

**MVME1X2P4  
VME Embedded Controller**

**Programmer's Reference  
Guide**

**V1X2P4A/PG2**

Edition of November 2000

© Copyright 2000 Motorola, Inc.

All rights reserved.

Printed in the United States of America.

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

MC68040™ and MC68060™ are trademarks of Motorola, Inc.

IndustryPack™ and IP™ are trademarks of GreenSpring Computers, Inc.

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

## Safety Summary

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

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

### **Ground the Instrument.**

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

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

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

### **Keep Away From Live Circuits Inside the Equipment.**

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

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

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

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

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

### **Observe Warnings in Manual.**

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



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

## Flammability

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

## EMI Caution



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

## Lithium Battery Caution

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



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



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



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

## **CE Notice (European Community)**

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

EN55022 “Limits and Methods of Measurement of Radio Interference Characteristics of Information Technology Equipment”; this product tested to Equipment Class B

EN50082-1:1997 “Electromagnetic Compatibility—Generic Immunity Standard, Part 1. Residential, Commercial and Light Industry”

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

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

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

### **Notice**

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.

Electronic versions of this material may be read online, downloaded for personal use, or referenced in another document as a URL to the Motorola Computer Group website. The text itself may not be published commercially in print or electronic form, edited, translated, or otherwise altered without the 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 available 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.

## **Limited and Restricted Rights Legend**

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

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

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

# Contents

---

## About This Manual

Summary of Changes .....	xxii
Overview of Contents .....	xxii
Comments and Suggestions .....	xxiii
Conventions Used in This Manual .....	xxiv

## CHAPTER 1 Board Description and Memory Maps

Introduction .....	1-1
Overview .....	1-2
The Petra ASIC and Second-Generation MVME1X2 Boards .....	1-2
I/O Routing .....	1-3
VMEbus Interface .....	1-5
Petra ASIC .....	1-5
Applicable Industry Standards .....	1-7
Block Diagram .....	1-7
Functional Description .....	1-9
VMEbus Interface and VMEchip2 .....	1-9
Memory Maps .....	1-11
Local Bus Memory Map .....	1-11
Normal Address Range .....	1-11
Detailed I/O Memory Maps .....	1-17
BBRAM/TOD Clock Memory Map .....	1-33
Interrupt Acknowledge Map .....	1-39
VMEbus Memory Map .....	1-39
VMEbus Accesses to the Local Bus .....	1-40
VMEbus Short I/O Memory Map .....	1-40
Software Support Considerations .....	1-40
Interrupts .....	1-40
Cache Coherency .....	1-41
Sources of Local BERR* .....	1-41
Local Bus Time-Out .....	1-41
VMEbus Access Time-Out .....	1-42
VMEbus BERR* .....	1-42
Local DRAM Parity Error .....	1-42

---

VMEchip2 .....	1-42
Bus Error Processing .....	1-43
Description of Error Conditions on the MVME1X2P4.....	1-43
MPU Parity Error .....	1-43
MPU Offboard Error .....	1-44
MPU TEA - Cause Unidentified .....	1-44
MPU Local Bus Timeout .....	1-44
DMAC VMEbus Error.....	1-45
DMAC Parity Error.....	1-45
DMAC Offboard Error.....	1-46
DMAC LTO Error.....	1-46
DMAC TEA - Cause Unidentified.....	1-47
LAN Parity Error.....	1-47
LAN Offboard Error .....	1-48
LAN LTO Error .....	1-48
SCSI Parity Error .....	1-49
SCSI Offboard Error .....	1-49
SCSI LTO Error .....	1-50
Proper Use of Bus Timers .....	1-50
MVME1X2P4 MC680x0 Indivisible Cycles .....	1-51
Illegal Access to IP Modules from External VMEbus Masters .....	1-52

## CHAPTER 2 VMEchip2

Introduction .....	2-1
Functional Blocks .....	2-4
Local-Bus-to-VMEbus Interface .....	2-4
Local-Bus-to-VMEbus Requester .....	2-7
VMEbus-to-Local-Bus Interface .....	2-9
Local-Bus-to-VMEbus DMA Controller .....	2-10
No-Address-Increment DMA Transfers .....	2-12
DMAC VMEbus Requester .....	2-13
Tick and Watchdog Timers.....	2-14
Prescaler .....	2-14
Tick Timers .....	2-15
Watchdog Timer.....	2-15
VMEbus Interrupter .....	2-16
VMEbus System Controller .....	2-17
Arbiter .....	2-17
IACK Daisy-Chain Driver .....	2-17
Bus Timer.....	2-17
Reset Driver .....	2-18



