memory Map* section. The following figure illustrates how the VMEbus master mapping is accomplished.



11553.00 9609

**Figure 1-2. VMEbus Master Mapping**

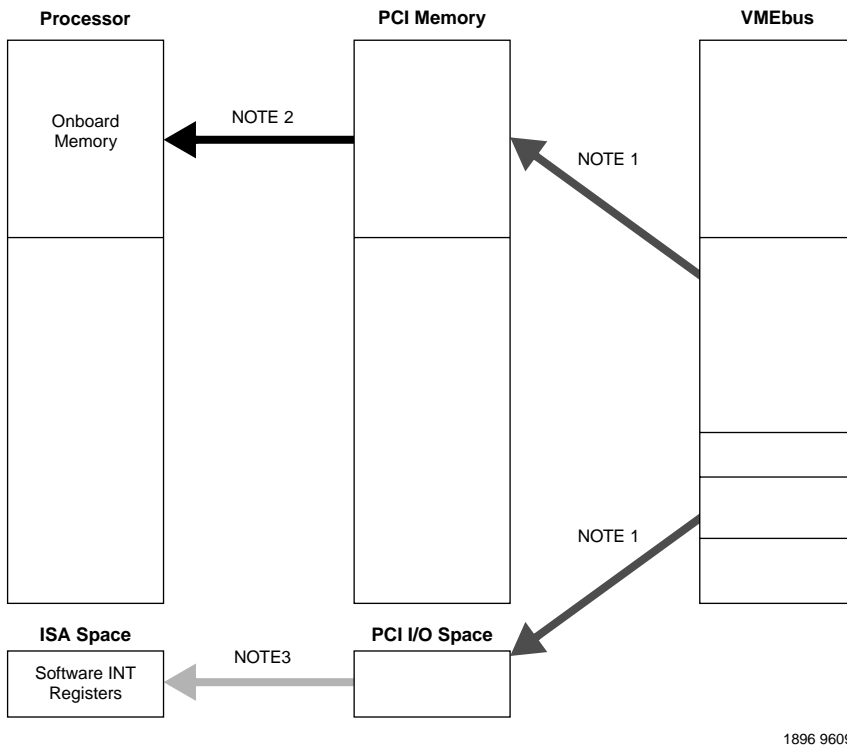
Notes:

1. Programmable mapping done by the Raven ASIC.

2. Programmable mapping via the four PCI Slave Images in the Universe ASIC.
3. Programmable mapping via the Special Slave Image (SLSI) in the Universe ASIC.

### **VMEbus Slave Map**

The four programmable VME Slave Images in the Universe ASIC allow other VMEbus masters to get to any devices on the MVME2600 series. The combination of the four Universe VME Slave Images and the four Raven PCI Slave Decoders offers a lot of flexibility for mapping the system resources as seen from the VMEbus. In most applications, the VMEbus only needs to see the system memory and, perhaps, the software interrupt registers (SIR1 and SIR2 registers). An example of the VMEbus slave map is shown below:



1896 9609

**Figure 1-3. VMEbus Slave Mapping**

Notes:

1. Programmable mapping via the four VME Slave Images in the Universe ASIC.
2. Programmable mapping via PCI Slave Images in the Raven ASIC.
3. Fixed mapping via the PIB device.

The following table shows the programmed values for the associated Universe registers for the VMEbus slave function.

**Table 1-13. Universe PCI Register Values for VMEbus Slave Map Example**

Configuration Address Offset	Configuration Register Name	Register Value (CHRP)	Register Value (PREP)
\$F00	VS10_CTL	C0F2 0001	C0F2 0001
\$F04	VS10_BS	4000 0000	4000 0000
\$F08	VS10_BD	4000 1000	4000 1000
\$F0C	VS10_TO	C000 1000	C000 1000
\$F14	VS11_CTL	E0F2 00C0	E0F2 00C0
\$F18	VS11_BS	1000 0000	1000 0000
\$F1C	VS11_BD	2000 0000	2000 0000
\$F20	VS11_TO	F000 0000	7000 0000
\$F28	VS12_CTL	0000 0000	0000 0000
\$F2C	VS12_BS	XXXX XXXX	XXXX XXXX
\$F30	VS12_BD	XXXX XXXX	XXXX XXXX
\$F34	VS12_TO	XXXX XXXX	XXXX XXXX
\$F3C	VS13_CTL	0000 0000	0000 0000
\$F40	VS13_BS	XXXX XXXX	XXXX XXXX
\$F44	VS13_BD	XXXX XXXX	XXXX XXXX
\$F48	VS13_TO	XXXX XXXX	XXXX XXXX

The above register values yield the following VMEbus slave map:



**Table 1-14. VMEbus Slave Map Example**

VMEbus Address		Size	CHRP Map	PREP Map
Range	Mode			
4000 0000 - 4000 0FFF	A32 U/S/P/D D08/16/32	4K	PCI/ISA I/O Space: 0000 1000 - 0000 1FFF	PCI/ISA I/O Space: 0000 1000 - 0000 1FFF
1000 0000 - 1FFF FFFF	A32 U/S/P/D D08/16/32/64 RMW	256M	PCI/ISA Memory Space (On-board DRAM) 0000 0000 - 0FFF FFFF	PCI/ISA Memory Space (On-board DRAM) 8000 0000 - 8FFF FFFF

## Falcon-Controlled System Registers

The Falcon chipset latches the states of the DRAM data lines onto the PR\_STAT1 and PR\_STAT2 registers. The MVME2600 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 utilized by the MVME2600 series to provide the system control registers.

**Table 1-15. 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	<i>System Configuration Register (Upper Falcon's PR_STAT1)</i>																															
FEF80404	<i>Memory Configuration Register (Lower Falcon's PR_STAT1)</i>																															
FEF88000	<u>System External Cache Control Register</u>																															
FEF88300	<u>CPU Control Register</u>																															

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:

REG	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				POSTAT			P1STAT													
OPER	READ ONLY																															
RESET	\$FE							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 MVME2600 series, this field returns a value of \$FE.

**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	50MHz	25MHz
0b1110	60MHz	30MHz
0b1111	66.66MHz	33.33MHz

**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	1M
0b1101	512K
0b1110	256K
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	1M
0b0101	YES	512K
0b0110	YES	256K
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 to provide some information about the system memory. Configuration is accomplished with external pull-down resistors. This 32-bit read-only Register is defined as follows:

REG	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																
OPER																																
RESET	1	1	1	X	1	1	X	X	1	X	X	X	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	

**M\_FREF** Block A/B/C/D Fast Refresh. When this bit is set, it indicates that a DRAM block requires faster refresh rate. If any of the four blocks requires faster refresh rate then the **ram ref** control bit should be set.

**M\_SPD[0:1]** Memory Speed. This field relays the memory speed information as follows:

M_SPD[0:1]	DRAM Speed	DRAM Type
0b00	70ns	Past Page
0b01	60ns	Fast Page
0b10	Reserved	Reserved
0b11	50ns	EDO

These two bits reflect the combined status of the four blocks of DRAM. Initialization software uses this information to program the **ram\_spd0** and **ram\_spd1** control bits in the Falcon's Chip Revision Register.

**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 16-bit wide FLASH with 16K Bottom Boot Block
0b111	Unknown type (i.e. 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.

### 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:

REG	System External Cache Control Register - \$FEF88000							
BIT	0	1	2	3	4	5	6	7
FIELD	SXC_DIS_	SXC_RST_	SXC_ML_	SXC_FLSH_				
OPER	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. *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 become MPC bus master.*

**SXC\_RST\_**System External Cache Reset. When this bit is cleared, it invalidates all tags and holds the cache in a reset condition. *There is a bug in Glance - It really does not hold the chip in a reset condition. The tag invalidate still works okay though.*

**SXC\_MI\_** System External Cache Miss Inhibit. When this bit is cleared, it prevents line fills on cache misses.



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.

## 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:

REG	CPU Control Register - \$FEF88300							
BIT	0	1	2	3	4	5	6	7
FIELD		LEMODE	P1_TBEN	P0_TBEN				
OPER	R	R	R/W	R/W	R	R	R	R
RESET	X	0	1	1	X	X	X	X

**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 actually accessible from the PCI bus. Refer to the W83C553 Data Book for details.

## PC87308VUL Super I/O (ISASIO) Strapping

The PC87308VUL Super I/O (ISASIO) provides the following functions to the MVME2600 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 1-16. 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 \$002E.
SELCS	1 - CS0# on CS0# pin.

## NVRAM/RTC & Watchdog Timer Registers

The MK48T59/559 provides the MVME2600 series with 8K of non-volatile SRAM, a time-of-day clock, and a watchdog timer. Accesses to the MK48T59559 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 1-17. 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 for additional details and programming information.

## Module Configuration and Status Registers

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

**Table 1-18. 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 MVME2600 series to provide some backward compatibility with older MVME1600 products. 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.

REG	Old CPU Configuration Register - \$FE000800							
BIT	SD7	SD6	SD5	SD4	SD3	SD2	SD1	SD0
FIELD	CPUTYPE							
OPER	R				R			
RESET	\$E				\$F			

**CPUTYPE** CPU Type. This field will always read as \$E for the MVME2600 series. 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 Genesis Single Board Computer. This read-only register is located at ISA I/O address x0802.

REG	Base Module Feature Register - Offset \$0802							
BIT	SD7	SD6	SD5	SD4	SD3	SD2	SD1	SD0
FIELD		SCCP_	PMC2P_	PMC1P_	VMEP_	GFXP_	LANP_	SCSIP_
OPER	R	R	R	R	R	R	R	R
RESET	X	X	X	X	X	X	X	X

- 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.
- PMC2P\_** PMC/PCIX Slot 2 Present. If set, there is no PMC/PCIX device installed in the PMC/PCIX Slot 2. If cleared, the PMC/PCIX Slot 2 contains a PCI Mezzanine Card or a PCI device.
- PMC1P\_** PMC Slot 1 Present. If set, there is no PCI Mezzanine Card installed in the PMC Slot 1. If cleared, the PMC Slot 1 contains a PMC.
- VMEP\_** VMEbus Present. If set, there is no VMEbus interface. If cleared, VMEbus interface is supported.
- GFXP\_** Graphics Present. If set, there is no on-board Graphics interface. If cleared, there is an on-board graphics capability.
- 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.

REG	Base Module Status Register - Offset \$0803							
BIT	SD7	SD6	SD5	SD4	SD3	SD2	SD1	SD0
FIELD	BASE_TYPE							
OPER	R							
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 \$F9	Reserved
\$FA	Reserved -- Special with MVME712M I/O and 100BaseT4 on row Z
\$FB	MVME2600 with MVME712M I/O
\$FC	MVME2600 with MVME761 I/O
\$FD	MVME3600 with MVME712M I/O
\$FE	MVME3600 with MVME761 I/O
\$FF	MVME1600-001 or MVME1600-011

## 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.

REG	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]			
OPER	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.

## VME Registers

The following registers provide the following functions for the VMEbus interface: a software interrupt capability, a location monitor function, and a geographical address status. For these registers to be accessible from the VMEbus, the Universe ASIC must be programmed to map the VMEbus Slave Image 0 into the appropriate PCI I/O address range. Refer to the VMEbus Slave Map section for additional details. The following table shows the registers provided for various VME functions:

**Table 1-19. VME Registers**

PCI I/O Address	Function
0000 1000	SIG/LM Control Register
0000 1001	SIG/LM Status Register
0000 1002	VMEbus Location Monitor Upper Base Address
0000 1003	VMEbus Location Monitor Lower Base Address
0000 1004	VMEbus Semaphore Register 1
0000 1005	VMEbus Semaphore Register 2
0000 1006	VMEbus Geographical Address Status

These registers are described in the following sub-sections.

### LM/SIG Control Register

The LM/SIG Control Register is an 8-bit register located at ISA I/O address x1000. This register provides a method to generate software interrupts. The Universe ASIC is programmed so that this register can be accessed from the VMEbus to generate software interrupts to the processor(s).

REG	LM/SIG Control Register - Offset \$1000							
BIT	SD7	SD6	SD5	SD4	SD3	SD2	SD1	SD0
FIELD	SET SIG1	SET SIG0	SET LM1	SET LM0	CLR SIG1	CLR SIG0	CLR LM1	CLR LM0
OPER	WRITE-ONLY							
RESET	0	0	0	0	0	0	0	0

**SET\_SIG1** Writing a 1 to this bit will set the SIG1 status bit.

**SET\_SIG0** Writing a 1 to this bit will set the SIG0 status bit.

**SET\_LM1** Writing a 1 to this bit will set the LM1 status bit.

**SET\_LM0** Writing a 1 to this bit will set the LM0 status bit.

**CLR\_SIG1** Writing a 1 to this bit will clear the SIG1 status bit.

**CLR\_SIG0** Writing a 1 to this bit will clear the SIG0 status bit.

**CLR\_LM1** Writing a 1 to this bit will clear the LM1 status bit.

**CLR\_LM0** Writing a 1 to this bit will clear the LM0 status bit.

### LM/SIG Status Register

The LM/SIG Status Register is an 8-bit register located at ISA I/O address x1001. This register, in conjunction with the LM/SIG Control Register, provides a method to generate interrupts. The Universe ASIC is programmed so that this register can be accessed from the VMEbus to provide a capability to generate software interrupts to the onboard processor(s) from the VMEbus.

REG	LM/SIG Status Register - Offset \$1001							
<b>BIT</b>	SD7	SD6	SD5	SD4	SD3	SD2	SD1	SD0
<b>FIELD</b>	EN SIG1	EN SIG0	EN LM1	EN LM0	SIG1	SIG0	LM1	LM0
<b>OPER</b>	R/W				READ-ONLY			
<b>RESET</b>	0	0	0	0	0	0	0	0

**EN\_SIG1** When the EN\_SIG1 bit is set, a LM/SIG Interrupt 1 is generated if the SIG1 bit is asserted.

**EN\_SIG0** When the EN\_SIG0 bit is set, a LM/SIG Interrupt 0 is generated if the SIG0 bit is asserted.

**EN\_LM1** When the EN\_LM1 bit is set, a LM/SIG Interrupt 1 is generated and the LM1 bit is asserted.

**EN\_LM0** When the EN\_LM0 bit is set, a LM/SIG Interrupt 0 is generated and the LM0 bit is asserted.



- SIG1** SIG1 status bit. This bit can only be set by the SET\_LM1 control bit. It can only be cleared by a reset or by writing a 1 to the CLR\_LM1 control bit.
- SIG0** SIG0 status bit. This bit can only be set by the SET\_LM0 control bit. It can only be cleared by a reset or by writing a 1 to the CLR\_LM0 control bit.
- LM1** LM1 status bit. This bit can be set by either the location monitor function or the SET\_LM1 control bit. LM1 correspond to offset 3 from the location monitor base address. This bit can only be cleared by a reset or by writing a 1 to the CLR\_LM1 control bit.
- LM0** LM0 status bit. This bit can be set by either the location monitor function or the SET\_LM0 control bit. LM0 correspond to offset 1 from the location monitor base address. This bit can only be cleared by a reset or by writing a 1 to the CLR\_LM0 control bit.

### Location Monitor Upper Base Address Register

The Location Monitor Upper Base Address Register is an 8-bit register located at ISA I/O address x1002. The Universe ASIC is programmed so that this register can be accessed from the VMEbus to provide VMEbus location monitor function.

REG	Location Monitor Upper Base Address Register - Offset \$1002							
BIT	SD7	SD6	SD5	SD4	SD3	SD2	SD1	SD0
FIELD	VA15	VA14	VA13	VA12	VA11	VA10	VA9	VA8
OPER	R/W							
RESET	0	0	0	0	0	0	0	0

**VA[15:8]** Upper Base Address for the location monitor function.

### Location Monitor Lower Base Address Register

The Location Monitor Lower Base Address Register is an 8-bit register located at ISA I/O address x1003. The Universe ASIC is programmed so that this register can be accessed from the VMEbus to provide VMEbus location monitor function.

REG	Location Monitor Lower Base Address Register - Offset \$1003							
BIT	SD7	SD6	SD5	SD4	SD3	SD2	SD1	SD0
FIELD	VA7	VA6	VA5	VA4	LMEN			
OPER	R/W					R	R	R
RESET	0	0	0	0	0	0	0	0

**VA[7:4]** Lower Base Address for the location monitor function.

**LMEN** This bit must be set to enable the location monitor function.

### Semaphore Register 1

The Semaphore Register 1 is an 8-bit register located at ISA I/O address x1004. The Universe ASIC is programmed so that this register can be accessible from the VMEbus. This register can only be updated if bit 7 is low or if the new value has the most significant bit cleared. When bit 7 is high, this register will not latch in the new value if the new value has the most significant bit set.

REG	Semaphore Register 1 - Offset \$1004							
BIT	SD7	SD6	SD5	SD4	SD3	SD2	SD1	SD0
FIELD	SEM1							
OPER	R/W							
RESET	0	0	0	0	0	0	0	0

## Semaphore Register 2

The Semaphore Register 2 is an 8-bit register located at ISA I/O address x1005. The Universe ASIC is programmed so that this register can be accessible from the VMEbus. This register can only be updated if bit 7 is low or if the new value has the most significant bit cleared. When bit 7 is high, this register will not latch in the new value if the new value has the most significant bit set.

REG	Semaphore Register 2 - Offset \$1005							
BIT	SD7	SD6	SD5	SD4	SD3	SD2	SD1	SD0
FIELD	SEM2							
OPER	R/W							
RESET	0	0	0	0	0	0	0	0

## VME Geographical Address Register (VGAR)

The VME Geographical Address Register is an 8-bit read-only register located at ISA I/O address x1006. This register reflects the states of the geographical address pins at the 5-row, 160-pin P1 connector.

REG	VME Geographical Address Register - Offset \$1006							
BIT	SD7	SD6	SD5	SD4	SD3	SD2	SD1	SD0
FIELD			GAP#	GA4#	GA3#	GA2#	GA1#	GA0#
OPER	READ ONLY							
RESET	X	X	X	X	X	X	X	X

## Z85230 ESCC and Z8536 CIO Registers and Port Pins

The Z85230 ESCC is used to provide the two sync/asynch serial ports on some MVME2600 series models. The PCLK which can be used to derive the baud rates, is 10 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 and a method to inquire the module ID of the two sync/asynch serial ports that reside on the MVME761 module. Refer to the Z8536 Data Sheet 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 1-20. 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 1-21. Z8536 CIO Port Pins Assignment**

Port Pin	Signal Name	Direction	Descriptions
PA0	TM3_MID0	Input	Port 3 Test Mode when IDREQ_ = 1; Module ID Bit 0 when IDREQ_ = 0.
PA1	DSR3_MID1	Input	Port 3 Data Set Ready when IDREQ_ = 1; Module ID Bit 1 when IDREQ_ = 0.
PA2	RI3_	Input	Port 3 Ring Indicator
PA3	LLB3_MODSEL	Output	Port 3 Local Loopback (IDREQ_ = 1) or Port Select (IDREQ_ = 0): IDREQ_ = 0 & MODSEL = 0 => Port 3 ID Select IDREQ_ = 0 & MODSEL = 1 => Port 4 ID Select
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	IDREQ_	Output	Module ID Request - low true
PB0	TM4_MID2	Input	Port 4 Test Mode when IDREQ_ = 1; Module ID Bit 2 when IDREQ_ = 0.
PB1	DSR4_MID3	Input	Port 4 Data Set Ready when IDREQ_ = 1; Module ID Bit 3 when IDREQ_ = 0.
PB2	RI4_	Input	Port 4 Ring Indicator
PB3	LLB4_	Output	Port 4 Local Loopback
PB4	RLB4_	Output	Port 4 Remote Loopback
PB5	DTR4_	Output	Port 4 Data Terminal Ready
PB6	FUSE	Input	FUSE = 1 means that at least one of the fuses or polyswitches is open.
PB7	ABORT_	Input	Status of ABORT# signal

**Table 1-21. Z8536 CIO Port Pins Assignment (Continued)**

<b>Port Pin</b>	<b>Signal Name</b>	<b>Direction</b>	<b>Descriptions</b>
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.

The module ID signals, which are only valid when IDREQ\_ is asserted, indicate the type of the serial module that is installed on either Port 3 or Port 4. The following table shows how to interpret the MID3-MID0 signals:

**Table 1-22. Interpretation of MID3-MID0**

IDREQ_	LLB3_ MODSEL	MID3	MID2	MID1	MID0	Serial Module Type	Module Assembly Number
1	X	X	X	X	X	Invalid module ID	
0	0	0	0	0	0	Module 3: EIA232 DCE	01-W3876B01
0	0	0	0	0	1	Module 3: EIA232 DTE	01-W3877B01
0	0	0	0	1	0	Module 3: EIA530 DCE	01-W3878B01
0	0	0	0	1	1	Module 3: EIA530 DTE	01-W3879B01
0	0	1	1	1	1	Module 3 Not Installed	
0	1	0	0	0	0	Module 4: EIA232 DCE	01-W3876B01
0	1	0	0	0	1	Module 4: EIA232 DTE	01-W3877B01
0	1	0	0	1	0	Module 3: EIA530 DCE	01-W3878B01
0	1	0	0	1	1	Module 3: EIA530 DTE	01-W3879B01
0	1	1	1	1	1	Module 4 Not Installed	

**Note** Because IDREQ\_ and MID3-MID0 signals go through the P2MX (P2 multiplexing) function used on MVME2600 series modules configured for the MVME761-type transition module, software must wait for the MID3-MID0 to become valid after asserting IDREQ\_. The waiting time should be about 4 microseconds because the sampling rate is about 1.6 microsecond with a 10MHz MXCLK clock.

## 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 1-23. 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		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		Not Used

**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 Chip

---

2

## Introduction

### Overview

This document describes the architecture and usage of the Raven, a PowerPC to PCI Local Bus Bridge ASIC. The Raven is intended to provide MPC60x compliant devices access to devices residing on the PCI Local Bus in a very efficient manner. In the remainder of this chapter, the MPC60x bus will be referred to as the MPC bus and the PCI Local Bus as PCI.

No manufacturer currently has plans to support the MPC 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 MByte/sec in 32-bit mode or 264 MByte/sec in 64-bit mode. While the PCI specification is relatively new, it has received overwhelming support among PC clone manufacturers. Many peripheral device manufacturers have designed PCI Local Bus compliant products. NCR tested a PCI SCSI controller and a PCI Ethernet controller. Ethernet controllers from AMD and National Semiconductor are available. Graphics controllers are available from Trident, S3, Oak Technologies, Weitek, Chips and Technologies, Headland Technologies, and NCR.

## Requirements

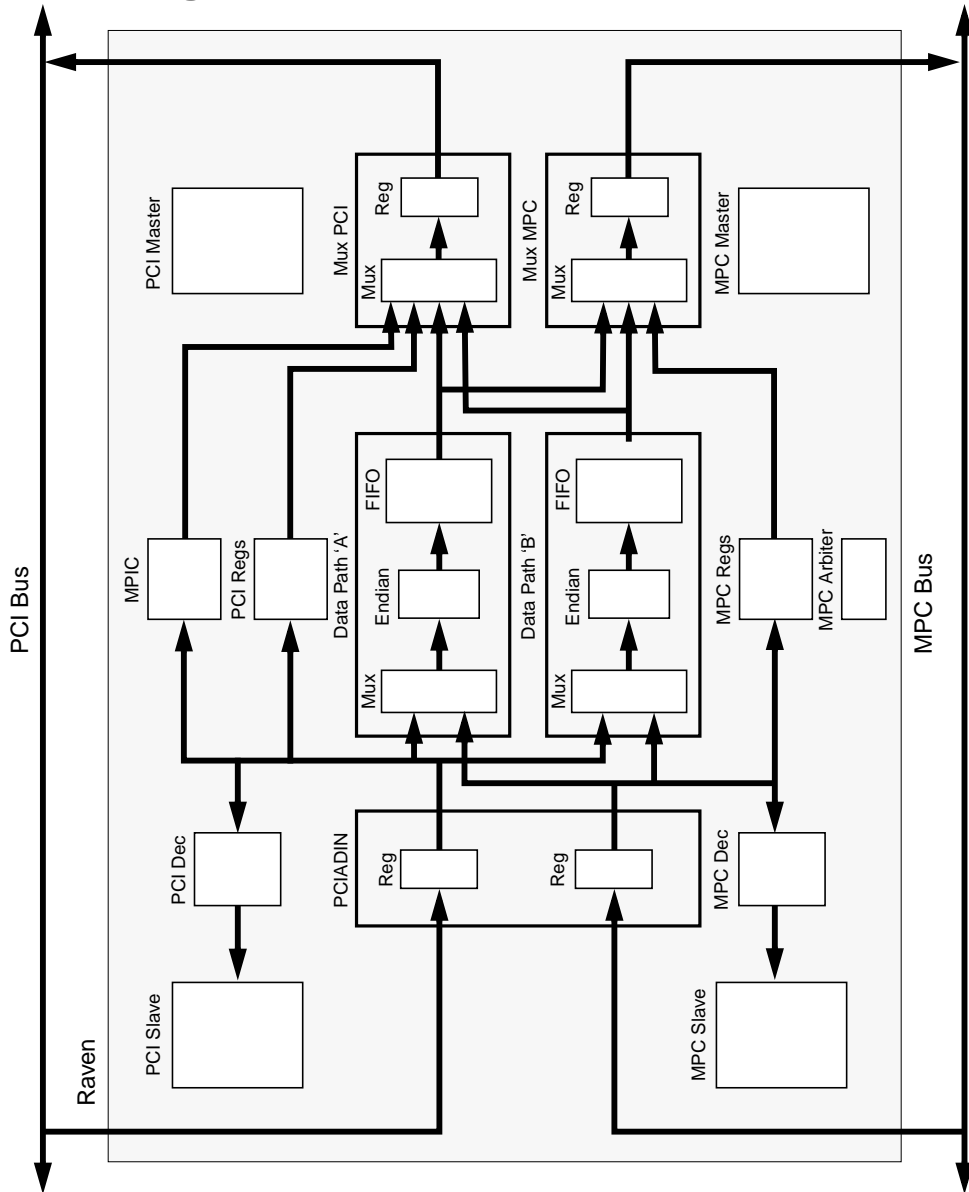
The Raven must provide a high throughput interface between multiple MPC60x processors and 32/64-bit PCI local bus. It must be capable of supporting up to two MPC60x processors and contain a multiprocessing interrupt structure to efficiently distribute interrupts dynamically between these processors.

## Features

- MPC Bus Interface
  - Direct interface to MPC601, MPC603 or MPC604 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 MPC 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 MPC bus.
  - Read-ahead buffer for reads from the MPC 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



1914 9610

Figure 2-1. Raven Block Diagram

# Functional Description

## MPC Bus Interface

The MPC Bus Interface is designed to be coupled directly to up to two MPC601, MPC603, or MPC604 microprocessors as well as a memory / cache subsystem. It uses a subset of the capabilities of the MPC60x bus protocol.

## MPC Arbiter

The MPC Arbiter is an optional feature in the Raven. The Raven MPC Arbiter is enabled if both CPUID pins are sampled high on the rising edge of RST\*. When this feature is enabled, the MARB bit in the General Control/Status Register (GCSR) will be set. When this feature is not enabled, the MARB bit will be cleared and the Raven will be configured for external arbitration.

The MPC Arbiter function is responsible for determining address bus ownership. The MPC Arbiter function does not provide data bus arbitration. Determining data bus ownership is the responsibility of each MPC master and follows the ownership ordering established on the address bus.

The MPC Arbiter supports a total of four participants. One participant is the Raven MPC master function, which represents PCI initiated MPC transactions. The remaining three participants are external to the Raven, and are represented as request / grant signal pairs.

The MPC Arbiter supports a mixture of Fixed Priority and Round-Robin Priority arbitration schemes. PCI initiated transactions will always have the highest priority over all external requests. The external requests will be honored in a Round-Robin algorithm. This algorithm enforces fairness for bus ownership by assigning the lowest priority to the most recently granted bus requester. If a bus requester is not granted bus ownership during an arbitration event, the priority of that requester will be increased for the next arbitration event.

Each external request can be individually enabled or disabled using control bits in the MPC Arbiter Control Register (MARB). Following reset, all requests will be enabled.

The MPC Arbiter supports optional bus parking in order to reduce arbitration latency. If bus parking is enabled, then one of two modes may be selected. The arbiter can be configured to grant the bus to the last selected master, or it can be configured to grant the bus to one “default” device. The parking enable, parking mode and default master information is represented as control bits within the MARB register.

There is a special ‘Glance Mode’ incorporated into the MPC Arbiter design. This mode was designed to compensate for an undesirable characteristic associated with early release Glance look-aside cache chips. Under certain conditions, the Glance would not maintain single-level pipelined operation when the Raven was MPC bus master. When Glance Mode is enabled, the MPC Arbiter tracks address and data tenures and will grant bus ownership to non-Raven bus masters in a manner that guarantees single-level pipelined depth. This mode is controlled by the GLMD bit within the MPC Arbiter Control register. The default state is to have this mode disabled.

There is another special mode called ‘Benign Address Retry Mode’. This mode was designed to compensate for an anomaly discovered when running a PPC603 and a pair of early release Falcon memory controllers. It is possible that contention of the PPC603 single port cache tag memory causes the PPC603 to issue a false (benign) address retry. Under certain circumstances, these benign address retry cycles would create problems with Falcon. When the Benign Address Retry Mode is enabled, the MPC Arbiter will be watching for benign address retry cycles. When one is detected, the arbiter will hold-off all non-Raven bus masters for a short period of time. This mode is controlled by the BAMD bit within the MPC Arbiter Control register. The default state is to have this mode enabled.

A side benefit of the Benign Address Retry Mode is that it provides a possible solution to a known live-lock condition that can happen when the 603 is executing a tight loop out of cache and another master is attempting to transfer a coherent cache line to or from memory. The Benign Address Retry Mode effectively introduces jitter between the contesting participants.

## MPC 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 MPC bus addresses to which the Raven will respond: the MPC 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**

MPC Address	Function
\$00000000-\$7FFFFFFF	System Memory (2G)
\$80000000-\$FCFFFFFF	PCI Memory (2G - 48M)
\$FD000000-\$FDFFFFFF	ISA Memory (16M)
\$FE000000-\$FE7FFFFFF	Discontiguous PCI IO (8M)
\$FE800000-\$FEBFFFFFF	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 MPC Register File decoder determines the address location of the Raven's MPC registers from the MPC bus. These registers may be accessed using only 1-, 2-, 3-, 4-, or 8-byte operations. The location of the MPC register file is fixed beginning at MPC 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 MPC register file will start at address \$FEFE0000. If the EXT01 pin is sampled in the high state, the MPC register file will start at address \$FEFF0000. All references to the MPC 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 MPC bus to the PCI bus. These decoders provide a window into the PCI bus from the MPC bus. The most significant 16 bits of the MPC 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 MPC address, and the result is used as the PCI address. This offset allows PCI devices to reside at any PCI address, independent of the MPC address map.

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

## **MPC Write Posting**

The MPC 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 MPC write transfer to the PCI bus and immediately acknowledges the transaction on the MPC bus. This frees the MPC bus from waiting for the potentially long PCI arbitration and transfer. The MPC 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.

## MPC Master

Wherever possible, the MPC 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 MPC master will attempt singular burst transfers. The MPC 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 MPC 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 MPC master can be moving another cache line worth onto the MPC bus. This will allow the PCI slave to receive long block transfers without stalling.

When programmed in "read ahead" mode (the RAEN bit in the PSATTx register is set) and the PCI slave receives a Memory Read Line or Memory Read Multiple command, the MPC master will fetch data in bursts and store it in the FIFO. The contents of the FIFO will then be used to attempt to satisfy the data requirements for the remainder of the PCI block transaction. If the data requested is not in the FIFO, the MPC master will read another cache line. The contents of the FIFO are "invalidated" at the end of each PCI block transaction.

- Notes**
1. Read ahead mode should not be used when data coherency may be a problem as there is no way to snoop all MPC bus transactions and invalidate the contents of the FIFO.
  2. Accesses near the top of local memory with read-ahead mode enabled could cause the MPC master to perform reads beyond the top of local memory which could result in an MPC bus timeout error.

The MPC bus transfer types generated by the MPC master depend on the PCI command code and the INV bit in the PSATTx registers.

**Table 2-2. MPC Transfer Types**

PCI Command Code	INV	MPC Transfer Type	MPC 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 MPC master incorporates an optional operating mode called Bus Hog. When Bus Hog is enabled, the MPC master will continually request the MPC bus for the entire duration of each PCI transfer. When Bus Hog is not enabled, the MPC 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 MPC 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.

## MPC Bus Timer

The MPC 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 MPC Error Status Register, latch the MPC address in the MPC 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 MPC 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 MPC 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 MPC 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 MPC address. This offset allows devices to reside at any MPC 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 MPC 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 MPC 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 MPC transfer type, TBST\*, and the MEM field in the MSATTx registers.

**Table 2-3. PCI Command Codes**

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

## 2 Generating PCI Memory and I/O Cycles

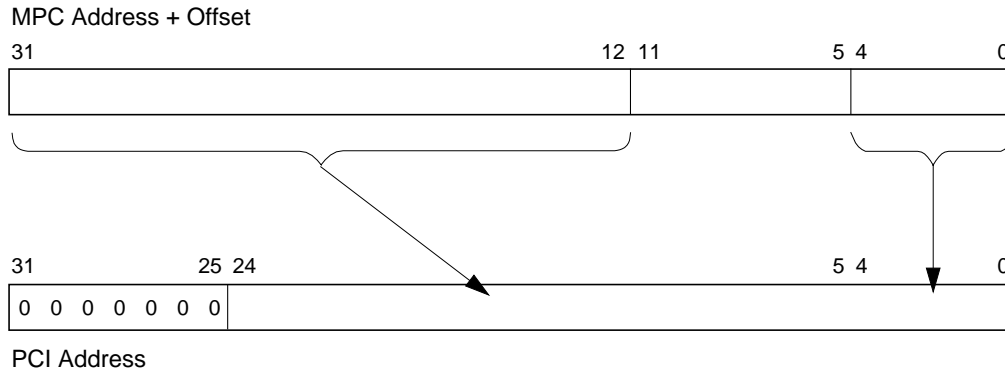
Each programmable slave may be configured to generate PCI I/O or memory accesses through the MEM and IOM fields in its Attribute register as shown below.

<b>MEM</b>	<b>IOM</b>	<b>PCI Cycle Type</b>
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 MPC 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 MPC address, apply the offset specified in the MSOFFx register, and map the result to PCI as shown in Figure 2-2.



1915 9610

**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 MPC 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 MPC Slave Address Registers is used to gain access to \$CF8 and \$CFC. Note that MSADD3 is initialized at reset to access PCI I/O space with an MPC 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.

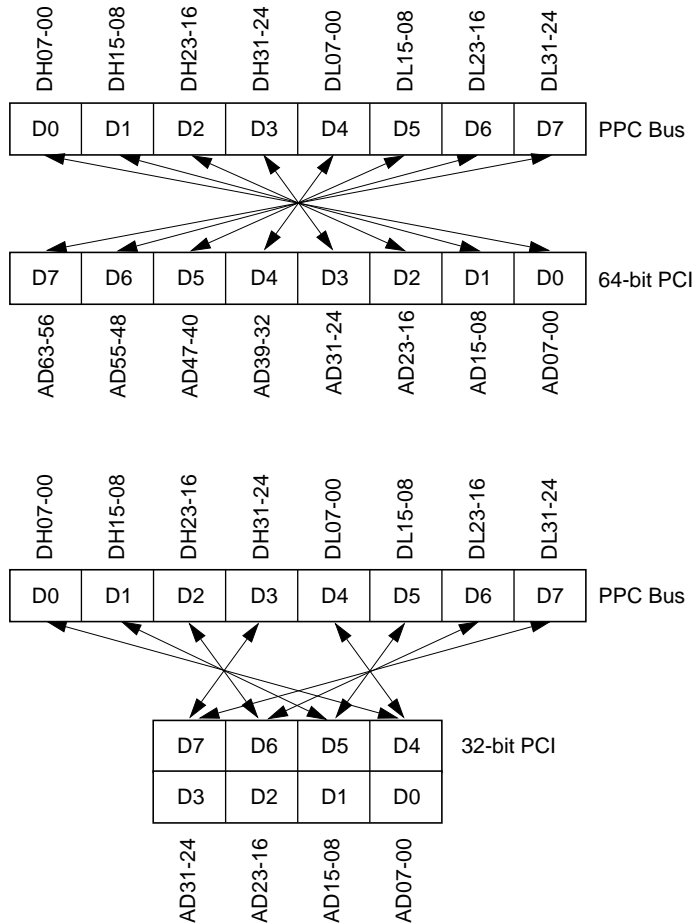


## Endian Conversion

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

### When MPC Devices are Big-Endian

When all MPC devices are operating in Big-Endian mode, all data to/from PCI must be swapped such that PCI looks big endian from the MPC bus's perspective. This is shown in Figure 2-3.



1916 9610

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

**When MPC Devices are Little Endian**

When all MPC devices are operating in little endian mode, the MPC address must be modified to remove the exclusive-ORing applied by MPC60x 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 MCP60x processor does in little endian mode. The modification will be the same as that described in Section 3.7.2 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 MPC bus.

### Error Handling

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

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

Error Status	Error Address and Attributes
MATO	From MPC 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 MPC Error Enable Register on a source by source basis. When a machine check is enabled, either the MID field in the MPC 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 MPC 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 MPC master would be a PCI system error.

## PCI/MPC Contention Handling

The Raven has a stall detection mechanism that detects when there is a possible resource contention problem (i.e. deadlock) as a result of overlapping MPC and PCI initiated transactions. The MPC 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 MPC Slave function whenever it is currently processing a transaction that must have control of the MPC 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 MPC Slave function determines its future actions based on the stall signal and the current MPC bus activity. If the MPC Slave function determines there will be contention between a cycle completing on the MPC bus and an incoming PCI cycle, the MPC Slave will issue a retry for the current MPC transaction. This retry will free up the MPC bus and allow the PCI initiated transaction to complete. An idle MPC bus obviously gives immediate access to the pending PCI initiated transaction.

If the MPC bus is currently supporting a read cycle of any type, the cycle will be terminated with a retry. Note that if the read cycle is across a mod-4 address boundary (i.e. 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 MPC 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 MPC 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 MPC 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 MPC 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.

## Registers

This chapter provides a detailed description of all Raven registers. These registers are broken into two groups: the MPC Registers and the PCI Configuration Registers. The MPC Registers are accessible only from the MPC bus using any valid transfer size. The PCI Configuration Registers reside in PCI configuration space. They are accessible from the MPC bus through the Raven. The MPC Registers are described first; the PCI Configuration Registers are described next.

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.

## MPC Registers

The Raven MPC register map is shown in Table 2-5.

**Table 2-5. Raven MPC Register Map**

Bit --->	0	1	2	3	4	5	6	7	8	9	0	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	3	3
\$FEFF0000	VENID										DEVID																			
\$FEFF0004									REVID																					
\$FEFF0008	GCSR										FEAT																			
\$FEFF000C											MARB																			
\$FEFF0010																					PADJ									
\$FEFF0014																														
\$FEFF0018																														
\$FEFF001C																														
\$FEFF0020											MEREN																			
\$FEFF0024																					MERST									

**Table 2-5. Raven MPC Register Map (Continued)**

Bit ---->	0	1	2	3	4	5	6	7	8	9	0	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	3	3		
<b>\$FEFF0028</b>	MERAD																															
<b>\$FEFF002C</b>																	MERAT															
<b>\$FEFF0030</b>	PIACK																															
<b>\$FEFF0034</b>																																
<b>\$FEFF0038</b>																																
<b>\$FEFF003C</b>																																
<b>\$FEFF0040</b>	MSADD0																															
<b>\$FEFF0044</b>	MSOFF0																MSATT0															
<b>\$FEFF0048</b>	MSADD1																															
<b>\$FEFF004C</b>	MSOFF1																MSATT1															
<b>\$FEFF0050</b>	MSADD2																															
<b>\$FEFF0054</b>	MSOFF2																MSATT2															
<b>\$FEFF0058</b>	MSADD3																															
<b>\$FEFF005C</b>	MSOFF3																MSATT3															
<b>\$FEFF0060</b>																																
<b>\$FEFF0064</b>																																
<b>\$FEFF0068</b>																																
<b>\$FEFF006C</b>																																
<b>\$FEFF0070</b>																	GPREG0(Upper)															
<b>\$FEFF0074</b>																	GPREG0(Lower)															
<b>\$FEFF0078</b>																	GPREG1(Upper)															
<b>\$FEFF007C</b>																	GPREG1(Lower)															



## Vendor ID/Device ID Registers

Address	\$FEFF0000																																
Bit	0	1	2	3	4	5	6	7	8	9	0	1	1	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2	3	3
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	0	1	1	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2	2	3	3
Name									REVID																									
Operation	R								R								R								R									
Reset	\$00								\$02								\$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	0	1	2	3	4	5	6	7	8	9	0	1	2	2	2	2	2	2	2	3	3	
Name	GCSR																FEAT															
	LEND				BHOG	FLBRD	MBT1	MBT0	P64	MARB	MPIC			MID1	MID0		EXT14	EXT13	EXT12	EXT11	EXT10	EXT09	EXT08	EXT07	EXT06	EXT05	EXT04	EXT03	EXT02	EXT01	EXT00	
Operation	RW	R	R	R	RW	RW	RW	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	
Reset	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 MPC bus is operating in little endian mode. The MPC address will be modified as described in the section *When MPC Devices are Little Endian*. When LEND is clear, the MPC bus is operating in big endian mode, and all data to/from PCI is swapped as described in the section *When MPC Devices are Big-Endian*.
- FLBRD** **Flush Before Read.** If set, the Raven will guarantee that all PCI initiated posted write transactions will be completed before any MPC initiated read transactions will be allowed to complete. When FLBRD is clear, there will be no correlation between these transaction types and their order of completion. Please refer to the section on *PCI/MPC Contention Handling* for more information.
- BHOG** **Bus Hog.** If set, the Raven MPC master will operate in the Bus Hog mode. Bus Hog mode means the MPC master will continually request the MPC bus for the entire duration of each PCI transfer. If Bus Hog is not enabled, the MPC master will request the bus in a normal manner. Please refer to the section on *MPC Master* for more information.

**MBTx** **MPC Bus Time-out.** This field specifies the MPC bus time-out length. The time-out length is encoded as follows:

MBT	Time Out Length
00	256 $\mu$ sec
01	64 $\mu$ sec
10	8 $\mu$ sec
11	disabled

**P64** **64-bit PCI Mode Enable.** If set, the Raven is connected to a 64-bit PCI bus. This bit is set if REQ64\* is asserted on the rising edge of RESET\*.

**MARB** **MPC Arbiter Enable.** If set, the Raven internal MPC Arbiter is enabled. This bit is set if CPUID is %111 on the rising edge of RESET\*.

**MPIC** **Multi-Processor Interrupt Controller Enable.** If set, the Raven internal MPIC interrupt controller is enabled. This bit is set if EXT15 is high on the rising edge of RESET\*. 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 MPC bus master. When the internal MPC arbiter is enabled (MARB is set), these bits are controlled by the internal arbiter. When the internal arbiter is disabled (MARB is clear) these bits reflect the status of the CPUID pins. In a multiprocessor environment, these bits allow software to determine on which processor it is currently running. The internal MPC arbiter encodes this field as follows:

MID	Current MPC 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.

## MPC Arbiter Control Register

Address	\$FEFF000C																																						
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	32						
Name																	MARB																						
Operation	R								R								R	R	R	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	0	0	0	0	0	0	0

**BREN2 Bus Request 2 Enable.** If set, the processor bus request signal ABR2\* is enabled. If cleared, ABR2\* is not enabled, and ABG2\* will never be asserted.

**BREN1 Bus Request 1 Enable.** If set, the processor bus request signal ABR1\* is enabled. If cleared, ABR1\* is not enabled, and ABG1\* will never be asserted.

**BREN0 Bus Request 0 Enable.** If set, the processor bus request signal ABR0\* is enabled. If cleared, ABR0\* is not enabled, and ABG0\* will never be asserted.

**PKEN Bus Parking Enable.** If set, the MPC arbiter will park an MPC master on the bus when no bus requests are pending. If cleared, no MPC master will be granted the bus without first asserting its ABRx\*.

**PKMD Bus Parking Mode.** When bus parking is enabled (PKEN is set), this bit defines the method used to determine which MPC master is parked on the bus. If set, the master specified in the DEFM field is parked on the bus when no bus requests are pending. If cleared, the last active MPC master is

parked on the bus when no bus requests are pending. This field has no meaning when PKEN is clear.

**GLMD** **Glance Mode.** If set, the MPC arbiter will operate in the Glance mode. Please refer to the section on the *MPC Arbiter* for more information.

**BAMD** **Benign Address Retry Mode.** If set, the MPC arbiter will operate in the benign address retry mode. Please refer to the section on the *MPC Arbiter* for more information.

**DEFM<sub>x</sub>** **Default Master.** This field specifies which MPC bus master will be parked on the bus when default parking is enabled. DEFM only has meaning when PRKEN and PRKMD are both set.

DEFM1	DEFM0	Parked Device
0	0	device on ABR0*
0	1	device on ABR1*
1	0	device on ABR2*
1	1	Raven

### 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								\$B4							

**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:

$$\text{PADJ} = 256 - \text{Clk},$$

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	\$B4
50	\$CE
40	\$D8
33	\$DF
25	\$E7

### MPC Error Enable Register

Address	\$FEFF0020																																	
Bit	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	2	2	2	2	2	2	2	3	3		
Name																	MEREN																	
																	DFLT	MATOM		PERRM	SERRM	SMAM	RTAM			MATOLL		PERRI	SERRI	SMARI	RTARI			
Operation	R								R								R	RW	RW	R	RW	RW	RW	RW	RW	R	R	RW	R	RW	RW	RW	RW	RW
Reset	\$00								\$00								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- DFLT**     **Default MPC Master ID.** This bit determines which MCHK\* pin will be asserted for error conditions in which the MPC master ID can not be determined or the Raven was the MPC master. For example, in event of a PCI parity error for a transaction in which the Raven's PCI master was not involved, the MPC master ID can not be determined. When DFLT is set, MCHK1\* is used. When DFLT is clear, MCHK0\* will be used.
- MATOM**   **MPC Address Bus Time-out Machine Check Enable.** When this bit is set, the MATO bit in the MERST 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 MERST 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 MERST 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 MERST 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 MERST 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.



- MATOI**    **MPC Address Bus Time-out Interrupt Enable.**When this bit is set, the MATO bit in the MERST register will be used to assert an interrupt through the MPIC 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 MERST register will be used to assert an interrupt through the MPIC 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 MERST register will be used to assert an interrupt through the MPIC 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 MERST register will be used to assert an interrupt through the MPIC 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 MERST register will be used to assert an interrupt through the MPIC interrupt controller. When this bit is clear, no interrupt will be asserted.

## MPC Error Status Register

Address	\$FEFF0024																															
Bit	0	1	2	3	4	5	6	7	8	9	0	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	3	3			
Name																	MERST															
																	OVF	MATO	PERR	SERR	SMA	RTA										
Operation	R								R								R/C	R	R/C	R	R/C	R/C	R/C									
Reset	\$00								\$00								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.

**MATO**      **MPC Address Bus Time-out.** This bit is set when the MPC 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 MATOM bit in the MEREN register is set, the assertion of this bit will assert MCHK to the master designated by the MID field in the MERAT register. When the MATOI bit in the MEREN register is set, the assertion of this bit will assert an interrupt through the MPIC 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 MEREN register is set, the assertion of this bit will assert MCHK to the master designated by the DFLT bit in the MERAT register. When the PERRI bit in the MEREN register is set, the assertion of this bit will assert an interrupt through the MPIC 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 MEREN register is set, the assertion of this bit will assert MCHK to the master designated by the DFLT bit in the MERAT register. When the SERRI bit in the MEREN register is set, the assertion of this bit will assert an interrupt through the MPIC 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 MEREN register is set, the assertion of this bit will assert MCHK to the master designated by the MID field in the MERAT register. When the SMAI bit in the MEREN register is set, the assertion of this bit will assert an interrupt through the MPIC 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 MEREN register is set, the assertion of this bit will assert MCHK to the master designated by the MID field in the MERAT register. When the RTAI bit in the MEREN register is set, the assertion of this bit will assert an interrupt through the MPIC interrupt controller.