---

Local Bus Interrupter and Interrupt Handler .....	2-18
Global Control and Status Registers.....	2-20
LCSR Programming Model .....	2-20
Programming the VMEbus Slave Map Decoders.....	2-26
VMEbus Slave Ending Address Register 1 .....	2-28
VMEbus Slave Starting Address Register 1 .....	2-28
VMEbus Slave Ending Address Register 2 .....	2-29
VMEbus Slave Starting Address Register 2 .....	2-29
VMEbus Slave Address Translation Address Offset Register 1 .....	2-29
VMEbus Slave Address Translation Select Register 1 .....	2-30
VMEbus Slave Address Translation Address Offset Register 2 .....	2-31
VMEbus Slave Address Translation Select Register 2 .....	2-31
VMEbus Slave Write Post / Snoop Control Register 2.....	2-32
VMEbus Slave Address Modifier Select Register 2 .....	2-33
VMEbus Slave Write Post and Snoop Control Register 1 .....	2-35
VMEbus Slave Address Modifier Select Register 1 .....	2-36
Programming the Local-Bus-to-VMEbus Map Decoders .....	2-37
Local Bus Slave (VMEbus Master) Ending Address Register 1.....	2-40
Local Bus Slave (VMEbus Master) Starting Address Register 1.....	2-40
Local Bus Slave (VMEbus Master) Ending Address Register 2.....	2-40
Local Bus Slave (VMEbus Master) Starting Address Register 2.....	2-41
Local Bus Slave (VMEbus Master) Ending Address Register 3 .....	2-41
Local Bus Slave (VMEbus Master) Starting Address Register 3 .....	2-41
Local Bus Slave (VMEbus Master) Ending Address Register 4 .....	2-42
Local Bus Slave (VMEbus Master) Starting Address Register 4 .....	2-42
Local Bus Slave (VMEbus Master)	
Address Translation Address Register 4 .....	2-42
Local Bus Slave (VMEbus Master)	
Address Translation Select Register 4 .....	2-43
Local Bus Slave (VMEbus Master) Attribute Register 4 .....	2-43
Local Bus Slave (VMEbus Master) Attribute Register 3 .....	2-44
Local Bus Slave (VMEbus Master) Attribute Register 2 .....	2-45
Local Bus Slave (VMEbus Master) Attribute Register 1 .....	2-46
VMEbus Slave GCSR Group Address Register .....	2-47
VMEbus Slave GCSR Board Address Register .....	2-48
Local-Bus-to-VMEbus Enable Control Register .....	2-49
Local-Bus-to-VMEbus I/O Control Register .....	2-50
ROM Control Register .....	2-51
Programming the VMEchip2 DMA Controller .....	2-51
DMAC Registers.....	2-53
EPROM Decoder, SRAM and DMA Control Register .....	2-53
Local-Bus-to-VMEbus Requester Control Register .....	2-54

---

DMAC Control Register 1 (bits 0-7) .....	2-55
DMAC Control Register 2 (bits 8-15) .....	2-57
DMAC Control Register 2 (bits 0-7) .....	2-58
DMAC Local Bus Address Counter .....	2-59
DMAC VMEbus Address Counter .....	2-60
DMAC Byte Counter .....	2-60
Table Address Counter .....	2-60
VMEbus Interrupter Control Register .....	2-61
VMEbus Interrupter Vector Register .....	2-62
MPU Status and DMA Interrupt Count Register .....	2-62
DMAC Status Register .....	2-63
Programming the Tick and Watchdog Timers.....	2-64
VMEbus Arbiter Time-Out Control Register .....	2-64
DMAC Ton/Toff Timers and	
VMEbus Global Time-out Control Register .....	2-65
VME Access, Local Bus, and Watchdog Time-out Control Register .....	2-66
Prescaler Control Register .....	2-67
Tick Timer 1 Compare Register .....	2-68
Tick Timer 1 Counter .....	2-68
Tick Timer 2 Compare Register .....	2-69
Tick Timer 2 Counter .....	2-69
Board Control Register .....	2-70
Watchdog Timer Control Register .....	2-71
Tick Timer 2 Control Register .....	2-72
Tick Timer 1 Control Register .....	2-73
Prescaler Counter .....	2-73
Programming the Local Bus Interrupter.....	2-74
Local Bus Interrupter Status Register (bits 24-31) .....	2-77
Local Bus Interrupter Status Register (bits 16-23) .....	2-78
Local Bus Interrupter Status Register (bits 8-15) .....	2-79
Local Bus Interrupter Status Register (bits 0-7) .....	2-80
Local Bus Interrupter Enable Register (bits 24-31) .....	2-81
Local Bus Interrupter Enable Register (bits 16-23) .....	2-82
Local Bus Interrupter Enable Register (bits 8-15) .....	2-83
Local Bus Interrupter Enable Register (bits 0-7) .....	2-84
Software Interrupt Set Register (bits 8-15) .....	2-85
Interrupt Clear Register (bits 24-31) .....	2-85
Interrupt Clear Register (bits 16-23) .....	2-86
Interrupt Clear Register (bits 8-15) .....	2-87
Interrupt Level Register 1 (bits 24-31) .....	2-87
Interrupt Level Register 1 (bits 16-23) .....	2-88
Interrupt Level Register 1 (bits 8-15) .....	2-88