### MPC Error Address Register

Address	\$FEFF0028																															
Bit	0	1	2	3	4	5	6	7	8	9	0	1	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2	3	3
Name	MERAD																															
Operation	R																															
Reset	\$00000000																															

**MERAD MPC Error Address.** This register captures the MPC address when the MATO bit is set in the MERST register. It captures the PCI address when the SMA or RTA bits are set in the MERST register. Its contents are not defined when the PERR or SERR bits are set in the MERST register.

### MPC Error Attribute Register - MERAT

If the 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	0	1	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2	3	3				
Name																	MERAT																			
Operation	R								R								R	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	0	0	0	0

- MID<sub>x</sub>** **MPC Master ID.** This field contains the ID of the MPC 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.
- TSIZ<sub>x</sub>** **Transfer Size.** This field contains the transfer size of the MPC transfer in which the error occurred.
- TT<sub>x</sub>** **Transfer Type.** This field contains the transfer type of the MPC transfer in which the error occurred.

If the SMA or RTA bit are 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	32				
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	R	R	R		
Reset	\$00								\$00								0	0	0	0	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<sub>x</sub>** **MPC Master ID.** This field contains the ID of the MPC master which originated the transfer in which the error occurred. The encoding scheme is identical to that used in the GCSR register.
- COMM<sub>x</sub>** **PCI Command.** This field contains the PCI command of the PCI transfer in which the error occurred.

**BYTE<sub>x</sub>** **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	0	1	1	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2	3	3
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.

### MPC 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 MPC bus, the MPC 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 MPC 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 MPC address.

**END End Address.** This field determines the end address of a particular memory area on the MPC 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 MPC address.

**MPC Slave Address (3) Register**

Address	MSADD3 - \$FEFF0058																															
Bit	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	2	2	2	2	2	2	2	3	3
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 (\$8000CF8) and Configuration Data (\$8000CFC) registers. Note that this implies that MSxxx3 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 MPC register initialization. Please see the description of the MSOFF3/MSATT3 Registers for additional information.

To initiate a PCI cycle from the MPC bus the MPC 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 MPC 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 MPC address.

**END**        **End Address.** This field determines the end address of a particular memory area on the MPC 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 MPC address.



## MPC 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																					
Reset	\$0000										\$00																					

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

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

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

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

**MEM PCI Memory Cycle.** If set, the corresponding MPC slave will generate transfers to or from PCI memory space. When clear, the corresponding MPC 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 MPC 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 MPC slave will generate PCI I/O cycles using contiguous addressing. This field only has meaning when the MEM bit is clear.

**MPC 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																RW	RW	R	RW	R	R	R	R	RW
Reset	\$8000																\$00																1	1	0	0	0	0	0	0	0

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

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

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

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

**IOM**     **PCI I/O Mode.** If set, the corresponding MPC 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 MPC 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, Revision 2.0. 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**

3	3	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	<--- Bit	
DEVID														VENID														\$00				
PSTAT														PCOMM														\$04				
CLASS														REVID														\$08				
																												\$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**

3	3	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	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	3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	1 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	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 MPC Registers.

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

## PCI Command/ Status Registers

Offset	\$04																																
Bit	3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	1 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0	
Name	PSTAT																PCOMM																
	RCV/PE	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	R/W	R	R/W	R	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	0	0	0	0

<b>IOSP</b>	<b>IO Space Enable.</b> 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.
<b>MEMSP</b>	<b>Memory Space Enable.</b> 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.
<b>MSTR</b>	<b>Bus Master Enable.</b> If set, the Raven may act as a master on PCI. If cleared, the Raven may not act as a PCI master.
<b>PERR</b>	<b>Parity Error Response.</b> 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.
<b>SERR</b>	<b>System Error Enable.</b> 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.
<b>FAST</b>	<b>Fast Back-to-Back Capable.</b> This bit indicates that the Raven is capable of accepting fast back-to-back transactions with different targets.
<b>DPAR</b>	<b>Data Parity Detected.</b> 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.
<b>SELTIM</b>	<b>DEVSEL Timing.</b> 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																\$02															

**REVID** **Revision ID.** This register identifies the Raven revision level. This register is duplicated in the MPC 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

**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.





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

Offset	PSADD0 - \$80 PSADD1 - \$88 PSADD2 - \$90 PSADD3 - \$98																															
Bit	3	3	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	0	9	8	7	6	5	4	3	2	1	0
Name	PSADDx																															
	START																END															
Operation	R/W																R/W															
Reset	\$0000																\$0000															

To initiate an MPC 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 MPC 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 MPC bus resources. The value of this field will be compared with the upper 16 bits of the incoming PCI address.



**PSOFFx 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 MPC address used for transfers from PCI to the MPC bus. This offset allows MPC resources to reside at addresses that would not normally be visible from PCI.

### CONFIG\_ADDRESS

The routing of the MPC 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 MPC 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																															
	LEND	BUS						DEV				FUN				REG																
Operation	R/W	R						R/W				R/W				R/W						0	0									
Reset	1	\$00						\$00				\$00				\$0						0	0									
LEND = 0		MPC[39:32]						MPC[47:40]				MPC[55:48]				MPC[63:56]																
LEND = 1		MPC[31:24]						MPC[23:16]				MPC[15:8]				MPC[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 (RavenMPIC) 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 RavenMPIC 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 RavenMPIC 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 RavenMPIC 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 RavenMPIC provides a mechanism to support PC-AT compatible chip sets using the 8259 interrupt controller architecture. After power-on reset, the RavenMPIC 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 RavenMPIC) 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 RavenMPIC.

The RavenMPIC 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. The Raven Error Vector-Priority Register should be programmed for high true level sensitive activation.

For system implementations where the RavenMPIC 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 that 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.

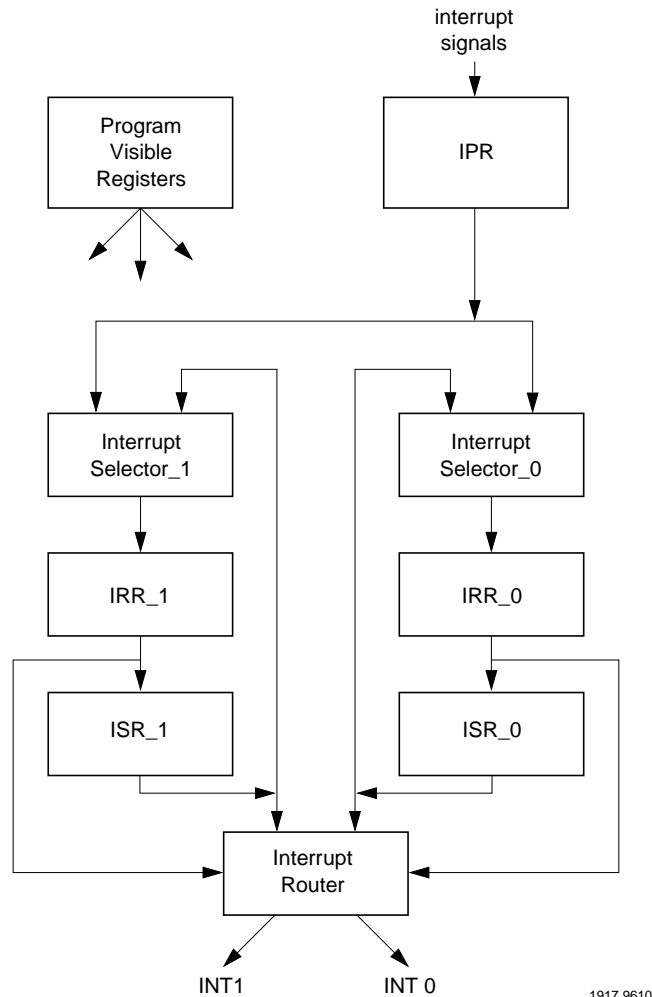
In the direct delivery mode, the interrupt is directed to one or both processors. If it is directed to two processors (i.e. multicast), it will be delivered to two processors. The interrupt is delivered to the processor when the priority of the interrupt is greater than the priority contained in the 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.

## 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. RavenMPIC 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 RavenMPIC 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 in 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 is always delivered to processor 0. 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.

## RavenMPIC Registers

The RavenMPIC register map is shown in the following table. The Off field is the address offset from the base address of the RavenMPIC registers in the MPC-IO or MPC-MEMORY space. Note that this map does not depict linear addressing. The Raven PCI-SLAVE has two decoders for generating the RavenMPIC select. These decoders will generate a select and acknowledge all accesses which are in a reserved 256K byte range. If the index into that 256K block does not decode a valid RavenMPIC register address, the logic will return \$00000000.

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



**Table 2-8. RavenMPIC Register Map**

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	1 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	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 1 VECTOR-PRIORITY REGISTER																											\$01160					
TIMER 1 DESTINATION 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					

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	1 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0	Off
TIMER 3 VECTOR-PRIORITY REGISTER																															\$011e0	
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	



2

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	1 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0	Off
																											IACK REGISTER P1	\$210a0				
																											EOI REGISTER P1	\$210b0				

### Feature Reporting Register

Offset	\$01000																																							
Bit	3	3	2	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1			
Name	FEATURE REPORTING																																							
									NIRQ																NCPU								VID							
Operation	R								R								R								R								R							
Reset	\$0								\$00F								\$0								\$01								\$02							

**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 02 is used for the initial 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																														
Operation	C	R	R/W	R																				R								
Reset	0	0	0	\$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. MPIC is essentially disabled. In the mixed mode, 8259 interrupts are delivered using the priority and distribution mechanism of 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



**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.





### Timer Current Count Registers

Offset	Timer 0 - \$01100 Timer 1 - \$01140 Timer 2 - \$01180 Timer 3 - \$011C0																															
Bit	3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	0 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	9 8	8 7	7 6	6 5	5 4	4 3	3 2	2 1	1 0		
Name	TIMER CURRENT COUNT																															
	CC																															
Operation	R																															
Reset	\$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.

### Timer Basecount Registers

Offset	Timer 0 - \$01110 Timer 1 - \$01150 Timer 2 - \$01190 Timer 3 - \$011D0																															
Bit	3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	0 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	9 8	8 7	7 6	6 5	5 4	4 3	3 2	2 1	1 0		
Name	TIMER BASECOUNT																															
	BC																															
Operation	R/W																															
Reset	\$00000000																															

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



**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/W	R/W		
Reset	\$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.



**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							RW	RW		
Reset	\$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 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																																	
Operation	R								R								R								R								P1	P0
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																																	
Operation	R								R								R								R								P1	P0
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





### Interrupt Acknowledge Registers

Offset	Processor 0 \$200A0 Processor 1 \$210A0																															
Bit	3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	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	3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	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 RavenMPIC 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). RavenMPIC 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 within the RavenMPIC 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 RavenMPIC (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 RavenMPIC 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 RavenMPIC.

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.





## Introduction

The Falcon DRAM controller ASIC is designed for the MVME2600 family of boards. It is used in sets of two to provide the interface between the PowerPC 60x 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).

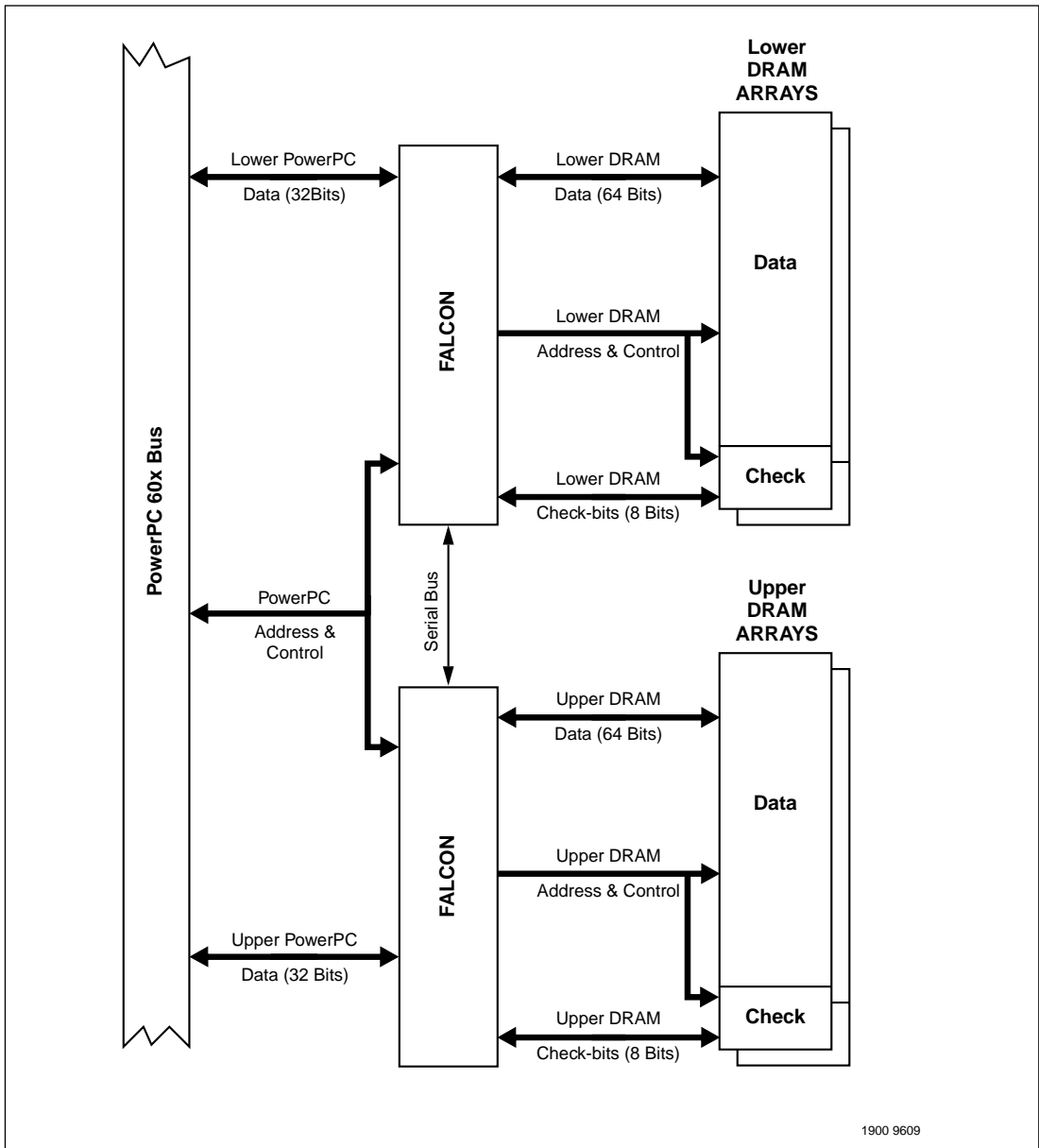
## Features

- ❑ 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.
  - Programmable sequencer for fast DRAM tests.
- ❑ Error Notification for DRAM

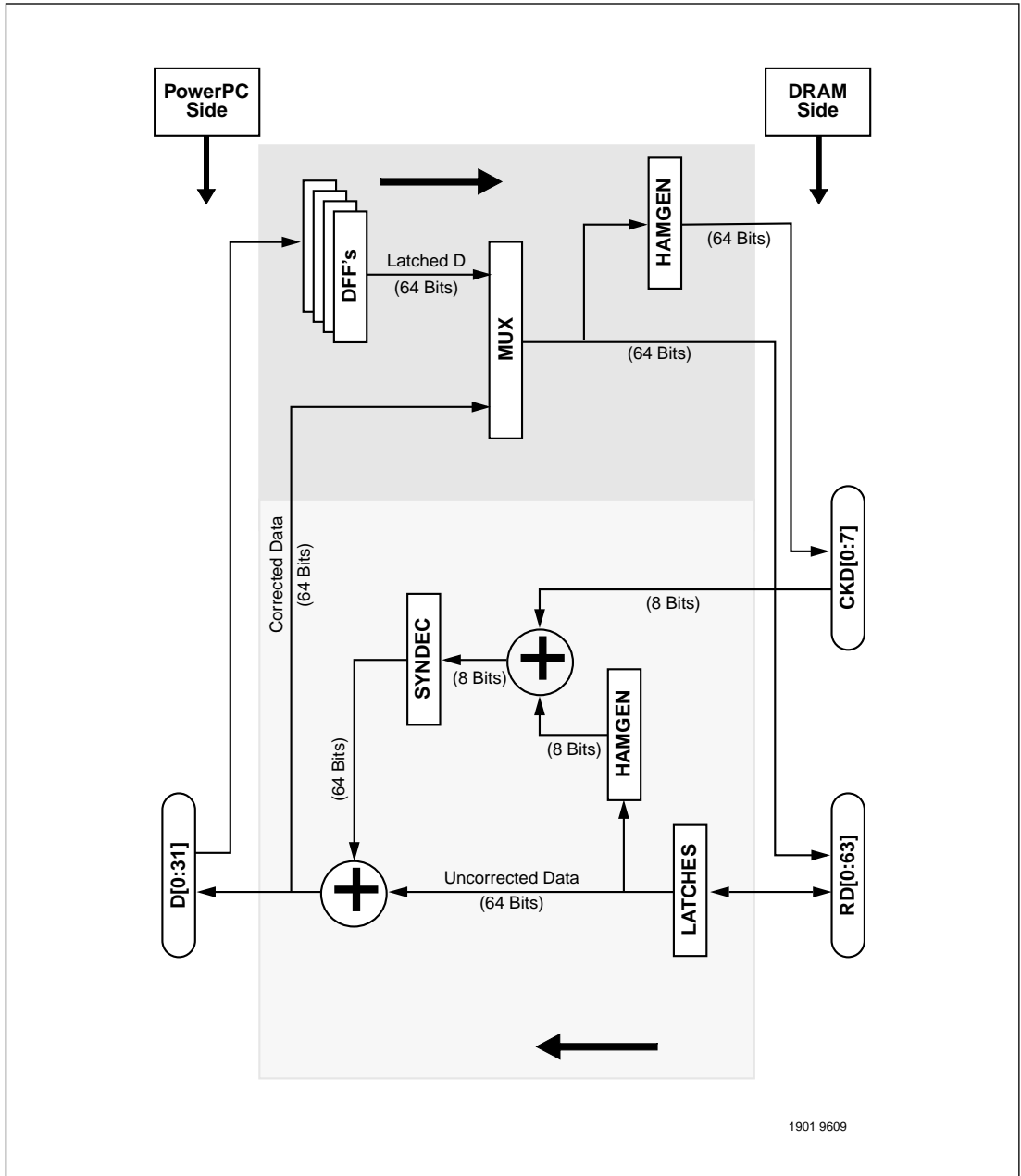
- Software programmable Interrupt on Single/Double-Bit Error.
  - Error address and Syndrome Log Registers for Error Logging.
  - *Does not provide TEA\_ on Double-Bit Error.* (Chip has no TEA\_ pin.)
- ROM/Flash Interface
- Two blocks with two 8-bit devices, or two 32-bit devices per block.

## 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.

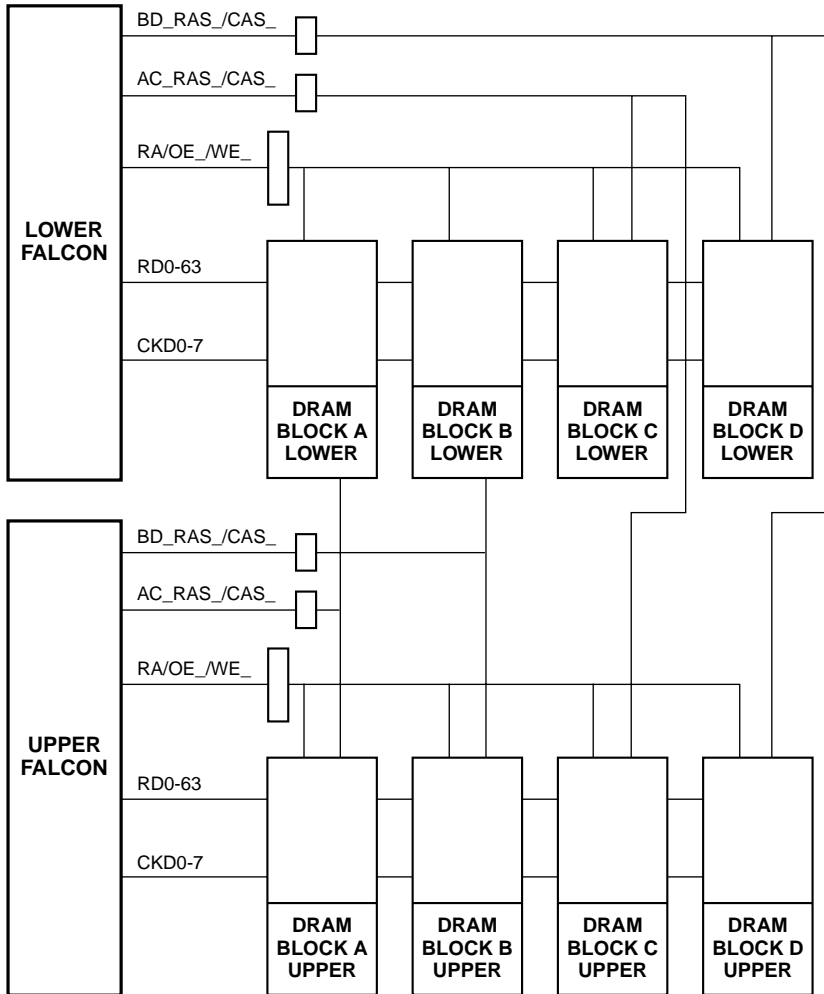


**Figure 3-1. Falcon Pair Used with DRAM in a System**



1901 9609

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



1902 9609

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 60x 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 60x bus at 66MHz. This is done by providing a two-way interleave between the 64-bit PowerPC 60x data bus and the 128-bit (144 with check-bits) DRAM bus. When a PowerPC 60x 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 60x bus master, but so is the next. While the Falcon pair is presenting the first two double-words to the PowerPC 60x 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 60x bus after 0 or more idle clocks.

The Falcon pair also takes advantage of the fact that PowerPC 60x processors can do address pipelining. Many times while a data cycle is finishing, the PowerPC 60x 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 60x bus do not achieve data rates as high as do four-beat cycles. The Falcon pair does take advantage of the PowerPC 60x 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. When the Falcon pair can take advantage of address pipelining, back-to-back single-beat writes take 10 clocks to complete.

### 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 Hyper-Page parts. When the Falcon pair is configured for 70ns DRAMs it assumes that the devices are Page parts. When the pair is configured for 60ns DRAMs, it allows the devices to be either Page or Hyper-Page parts. Performance summaries using the different devices are shown in Tables 3-1, 3-2, and 3-3.

**Table 3-1. PowerPC 60x Bus to DRAM Access Timing When Configured for 70ns 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 <sup>1</sup>	1	3	1	14/8
4-Beat Read after 4-Beat Read (misaligned)	7/2 <sup>1</sup>	4	1	1	13/8
4-Beat Write after Idle	4	1	1	1	7
4-Beat Write after 4-Beat Write (Quad-word aligned)	10/6 <sup>1</sup>	1	1	1	13/9
1-Beat Read after Idle	10	-	-	-	10
1-Beat Read after 1-Beat Read	11/7 <sup>1</sup>	-	-	-	11/7
1-Beat Write after Idle	4	-	-	-	4
1-Beat Write after 1-Beat Write	15/11 <sup>1</sup>	-	-	-	15/11

## Notes:

1. These numbers assume that the PowerPC 60x 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 hit/page miss.
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 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 <sup>1</sup>	1	2	1	11/7
4-Beat Read after 4-Beat Read (misaligned)	6/2 <sup>1</sup>	3	1	1	11/7
4-Beat Write after Idle	4	1	1	1	7
4-Beat Write after 4-Beat Write (Quad-word aligned)	7/3 <sup>1</sup>	1	1	1	10/6
1-Beat Read after Idle	9	-	-	-	9
1-Beat Read after 1-Beat Read	9/6 <sup>1</sup>	-	-	-	9/6
1-Beat Write after Idle	4	-	-	-	4
1-Beat Write after 1-Beat Write	13/10 <sup>1</sup>	-	-	-	13/10

## Notes:

1. These numbers assume that the PowerPC 60x 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 hit/page miss.
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 Hyper 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 <sup>1</sup>	1	1	1	8/5
4-Beat Read after 4-Beat Read (misaligned)	4/2 <sup>1</sup>	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 <sup>1</sup>	1	1	1	7/6
1-Beat Read after Idle	8	-	-	-	8
1-Beat Read after 1-Beat Read	7/5 <sup>1</sup>	-	-	-	7/5
1-Beat Write after Idle	4	-	-	-	4
1-Beat Write after 1-Beat Write	9/7 <sup>1</sup>	-	-	-	9/7

## Notes:

1. These numbers assume that the PowerPC 60x 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 hit/page miss.
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. Each block can address up to 64Mbytes of memory depending on the type of the devices implemented for that block (8-bit, 32-bit, or 64-bit devices). The access times for ROM/Flash are shown in Tables 3-4 and 3-5.

**Table 3-4. PowerPC 60x Bus to ROM/Flash Access Timing When Configured for 32/64-bit Devices**

ACCESS TYPE	CLOCK PERIODS REQUIRED FOR:				Total Clocks
	1st Beat	2nd Beat	3rd Beat	4th Beat	
4-Beat Read	20	16	16	16	68
4-Beat Write	N/A	N/A	N/A	N/A	N/A
1-Beat Read	20	-	-	-	20
1-Beat Write	19	-	-	-	19

**Table 3-5. PowerPC 60x Bus to ROM/Flash Access Timing When Configured for 8-bit Devices**

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

### 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 60x 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 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 60x bus as soon as the entity has data ready and the PowerPC 60x 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 60x data bus has been granted.

### Cache Coherency

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

## Cache Coherency Restrictions

The PowerPC 60x 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 60x 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 ( $2 \times [64+8]$ ).

## Cycle Types

To support ECC, the Falcon pair always deals with DRAM using full width (144-bit) accesses. When the PowerPC 60x bus master requests any size read of DRAM, the Falcon pair reads 144 bits at least once. When the PowerPC 60x bus master requests a four-beat write to DRAM, the Falcon pair writes all 144 bits twice. When the PowerPC 60x 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 60x 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 3-6 shows the actions it takes for different errors during these accesses.

Note that 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 3-6. Error Reporting**

Error Type	Single-Beat/Four-Beat Read	Single-Beat Write	Four-Beat Write	Scrub
<i>Single-Bit Error</i>	<p>Terminate the PowerPC 60x bus cycle normally.</p> <p>Provide corrected data to the PowerPC 60x bus master.</p> <p>Assert INT_ if so enabled.</p>	<p>Terminate the PowerPC 60x bus cycle normally.</p> <p>Correct the data read from DRAM, merge with the write data, and write the corrected, merged data to DRAM.</p> <p>Assert INT_ if so enabled.</p>	N/A <sup>1</sup>	<p>This cycle is not seen on the PowerPC 60x bus.</p> <p>Write corrected data back to DRAM if so enabled.</p> <p>Assert INT_ if so enabled.</p>
<i>Double-Bit Error</i>	<p>Terminate the PowerPC 60x bus cycle normally.</p> <p>Provide miss-corrected, raw DRAM data to the PowerPC 60x bus master.</p> <p>Assert INT_ if so enabled.</p> <p>Assert MCP_ if so enabled.</p>	<p>Terminate the PowerPC 60x bus cycle normally.</p> <p>Do not perform the write portion of the read-modify-write cycle to DRAM.</p> <p>Assert INT_ if so enabled.</p> <p>Assert MCP_ if so enabled.</p>	N/A <sup>1</sup>	<p>This cycle is not seen on the PowerPC 60x bus.</p> <p>Do not perform the write portion of the read-modify-write cycle to DRAM.</p> <p>Assert INT_ if so enabled.</p>
<i>Triple- (or greater) Bit Error</i>	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.

## 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* in this chapter.

## DRAM Tester



### Caution

The DRAM tester is for in-house manufacturing testing purposes only and should not be used by customers.

## ROM/Flash Interface

The Falcon pair provides the interface for two blocks of ROM/Flash. Each block provides addressing and control for up to 16Mbytes of memory using two 8-bit devices, or up to 64Mbytes using two 32-bit devices or one 64-bit device. Note that no error checking (ECC or Parity) 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 two software programmable control register bits. One bit is an overall enable. The other is a write enable to be used in programming Flash devices. The overall enable bits are always cleared at reset.

Each block also has a reset vector enable bit that controls whether it is enabled at \$FFF00000-\$FFFFFFFF. This 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.

The write enable bit is cleared at reset for both blocks.

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 assumed device type 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 device type status bit is cleared, the block's ROM/Flash is considered to be two 8-bit devices with one device connected to each Falcon. In this mode, the following rules are enforced:



- a. both devices must be the same,
- b. only single-byte writes are allowed (all other sizes are ignored), and
- c. all reads are allowed (multiple accesses are performed to the ROM/Flash devices when the read is for greater than one byte).

When the device type status bit is set, the block's ROM/Flash is considered to be two 32-bit read/write devices or one 64-bit read-only device. If two 32-bit devices are being used, then one device is connected to each Falcon and the following rules are enforced:

- a. only aligned, 4-byte writes should be attempted (all other sizes are ignored), and
- b. all reads are allowed (multiple accesses to the ROM/Flash device are performed for burst reads).

If one 64-bit device is being used, then control for the device is connected to either Falcon while half of the data is connected to the upper Falcon and half of the data is connected to the lower Falcon. The following rules are enforced:

- a. no writes should be attempted (aligned, 4-byte, writes are allowed but produce undefined results, all other sizes are ignored), and
- b. 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 this chapter.

In order to place code correctly in the ROM/Flash devices, address mapping information is required. Table 3-7 shows how PowerPC 60x addresses map to the ROM/Flash addresses with two 8-bit devices. Table 3-8 shows how they map with two 32-bit devices or one 64-bit (read-only) device.

**Table 3-7. PowerPC 60x to ROM/Flash Address Mapping with Two 8-bit Devices**

PowerPC 60x A0-A31	ROM/Flash A22-A0	ROM/Flash Device 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
⋮	⋮	⋮
⋮	⋮	⋮
⋮	⋮	⋮
\$XXFFFFFF8	\$7FFFFFFC	Upper
\$XXFFFFFF9	\$7FFFFFFD	Upper
\$XXFFFFFFA	\$7FFFFFFE	Upper
\$XXFFFFFFB	\$7FFFFFFF	Upper
\$XXFFFFFFC	\$7FFFFFFC	Lower
\$XXFFFFFFD	\$7FFFFFFD	Lower
\$XXFFFFFFE	\$7FFFFFFE	Lower
\$XXFFFFFFF	\$7FFFFFFF	Lower

**Table 3-8. PowerPC 60x Address to ROM/Flash Address Mapping with Two 32-bit or One 64-bit Device(s)**

PowerPC 60x A0-A31	ROM/Flash A22-A0	ROM/Flash Device Selected (Applies only when using two 32-bit Devices)
\$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
.	.	.
.	.	.
.	.	.
\$X3FFFFFF0	\$7FFFFFFE	Upper
\$X3FFFFFF1	\$7FFFFFFE	Upper
\$X3FFFFFF2	\$7FFFFFFE	Upper
\$X3FFFFFF3	\$7FFFFFFE	Upper
\$X3FFFFFF4	\$7FFFFFFE	Lower
\$X3FFFFFF5	\$7FFFFFFE	Lower
\$X3FFFFFF6	\$7FFFFFFE	Lower
\$X3FFFFFF7	\$7FFFFFFE	Lower

PowerPC 60x A0-A31	ROM/Flash A22-A0	ROM/Flash Device Selected (Applies only when using two 32-bit Devices)
\$X3FFFFFF8	\$7FFFFFF	Upper
\$X3FFFFFF9	\$7FFFFFF	Upper
\$X3FFFFFFA	\$7FFFFFF	Upper
\$X3FFFFFFB	\$7FFFFFF	Upper
\$X3FFFFFFC	\$7FFFFFF	Lower
\$X3FFFFFFD	\$7FFFFFF	Lower
\$X3FFFFFFE	\$7FFFFFF	Lower
\$X3FFFFFFF	\$7FFFFFF	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 60 $\mu$ s. This increases to once every 30 $\mu$ s 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 30 $\mu$ s. This increases to once every 15 $\mu$ s 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.

Note that an entire refresh of DRAM is achieved every time the row address rolls over, and that 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 60x bus master is notified via interrupt.

## DRAM Arbitration

The Falcon pair has 3 different entities that can request use of the DRAM cycle controller:

- ❑ The PowerPC 60x bus master
- ❑ The tester
- ❑ The refresher/scrubber

The Falcon pair's arbiters assign priority with the refresher/scrubber highest, the tester next, and the PowerPC 60x bus lowest. When no requests are pending, the arbiter defaults to providing a PowerPC 60x bus grant. This provides fast response for PowerPC 60x bus cycles. Although the arbiter operates on a priority basis, it also performs a pseudo round-robin algorithm in order to prevent starving any of the requesting entities. Note that PowerPC DRAM or ROM/Flash accesses should not be attempted while the tester is in operation.

## 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* in this chapter 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 and SRAM only. No duplication is performed for writes to DRAM, ROM/Flash, or the External Register set.

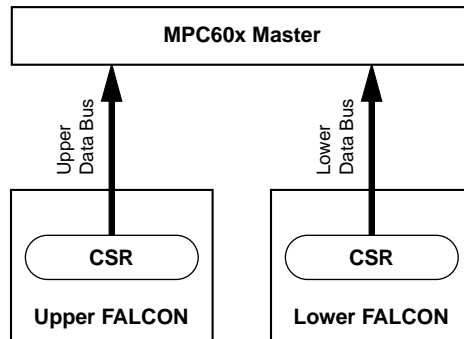
# Programming Model

## 3

### CSR Architecture

The CSR (control and status register set) consists of the chip's internal register set, its test SRAM, 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 Figure 3-4.)



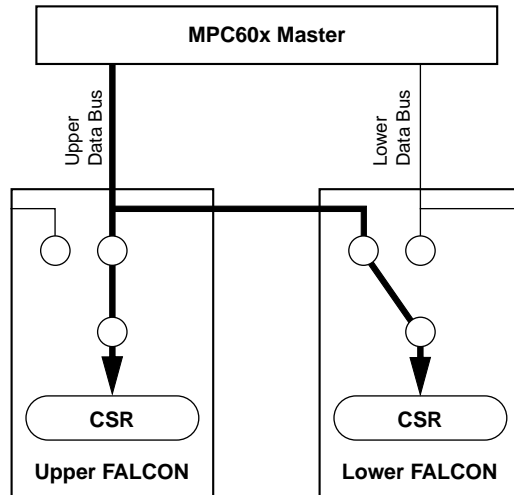
1903 9609

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

For writes, internal register or test SRAM 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 or test



SRAM 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 Figure 3-5.).

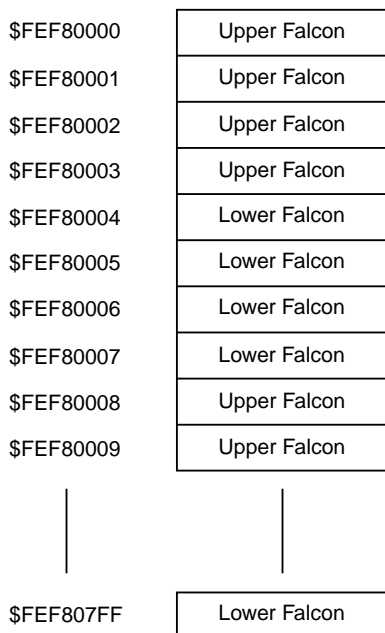


1904 9609

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

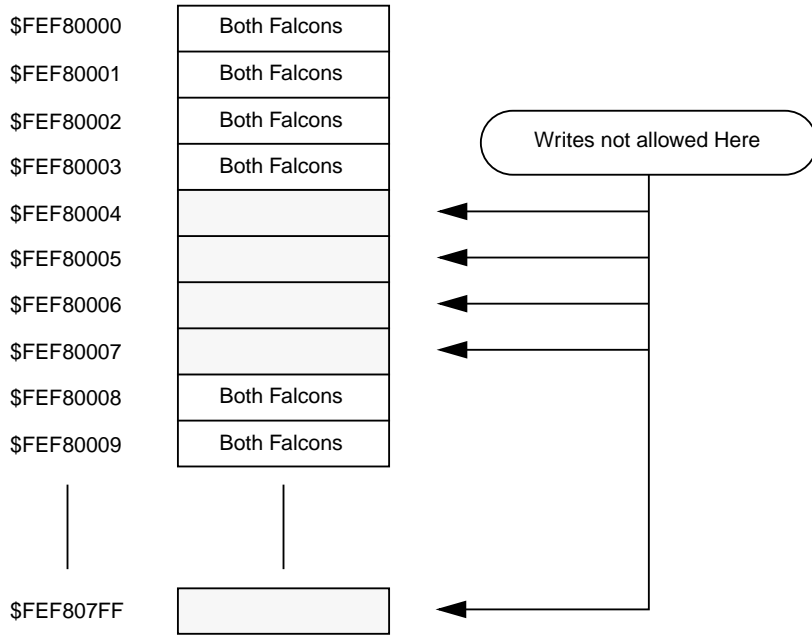
External register data that is written on the upper data bus goes through the upper Falcon, while data that is written on the lower data bus goes through the lower Falcon. There is no automatic copying of upper data to lower data for the external register set.

CSR read accesses can have a size of 1, 2, 4, or 8 bytes with any alignment. CSR write accesses must have a size of 1 or 4 bytes and they must be aligned. Some Tester registers are limited to 4-byte only accesses. Figures 3-6, 3-7, 3-8, and 3-9 show the memory map for the different kinds of access.



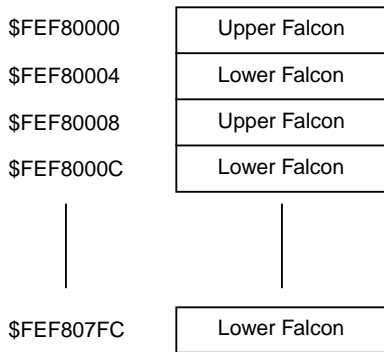
1905 9609

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



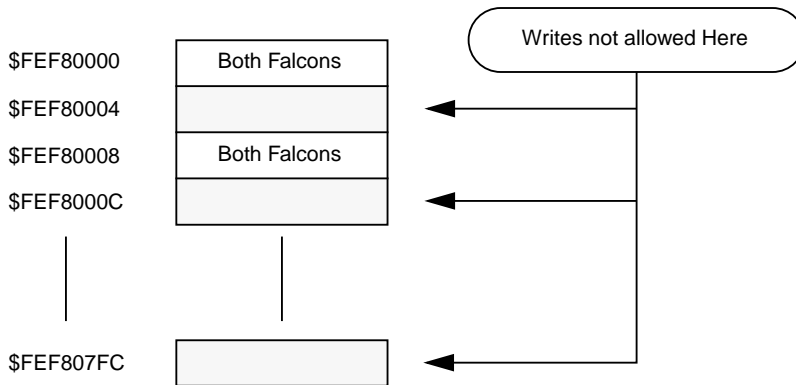
1906 9609

**Figure 3-7. Memory Map for Byte Writes to the Internal Register Set and Test SRAM**



1907 9609

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



1908 9609

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

## Register Summary

Table 3-9 on the following page shows a summary of the CSR. Note that 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 and test SRAM 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 3-9. Register Summary**

3

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										isa_hole				adlis		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					rdldis		rvwcb		dtrc						scien		slen		tlen		mlen										mcken								
FEF80030	elag		esch				esen		embt		esbr		ERROR_SYNDROME															esbrk0		esbrk1		scdf		SBE COUNT					
FEF80038	ERROR_ADDRESS																																						
FEF80040	scb0		scb1						swen						rtes10		rtes11		rtes12																				
FEF80048	ROW ADDRESS															COL ADDRESS																							
FEF80050	ROM A BASE										rom_a_64		ROM A SIZ										rom_a_rv		rom_a_en		rom_a_we												
FEF80058	ROM B BASE										rom_b_64		ROM B SIZ										rom_b_rv		rom_b_en		rom_b_we												
FEF80060	trun		tsse						tpass		trail						tzbit														th0		th1						
FEF80068	TEST PC								TEST IR																														
FEF80070	TEST A0																																						
FEF80078	TEST A1																																						
FEF80080																																							
FEF80088	TEST D0 (Upper 8 Bits)																																						
FEF80090	TEST D0 (Middle 32 Bits)																																						
FEF80098	TEST D0 (Lower 32 Bits)																																						

**Table 3-9. Register Summary (Continued)**

FEF800A0								
FEF800A8							<i>TEST D1 (Upper 8 Bits)</i>	
FEF800B0	<i>TEST D1 (Middle 32 Bits)</i>							
FEF800B8	<i>TEST D1 (Lower 32 Bits)</i>							
FEF800C0								
FEF800C8							<i>TEST D2 (Upper 8 Bits)</i>	
FEF800D0	<i>TEST D2 (Middle 32 Bits)</i>							
FEF800D8	<i>TEST D2 (Lower 32 Bits)</i>							
FEF800E0								
FEF800E8							<i>TEST D3 (Upper 8 Bits)</i>	
FEF800F0	<i>TEST D3 (Middle 32 Bits)</i>							
FEF800F8	<i>TEST D3 (Lower 32 Bits)</i>							
FEF80100	<i>CTR32</i>							
FEF80200								
.								
FEF803F8								
FEF80400	<i>PR_STAT1</i>							
FEF80408								
.								
FEF804F8								
FEF80500	<i>PR_STAT2</i>							
FEF80508								
.								
FEF807F8								
FEF80800					<i>TEST SRAM</i>			
.								
FEF80BF8								







***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 7.8 $\mu$ s per row rather than the standard 15.6 $\mu$ s 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 3-10. *ram\_spd1,ram\_spd0* and DRAM Type**

<b><i>ram_spd0, ram_spd1</i></b>	<b>DRAM Speed</b>	<b>DRAM Type</b>
%00	70ns	Page Mode
%01	60ns	Page Mode
%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** *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 60x 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 60x 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					ram a siz0	ram a siz1	ram a siz2	ram b en					ram b siz0	ram b siz1	ram b siz2	ram c en					ram c siz0	ram c siz1	ram c siz2	ram d en					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

**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 3-11 shows the block configuration assumed by the Falcon pair for each value of **ram siz0-ram siz2**.

**Table 3-11. Block\_A/B/C/D Configurations**

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

### 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 60x 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 that 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																								PDR							
OPERATION	READ/WRITE								READ ZERO								READ ZERO								R							
RESET	42 P								X								X								X							

**CLK FREQUENCY** These bits should be programmed with the hexadecimal value of the operating CLOCK frequency in MHz (i.e. \$42 for 66MHz). When these bits are programmed this way, the chip's prescale counter produces a 1MHz 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	deic	0	0	0	0	scien	tien	sien	mien																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	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	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*

*rwcb*, when set, enables the data from the 8 check-bits in this Falcon to be written and read on the PowerPC 60x data bus (DH24-31 for upper Falcon, DL24-31 for lower Falcon). This bit should be cleared for normal system operation. Note that if test software forces a single-bit error to a location using this function, the scrubber may correct the location before the test software gets a chance to check for the single-bit error at that location. 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. Stop all DRAM Tester operations by clearing the **trun** bit.
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 60x data bus unaltered 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** 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.

**tien** When **tien** is set, the setting of the *tpass* or the *tfail* bit 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 assert its machine check interrupt request pin (MCP\_). When it 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-pollled 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	<i>elog</i>	0	0	0	<i>escb</i>	<i>esen</i>	<i>embt</i>	<i>esbt</i>	<b>ERROR_SYNDROME</b>									<i>scof</i>	<b>SBE COUNT</b>																				
OPERATION	R/C	R	R	R	R	R/W	R	R	READ ONLY									R	R	R	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	X	X	X	0 P	0 P										

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

***escb*** *escb* indicates the entity that was accessing DRAM at the last logging of a single- or double-bit error. 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. Note that the DRAM Tester cannot cause an error to be logged.

***esen*** When set, *esen* allows errors to be logged during scrubs. When *esen* is cleared, errors that occur during a scrub are not logged.

***embt*** *embt* is set when the last error logged was due to a multiple-bit error. It is cleared when a 1 is written to the *elog* bit. The syndrome code is meaningless if *embt* is set.

*esbt*      *esbt* is set when the last error logged was due to a single-bit error. It is cleared when a 1 is written to the *elog* bit or when a multiple-bit error is logged. The syndrome code reflects the bit in error. (Refer to the section on *ECC Codes*.)

***ERROR\_SYNDROME***      ***ERROR\_SYNDROME*** reflects the syndrome value at the last logging of an error. 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 read-only bits indicate which block of DRAM was being accessed at the last logging of 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*      *scof* is set by the ***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, the 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 60x address bus at the last logging of 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 3-23 in the subsection on *Sizing DRAM* in the section on *Software Considerations*. 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	itest0	itest1	itest2																				
OPERATION	R	R	R	R	R	R	R	R/W	R	R	R	R	R	R	R/W	R/W	READ ZERO										READ ZERO									
RESET	0P	0P	X	X	X	X	X	0P	X	X	X	X	X	0P	0P	0P	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. They are 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 3-12 shows their encodings. Note that these test modes are not intended to be used once the chip is in a system.

**Table 3-12. rtest encodings**

<b>rtest0,rtest1,rtest2</b>	<b>Test Mode selected</b>
%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**

<b>ADDRESS</b>	<b>\$FEF80048</b>																															
<b>BIT</b>	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
<b>NAME</b>	0	0	0	<i>ROW ADDRESS</i>													0	0	0	<i>COL ADDRESS</i>												
<b>OPERATION</b>	R	R	R	READ/WRITE													R	R	R	READ/WRITE												
<b>RESET</b>	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 that 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 that 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_size0	rom_a_size1	rom_a_size2												rom_a_tv	rom_a_en	rom_a_we					
OPERATION	READ/WRITE											R	RW	RW	RW	READ ZERO											R	R	R	R	R	RW	RW	RW
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 60x 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 that in addition to the programmed address, the first 1Mbyte of Block A also appears at \$FFF00000 - \$FFFFFFF if the *rom\_a\_tv* bit is set and the *rom\_b\_tv* bit is cleared.

Also note that the combination of **ROM\_A\_BASE** and **rom\_a\_siz** should never be programmed such that ROM 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** *rom\_a\_64* indicates the type of ROM/FLASH device/ devices being used for Block A. When *rom\_a\_64* is cleared, Block A consists of two 8-bit ROM/Flash devices with one device connected to the upper Falcon and the other device connected to the lower Falcon. When *rom\_a\_64* is set, Block A consists of either two 32-bit devices or one 64-bit read-only device (such as a 64-bit ROM SIMM). With two 32-bit devices, one device is connected to the upper Falcon and the other device connected to the lower Falcon. With one 64-bit device, the device is controlled by either Falcon but connects to data signals from both the upper and lower Falcons. *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 Table 3-13.

**Table 3-13. ROM Block A Size Encoding**

<b>rom_a_siz</b>	<b>BLOCK SIZE</b>
%000	1MB
%001	2MB
%010	4MB
%011	8MB
%100	16MB
%101	32MB
%110	64MB
%111	Reserved

***rom\_a\_rv*** *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 - \$FFFFFFF as shown in the table below.

**Table 3-14. *rom\_a\_rv* and *rom\_b\_rv* encoding**

<i>rom_a_rv</i>	<i>rom_b_rv</i>	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 3-15 for details of ROM/Flash accesses.

**Table 3-15. Read/Write to ROM/Flash**

Cycle	Transfer Size	Alignment	<i>rom_x_64</i>	<i>rom_x_we</i>	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											<i>rom_b_64</i>	<i>rom_b_siz0</i>	<i>rom_b_siz1</i>	<i>rom_b_siz2</i>								<i>rom_b_iv</i>	<i>rom_b_en</i>	<i>rom_b_we</i>														
OPERATION	READ/WRITE											R	R/W	R/W	R/W	READ ZERO							R	R	R	R	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
RESET	\$FF4 PL											V P	0 PL	0 PL	0 PL	X							X	X	X	X	X	X	V P	0 PL	0 PL	0 PL	0 PL	0 PL	0 PL	0 PL	0 PL	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 60x 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 that in addition to the programmed address, the first 1Mbyte of Block B also appears at \$FFF00000 - \$FFFFFFF 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 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* *rom\_b\_64* indicates the type of ROM/FLASH device/devices being used for Block B. When *rom\_b\_64* is cleared, Block B consists of two 8-bit ROM/Flash devices with one device connected to the upper Falcon and the other device connected to the lower Falcon. When *rom\_b\_64* is set, Block B consists of either two 32-bit devices or one 64-bit read-only device (such as a 64-bit ROM SIMM). With two 32-bit devices, one device is connected to the upper Falcon and the other device connected to the lower Falcon. With one 64-bit device, the device is controlled by either Falcon but connects to data signals from both the upper and lower Falcons. *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 Table 3-16.