---

Interrupt Level Register 1 (bits 0-7) .....	2-89
Interrupt Level Register 2 (bits 24-31) .....	2-89
Interrupt Level Register 2 (bits 16-23) .....	2-90
Interrupt Level Register 2 (bits 8-15) .....	2-90
Interrupt Level Register 2 (bits 0-7) .....	2-91
Interrupt Level Register 3 (bits 24-31) .....	2-91
Interrupt Level Register 3 (bits 16-23) .....	2-92
Interrupt Level Register 3 (bits 8-15) .....	2-92
Interrupt Level Register 3 (bits 0-7) .....	2-93
Interrupt Level Register 4 (bits 24-31) .....	2-93
Interrupt Level Register 4 (bits 16-23) .....	2-94
Interrupt Level Register 4 (bits 8-15) .....	2-94
Interrupt Level Register 4 (bits 0-7) .....	2-95
Vector Base Register .....	2-95
I/O Control Register 1 .....	2-96
I/O Control Register 2 .....	2-97
I/O Control Register 3 .....	2-97
Miscellaneous Control Register .....	2-98
GCSR Programming Model.....	2-100
Programming the GCSR.....	2-102
VMEchip2 Revision Register .....	2-103
VMEchip2 ID Register.....	2-104
VMEchip2 LM/SIG Register .....	2-104
VMEchip2 Board Status/Control Register .....	2-106
General Purpose Register 0 .....	2-107
General Purpose Register 1 .....	2-107
General Purpose Register 2 .....	2-107
General Purpose Register 3 .....	2-108
General Purpose Register 4 .....	2-108
General Purpose Register 5 .....	2-108

## CHAPTER 3 MC2 Functions

Introduction.....	3-1
Summary of Major Features .....	3-1
Functional Description.....	3-2
Petra/MC2 Chip Initialization.....	3-2
Flash and EPROM Interface .....	3-2
BBRAM Interface.....	3-3
82596CA LAN Interface .....	3-3
MPU Port and MPU Channel Attention.....	3-3

---

MC680x0-Bus Master Support for 82596CA .....	3-4
LANC Bus Error .....	3-4
LANC Interrupt .....	3-5
53C710 SCSI Controller Interface .....	3-5
SRAM Memory Controller .....	3-5
SDRAM Memory Controller.....	3-6
Z85230 SCC Interface.....	3-7
Tick Timers.....	3-8
Watchdog Timer .....	3-9
Local Bus Timer .....	3-9
Petra/MC2 Registers: Memory Map.....	3-10
Programming Model.....	3-11
MC2 Sector ID Register .....	3-12
MC2 Sector Revision Register .....	3-12
General Control Register .....	3-12
Interrupt Vector Base Register .....	3-13
Programming the Tick Timers.....	3-15
Tick Timer 1 and 2 Compare and Counter Registers.....	3-15
LSB Prescaler Count Register.....	3-17
Prescaler Clock Adjust Register.....	3-17
Tick Timer 1 and 2 Control Registers.....	3-18
Tick Timer Interrupt Control Registers.....	3-19
DRAM Parity Error Interrupt Control Register .....	3-21
SCC Interrupt Control Register .....	3-22
Tick Timer 3 and 4 Control Registers .....	3-23
DRAM and SRAM Memory Controller Registers.....	3-24
DRAM Space Base Address Register .....	3-24
SRAM Space Base Address Register.....	3-25
DRAM Space Size Register .....	3-25
DRAM/SRAM Options Register .....	3-27
SRAM Space Size Register.....	3-28
LANC Error Status Register.....	3-29
82596CA LANC Interrupt Control Register .....	3-30
LANC Bus Error Interrupt Control Register .....	3-31
SCSI Error Status Register .....	3-32
General-Purpose Inputs Register .....	3-33
MVME1X2P4 Version Register .....	3-34
SCSI Interrupt Control Register .....	3-35
Tick Timer 3 and 4 Compare and Counter Registers .....	3-36
Bus Clock Register.....	3-37
EPROM Access Time Control Register .....	3-38
Flash Access Time Control Register .....	3-39

---

ABORT Switch Interrupt Control Register .....	3-40
RESET Switch Control Register.....	3-41
Watchdog Timer Control Register .....	3-42
Access and Watchdog Time Base Select Register .....	3-43
DRAM Control Register.....	3-44
MPU Status Register.....	3-46
32-bit Prescaler Count Register .....	3-47