**Table 3-16. ROM Block B Size Encoding**

<b><u>rom b siz</u></b>	<b>BLOCK SIZE</b>
%000	1Mbytes
%001	2Mbytes
%010	4Mbytes
%011	8Mbytes
%100	16Mbytes
%101	32Mbytes
%110	64Mbytes
%111	Reserved

**rom b rv** **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 - \$FFFFFFF as shown in Table 3-14.

**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 3-15 for more details.

## DRAM Tester Control Registers



The tester should not be used by software. The **trun** and **tsse** bits (bits 0 and 1 of the register at address \$FEF80060) should never be set.

### 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	<i>CTR32</i>																															
OPERATION	READ/WRITE																															
RESET	0 PL																															

***CTR32*** *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 66.67MHz, *CTR32* will count at a fractional amount faster or slower than 1MHz, depending on the programming of the CLK Frequency Register.

### Test SRAM

3

ADDRESS	\$FEF80800 - \$FEF80BF8																															
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																	<p style="text-align: center;"><u>TEST SRAM</u> (64 Locations)</p>															
OPERATION																	READ ZERO								READ ZERO							
RESET	X PL								X PL								X PL															

**TEST SRAM** The code that is executed by the DRAM Tester is contained in **TEST SRAM**. **TEST SRAM** is 16 bits wide and 64 locations deep.

Software can read and write to **TEST SRAM**. The DRAM Tester can read **TEST SRAM** as it fetches instructions. It cannot write to it.

### 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** 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 MVME2600 series boards, refer to the *Falcon-Controlled System Registers* in Chapter 1, especially the *System Configuration Register* and the *Memory Configuration Register*.

### 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** 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	<u>EXTERNAL REGISTER SET</u>																															
OPERATION	READ/WRITE																															
RESET	X PL																															

**EXTERNAL REGISTER SET**

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 MVME2600 series boards, refer to the *Falcon-Controlled System Registers* in Chapter 1, especially the *System External Cache Control Register* and the *CPU Control Register*.

---

## 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 or data buses. Because of this, the appropriate registers in the MPC60x should be programmed to disable parity checking for the address bus and 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 3-7 and in Table 3-8. When using 8-bit or 32-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 - \$FFFFFFF).

## 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, tien, 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 3-17. Table 3-18 also 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.

**Table 3-17. Sizing Addresses**

1024MB	256MB	128MB	64MB	32MB	16MB
\$00000000	\$00000000,	\$00000000	\$00000000	\$00000000	\$00000000
\$20000000	\$02000000,	\$00002000	\$00002000	\$00001000	
	\$08000000,	\$02000000	\$02000000	\$00002000	
	\$0A000000	\$02002000	\$02002000	\$00003000	
		\$04000000		\$01000000	
		\$04002000		\$01001000	
		\$06000000		\$01002000	
		\$06002000		\$01003000	

**Table 3-18. PowerPC 60x Address to DRAM Address Mappings**

RA ----> Block Size   V		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 3-19 shows the syndrome for each possible single bit error. Table 3-20 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 pairs positions PowerPC data in DRAM. See the section on *Data Paths* for an explanation of this.

Note that Tables 3-19 and 3-20 are the same whether the Falcon is configured as upper or as lower.

**Table 3-19. Syndrome Codes Ordered by Bit in Error**

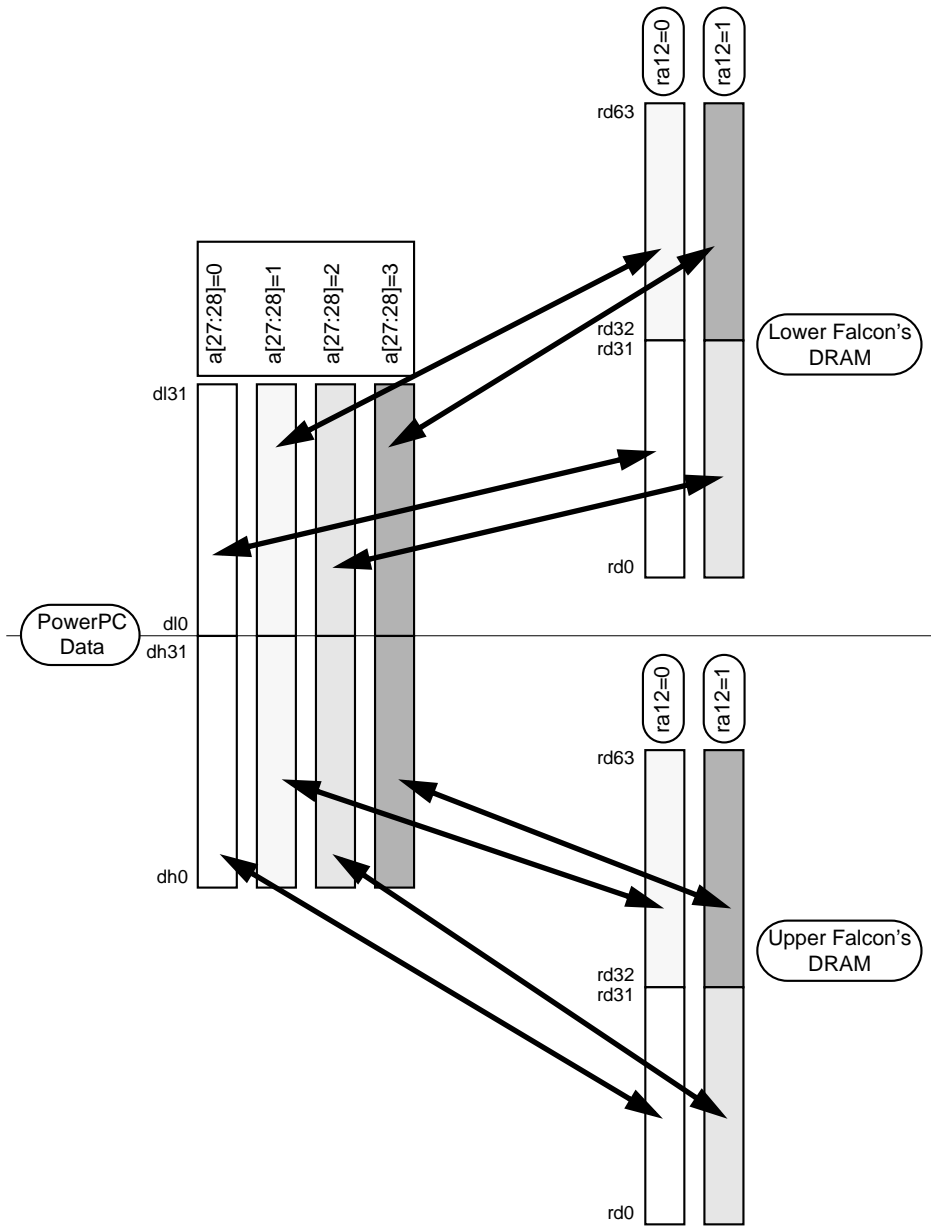
Bit	Syndrome	Bit	Syndrome	Bit	Syndrome	Bit	Syndrome	Bit	Syndrome
rd0	\$4A	rd16	\$92	rd32	\$A4	rd48	\$29	ckd0	\$01
rd1	\$4C	rd17	\$13	rd33	\$C4	rd49	\$31	ckd1	\$02
rd2	\$2C	rd18	\$0B	rd34	\$C2	rd50	\$B0	ckd2	\$04
rd3	\$2A	rd19	\$8A	rd35	\$A2	rd51	\$A8	ckd3	\$08
rd4	\$E9	rd20	\$7A	rd36	\$9E	rd52	\$A7	ckd4	\$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
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 3-20. Single-Bit Errors Ordered by Syndrome Code**

Syn-drome	Bit	Syn-drome	Bit	Syn-drome	Bit	Syn-drome	Bit	Syn-drome	Bit	Syn-drome	Bit	Syn-drome	Bit	Syn-drome	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	-
\$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 3-21 shows the same information in tabular format.



1909 9609

**Figure 3-10. PowerPC Data to DRAM Data Correspondence**

**Table 3-21. PowerPC Data to DRAM Data Mapping****3**

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]	-
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]





**Note** All of the information in this chapter (except the section *Universe Chip Problems after a PCI Reset*) is taken from the *Universe User Manual*, which is listed in *Manufacturers' Documents in Appendix A*. Refer to that manual for complete information.

## General Information

### Introduction

The Universe VMEbus interface chip (CA91C042) provides a reliable high performance 64-bit VMEbus to PCI interface in one device.

Designed by Tundra Semiconductor Corporation in consultation with Motorola, the Universe is compliant with the VME64 specification and is tuned to the new generation of high speed processors.

The Universe is ideally suited for CPU boards acting as both master and slave in the VMEbus system, and is particularly fitted for PCI local systems. The Universe is manufactured in a CMOS process.

### Product Overview - Features

- ❑ Fully compliant, 64-bit, 33 MHz PCI local bus interface
- ❑ Fully compliant, high performance 64-bit VMEbus interface
- ❑ Integral FIFOs for write posting to maximize bandwidth utilization
- ❑ Programmable DMA controller with linked list support

- ❑ VMEbus transfer rates of 60-70 MBytes/sec
- ❑ Complete suite of VMEbus address and data transfer modes
  - A32/A24/A16 master and slave
  - D64 (MBLT)/D32/D16/D08 master and slave
  - BLT, ADOH, RMW, LOCK
- ❑ Automatic initialization for slave-only applications
- ❑ Flexible register set, programmable from both the PCI bus and VMEbus ports
- ❑ Full VMEbus system controller functionality
- ❑ IEEE 1149.1 JTAG testability support, and
- ❑ Available in 313-pin Plastic BGA and 324-pin contact Ceramic BGA

## Functional Description

### Architectural Overview

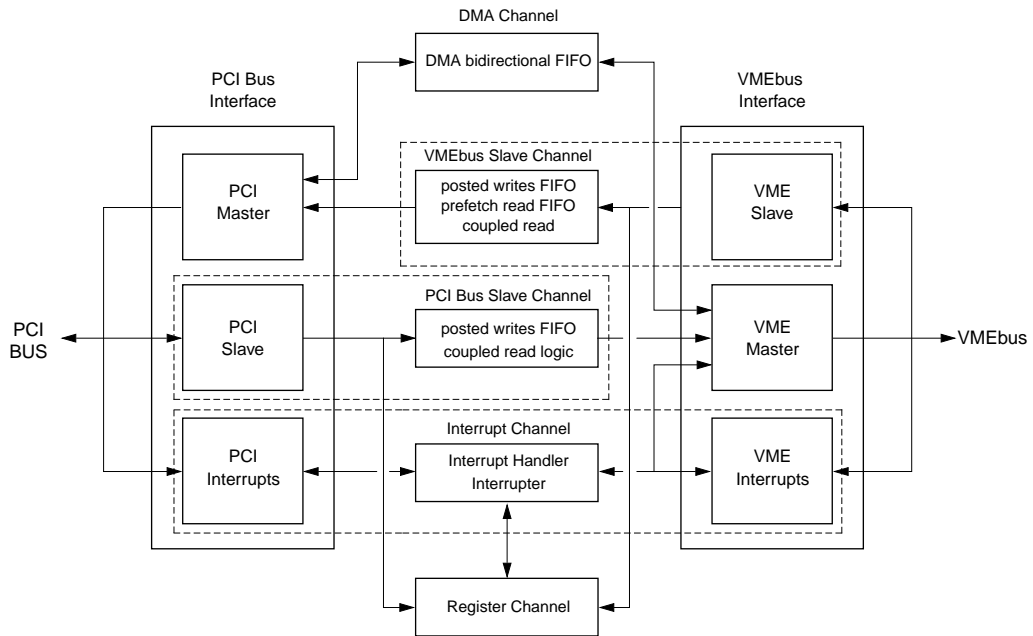
This section introduces the general architecture of the Universe. This description makes reference to the functional block diagram provided in Figure 4-1 that follows. Notice that for each of the interfaces, VMEbus and PCI bus, there are three functionally distinct modules: master module, slave module, and interrupt module. These modules are connected to the different functional channels operating in the Universe. These channels are:

- ❑ VME Slave Channel
- ❑ PCI Bus Slave Channel
- ❑ DMA Channel
- ❑ Interrupt Channel
- ❑ Register Channel

The Architectural Overview is organized into the following sections:

- ❑ VMEbus Interface
- ❑ PCI Bus Interface
- ❑ Interrupter and Interrupt Handler
- ❑ DMA Controller

These sections describe the operation of the Universe in terms of the different modules and channels illustrated in Figure 4-1.



1894 9609

**Figure 4-1. Architectural Diagram for the Universe**

## VMEbus Interface

### Universe as VMEbus Slave

The Universe VME Slave Channel accepts all of the addressing and data transfer modes documented in the VME64 specification (except A64 and those intended to support 3U applications, i.e. A40 and MD32). Incoming write transactions from the VMEbus may be treated as either coupled or posted, depending upon the programming of the VMEbus slave image. (Refer to *VME Slave Images* in the *Universe User Manual*.) With posted write transactions, data is written to a Posted Write Receive FIFO (RXFIFO), and the VMEbus master receives data acknowledgment from the Universe. Write data is transferred to the PCI resource from the RXFIFO without the involvement of the initiating VMEbus master (Refer to *Posted Writes* in the *Universe User Manual* for a full explanation of this operation.). With a coupled cycle, the VMEbus master only receives data acknowledgment when the transaction is complete on the PCI bus. This means that the VMEbus is unavailable to other masters while the PCI bus transaction is executed.

Read transactions may be prefetched or coupled. If enabled by the user, a prefetched read is initiated when a VMEbus master requests a block read transaction (BLT or MBLT) and this mode is enabled. When the Universe receives the block read request, it begins to fill its Read Data FIFO (RDFIFO) using burst transactions from the PCI resource. The initiating VMEbus master then acquires its block read data from the RDFIFO rather than from the PCI resources directly.

### Universe as VMEbus Master

The Universe becomes VMEbus master when the VME Master Interface is internally requested by the PCI Bus Slave Channel, the DMA Channel, or the Interrupt Channel. The Interrupt Channel always has priority over the other two channels. Several mechanisms are available to configure the relative priority that the PCI Bus Slave Channel and DMA Channel have over ownership of the VMEbus Master Interface.

The Universe's VME Master Interface generates all of the addressing and data transfer modes documented in the VME64 specification (except A64 and those intended to support 3U applications, i.e. A40 and MD32). The Universe is also compatible with all VMEbus modules conforming to pre-VME64 specifications. As VMEbus master, the Universe supports Read-Modify-Write (RMW), and Address-Only-with-Handshake (ADOH) but does not accept RETRY\* as a termination from the VMEbus slave. The ADOH cycle is used to implement the VMEbus Lock command allowing a PCI master to lock VMEbus resources.

## PCI Bus Interface

### Universe as PCI Slave

Read transactions from the PCI bus are always processed as coupled. Write transactions may be either coupled or posted, depending upon the setting of the PCI bus slave image. (Refer to *PCI Bus Slave Images* in the *Universe User Manual*.) With a posted write transaction, write data is written to a Posted Write Transmit FIFO (TXFIFO) and the PCI bus master receives data acknowledgment from the Universe with zero wait states. Meanwhile, the Universe obtains the VMEbus and writes the data to the VMEbus resource independent of the initiating PCI master. (Refer to *Posted Writes* in the *Universe User Manual* for a full description of this operation.)

To allow PCI masters to perform RMW and ADOH cycles, the Universe provides a Special Cycle Generator. The Special Cycle Generator can be used in combination with a VMEbus ownership function to guarantee PCI masters exclusive access to VMEbus resources over several VMEbus transactions, and. (Refer to *Exclusive Accesses* and *RMW and ADOH Cycles* in the *Universe User Manual* for a full description of this functionality.)

## Universe as PCI Master

The Universe becomes PCI master when the PCI Master Interface is internally requested by the VME Slave Channel or the DMA Channel. There are mechanisms provided which allow the user to configure the relative priority of the VME Slave Channel and the DMA Channel.

## Interrupter and Interrupt Handler

### Interrupter

The Universe interrupt channel provides a flexible scheme to map interrupts to either the PCI bus or VMEbus interface. Interrupts are generated from either hardware or software sources (Refer to *Interrupter* in the *Universe User Manual* for a full description of hardware and software sources.). Interrupt sources can be mapped to any of the PCI bus or VMEbus interrupt output pins. Interrupt sources mapped to VMEbus interrupts are generated on the VMEbus interrupt output pins VIRQ#[7:1]. When a software and hardware source are assigned the same VIRQn# pin, the software source always has higher priority.

Interrupt sources mapped to PCI bus interrupts are generated on one of the INT#[7:0] pins. To be fully PCI compliant, all interrupt sources must be routed to a single INT# pin.

For VMEbus interrupt outputs, the Universe interrupter supplies an 8-bit STATUS/ID to a VMEbus interrupt handler during the IACK cycle, and optionally generates an internal interrupt to signal that the interrupt vector has been provided. (Refer to *VMEbus Interrupt Generation* in the *Universe User Manual*.)

Interrupts mapped to PCI bus outputs are serviced by the PCI interrupt controller. The CPU determines which interrupt sources are active by reading an interrupt status register in the Universe. The source negates its interrupt when it has been serviced by the CPU. (Refer to *PCI Interrupt Generation* in the *Universe User Manual*.)

## VMEbus Interrupt Handling

A VMEbus interrupt triggers the Universe to generate a normal VMEbus IACK cycle and generate the specified interrupt output. When the IACK cycle is complete, the Universe releases the VMEbus and the interrupt vector is read by the PCI resource servicing the interrupt output. Software interrupts are ROAK, while hardware, and internal interrupts are RORA.

## DMA Controller

The Universe provides an internal DMA controller for high performance data transfer between the PCI and VMEbus. DMA operations between the source and destination bus are decoupled through the use of a single bidirectional FIFO (DMAFIFO). Parameters for the DMA transfer are software configurable in the Universe registers. (Refer to *DMA Controller* in the *Universe User Manual*.)

The principal mechanism for DMA transfers is the same for operations in either direction (PCI to VME, or VME to PCI), only the relative identity of the source and destination bus changes. In a DMA transfer, the Universe gains control of the source bus and reads data into its DMAFIFO. Following specific rules of DMAFIFO operation (refer to *FIFO Operation and Bus Ownership* in the *Universe User Manual*), it then acquires the destination bus and writes data from its DMAFIFO.

The DMA controller can be programmed to perform multiple blocks of transfers using entries in a linked list. The DMA will work through the transfers in the linked-list following pointers at the end of each linked-list entry. Linked-list operation is initiated through a pointer in an internal Universe register, but the linked list itself resides in PCI bus memory.

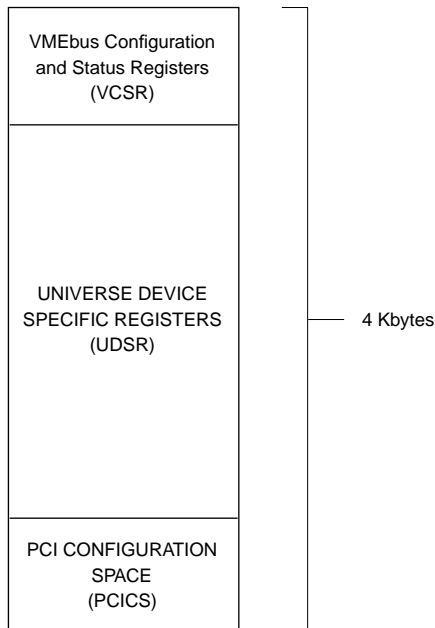
## Registers - Universe Control and Status Registers (UCSR)

The Universe Control and Status Registers (UCSR) facilitate host system configuration and allow the user to control Universe operational characteristics. The UCSRs are divided into three groups:

- ❑ PCI Configuration Space (PCICS)
- ❑ VMEbus Control and Status Registers (VCSR), and
- ❑ Universe Device Specific Status Registers (UDSR)

The Universe registers are little-endian.

Figure 4-2 below summarizes the supported register access mechanisms.



1895 9609

**Figure 4-2. UCSR Access Mechanisms**



## Universe Register Map

Table 4-1 below lists the Universe registers by address offset. Tables in the *Universe User Manual* provide detailed descriptions of each register.

Address offsets in Table 4-1 below apply to accesses from the PCI bus and to accesses from the VMEbus side using the VMEbus Register Access Image (Refer to *Registers* in the *Universe User Manual*). For register accesses in CR/CSR space, be sure to add 508 KBytes (0x7F00) to the address offsets provided in the table.

**Table 4-1. Universe Register Map**

Offset	Register	Name
000	PCI Configuration Space ID Register	PCI_ID
004	PCI Configuration Space Control and Status Register	PCI_CSR
008	PCI Configuration Class Register	PCI_CLASS
00C	PCI Configuration Miscellaneous 0 Register	PCI_MISC0
010	PCI Configuration Base Address Register	PCI_BS
014	PCI Unimplemented	
018	PCI Unimplemented	
01C	PCI Unimplemented	
020	PCI Unimplemented	
024	PCI Unimplemented	
028	PCI Reserved	
02C	PCI Reserved	
030	PCI Unimplemented	
034	PCI Reserved	
038	PCI Reserved	
03C	PCI Configuration Miscellaneous 1 Register	PCI_MISC1
040 - 0FF	PCI Unimplemented	

**Table 4-1. Universe Register Map (Continued)**

Offset	Register	Name
100	PCI Slave Image 0 Control	LSI0_CTL
104	PCI Slave Image 0 Base Address Register	LSI0_BS
108	PCI Slave Image 0 Bound Address Register	LSI0_BD
10C	PCI Slave Image 0 Translation Offset	LSI0_TO
110	Universe Reserved	
114	PCI Slave Image 1 Control	LSI1_CTL
118	PCI Slave Image 1 Base Address Register	LSI1_BS
11C	PCI Slave Image 1 Bound Address Register	LSI1_BD
120	PCI Slave Image 1 Translation Offset	LSI1_TO
124	Universe Reserved	
128	PCI Slave Image 2 Control	LSI2_CTL
12C	PCI Slave Image 2 Base Address Register	LSI2_BS
130	PCI Slave Image 2 Bound Address Register	LSI2_BD
134	PCI Slave Image 2 Translation Offset	LSI2_TO
138	Universe Reserved	
13C	PCI Slave Image 3 Control	LSI3_CTL
140	PCI Slave Image 3 Base Address Register	LSI3_BS
144	PCI Slave Image 3 Bound Address Register	LSI3_BD
148	PCI Slave Image 3 Translation Offset	LSI3_TO
14C - 16C	Universe Reserved	
170	Special Cycle Control Register	SCYC_CTL
174	Special Cycle PCI bus Address Register	SCYC_ADDR
178	Special Cycle Swap/Compare Enable Register	SCYC_EN
17C	Special Cycle Compare Data Register	SCYC_CMP
180	Special Cycle Swap Data Register	SCYC_SWP

**Table 4-1. Universe Register Map (Continued)**

Offset	Register	Name
184	PCI Miscellaneous Register	LMISC
188	Special PCI Slave Image	SLSI
18C	PCI Command Error Log Register	L_CMDERR
190	PCI Address Error Log	LAERR
194 - 1FC	Universe Reserved	
200	DMA Transfer Control Register	DCTL
204	DMA Transfer Byte Count Register	DTBC
208	DMA PCI bus Address Register	DLA
20C	Universe Reserved	
210	DMA VMEbus Address Register	DVA
214	Universe Reserved	
218	DMA Command Packet Pointer	DCPP
21C	Universe Reserved	
220	DMA General Control and Status Register	DGCS
224	DMA Linked List Update Enable Register	D_LLUE
228 - 2FC	Universe Reserved	
300	PCI Interrupt Enable	LINT_EN
304	PCI Interrupt Status	LINT_STAT
308	PCI Interrupt Map 0	LINT_MAP0
30C	PCI Interrupt Map 1	LINT_MAP1
310	VMEbus Interrupt Enable	VINT_EN
314	VMEbus Interrupt Status	VINT_STAT
318	VMEbus Interrupt Map 0	VINT_MAP0
31C	VMEbus Interrupt Map 1	VINT_MAP1
320	Interrupt Status/ID Out	STATID

**Table 4-1. Universe Register Map (Continued)**

Offset	Register	Name
324	VIRQ1 STATUS/ID	V1_STATID
328	VIRQ2 STATUS/ID	V2_STATID
32C	VIRQ3 STATUS/ID	V3_STATID
330	VIRQ4 STATUS/ID	V4_STATID
334	VIRQ5 STATUS/ID	V5_STATID
338	VIRQ6 STATUS/ID	V6_STATID
33C	VIRQ7 STATUS/ID	V7_STATID
340 - 3FC	Universe Reserved	
400	Master Control	MAST_CTL
404	Miscellaneous Control	MISC_CTL
408	Miscellaneous Status	MISC_STAT
40C	User AM Codes Register	USER_AM
410 - EFC	Universe Reserved	
F00	VMEbus Slave Image 0 Control	VSI0_CTL
F04	VMEbus Slave Image 0 Base Address Register	VSI0_BS
F08	VMEbus Slave Image 0 Bound Address Register	VSI0_BD
F0C	VMEbus Slave Image 0 Translation Offset	VSI0_TO
F10	Universe Reserved	
F14	VMEbus Slave Image 1 Control	VSI1_CTL
F18	VMEbus Slave Image 1 Base Address Register	VSI1_BS
F1C	VMEbus Slave Image 1 Bound Address Register	VSI1_BD
F20	VMEbus Slave Image 1 Translation Offset	VSI1_TO
F24	Universe Reserved	
F28	VMEbus Slave Image 2 Control	VSI2_CTL
F2C	VMEbus Slave Image 2 Base Address Register	VSI2_BS

**Table 4-1. Universe Register Map (Continued)**

Offset	Register	Name
F30	VMEbus Slave Image 2 Bound Address Register	VSI2_BD
F34	VMEbus Slave Image 2 Translation Offset	VSI2_TO
F38	Universe Reserved	
F3C	VMEbus Slave Image 3 Control	VSI3_CTL
F40	VMEbus Slave Image 3 Base Address Register	VSI3_BS
F44	VMEbus Slave Image 3 Bound Address Register	VSI3_BD
F48	VMEbus Slave Image 3 Translation Offset	VSI3_TO
F4C - F6C	Universe Reserved	
F70	VMEbus Register Access Image Control Register	VRAI_CTL
F74	VMEbus Register Access Image Base Address	VRAI_BS
F78	Universe Reserved	
F7C	Universe Reserved	
F80	VMEbus CSR Control Register	VCSR_CTL
F84	VMEbus CSR Translation Offset	VCSR_TO
F88	VMEbus AM Code Error Log	V_AMERR
F8C	VMEbus Address Error Log	VAERR
F90 - FEC	Universe Reserved	
FF0	VME CR/CSR Reserved	
FF4	VMEbus CSR Bit Clear Register	VCSR_CLR
FF8	VMEbus CSR Bit Set Register	VCSR_SET
FFC	VMEbus CSR Base Address Register	VCSR_BS



Register space marked as “Reserved” should not be overwritten. Unimplemented registers return a value of 0 on reads; writes complete normally.

## Universe Chip Problems after a PCI Reset

Under the conditions described below, there are problems with the Universe chip after a PCI reset.

### 4

#### Problem Description

The Universe chip is being enabled on the PCI bus after a PCI reset. The problem does not occur after a board reset or power up.

The Universe is causing the "bye" command to hang the system. The Universe Master Enable and Memory Enable in the PCI\_CSR (Configuration Space register) are enabled, even before the PCI\_BS register has been initialized. The symptoms can be alleviated by modifying the PCI probe list such that the Universe PCI configuration is done first.

The Configuration Space enables are not the only things enabled after a PCI reset. The LSI0 image may also not be disabled by a PCI reset, regardless of the enable bit's Power Up condition. If the image is active at the time the reset occurs, it will remain enabled through the reset. Additionally, the image is not staying at the same VME or PCI address range.

MCG is not sure how many of the Power Up (P/U) Option bits actually get latched as the Universe manual indicates they should. At least the EN bit in the LSI0\_CTL bit is not latched; in our board, its P/U state is disabled but is not honored.

The software cannot completely correct this problem, but it can minimize it. Two methods are presented:

##### Method 1:

1. Modify the PCI probe code to disable each PCI device prior to writing its configuration space BS registers. This will prevent the Universe from being active on the PCI bus while it has a base addr of 0.
2. Once the Universe has been assigned a valid PCI Base Address, enable register space access and disable the LSI0 slave image by clearing the EN bit of the LSI0\_CTL register.

### Method 2:

The port 92 reset code can be modified to disable the LSIO image prior to propagating the reset. This will cause the LSIO image to come up disabled.

We at MCG understand that both of these methods are awkward, because the PCI probe/reset code should not contain any device-specific patches. It should only need to follow the probing conventions of the PCI specification. However, the only other option appears to be avoidance of the LSIO image altogether.

Tundra engineering describes the problem thus:

"Following are the most recently discovered bugs which will be addressed in Universe 1.1.

#### 1. LSIO image

Description: After a PCI reset, the LSIO image is still enabled, but the base, bound, and translation offset changes value.

Workaround: None."

Motorola Firmware Engineering has implemented Method 1 in the BSP (Board Support Package which runs before Open Firmware). Motorola Firmware Engineering has implemented this firmware work-around for this hardware problem prior to the PPCOF2.0 RM01 release.

The above notes describing this Universe chip PCI reset problem are for those customers who replace the MCG firmware with their own. Many of these customers replace Motorola's firmware, or overwrite the firmware settings for the Universe chip. These customers may run into this problem.

Customers should not encounter any problems if they leave Motorola's Open Firmware intact.

## Examples

### Example 1: MVME2600 Series Board Exhibits Problem

Use an MVME2600 series board to exhibit the problem.

Conditions:

MVME260x  
running PPCOF2.0 Ir05  
all the Universe code which initializes the Universe has been  
disabled (not the PCI code, the driver code)  
modified probe list to d,c,e,f,10  
env parameters set such that LSI0 image will be enabled

1. Manually call each Universe initialize word, to duplicate typical operation.

With the LSI0 enabled, the registers are:

CTL	BS	BD	TO
80821000	1012000	21012000	3efee000

2. Execute a bye command.
3. Manually enable access to the Universe register set, and read the LSI0 registers:

CTL	BS	BD	TO
80820000	0	20000000	0

This means that the PCI reset changed the image as follows:

from supervisor address modifier to user  
from PCI space base address 1012000 to 0 (size of 2000.0000 constant)  
from VME address range 4000.0000 thru 5fff.ffff to a new VMEbus range of 0 thru 1fff.ffff

It is still enabled.



**Example 2: MVME3600 Series Board Acts Differently**

Repeat portions of the earlier example on an MVME3600 series board. This board had not previously been seen to hang upon PCI reset. This particular board had customized values for the LSI0 setup parameters.

The Universe register init code is still disabled, and must be manually called. The PCI init code is enabled, so the Universe PCI memory space requirement defined by its Configuration Space register at offset 0x10 is being accommodated and enabled during PCI probing. PCI probe list d,c,e,f,10

1. After a P/U reset, before the init code has written the registers, the LSI0 register settings are:

CTL	BS	BD	TO
800000	0	0	0

2. Run the init code and the LSI0 registers become:

CTL	BS	BD	TO
80821000	3000000	300a000	4d000000

3. After a bye, before the init code has run:

CTL	BS	BD	TO
80820000	0	0	0

Therefore the PCI reset caused the following changes in the LSI0 image:

- from supervisor to user
- from PCI space base address 300.0000 to 0
- from PCI space size of a000 to size of 0
- from a VME base address of 5000.0000 to 0

This explains why it has never been a problem on this particular MVME360x. The fact that the PCI base and PCI bound registers are both 0 makes the effective size of the image 0 bytes. Therefore this "enabled" image will never utilize any PCI address space.

4. Now try modifying the LSI0 env parameters, to be the same as those on the MVME260x which failed:

```
printenv
vme3_lsi0_vmeaddr    1073741824    1073741824
vme3_lsi0_size      536870912    536870912
vme3_lsi0_phi       77          77
```

After a power-up, before the init code ran, the LSI0 values are:

```
800000 0 0 0
```

5. Do NOT run the init code, but press push-button RESET, and the values become:

```
830001 f0000000 f0000000 0
```

6. Run the vme3 init code, and the values are set to accommodate env parameters:

```
80821000 3000000 23000000 3d000000
```

7. Do a bye

The values before the init code runs are:

```
80820000 0 20000000 0
```

This gets the same results with the MVME360x as with the MVME260x, and the difference seen earlier is related to the values of the LSI0 slave at the time the PCI reset occurred.

### Example 3: Universe Chip is Checked at Tundra

An engineer at Tundra Semiconductor Corporation had run a simulation on the LSI0\_CTL register, and could see that it was going to be enabled after a port 92 reset. Motorola engineers mentioned that the problem is primarily with the \_BS, \_BD, and \_TO registers. He said he would run more simulations to look at the outcome on those registers. Motorola engineers explained what they had seen.

The engineer at Tundra re-ran the simulation based on the information given him. He saw exactly what the Motorola engineers had seen, . i.e., that the LSI0\_BS, LSI0\_BD, and LSI0\_TO values change, as well as the LSI0\_CTL fields for program, super, and vct. He checked to see if this is in fact what the Universe is supposed to do.

The following are his results:

Register	Before RST#	After RST#
-----	-----	-----
LSI0_CTL	8082_5FFF	8082_0001
LSI0_BS	FFFF_FFFF	F000_0000
LSI0_BD	FFFF_FFFF	F000_0000
LSI0_TO	FFFF_FFFF	0000_0000

Explanation:

All the fields in the LSI0 registers which are "Power-up Options" cannot be reset by assertion of RST# (PCI reset).

The following fields in the LSI0 registers cannot be reset by a PCI reset:

LSI0\_CTL register: EN, VAS, LAS

LSI0\_BS register: Bits [31:28]

LSI0\_BD register: Bits [31:28]

All the other fields in the LSI0 registers are reset to 0, which explains why the PGM and SUPER fields changed, the translation offset reset to 0, etc.



## Introduction

This chapter contains details of several programming functions that are not tied to any specific ASIC chip.

## PCI Arbitration

PCI arbitration is performed by the PCI-to-ISA Bridge (PIB) which supports six PCI external PCI masters. The PIB can also be a PCI master for ISA DMA functions. The arbitration assignments on the MVME2600 series are as follows:

**Table 5-1. PCI Arbitration Assignments**

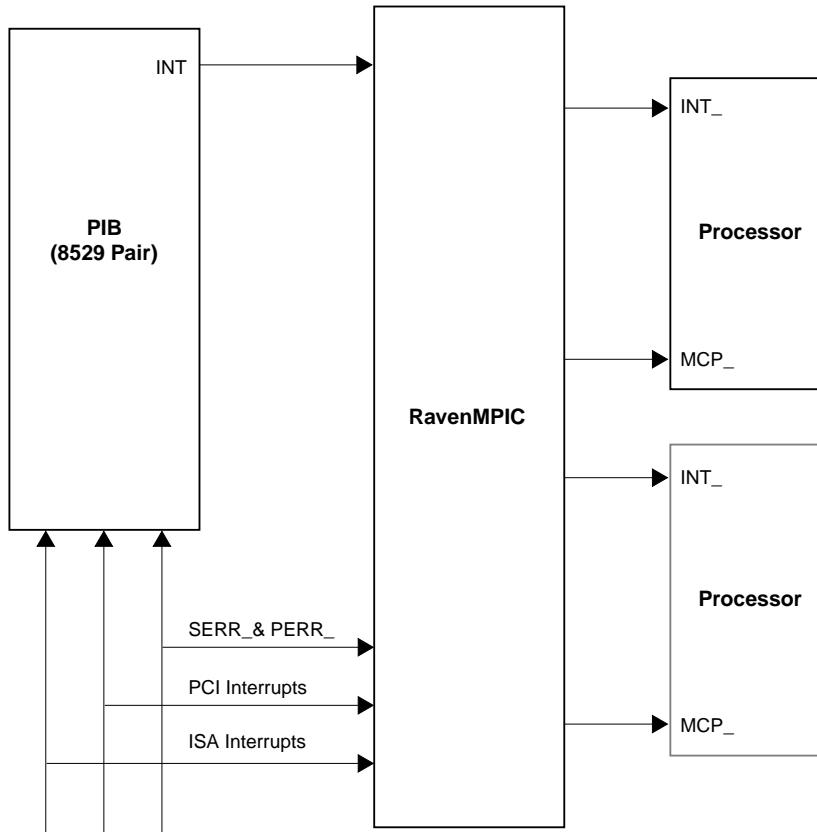
PCI BUS REQUEST	PCI Master(s)
PIB (internal)	PIB
CPU	Secondary Ethernet Secondary SCSI Raven ASIC
Request 0	PMC Slot 2 (PCIX)
Request 1	PMC Slot 1
Request 2	Ethernet
Request 3	SCSI
Request 4	Universe ASIC (VMEbus)

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 for additional details.

# Interrupt Handling

The interrupt architecture of the MVME2600 series SBC is shown in the following figure:

5



11559.00 9609

**Figure 5-1. MVME2600 Series Interrupt Architecture**

## RavenMPIC

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 on the Raven for details on the RavenMPIC. The following table shows the interrupt assignments for the RavenMPIC on the MVME2600 series:

**Table 5-2. RavenMPIC 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	4
IRQ3	Level	Low	PCI-SCSI	4
IRQ4	Level	Low	PCI-Graphics	4
IRQ5	Level	Low	PCI-VME INT 0 (Universe LINT0#)	3, 4
IRQ6	Level	Low	PCI-VME INT 1 (Universe LINT1#)	3
IRQ7	Level	Low	PCI-VME INT 2 (Universe LINT2#)	3
IRQ8	Level	Low	PCI-VME INT 3 (Universe LINT3#)	3
IRQ9	Level	Low	PCI-PMC1/PMC2 INTA#	4
IRQ10	Level	Low	PCI-PMC1/PMC2 INTB#	
IRQ11	Level	Low	PCI-PMC1/PMC2 INTC#	
IRQ12	Level	Low	PCI-PMC1/PMC2 INTD#	
IRQ13	Level	Low	LM/SIG Interrupt 0	4

**Table 5-2. RavenMPIC Interrupt Assignments (Continued)**

MPIC IRQ	Edge/ Level	Polarity	Interrupt Source	Notes
IRQ14	Level	Low	LM/SIG Interrupt 1	4
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. The mapping of interrupt sources from the VMEbus and Universe internal interrupt sources is programmable via the Local Interrupt Map 0 Register and the Local Interrupt Map1 Register in the Universe ASIC.
4. These interrupts also appear at the PIB for backward compatibility with older MVME1600 and PM603/4 modules.

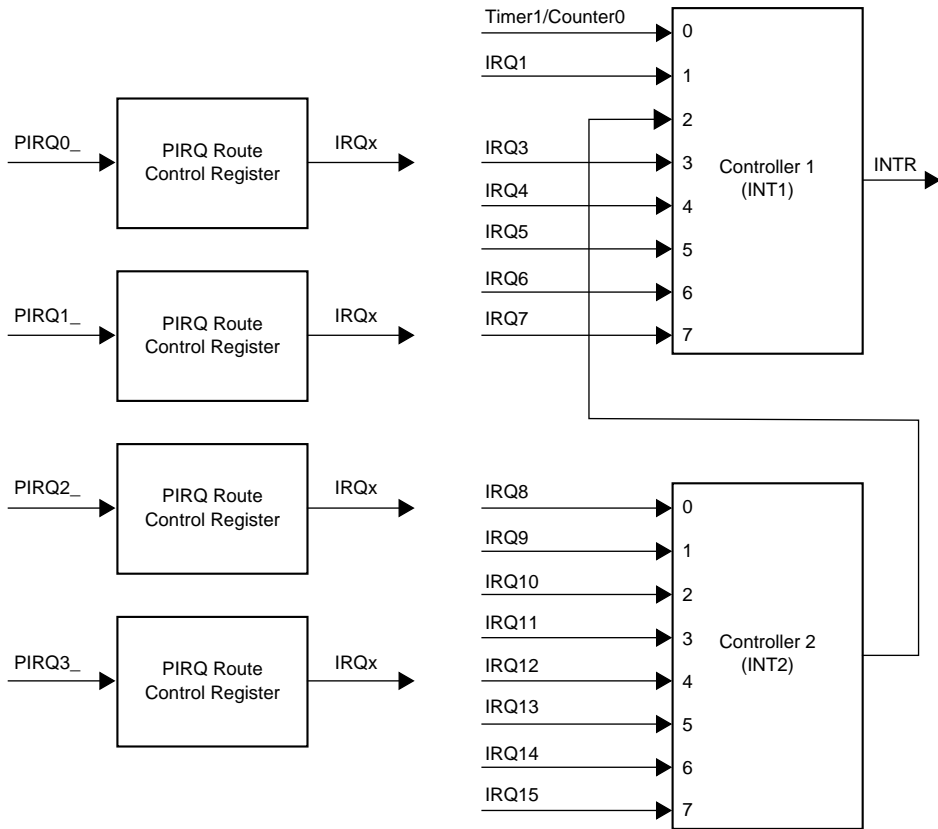
## 8259 Interrupts

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 interrupts can be routed to the same ISA IRQ line. The PIB can be programmed to handle the PCI interrupts if the RavenMPIC is either not present or not used.



The following figure shows the interrupt structure of the PIB.



1897 9609

**Figure 5-2. PIB Interrupt Handler Block Diagram**

The assignments of the PCI and ISA interrupts supported by the PIB are as follows:

**Table 5-3. 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 Z85230 ESCC	3,4	
5	IRQ10		PIRQ0_		Level	Low	PCI-Ethernet Interrupt	3,5,6
6	IRQ11		PIRQ1_		Level	Low	Universe Interrupt (LINT0#)	3,5,6
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	PCI-Graphics Interrupt	3,5,6	
						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	LM/SIG Interrupt 0/1	6
14	IRQ6			Edge	High	Floppy Interrupt		
15	IRQ7			Edge	High	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 RavenMPIC is either not present or not used.
6. The RavenMPIC, when present, should be used for these interrupts.

**5**

## ISA DMA Channels

Refer to Chapter 1 for information on the ISA DMA channels.

# Exceptions

## Sources of Reset

There are nine potential sources of reset on the MVME2600 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
6. VMEbus SYSRESET# signal
7. Local software reset via the Universe ASIC (MISC\_CTL Register)
8. VME System Reset Via the Universe ASIC (MISC\_CTL Register)
9. VME CSR reset via the Universe ASIC (VCSR\_SET Register)

The following table shows which devices are affected by various reset sources:

**Table 5-4. Reset Sources and Devices Affected**

Device Affected	Processor (s)	Raven ASIC	Falcon Chipset	PCI Devices	ISA Devices	VMEbus (System Controller)
Power-On	✓	✓	✓	✓	✓	✓
Reset Switch	✓	✓	✓	✓	✓	✓
Watchdog (MK48T59)	✓	✓	✓	✓	✓	✓
VME System Reset (SYSRESET# Signal)	✓	✓	✓	✓	✓	✓
VME System Software Reset (MISC_CTL Register)	✓	✓	✓	✓	✓	✓
VME Local Software Reset (MISC_CTL Register)	✓	✓	✓	✓	✓	
VME CSR Reset (VCSR_SET Register)	✓	✓	✓	✓	✓	
Hot Reset (Port 92 Register)	✓	✓	✓	✓	✓	
PCI/ISA Reset (Clock Divisor Register)				✓	✓	

## Soft Reset

Software can assert the SRESET# pin of any processor by programming the Processor Init Register of the RavenMPIC appropriately.

## Universe Chip Problems after a PCI Reset

Under certain conditions, there can be problems with the Universe chip after a PCI reset. Refer to Chapter 4 for the details.

## Error Notification and Handling

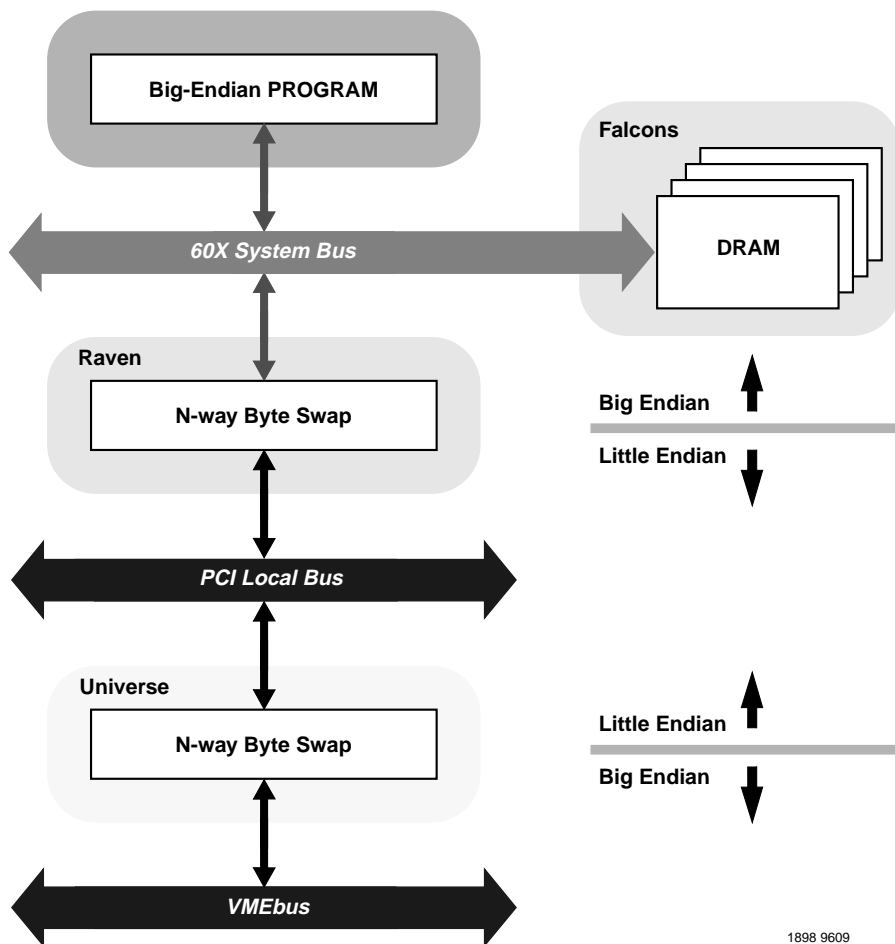
The Raven and Falcon chipset can detect certain hardware errors and can be programmed to report these errors via the RavenMPIC interrupts or Machine Check Interrupt. Note that the TEA\* signal is not used at all by the MVME2600 series. The following table summarizes how the hardware errors are handled by the MVME2600 series:

**Table 5-5. Error Notification and Handling**

Cause	Action
Single-bit ECC	<i>Store:</i> Write corrected data to memory <i>Load:</i> Present corrected data to the MPC master Generate interrupt via RavenMPIC if so enabled
Double-bit ECC	<i>Store:</i> Terminate the bus cycle normally without writing to DRAM <i>Load:</i> Present un-corrected data to the MPC master Generate interrupt via RavenMPIC if so enabled Generate Machine Check Interrupt to the Processor(s) if so enabled
MPC Bus Time Out	<i>Store:</i> Discard write data and terminate bus cycle normally <i>Load:</i> Present undefined data to the MPC master Generate interrupt via RavenMPIC if so enabled Generate Machine Check Interrupt to the Processor(s) if so enabled
PCI Target Abort	<i>Store:</i> Discard write data and terminate bus cycle normally <i>Load:</i> Return all 1's and terminate bus cycle normally Generate interrupt via RavenMPIC if so enabled Generate Machine Check Interrupt to the Processor(s) if so enabled
PCI Master Abort	<i>Store:</i> Discard write data and terminate bus cycle normally <i>Load:</i> Return all 1's and terminate bus cycle normally Generate interrupt via RavenMPIC if so enabled Generate Machine Check Interrupt to the Processor(s) if so enabled
PERR# Detected	Generate interrupt via RavenMPIC if so enabled Generate Machine Check Interrupt to the Processor(s) if so enabled
SERR# Detected	Generate interrupt via RavenMPIC if so enabled Generate Machine Check Interrupt to the Processor(s) if so enabled