## CHAPTER 4 IP2 Functions

Introduction.....	4-1
Summary of Major Features .....	4-1
IP2/IP1 Bug Fixes.....	4-2
Command Chain Table Update .....	4-2
DMA End .....	4-2
IP Bus Reset Compatibility .....	4-3
Functional Description.....	4-3
General Description .....	4-3
Cache Coherency .....	4-4
Local Bus to IndustryPack DMA Controllers.....	4-4
Clocking Environments and Performance .....	4-6
Programmable Clock .....	4-8
Error Reporting .....	4-8
Error Reporting as a Local Bus Slave .....	4-8
Error Reporting as a Local Bus Master .....	4-8
IndustryPack Error Reporting.....	4-9
Interrupts.....	4-9
Overall Memory Map .....	4-10
Programming Model .....	4-11
Chip ID Register .....	4-18
Chip Revision Register .....	4-18
Vector Base Register.....	4-19
Memory Base Address Registers: IP_a, IP_b, IP_c, IP_d.....	4-20
IP_a or Double-Size IP_ab Memory Base Address Registers .....	4-20
IP_b Memory Base Address Registers .....	4-21
IP_c or Double-Size IP_cd Memory Base Address Registers.....	4-21
IP_d Memory Base Address Registers .....	4-21
Memory Size Registers: IP_a, IP_b, IP_c, IP_d.....	4-22
IRQ0/IRQ1 Interrupt Control Registers: IP_a, IP_b, IP_c, IP_d.....	4-23
General Control Registers: IP_a, IP_b, IP_c, IP_d.....	4-24
IP Clock Register .....	4-28

---

DMA Arbitration Control Register .....	4-29
IP RESET Register .....	4-30
Programming the DMA Controllers.....	4-31
DMA Enable Function .....	4-34
DMA Control and Status Register Set Definition.....	4-34
Programming the Programmable Clock.....	4-44
Local Bus to IndustryPack Addressing .....	4-49
Eight-Bit Memory Space.....	4-50
16-Bit Memory Space .....	4-51
32-Bit Memory Space .....	4-52
IP_a I/O Space.....	4-53
IP_ab I/O Space.....	4-54
IP_a ID Space.....	4-55
IP to Local Bus Data Routing.....	4-56
Memory Space Accesses .....	4-56
I/O and ID Space Accesses .....	4-58

## CHAPTER 5 MCECC Functions

Introduction .....	5-1
Features.....	5-2
Functional Description .....	5-3
General Description.....	5-3
Performance.....	5-3
Cache Coherency.....	5-4
ECC .....	5-5
Cycle Types.....	5-5
Error Reporting .....	5-5
Error Logging .....	5-8
Scrub.....	5-8
Refresh.....	5-8
Arbitration .....	5-9
Chip Defaults.....	5-9
Programming Model.....	5-10
Chip ID Register.....	5-13
Chip Revision Register.....	5-13
Memory Configuration Register .....	5-14
Base Address Register.....	5-15
DRAM Control Register .....	5-15
BCLK Frequency Register .....	5-16
Data Control Register .....	5-17

---

Scrub Control Register.....	5-19
Scrub Period Register Bits 15-8.....	5-20
Scrub Period Register Bits 7-0.....	5-20
Chip Prescaler Counter .....	5-21
Scrub Time On/Time Off Register.....	5-21
Scrub Prescaler Counter (Bits 21-16).....	5-23
Scrub Prescaler Counter (Bits 15-8).....	5-23
Scrub Prescaler Counter (Bits 7-0).....	5-24
Scrub Timer Counter (Bits 15-8).....	5-24
Scrub Timer Counter (Bits 7-0).....	5-25
Scrub Address Counter (Bits 26-24).....	5-25
Scrub Address Counter (Bits 23-16).....	5-26
Scrub Address Counter (Bits 15-8).....	5-26
Scrub Address Counter (Bits 7-4).....	5-26
Error Logger Register .....	5-27
Error Address (Bits 31-24) .....	5-28
Error Address (Bits 23-16) .....	5-28
Error Address Bits (15-8) .....	5-29
Error Address Bits (7-4) .....	5-29
Error Syndrome Register .....	5-30
Defaults Register 1.....	5-30
Defaults Register 2.....	5-32
SDRAM Configuration Register .....	5-33
Initialization .....	5-34
Syndrome Decoding.....	5-36

## **APPENDIX A Summary of Changes**

Introduction.....	A-1
-------------------	-----

## **APPENDIX B Using Interrupts on the MVME1X2P4**

Example: Tick Timer 1 Periodic Interrupt .....	B-1
--	-----

## **APPENDIX C Related Documentation**

Motorola Computer Group Documents .....	C-1
Manufacturers' Documents.....	C-2
Related Specifications.....	C-3

---



# List of Figures

---

Figure 1-1. MVME1X2P4 Memory Control Options.....	1-4
Figure 1-2. MVME1X2P4 Block Diagram.....	1-8
Figure 2-1. VMEchip2 Block Diagram .....	2-5

---

# List of Tables

---

Table 1-1. MVME1X2P4 Features Summary .....	1-6
Table 1-2. Functions Duplicated in VMEchip2 and Petra ASICs.....	1-10
Table 1-3. MVME1X2P4 Local Bus Memory Map .....	1-12
Table 1-4. MVME1X2P4 Local I/O Devices Memory Map .....	1-14
Table 1-5. VMEchip2 Memory Map (Sheet 1 of 3).....	1-18
Table 1-6. Petra MC2 Functions: Register Map .....	1-23
Table 1-7. Petra IP2 Functions: Overall Memory Map.....	1-24
Table 1-8. Petra IP2 Memory Map — Control/Status Registers.....	1-25
Table 1-9. Z85230 SCC Register Addresses.....	1-31
Table 1-10. 82596CA Ethernet LAN Memory Map .....	1-31
Table 1-11. 53C710 SCSI Memory Map.....	1-32
Table 1-12. M48T58 BBRAM/TOD Clock Memory Map .....	1-33
Table 1-13. BBRAM Configuration Area Memory Map.....	1-34
Table 1-14. TOD Clock Memory Map.....	1-35
Table 2-1. Features of the VMEchip2 ASIC.....	2-1
Table 2-2. VMEchip2 Memory Map—LCSR Summary (Sheet 1 of 2) .....	2-22
Table 2-3. DMAC Command Packet Format.....	2-53
Table 2-4. Local Bus Interrupter Summary .....	2-75
Table 2-5. VMEchip2 Memory Map (GCSR Summary).....	2-103
Table 3-1. MC2 Functions on the Petra ASIC .....	3-1
Table 3-2. SDRAM Performance.....	3-7
Table 3-3. SCC Device Addressing .....	3-8
Table 3-4. Petra MC2 Functions: Register Map .....	3-10
Table 3-5. Interrupt Vector Base Register Encoding and Priority.....	3-14
Table 3-6. DRAM Size Control Bit Encoding .....	3-26
Table 3-7. DRAM Size Control Bit Encoding .....	3-27
Table 3-8. SRAM Size Control Bit Encoding.....	3-28
Table 3-9. SRAM Size Control Bit Encoding.....	3-29
Table 4-1. IP2 Functions on the Petra ASIC.....	4-1
Table 4-2. IP2 sector Clock Cycles .....	4-7
Table 4-3. IP2 Sector Overall Memory Map.....	4-10
Table 4-4. Petra IP2 Memory Map — Control/Status Registers.....	4-12
Table 5-1. MCECC Functions on the Petra ASIC .....	5-2
Table 5-2. Memory System Cycle Timing.....	5-4

---

Table 5-3. MCECC Sector Internal Register Memory Map .....	5-11
Table 5-4. Syndrome Bit Encoding .....	5-36
Table 5-5. Identifying SDRAM Bank in Error .....	5-37
Table A-1. List of Changes .....	A-1
Table C-1. Motorola Computer Group Documents .....	C-1
Table C-2. Manufacturers' Documents .....	C-2
Table C-3. Related Specifications .....	C-3

---

# About This Manual

This manual provides board-level information and detailed ASIC information, including register bit descriptions, for the MVME162P-244 and -344 as well as the MVME172P-644 series of embedded controllers, known collectively as the “MVME1X2P4” because they are equipped with the “Petra” chip and accommodate up to four IP modules.

The “Petra” chip that distinguishes MVME1X2P4 embedded controllers is an application-specific integrated circuit (ASIC) which combines the functions previously covered by the MC2 chip, the IP2 chip, and the MCECC chip in a single ASIC. As of the publication date, the information presented in this manual applies to the following MVME1X2P4 models:

Model Number	Characteristics
MVME162P-244L	25MHz 68LC040, 16MB SDRAM, 2 SIO, 4 DMA IP
MVME162P-244LE	25MHz 68LC040, 16MB SDRAM, 2 SIO, 4 DMA IP, Ethernet
MVME162P-244LSE	25MHz 68LC040, 16MB SDRAM, 2 SIO, 4 DMA IP, SCSI/Ethernet
MVME162P-344	32MHz 68040, 16MB SDRAM, 2 SIO, 4 DMA IP
MVME162P-344S	32MHz 68040, 16MB SDRAM, 2 SIO, 4 DMA IP, SCSI
MVME162P-344E	32MHz 68040, 16MB SDRAM, 2 SIO, 4 DMA IP, Ethernet
MVME162P-344SE	32MHz 68040, 16MB SDRAM, 2 SIO, 4 DMA IP, SCSI/Ethernet
MVME172P-644L	64MHz 68LC060, 16MB SDRAM, 2 SIO, 4 DMA IP
MVME172P-644LE	64MHz 68LC060, 16MB SDRAM, 2 SIO, 4 DMA IP, Ethernet
MVME172P-644LSE	60MHz 68060, 16MB SDRAM, 2 SIO, 4 DMA IP, SCSI/Ethernet

If the part number of your board includes a "PA" (for example: MVME162PA-244L), your board is equipped with a second-generation Petra ASIC. All other particulars of the board remain the same.

This manual is intended for anyone who designs OEM systems, adds capability to an existing compatible system, or works in a lab environment for experimental purposes. A basic knowledge of computers and digital logic is assumed. To use this manual, you may also wish to become familiar with the publications listed in [Appendix C, Related Documentation](#).

---

## Summary of Changes

This is the second edition of the *Programmer's Reference Guide*. It supersedes the July 2000 edition and incorporates the following updates.

Date	Description of Change
October 2000	The part number of the manual was changed from V1X2PFXA/PG1 to V1X2P4A/PG1 to better reflect board nomenclature.
October 2000	Descriptions of the snoop function on boards equipped with the MC68060 processor (as described on page 2-32, for example) have been updated.
October 2000	References to IP2 and MC2 revision levels at various points in the text (in Chapter 3 under <i>MC2 Sector Revision Register</i> , for example) were incremented from 02 to 03 to reflect this change in the Petra chip.
October 2000	References in Chapter 4 to an irregularity in the behavior of the DMA "halt" function across DMA channels A, B, C, and D (as described on page 4-32, for example) were dropped, as this anomaly has been corrected.

## Overview of Contents

[Chapter 1, \*Board Description and Memory Maps\*](#), describes the board-level hardware features of MVME1X2P4 single-board computers. It includes memory maps and a discussion of some general software considerations such as cache coherency, interrupts, and bus errors.

[Chapter 2, \*VMEchip2\*](#), describes the VMEchip2 ASIC, the local bus/VMEbus interface chip on MVME1X2P4 boards.

[Chapter 3, \*MC2 Functions\*](#), describes MC2 functionality as implemented in the Petra ASIC. The MC2 Memory Controller ASIC is one of three ASICs (the MC2 chip, the IP2 chip, and the MCECC chip) used on earlier MVME162/172 models whose functions are now incorporated into the Petra chip on the MVME1X2P4.

[Chapter 4, \*IP2 Functions\*](#), describes IP2 functionality as implemented in the Petra ASIC. The IP2 IndustryPack Interface Controller ASIC is the second of three ASICs used on earlier MVME162/172 models whose functions are now incorporated into the Petra chip on the MVME1X2P4.

---

[Chapter 5, \*MCECC Functions\*](#), describes MCECC functionality as implemented in the Petra ASIC. The ECC DRAM Controller ASIC (MCECC) is the third of three ASICs used on earlier MVME162/172 models whose functions are now incorporated into the Petra chip on the MVME1X2P4.

[Appendix A, \*Summary of Changes\*](#), lists the modifications that accompanied the introduction of the Petra ASIC on the MVME162P4 and MVME172P4.

[Appendix B, \*Using Interrupts on the MVME1X2P4\*](#), illustrates an approach to generating and handling a VMEchip2 tick timer interrupt on an MVME1X2P4 that has a VMEbus connection.

[Appendix C, \*Related Documentation\*](#), lists all documentation related to the MVME162P4 and MVME172P4.

## Comments and Suggestions

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

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

You can also submit comments to the following e-mail address:  
[reader-comments@mcg.mot.com](mailto:reader-comments@mcg.mot.com)

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

---

## Conventions Used in This Manual

The following typographical conventions are used in this document:

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

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.

### **bold**

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

### *italic*

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

### `courier`

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

### <Enter>, <Return> or <CR>

represent the carriage return or Enter key.

### **CTRL**

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



---

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 are defined as follows:

- ❑ A *byte* is eight bits, numbered 0 through 7, with bit 0 being the least significant.
- ❑ A *word* is 16 bits, numbered 0 through 15, with bit 0 being the least significant.
- ❑ A *longword* is 32 bits, numbered 0 through 31, with bit 0 being the least significant.

The terms *control bit*, *status bit*, *true*, and *false* 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.