## Endian Issues

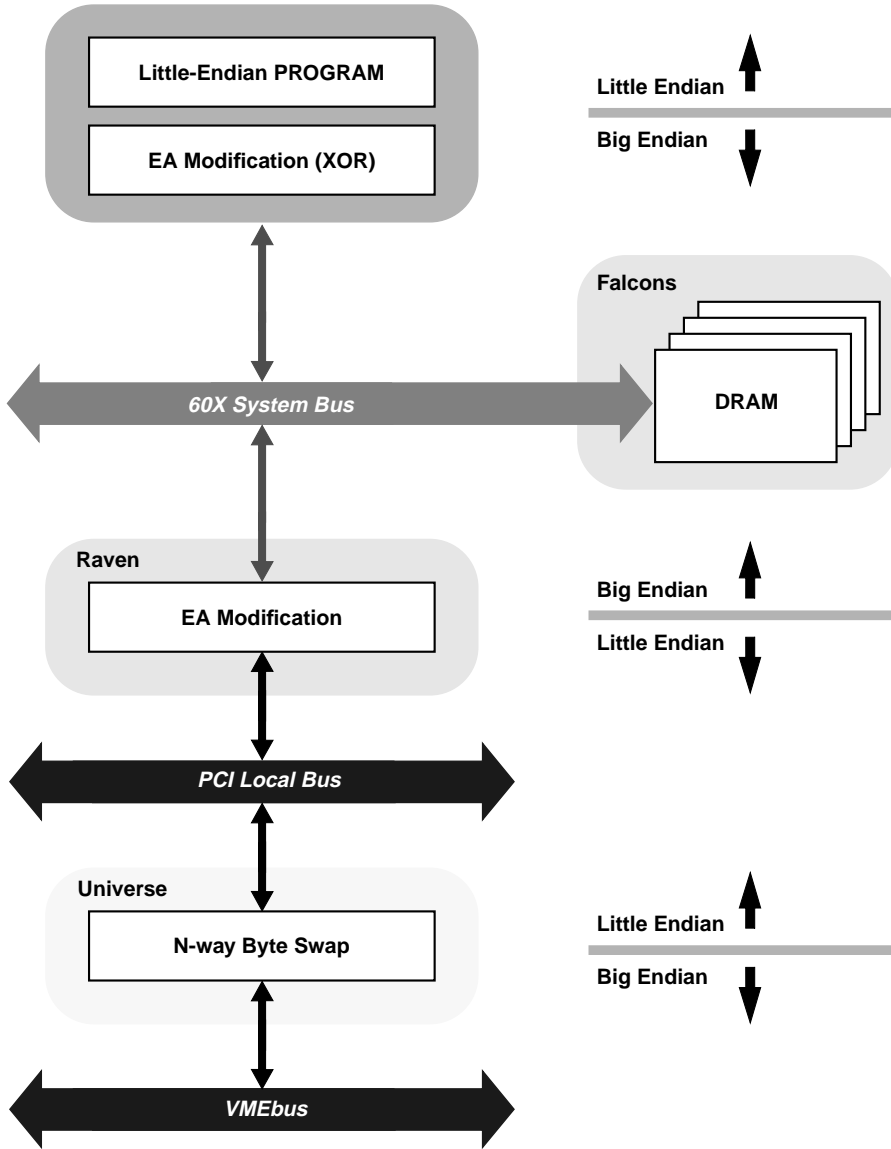
The MVME2600 series supports both little endian software (e.g. NT) and big-endian software (e.g. AIX). Because the PowerPC processor is inherently big endian, PCI is inherently little-endian, and the VMEbus is big endian, things do get rather confusing. The following figures show how the MVME2600 series handles the endian issue in big-endian and little-endian modes:



1898 9609

Figure 5-3. Big-Endian Mode





1899 9609

Figure 5-4. 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.

## PCI-Graphics

The effects of byte swapping on big-endian software must be considered by big-endian software.

**Note** There are graphics on the MVME3600 series boards, but no graphics on the MVME2600 series boards.

## Universe's Involvement

Since PCI is little-endian and the VMEbus is big-endian, the Universe performs byte swapping in both directions (from PCI to VMEbus and from VMEbus to PCI) to maintain address invariance, regardless of the mode of operation in the processor's domain.

## VMEbus Domain

The VMEbus is inherently big-endian and all devices connected directly to VMEbus are expected to operate in big-endian mode, regardless of the mode of operation in the processor's domain.

In big-endian mode, byte-swapping is performed by the Universe and then by the Raven. The result has the desirable effect by being transparent to the big-endian software.

In little-endian mode, however, software must be aware of the byte-swapping effect from the Universe and the address reverse-rearranging effect of the Raven.

## 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 5-6. ROM/FLASH Bank Default**

<i>rom_b_rv</i>	Default Mapping for FFF00000-FFFFFFFF
0	ROM/FLASH Bank A
1	ROM/FLASH Bank B

# Related Documentation

A

## Motorola Computer Group Documents

The publications listed below are on related products, and some may be referenced in this document. If not shipped with this product, manuals may be purchased by contacting your local Motorola sales office.

**Table A-1. Motorola Computer Group Documents**

Document Title	Publication Number
MVME2600 Series Single Board Computer Installation and Use *	V2600A/IH
MVME2600 Series Single Board Computer Programmer's Reference Guide *	V2600A/PG
PowerPC Open Firmware Quick Start Guide*	PPCOFWQSA/UG
PowerPC Open Firmware Command Dictionary Release 2.0 Quick Reference*	PPCOFWCDA/QR
PowerPC Open Firmware User's Manual (Volumes 1 and 2) *	PPCOFWA1/UM PPCOFWA2/UM
MVME712M Transition Module and P2 Adapter Board Installation and Use	VME712MA/IH
MVME761 Transition Module Installation and Use	VME761A/IH

**Notes** 1. Although not shown in the above list, each Motorola Computer Group manual publication number is suffixed with characters that represent the revision level of the document, such as "/xx2" (the second revision of a manual); a supplement bears the same number as the manual but has a suffix such as "/xx2A1" (the first supplement to the second revision of the manual).

2. Motorola documents marked with a \* in the above list can be purchased as a set under part number **LK-2600**.





**Table A-2. Manufacturers' Documents (Continued)**

Document Title and Source	Publication Number
<p>MPC2604GA Integrated Secondary Cache for PowerPC Microprocessors (Glance) Data Sheets</p> <p>Motorola Literature and Printing Distribution Services  P.O. Box 20924  Phoenix, Arizona 85036-0924  Telephone: (602) 994-6561  FAX: (602) 994-6430</p>	MPC2604GA
<p>Alpine™ VGA Family - CL-GD543X/'4X Technical Reference Manual Fourth Edition</p> <p>Cirrus Logic, Inc. (or nearest Sales Office)  3100 West Warren Avenue  Fremont, California 94538-6423  Telephone: (510) 623-8300  FAX: (510) 226-2180</p>	385439-004
<p>DECchip 21140 PCI Fast Ethernet LAN Controller Hardware Reference Manual</p> <p>Digital Equipment Corporation  Maynard, Massachusetts  DECchip Information Line  Telephone (United States and Canada): 1-800-332-2717  TTY (United States only): 1-800-332-2515  Telephone (outside North America): +1-508-568-6868</p>	EC-QN7NC-TE
<p>PC87308VUL (Super I/O™ Enhanced Sidewinder Lite) Floppy Disk Controller, Keyboard Controller, Real-Time Clock, Dual UARTs, IEEE 1284 Parallel Port, and IDE Interface</p> <p>National Semiconductor Corporation  Customer Support Center (or nearest Sales Office)  2900 Semiconductor Drive  P.O. Box 58090  Santa Clara, California 95052-8090  Telephone: 1-800-272-9959</p>	PC87308VUL
<p>MK48T59/559 CMOS 8K x 8 TIMEKEEPER™ SRAM Data Sheet</p> <p>SGS-Thomson Microelectronics Group  Marketing Headquarters (or nearest Sales Office)  1000 East Bell Road  Phoenix, Arizona 85022  Telephone: (602) 967-6100</p>	MK48T59



**Table A-2. Manufacturers' Documents (Continued)**

Document Title and Source	Publication Number
SYM 53CXX (was NCR 53C8XX) Family PCI-SCSI I/O Processors Programming Guide Symbios Logic Inc. 1731 Technology Drive, suite 600 San Jose, CA 95110 Telephone: (408) 441-1080 Hotline: 1-800-334-5454	J10931I
SCC (Serial Communications Controller) User's Manual (for Z85230 and other Zilog parts) Zilog, Inc. 210 East Hacienda Ave., mail stop C1-0 Campbell, California 95008-6600 Telephone: (408) 370-8016 FAX: (408) 370-8056	UM95SCC0100
Z8536 CIO Counter/Timer and Parallel I/O Unit Product Specification and User's Manual (in Z8000 <sup>®</sup> Family of Products Data Book) Zilog, Inc. 210 East Hacienda Ave., mail stop C1-0 Campbell, California 95008-6600 Telephone: (408) 370-8016 FAX: (408) 370-8056	DC-8319-00
W83C553 Enhanced System I/O Controller with PCI Arbiter (PIB) (was SL82C565) Winbond Electronics Corporation Winbond Systems Laboratory 2730 Orchard Parkway San Jose, CA 95134 Telephone: 1-408-943-6666 FAX: 1-408-943-6668	W83C553 (was 2565 version A.6)

**Table A-2. Manufacturers' Documents (Continued)**

<b>Document Title and Source</b>	<b>Publication Number</b>
Universe User Manual Tundra Semiconductor Corporation 603 March Road Kanata, ON K2K 2M5, Canada Telephone: 1-800-267-7231 Telephone: (613) 592-1320 OR 695 High Glen Drive San Jose, CA 95133, USA Telephone: (408) 258-3600 FAX: (408) 258-3659	Universe (Part Number 9000000.MD303.01)



**Table A-3. Related Specifications (Continued)**

<b>Document Title and Source</b>	<b>Publication Number</b>
IEEE - Common Mezzanine Card Specification (CMC) Institute of Electrical and Electronics Engineers, Inc. Publication and Sales Department 345 East 47th Street New York, New York 10017-21633 Telephone: 1-800-678-4333	P1386 Draft 2.0
IEEE - PCI Mezzanine Card Specification (PMC) Institute of Electrical and Electronics Engineers, Inc. Publication and Sales Department 345 East 47th Street New York, New York 10017-21633 Telephone: 1-800-678-4333	P1386.1 Draft 2.0
Bidirectional Parallel Port Interface Specification Institute of Electrical and Electronics Engineers, Inc. Publication and Sales Department 345 East 47th Street New York, New York 10017-21633 Telephone: 1-800-678-4333	IEEE Standard 1284
Peripheral Component Interconnect (PCI) Local Bus Specification, Revision 2.1 PCI Special Interest Group 2575 NE Kathryn St #17 Hillsboro, OR 97124 Telephone: (800) 433-5177 (inside the U.S.) or (503) 693-6232 (outside the U.S.) FAX: (503) 693-8344	PCI Local Bus Specification





# Glossary

---

## Abbreviations, Acronyms, and Terms to Know

This glossary defines some of the abbreviations, acronyms, and key terms used in this document.

<b>10Base-5</b>	An Ethernet implementation in which the physical medium is a doubly shielded, 50-ohm coaxial cable capable of carrying data at 10 Mbps for a length of 500 meters (also referred to as thicknet). Also known as thick Ethernet.
<b>10Base-2</b>	An Ethernet implementation in which the physical medium is a single-shielded, 50-ohm RG58A/U coaxial cable capable of carrying data at 10 Mbps for a length of 185 meters (also referred to as AUI or thinnet). Also known as thin Ethernet.
<b>10Base-T</b>	An Ethernet implementation in which the physical medium is an unshielded twisted pair (UTP) of wires capable of carrying data at 10 Mbps for a maximum distance of 185 meters. Also known as twisted-pair Ethernet.
<b>100Base-TX</b>	An Ethernet implementation in which the physical medium is an unshielded twisted pair (UTP) of wires capable of carrying data at 100 Mbps for a maximum distance of 100 meters. Also known as fast Ethernet.
<b>ACIA</b>	<b>A</b> synchronous <b>C</b> ommunications <b>I</b> nterface <b>A</b> dapter
<b>AIX</b>	<b>A</b> dvanced <b>I</b> nteractive <b>eX</b> ecutive (IBM version of UNIX)
<b>architecture</b>	The main overall design in which each individual hardware component of the computer system is interrelated. The most common uses of this term are 8-bit, 16-bit, or 32-bit architectural design systems.
<b>ASCII</b>	<b>A</b> merican <b>S</b> tandard <b>C</b> ode for <b>I</b> nformation <b>I</b> nterchange. This is a 7-bit code used to encode alphanumeric information. In the IBM-compatible world, this is expanded to 8-bits to encode a total of 256 alphanumeric and control characters.

<b>ASIC</b>	<b>Application-Specific Integrated Circuit</b>
<b>AUI</b>	<b>Attachment Unit Interface</b>
<b>BBRAM</b>	<b>Battery Backed-up Random Access Memory</b>
<b>bi-endian</b>	Having big-endian and little-endian byte ordering capability.
<b>big-endian</b>	A byte-ordering method in memory where the address <i>n</i> of a word corresponds to the most significant byte. In an addressed memory word, the bytes are ordered (left to right) 0, 1, 2, 3, with 0 being the most significant byte.
<b>BIOS</b>	<b>Basic Input/Output System.</b> This is the built-in program that controls the basic functions of communications between the processor and the I/O (peripherals) devices. Also referred to as ROM BIOS.
<b>BitBLT</b>	<b>Bit Boundary BLock Transfer.</b> A type of graphics drawing routine that moves a rectangle of data from one area of display memory to another. The data specifically need not have any particular alignment.
<b>BLT</b>	<b>BLock Transfer</b>
<b>board</b>	The term more commonly used to refer to a PCB (printed circuit board). Basically, a flat board made of nonconducting material, such as plastic or fiberglass, on which chips and other electronic components are mounted. Also referred to as a circuit board or card.
<b>bpi</b>	<b>bits per inch</b>
<b>bps</b>	<b>bits per second</b>
<b>bus</b>	The pathway used to communicate between the CPU, memory, and various input/output devices, including floppy and hard disk drives. Available in various widths (8-, 16-, and 32-bit), with accompanying increases in speed.
<b>cache</b>	A high-speed memory that resides logically between a central processing unit (CPU) and the main memory. This temporary memory holds the data and/or



---

	instructions that the CPU is most likely to use over and over again and avoids accessing the slower hard or floppy disk drive.
<b>CAS</b>	<b>Column Address Strobe.</b> The clock signal used in dynamic RAMs to control the input of column addresses.
<b>CD</b>	<b>Compact Disc.</b> A hard, round, flat portable storage unit that stores information digitally.
<b>CD-ROM</b>	<b>Compact Disk Read-Only Memory</b>
<b>CFM</b>	<b>Cubic Feet per Minute</b>
<b>CHRP</b>	See <b>Common Hardware Reference Platform (CHRP).</b>
<b>CHRP-compliant</b>	See <b>Common Hardware Reference Platform (CHRP).</b>
<b>CHRP Spec</b>	See <b>Common Hardware Reference Platform (CHRP).</b>
<b>CISC</b>	<b>Complex-Instruction-Set Computer.</b> A computer whose processor is designed to sequentially run variable-length instructions, many of which require several clock cycles, that perform complex tasks and thereby simplify programming.
<b>CODEC</b>	<b>COder / DECoder</b>
<b>Color Difference (CD)</b>	The signals of (R-Y) and (B-Y) without the luminance (-Y) signal. The Green signals (G-Y) can be extracted by these two signals.
<b>Common Hardware Reference Platform (CHRP)</b>	A specification published by the Apple, IBM, and Motorola which defines the devices, interfaces, and data formats that make up a CHRP-compliant system using a PowerPC processor.
<b>Composite Video Signal (CVS/CVBS)</b>	Signal that carries video picture information for color, brightness and synchronizing signals for both horizontal and vertical scans. Sometimes referred to as "Baseband Video".
<b>cpi</b>	<b>characters per inch</b>
<b>cpl</b>	<b>characters per line</b>

---

<b>CPU</b>	<b>Central Processing Unit.</b> The master computer unit in a system.
<b>DCE</b>	<b>Data Circuit-terminating Equipment.</b>
<b>DLL</b>	<b>Dynamic Link Library.</b> A set of functions that are linked to the referencing program at the time it is loaded into memory.
<b>DMA</b>	<b>Direct Memory Access.</b> A method by which a device may read or write to memory directly without processor intervention. DMA is typically used by block I/O devices.
<b>DOS</b>	<b>Disk Operating System</b>
<b>dpi</b>	<b>dots per inch</b>
<b>DRAM</b>	<b>Dynamic Random Access Memory.</b> A memory technology that is characterized by extreme high density, low power, and low cost. It must be more or less continuously refreshed to avoid loss of data.
<b>DTE</b>	<b>Data Terminal Equipment.</b>
<b>ECC</b>	<b>Error Correction Code</b>
<b>ECP</b>	<b>Extended Capability Port</b>
<b>EEPROM</b>	<b>Electrically Erasable Programmable Read-Only Memory.</b> A memory storage device that can be written repeatedly with no special erasure fixture. EEPROMs do not lose their contents when they are powered down.
<b>EISA (bus)</b>	<b>Extended Industry Standard Architecture (bus) (IBM).</b> An architectural system using a 32-bit bus that allows data to be transferred between peripherals in 32-bit chunks instead of 16-bit or 8-bit that most systems use. With the transfer of larger bits of information, the machine is able to perform much faster than the standard ISA bus system.
<b>EPP</b>	<b>Enhanced Parallel Port</b>
<b>EPROM</b>	<b>Erasable Programmable Read-Only Memory.</b> A memory storage device that can be written once (per erasure cycle) and read many times.
<b>ESCC</b>	<b>Enhanced Serial Communication Controller</b>
<b>ESD</b>	<b>Electro-Static Discharge/Damage</b>

---

<b>Ethernet</b>	A local area network standard that uses radio frequency signals carried by coaxial cables.
<b>Falcon</b>	The DRAM controller chip developed by Motorola for the MVME2600 and MVME3600 series of boards. It is intended to be used in sets of two to provide the necessary interface between the Power PC60x bus and the 144-bit ECC DRAM (system memory array) and /or ROM/Flash.
<b>fast Ethernet</b>	See 100Base-TX.
<b>FDC</b>	Floppy Disk Controller
<b>FDDI</b>	Fiber Distributed Data Interface. A network based on the use of optical-fiber cable to transmit data in non-return-to-zero, invert-on-1s (NRZI) format at speeds up to 100 Mbps.
<b>FIFO</b>	First-In, First-Out. A memory that can temporarily hold data so that the sending device can send data faster than the receiving device can accept it. The sending and receiving devices typically operate asynchronously.
<b>firmware</b>	The program or specific software instructions that have been more or less permanently burned into an electronic component, such as a ROM (read-only memory) or an EPROM (erasable programmable read-only memory).
<b>frame</b>	One complete television picture frame consists of 525 horizontal lines with the NTSC system. One frame consists of two Fields.
<b>graphics controller</b>	On EGA and VGA, a section of circuitry that can provide hardware assist for graphics drawing algorithms by performing logical functions on data written to display memory.
<b>HAL</b>	Hardware Abstraction Layer. The lower level hardware interface module of the Windows NT operating system. It contains platform specific functionality.
<b>hardware</b>	A computing system is normally spoken of as having two major components: hardware and software. Hardware is the term used to describe any of the physical embodiments

of a computer system, with emphasis on the electronic circuits (the computer) and electromechanical devices (peripherals) that make up the system.

<b>HCT</b>	<b>Hardware Conformance Test.</b> A test used to ensure that both hardware and software conform to the Windows NT interface.
<b>I/O</b>	<b>Input/Output</b>
<b>IBC</b>	<b>PCI/ISA Bridge Controller</b>
<b>IDC</b>	<b>Insulation Displacement Connector</b>
<b>IDE</b>	<b>Intelligent Device Expansion</b>
<b>IEEE</b>	<b>Institute of Electrical and Electronics Engineers</b>
<b>interlaced</b>	A graphics system in which the even scanlines are refreshed in one vertical cycle (field), and the odd scanlines are refreshed in another vertical cycle. The advantage is that the video bandwidth is roughly half that required for a non-interlaced system of the same resolution. This results in less costly hardware. It also may make it possible to display a resolution that would otherwise be impossible on given hardware. The disadvantage of an interlaced system is flicker, especially when displaying objects that are only a few scanlines high.
<b>IQ Signals</b>	Similar to the color difference signals (R-Y), (B-Y) but using different vector axis for encoding or decoding. Used by some USA TV and IC manufacturers for color decoding.
<b>ISA (bus)</b>	<b>Industry Standard Architecture (bus).</b> The de facto standard system bus for IBM-compatible computers until the introduction of VESA and PCI. Used in the reference platform specification. (IBM)
<b>ISASIO</b>	<b>ISA Super Input/Output device</b>
<b>ISDN</b>	<b>Integrated Services Digital Network.</b> A standard for digitally transmitting video, audio, and electronic data over public phone networks.
<b>LAN</b>	<b>Local Area Network</b>
<b>LED</b>	<b>Light-Emitting Diode</b>

---

<b>LFM</b>	<b>Linear Feet per Minute</b>
<b>little-endian</b>	A byte-ordering method in memory where the address $n$ of a word corresponds to the least significant byte. In an addressed memory word, the bytes are ordered (left to right) 3, 2, 1, 0, with 3 being the most significant byte.
<b>MBLT</b>	<b>Multiplexed BLock Transfer</b>
<b>MCA (bus)</b>	<b>Micro Channel Architecture</b>
<b>MCG</b>	<b>Motorola Computer Group</b>
<b>MFM</b>	<b>Modified Frequency Modulation</b>
<b>MIDI</b>	<b>Musical Instrument Digital Interface.</b> The standard format for recording, storing, and playing digital music.
<b>MPC</b>	<b>Multimedia Personal Computer</b>
<b>MPC105</b>	The PowerPC-to-PCI bus bridge chip developed by Motorola for the Ultra 603/Ultra 604 system board. It provides the necessary interface between the MPC603/MPC604 processor and the Boot ROM (secondary cache), the DRAM (system memory array), and the PCI bus.
<b>MPC601</b>	Motorola's component designation for the PowerPC 601 microprocessor.
<b>MPC603</b>	Motorola's component designation for the PowerPC 603 microprocessor.
<b>MPC604</b>	Motorola's component designation for the PowerPC 604 microprocessor.
<b>MPIC</b>	<b>Multi-Processor Interrupt Controller</b>
<b>MPU</b>	<b>MicroProcessing Unit</b>
<b>MTBF</b>	<b>Mean Time Between Failures.</b> A statistical term relating to reliability as expressed in power on hours (poh). It was originally developed for the military and can be calculated several different ways, yielding substantially different results. The specification is based on a large number of samplings in one place, running continuously, and the rate at which failure occurs. MTBF is not representative of how

long a device, or any individual device is likely to last, nor is it a warranty, but rather, of the relative reliability of a family of products.

**multisession**

The ability to record additional information, such as digitized photographs, on a CD-ROM after a prior recording session has ended.

**non-interlaced**

A video system in which every pixel is refreshed during every vertical scan. A non-interlaced system is normally more expensive than an interlaced system of the same resolution, and is usually said to have a more pleasing appearance.

**nonvolatile memory**

A memory in which the data content is maintained whether the power supply is connected or not.

**NTSC**

National Television Standards Committee (USA)

**NVRAM**

Non-Volatile Random Access Memory

**OEM**

Original Equipment Manufacturer

**OMPAC**

Over - Molded Pad Array Carrier

**OS**

Operating System. The software that manages the computer resources, accesses files, and dispatches programs.

**OTP**

One-Time Programmable

**palette**

The range of colors available on the screen, not necessarily simultaneously. For VGA, this is either 16 or 256 simultaneous colors out of 262,144.