---

# Board Description and Memory Maps

---

# 1

## Introduction

This manual supplies programming information for the MVME162P4 and MVME172P4 Embedded Controllers, known collectively as the “MVME1X2P4”. Extensive programming information is provided for the Application-Specific Integrated Circuit (ASIC) devices used on the board. Reference information is also included for the Large Scale Integration (LSI) devices used on the board and sources for additional information are provided as well.

This chapter briefly describes the board level hardware features of the MVME1X2P4 Embedded Controller. The chapter begins with a board-level overview and features list. Memory maps are next, and the chapter closes with some general software considerations such as cache coherency, interrupts, and bus errors. All programmable registers that reside in ASICs on the MVME1X2P4 are covered in the chapters devoted to those ASICs.

**Note** The MVME1X2P4’s new “Petra” ASIC combines the functions previously covered by the MC2 chip, the IP2 chip, and the MCECC chip in a single ASIC. For ease of use in conjunction with programming models and documentation developed for earlier boards, however, the structure of this manual preserves the functional distinctions that formerly characterized those three ASICs.

Chapter 2 describes the VMEchip2 ASIC. Chapters 3, 4, and 5 respectively cover MC2 chip functions, IP2 chip functions, and MCECC chip functions, all three of which are now implemented by the Petra ASIC. Appendix A describes the use of interrupts. Readers more interested in programmable register bit definitions than in hardware functionality may wish to focus on Chapters 2, 3, 4, and 5. The related background information found in Chapter 1 (this chapter) may be of use in some cases.

## Overview

The MVME162P4 is based on the MC68040 or MC68LC040 microprocessor. The MVME172P4 is based on the MC68060 or MC68LC060 microprocessor. The board artwork is otherwise the same for both MVME1X2P4 models.

Both models are available in various versions with the features listed in [Table 1-1 on page 1-6](#). No-IP, no-SCSI, and no-LAN options are available as well.

### The Petra ASIC and Second-Generation MVME1X2 Boards

*Due to rapid changes in technology, the production of certain ASICs used on first- and second-generation MVME162/172 series boards has ended. The discontinued ASICs are the MC, MC2, IPIC, IP2, and MCECC chips. The Petra chip was developed to replace these discontinued ASICs.*

*The Petra ASIC is functionally compatible with each of the components that it replaces. In cases where functionality between ASICs is exclusive, configuration switches or jumpers are provided to let you select the desired functionality. For instance, the MC and MC2 chips differed in their implementation of Flash write protection. Boards built with the Petra ASIC, accordingly, have a configuration switch so that you can determine which method to use.*

*In several areas of functionality, the configuration switches provide backward compatibility with earlier MVME162/172 implementations, but their settings can be overridden in software if desired. A “R/W” by the corresponding register table entry denotes instances where this capability is present.*

*Where the older technology supported “fast page” or “EDO” DRAM chips, the Petra memory controllers support SDRAM devices. The two memory controllers modeled in Petra duplicate the functionality of the “parity” memory controller found in the MC ASICs as well as that of the “single-bit error correcting/double-bit error detecting” memory controller found in the MCECC ASICs (see [Figure 1-1](#)).*

*In line with this flexibility in memory control, where previous MVME1X2LX-series (2IP) Programmer's Reference Guides documented only the "parity" memory controller model, this guide describes the "parity" model (in Chapter 3) as well as the MCECC model (in Chapter 5). You can enable the ECC function if you so desire.*

*There is logic to prevent you from using both SDRAM memory controllers at the same time. Regardless of which memory controller is enabled, the same SDRAM memory array is targeted. The SDRAM array is 32 data bits wide with 7 checkbits. The array architecture is a non-interleaved single bank for sizes below 32MB. For array sizes above 32MB, additional physical memory banks are added but the architecture remains non-interleaved. Since the 7 checkbits enable error detection and correction, the MC "parity memory controller" actually interfaces to an error-correcting memory array. In this case, a parity error is defined to be a multibit error that cannot be corrected. Memory scrubbing is not available with the "parity memory controller".*

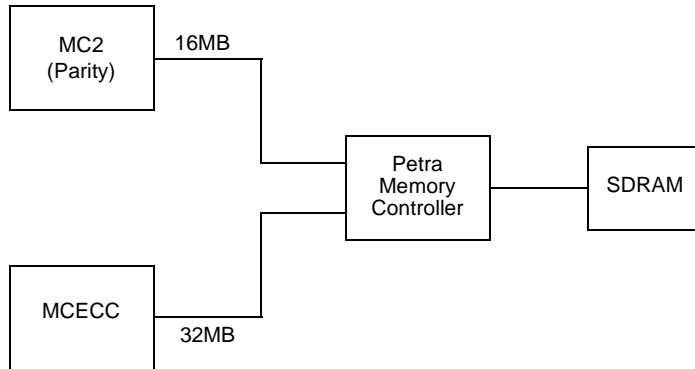
*A final note on the SDRAM implementation: The bandwidth between the SDRAM and local bus is greater than it was with the earlier DRAM array. As a result, software takes less time to execute. Applications that incorporate elapsed-time functions which are dependent on code execution may have problems.*