**parallel port**

A connector that can exchange data with an I/O device eight bits at a time. This port is more commonly used for the connection of a printer to a system.

**PCI (local bus)**

Peripheral Component Interconnect (local bus) (Intel). A high-performance, 32-bit internal interconnect bus used for data transfer to peripheral controller components, such as those for audio, video, and graphics.

---

<b>PCMCIA (bus)</b>	<b>Personal Computer Memory Card International Association (bus).</b> A standard external interconnect bus which allows peripherals adhering to the standard to be plugged in and used without further system modification.
<b>PCR</b>	<b>PCI Configuration Register</b>
<b>PHB</b>	<b>PCI Host Bridge</b>
<b>PDS</b>	<b>Processor Direct Slot</b>
<b>physical address</b>	A binary address that refers to the actual location of information stored in secondary storage.
<b>PIB</b>	<b>PCI-to-ISA Bridge</b>
<b>pixel</b>	An acronym for picture element, and is also called a pel. A pixel is the smallest addressable graphic on a display screen. In RGB systems, the color of a pixel is defined by some Red intensity, some Green intensity, and some Blue intensity.
<b>PLL</b>	<b>Phase-Locked Loop</b>
<b>PMC</b>	<b>PCI Mezzanine Card</b>
<b>POWER</b>	<b>Performance Optimized With Enhanced RISC architecture (IBM)</b>
<b>PowerPC™</b>	The trademark used to describe the <b>Performance Optimized With Enhanced RISC</b> microprocessor architecture for <b>Personal Computers</b> developed by the IBM Corporation. PowerPC is superscalar, which means it can handle more than one instruction per clock cycle. Instructions can be sent simultaneously to three types of independent execution units (branch units, fixed-point units, and floating-point units), where they can execute concurrently, but finish out of order. PowerPC is used by Motorola, Inc. under license from IBM.
<b>PowerPC 601™</b>	The first implementation of the PowerPC family of microprocessors. This CPU incorporates a memory management unit with a 256-entry buffer and a 32KB unified (instruction and data) cache. It provides a 64-bit data bus and a separate 32-bit address bus. PowerPC 601 is used by Motorola, Inc. under license from IBM.

<b>PowerPC 603™</b>	The second implementation of the PowerPC family of microprocessors. This CPU incorporates a memory management unit with a 64-entry buffer and an 8KB (instruction and data) cache. It provides a selectable 32-bit or 64-bit data bus and a separate 32-bit address bus. PowerPC 603 is used by Motorola, Inc. under license from IBM.
<b>PowerPC 604™</b>	The third implementation of the PowerPC family of microprocessors. PowerPC 604 is used by Motorola, Inc. under license from IBM.
<b>PowerPC Reference Platform (PRP)</b>	A specification published by the IBM Power Personal Systems Division which defines the devices, interfaces, and data formats that make up a PRP-compliant system using a PowerPC processor.
<b>PowerStack™ RISC PC (System Board)</b>	A PowerPC-based computer board platform developed by the Motorola Computer Group. It supports Microsoft's Windows NT and IBM's AIX operating systems.
<b>PRP</b>	See PowerPC Reference Platform (PRP).
<b>PRP-compliant</b>	See PowerPC Reference Platform (PRP).
<b>PRP Spec</b>	See PowerPC Reference Platform (PRP).
<b>PROM</b>	<b>Programmable Read-Only Memory</b>
<b>PS/2</b>	<b>Personal System /2 (IBM)</b>
<b>QFP</b>	<b>Quad Flat Package</b>
<b>RAM</b>	<b>Random-Access Memory.</b> The temporary memory that a computer uses to hold the instructions and data currently being worked with. All data in RAM is lost when the computer is turned off.
<b>RAS</b>	<b>Row Address Strobe.</b> A clock signal used in dynamic RAMs to control the input of the row addresses.



---

<b>Raven</b>	The PowerPC-to-PCI local bus bridge chip developed by Motorola for the MVME2600 and MVME3600 series of boards. It provides the necessary interface between the PowerPC 60x bus and the PCI bus, and acts as interrupt controller.
<b>Reduced-Instruction-Set Computer (RISC)</b>	A computer in which the processor's instruction set is limited to constant-length instructions that can usually be executed in a single clock cycle.
<b>RFI</b>	Radio Frequency Interference
<b>RGB</b>	The three separate color signals: <b>R</b> ed, <b>G</b> reen, and <b>B</b> lue. Used with color displays, an interface that uses these three color signals as opposed to an interface used with a monochrome display that requires only a single signal. Both digital and analog RGB interfaces exist.
<b>RISC</b>	See Reduced Instruction Set Computer (RISC).
<b>ROM</b>	Read-Only Memory
<b>RTC</b>	Real-Time Clock
<b>SBC</b>	Single Board Computer
<b>SCSI</b>	Small Computer Systems Interface. An industry-standard high-speed interface primarily used for secondary storage. SCSI-1 provides up to 5 Mbps data transfer.
<b>SCSI-2 (Fast/Wide)</b>	An improvement over plain SCSI; and includes command queuing. Fast SCSI provides 10 Mbps data transfer on an 8-bit bus. Wide SCSI provides up to 40 Mbps data transfer on a 16- or 32-bit bus.
<b>serial port</b>	A connector that can exchange data with an I/O device one bit at a time. It may operate synchronously or asynchronously, and may include start bits, stop bits, and/or parity.
<b>SIM</b>	Serial Interface Module
<b>SIMM</b>	Single Inline Memory Module. A small circuit board with RAM chips (normally surface mounted) on it designed to fit into a standard slot.

<b>SIO</b>	<b>Super I/O controller</b>
<b>SMP</b>	<b>Symmetric MultiProcessing.</b> A computer architecture in which tasks are distributed among two or more local processors.
<b>SMT</b>	<b>Surface Mount Technology.</b> A method of mounting devices (such as integrated circuits, resistors, capacitors, and others) on a printed circuit board, characterized by not requiring mounting holes. Rather, the devices are soldered to pads on the printed circuit board. Surface-mount devices are typically smaller than the equivalent through-hole devices.
<b>software</b>	A computing system is normally spoken of as having two major components: hardware and software. Software is the term used to describe any single program or group of programs, languages, operating procedures, and documentation of a computer system. Software is the real interface between the user and the computer.
<b>SRAM</b>	<b>Static Random Access Memory</b>
<b>SSBLT</b>	<b>Source Synchronous BLock Transfer</b>
<b>standard(s)</b>	A set of detailed technical guidelines used as a means of establishing uniformity in an area of hardware or software development.
<b>SVGA</b>	<b>Super Video Graphics Array (IBM).</b> An improved VGA monitor standard that provides at least 256 simultaneous colors and a screen resolution of 800 x 600 pixels.
<b>Teletext</b>	One way broadcast of digital information. The digital information is injected in the broadcast TV signal, VBI, or full field, The transmission medium could be satellite, microwave, cable, etc. The display medium is a regular TV receiver.
<b>thick Ethernet</b>	See 10Base-5.
<b>thin Ethernet</b>	See 10Base-2.
<b>twisted-pair Ethernet</b>	See 10Base-T.
<b>UART</b>	<b>Universal Asynchronous Receiver/Transmitter</b>

---

<b>Universe</b>	ASIC developed by Tundra in consultation with Motorola, that provides the complete interface between the PCI bus and the 64-bit VMEbus.
<b>UV</b>	UltraViolet
<b>UVGA</b>	<b>Ultra Video Graphics Array.</b> An improved VGA monitor standard that provides at least 256 simultaneous colors and a screen resolution of 1024 x 768 pixels.
<b>Vertical Blanking Interval (VBI)</b>	The time it takes the beam to fly back to the top of the screen in order to retrace the opposite field (odd or even). VBI is in the order of 20 TV lines. Teletext information is transmitted over 4 of these lines (lines 14-17).
<b>VESA (bus)</b>	<b>Video Electronics Standards Association (or VL bus).</b> An internal interconnect standard for transferring video information to a computer display system.
<b>VGA</b>	<b>Video Graphics Array (IBM).</b> The third and most common monitor standard used today. It provides up to 256 simultaneous colors and a screen resolution of 640 x 480 pixels.
<b>virtual address</b>	A binary address issued by a CPU that indirectly refers to the location of information in primary memory, such as main memory. When data is copied from disk to main memory, the physical address is changed to the virtual address.
<b>VL bus</b>	See VESA Local bus (VL bus).
<b>VMEchip2</b>	MCG second generation VMEbus interface ASIC (Motorola)
<b>VME2PCI</b>	MCG ASIC that interfaces between the PCI bus and the VMEchip2 device.
<b>volatile memory</b>	A memory in which the data content is lost when the power supply is disconnected.
<b>VRAM</b>	<b>Video (Dynamic) Random Access Memory.</b> Memory chips with two ports, one used for random accesses and the other capable of serial accesses. Once the serial port has been initialized (with a transfer cycle), it can operate independently of the random port. This frees the random

port for CPU accesses. The result of adding the serial port is a significantly reduced amount of interference from screen refresh. VRAMs cost more per bit than DRAMs.

**Windows NT™**

The trademark representing **Windows New Technology**, a computer operating system developed by the Microsoft Corporation.

**XGA**

**EXtended Graphics Array**. An improved IBM VGA monitor standard that provides at least 256 simultaneous colors and a screen resolution of 1024 x 768 pixels.

**Y Signal**

Luminance. This determines the brightness of each spot (pixel) on a CRT screen either color or B/W systems, but not the color.

## Numerics

8259 compatibility 2-57  
8259 interrupts 5-4  
8259 mode 2-86

## A

A0-A31 3-5  
abbreviations, acronyms, and terms to know GL-1  
Access Timing (DRAM) 3-8, 3-9, 3-10  
Access Timing (ROM) 3-11  
address modification for little endian transfers 2-19  
Address Pipelining 3-6  
Address Transfers 3-12  
Application-Specific Integrated Circuit (ASIC) 1-1  
architectural diagram for the Universe 4-3  
architectural notes 2-87  
architectural overview 4-2  
architecture 2-55  
ARTRY\_ 3-12  
assertion, definition 1-2  
asterisk (\*) 1-2

## B

Base Module Feature Register 1-36  
Base Module Status Register (BMSR) 1-37  
big to little endian data swap 2-18  
big-endian 1-2  
big-endian mode 5-12  
binary number 1-1

Bit Descriptions 3-32  
block diagram 2-4  
block diagram description 2-60  
Bus Interface (60x) 3-12  
byte ordering 1-2  
byte, definition 1-2

## C

Cache Coherency 3-12  
CHRP compliant memory map 2-7  
CHRP memory map example 1-9  
CLK FREQUENCY 3-37  
CLK Frequency Register 3-37  
Clock Frequency 3-37  
Column Address 3-45  
CONFIG\_ADDRESS 2-52  
Control Bit Descriptions 3-32  
control bit, definition 1-2  
conventions, manual 1-1  
CPU Configuration Register 1-35  
CPU Control Register 1-32  
CSR Accesses 3-23  
CSR Architecture 3-24  
CSR Base Address 3-24  
CSR Reads and Writes 3-24  
CSR's readability 2-56  
current task priority level 2-86  
cycles originating from PCI 2-19

## D

Data Path Diagram 3-62  
Data Path Mapping 3-63  
Data Paths 3-61

Data Transfers 3-12  
decimal number 1-1  
default PCI memory map 1-13  
default processor memory map 1-8  
derc 3-39  
Disable Error Correction Control Bit 3-39  
DMA controller 4-7  
double word, definition 1-2  
DRAM Addressing 3-58  
DRAM Addressing Examples 3-64  
DRAM Attributes Register 3-35  
DRAM Base Register 3-37  
DRAM Connection Diagram 3-5  
DRAM Enable Bits 3-35  
DRAM Size Control Bits 3-36  
DRAM Speed Control Bits 3-34  
DRAM Speeds 3-7  
DRAM Tester 3-15  
DRAM Tester Control Register 3-51  
DRAM tester control registers 3-51  
dynamically changing I/O interrupt configuration 2-85

## E

ECC Control Register 3-38  
elog 3-41  
embt 3-41  
endian conversion 2-17  
endian issues 5-11  
End-of-Interrupt Registers 2-82  
EOI Register 2-86  
Error Address Register 3-43  
Error Correction 3-13  
Error Correction Codes 3-59  
Error Detection 3-13  
error handling 2-19  
Error Logger Register 3-41  
Error Logging 3-15  
error notification and handling 5-10  
Error Reporting 3-14  
ERROR\_ADDRESS 3-43  
ERROR\_SYNDROME 3-42

esbt 3-42  
escb 3-41  
esen 3-41  
exceptions 5-8  
external interrupt service 2-83  
External Register Set 3-23, 3-54  
External Register Set Reads and Writes 3-24  
External Source Destination Registers 2-78  
External Source Vector/Priority Registers 2-77

## F

Falcon-controlled system registers 1-26  
false, definition 1-3  
Fast Refresh Control Bit 3-34  
Feature Reporting Register 2-69  
Features 3-1  
features 2-2  
Flash (See ROM/Flash) 3-16  
functional description 1-6, 2-5, 4-2

## G

General Control Register 3-33  
General Control-Status/Feature Registers 2-26  
general information 4-1  
General Purpose Registers 2-43  
generating PCI configuration cycles 2-15  
generating PCI interrupt acknowledge cycles 2-16  
generating PCI memory and I/O cycles 2-14  
generating PCI special cycles 2-16  
Global Configuration Register 2-70  
glossary GL-1

## H

half-word, definition 1-2  
hexadecimal character 1-1

---

## I

- I/O Base Register 2-48
- In-Service Register (ISR) 2-62
- interpretation of MID3-MID0 1-47
- Interprocessor Interrupt Dispatch Registers 2-80
- interprocessor interrupts 2-85
- interprocessor interrupts (IPI) 2-57
- Interrupt Acknowledge Register 2-86
- Interrupt Acknowledge Registers 2-82
- interrupt delivery modes 2-58
- Interrupt Enable Control Bits 3-40
- interrupt handling 5-2
- Interrupt Pending Register (IPR) 2-61
- Interrupt Request Register (IRR) 2-62
- interrupt router 2-62
- interrupt selector (IS) 2-61
- interrupt source priority 2-56
- Interrupt Task Priority Registers 2-81
- interrupter 4-6
- interrupter and interrupt handler 4-6
- introduction 2-1, 2-55, 4-1, 5-1
- IPI Vector/ Priority Registers 2-72
- ISA DMA channels 1-48, 5-7
- ISA local resource bus 1-33

## L

- L2 Cache Support 3-13
- L2CLM\_ 3-13
- Large Scale Integration (LSI) 1-1
- little-endian mode 5-13
- LM/SIG Control Register 1-39
- LM/SIG Status Register 1-40
- Location Monitor Lower Base Address Register 1-42
- Location Monitor Upper Base Address Register 1-41

## M

- manual terminology 1-2
- manufacturers' documents A-2
- mcken 3-40

- Memory Base Register 2-49
- Memory Configuration Register (MEM-CR) 1-29
- memory maps 1-7
- mien 3-40
- MK48T59 access registers 1-34
- module configuration and status registers 1-34
- Motorola Computer Group documents A-1
- MPC arbiter 2-5
- MPC Arbiter Control Register 2-29
- MPC bus interface 2-5
- MPC bus timer 2-10
- MPC Error Address Register 2-36
- MPC Error Attribute Register - MERAT 2-36
- MPC Error Enable Register 2-31
- MPC Error Status Register 2-34
- MPC map decoders 2-7
- MPC master 2-9
- MPC registers 2-23
- MPC Slave Address (0,1 and 2) Registers 2-39
- MPC Slave Address (3) Register 2-40
- MPC Slave Offset/ Attribute (0,1 and 2) Registers 2-41
- MPC Slave Offset/ Attribute (3) Registers 2-42
- MPC transfer types 2-10
- MPC write posting 2-8
- MPIC registers 2-64
- MVME2600 series 1-1
- MVME2600 series features summary 1-3
- MVME2600 series interrupt architecture 5-2
- MVME2600 series system block diagram 1-5
- MVME712M mode 1-6
- MVME761 mode 1-6

**N**

negation, definition 1-2  
 nesting of interrupt events 2-56  
 NVRAM/RTC & Watchdog Timer Registers 1-33

**O**

operation 2-85  
 overview 1-3, 2-1

**P**

Parity Checking 3-55  
 PC87308VUL Super I/O (ISASIO) strapping 1-33  
 PCI arbitration 5-1  
 PCI arbitration assignments 5-1  
 PCI bus interface 4-5  
 PCI CHRP memory map 1-13  
 PCI command codes 2-13  
 PCI Command/ Status Registers 2-45  
 PCI configuration access 1-12  
 PCI configuration space 2-12  
 PCI domain 5-14  
 PCI I/O CONFIG\_ADDRESS Register 2-52  
 PCI I/O CONFIG\_DATA Register 2-54  
 PCI interface 2-11  
 PCI Interrupt Acknowledge Register 2-38  
 PCI map decoders 2-11  
 PCI master 2-13  
 PCI memory maps 1-13  
 PCI PREP memory map 1-17  
 PCI registers 2-43  
 PCI Slave Address (0,1,2 and 3) Registers 2-50  
 PCI Slave Attribute/ Offset (0,1,2 and 3) Registers 2-51  
 PCI spread I/O cycle mapping 2-15  
 PCI write posting 2-12  
 PCI/MPC contention handling 2-21  
 PCI-Ethernet 5-15

PCI-graphics 5-15  
 PCI-SCSI 5-14  
 Performance 3-6  
 PIB DMA channel assignments 1-48  
 PIB interrupt handler block diagram 5-5  
 PIB PCI/ISA interrupt assignments 5-6  
 Power-Up Reset Status Bit 3-38  
 Power-Up Reset Status Register 3-53  
 PR\_STATL 3-53  
 PR\_STATU 3-53  
 PREP memory map example 1-11  
 Prescaler Adjust Register 2-30  
 processor CHRP memory map 1-9  
 Processor Init Register 2-71  
 processor memory maps 1-7  
 processor PREP memory map 1-11  
 processor/memory domain 5-14  
 processor's current task priority 2-56  
 product overview - features 4-1  
 program visible registers 2-61  
 programming details 5-1  
 programming model 1-7  
 programming notes 2-83  
 Programming ROM/Flash 3-55, 3-56

**R**

RAM A BASE 3-37  
 RAM B BASE 3-37  
 RAM C BASE 3-37  
 RAM D BASE 3-37  
 Raven block diagram 2-4  
 Raven interrupt controller (RavenMPIC) features 2-55  
 Raven interrupt controller implementation 2-55  
 Raven MPC register map 2-23  
 Raven MPC register values for CHRP memory map 1-10  
 Raven MPC register values for PREP memory map 1-12  
 Raven PCI configuration register map 2-44



---

Raven PCI Host Bridge & Multi-Processor Interrupt Controller chip 2-1  
 Raven PCI I/O register map 2-44  
 Raven PCI register values for CHRP memory map 1-15  
 Raven PCI register values for PREP memory map 1-18  
 Raven's involvement 5-14  
 Raven-detected errors 2-58  
 Raven-Detected Errors Destination Register 2-80  
 Raven-Detected Errors Vector/Priority Register 2-79  
 RavenMPIC 5-3  
 RavenMPIC block diagram 2-60  
 RavenMPIC interrupt assignments 5-3  
 RavenMPIC register map 2-65  
 RavenMPIC registers 2-64  
 Read/Write Checkbits Control Bit 3-38  
 Reads/Writes to ROM/Flash 3-48  
 Refresh Counter Test Control Bits 3-44  
 Refresh/Scrub 3-20  
 Refresh/Scrub Address Register 3-44  
 Register Bit Descriptions 3-32  
 Register Summary 3-29  
 registers 2-23  
 registers - Universe Control and Status Registers (UCSR) 4-8  
 related documentation A-1  
 related specifications A-7  
 requirements 2-2  
 reset sources and devices affected 5-9  
 reset state 2-84  
 Revision ID 3-33  
 Revision ID Register 2-25  
 Revision ID/Class Code Registers 2-47  
 ROM/Flash 3-16  
 ROM/Flash A Base Address Control Bits 3-45  
 ROM/Flash A Base/Size Register 3-45  
 ROM/Flash A Size Encoding 3-46  
 ROM/Flash A Width Control Bit 3-46  
 ROM/Flash B Base Address Control Bits 3-48  
 ROM/Flash B Base/Size Register 3-48  
 ROM/Flash B Width Control Bit 3-49  
 ROM/FLASH bank default 5-16  
 ROM/Flash initialization 5-16  
 ROM/Flash Speed 3-11  
 rom\_a\_64 3-46  
 ROM\_A\_BASE 3-45  
 rom\_a\_en 3-47  
 rom\_a\_rv 3-47  
 rom\_a\_siz 3-46  
 rom\_a\_we 3-47  
 rom\_b\_64 3-49  
 ROM\_B\_BASE 3-48  
 rom\_b\_en 3-50  
 rom\_b\_rv 3-50  
 rom\_b\_siz 3-50  
 rom\_b\_we 3-50  
 Row Address 3-44  
 rtest0-rtest2 3-44  
 rwcB 3-38

**S**

SBE\_COUNT 3-42  
 scb0,scb1 3-43  
 scien 3-38, 3-40  
 scof 3-42  
 Scrub Counter 3-43  
 Scrub Write Enable Control Bit 3-44  
 Scrub/Refresh Register 3-43  
 Semaphore Register 1 1-42  
 Semaphore Register 2 1-43  
 Seven-Segment Display Register 1-38  
 sien 3-40  
 Single Bit Error Counter 3-42  
 single word, definition 1-2  
 Single-beat Writes 3-7  
 small-endian 1-2  
 soft reset 5-9  
 Software Considerations 3-55  
 sources of reset 5-8

spurious vector generation 2-57  
 Spurious Vector Register 2-73  
 SRAM Base Address 3-24  
 SRAM Reads and Writes 3-24  
 Status Bit Descriptions 3-32  
 status bit, definition 1-2  
 strap pins configuration for the  
 PC87308VUL 1-33  
 swen 3-44  
 Syndrome Codes 3-59  
 System Configuration Register (SYSCR)  
 1-27  
 System External Cache Control Register  
 (SXCCR) 1-30  
 system register summary 1-26

## T

Test SRAM 3-52  
 tester control registers 3-51  
 Tester Description 3-15  
 tien 3-40  
 Timer Basecount Registers 2-74  
 Timer Current Count Registers 2-74  
 Timer Destination Registers 2-76  
 Timer Frequency Register 2-73  
 Timer Vector/Priority Registers 2-75  
 timers 2-58  
 Timing (DRAM Access) 3-8, 3-9, 3-10  
 Timing (ROM/Flash Access) 3-11  
 true, definition 1-2  
 trun 3-51  
 tsse 3-51

## U

UCSR access mechanisms 4-8  
 Universe (VMEbus to PCI) chip 4-1  
 Universe as PCI master 4-6  
 Universe as PCI slave 4-5  
 Universe as VMEbus master 4-4  
 Universe as VMEbus slave 4-4  
 Universe PCI Register values for CHRP  
 memory map 1-15

Universe PCI register values for PREP  
 memory map 1-19  
 Universe PCI register values for VMEbus  
 slave map example 1-24  
 Universe register map 4-9  
 Universe's involvement 5-15  
 Upper/Lower Chip Status Bit 3-35

## V

Vendor ID/ Device ID Registers 2-45  
 Vendor ID/Device ID Registers 2-25  
 Vendor Identification Register 2-71  
 Vendor/Device Register 3-33  
 VME Geographical Address Register  
 (VGAR) 1-43  
 VME registers 1-38, 1-39  
 VMEbus domain 5-15  
 VMEbus interface 4-4  
 VMEbus interrupt handling 4-7  
 VMEbus mapping 1-20  
 VMEbus master map 1-20  
 VMEbus master mapping 1-21  
 VMEbus slave map 1-22  
 VMEbus slave map example 1-25  
 VMEbus slave mapping 1-23

## W

W83C553 PIB registers 1-33  
 when MPC devices are big-endian 2-17  
 when MPC devices are little endian 2-18  
 word, definition 1-2  
 Writing to the Control Registers 3-55

## Z

Z85230 ESCC and Z8536 CIO registers  
 and port pins 1-44  
 Z8536 CIO port pins 1-45  
 Z8536 CIO port pins assignment 1-45  
 Z8536/Z85230 access registers 1-44  
 Z8536/Z85230 registers 1-44