*For readers who need to know the ASIC-specific differences between the MVME1X2P2 programming models in detail, certain areas of the text in this manual are printed in italics and marked with change bars (as is done here). Readers should compare those sections to the corresponding sections of the first-generation manuals.*

## I/O Routing

Two DB25 front panel I/O connectors furnish the I/O connection for the serial ports. The balance of the I/O is routed to the VMEbus P2 connector. A P2 transition board and cables link the main board to a user-selected transition board.

Previous memory model (emulates parity memory) - may be desirable for compatibility with existing implementations.



New memory model (has error correction circuitry) - may be desirable for the improved performance, or in applications where memory requirements exceed 16MB.

2610 9908

### Figure 1-1. MVME1X2P4 Memory Control Options

The MVME1X2P4 supports the MVME712M and MVME712B transition boards. It is also compatible with older, discontinued boards of the MVME712 series: MVME712-12, MVME712-13, MVME712A, or MVME712AM (all referred to in this manual collectively as MVME712x, unless separately specified). These transition boards provide configuration headers, serial port drivers, and industry-standard connectors for the I/O devices. Though originally designed to support MVME167 boards, the MVME712 series transition boards are compatible with the MVME1X2P4 as well. (Special precautions apply in a few cases. Refer to the section on the Serial Communications Interface in the *MVME1X2P4 Installation and Use* manual for more information.)

## VMEbus Interface

The VMEbus interface is implemented with an ASIC called the VMEchip2. The VMEchip2 includes:

- ❑ Two tick timers
- ❑ A watchdog timer
- ❑ Programmable map decoders for the master and slave interfaces
- ❑ A VMEbus to/from local bus DMA controller
- ❑ A VMEbus to/from local bus non-DMA programmed access interface
- ❑ A VMEbus interrupter, a VMEbus system controller, a VMEbus interrupt handler, and a VMEbus requester

Processor-to-VMEbus transfers can be D8, D16, or D32. VMEchip2 DMA transfers to the VMEbus, however, can be D16, D32, D16/BLT, D32/BLT, or D64/MBLT.

## Petra ASIC

The “Petra” ASIC combines the functions of the MC2 chip, the IP2 chip, and the MCECC chip in a single device.

Former MC2 chip functions now implemented by the Petra ASIC include four tick timers; the interfaces to the LAN chip, SCSI chip, serial port chip, and BBRAM; the programmable interface for the DRAM and/or SRAM; and Flash write enable.

Former IndustryPack Interface Controller (IP2 chip) functions now implemented by the Petra ASIC include control and status information, including DMA control, for up to two single-size IndustryPacks (IPs) or one double-size IP that can be plugged into the MVME1X2P4 main module.

The Petra ASIC also performs the memory control functions previously carried out by the MCECC chip Memory Controller ASIC: It supplies the programmable interface for the ECC-protected 16/32MB DRAM emulation.

The following table summarizes the features of the MVME162P4 and MVME172P4 Embedded Controllers.

**Table 1-1. MVME1X2P4 Features Summary**

Feature	MVME162P4	MVME172P4
Processor	25/32MHz 32-bit MC68040 or MC68LC040 microprocessor	60MHz 32-bit MC68060 microprocessor, or 64MHz 32-bit MC68LC060 microprocessor
DRAM	16/32MB synchronous DRAM (SDRAM). Configurable to emulate 1/4/8/16MB parity-protected DRAM or 4/8/16/32MB ECC-protected DRAM <i>MVME1X2P4 boards use SDRAM (Synchronous DRAM) in place of DRAM.</i>	
SRAM	512KB SRAM with battery backup	
EPROM	One JEDEC standard 32-pin PLCC EPROM socket (EPROMs may be shipped separately)	
Flash	One Intel 28F008SA 1Mx8 or 2Mx8 Flash memory device with optional write protection	One Intel 28F160S5 1Mx8 or 2Mx8 Flash memory device with optional write protection
NVRAM and RTC	8K by 8 Non-Volatile RAM (NVRAM) and Real-Time Clock (RTC) with battery backup and watchdog function (SGS-Thomson M48T58)	
Timers	Four 32-bit tick timers and watchdog timer in Petra ASIC	
	Two 32-bit tick timers and watchdog timer in VMEchip2 ASIC	
Software Interrupts	Eight software interrupts (in MVME1X2P4 versions equipped with the VMEchip2 ASIC)	
I/O	Two EIA-232-D, EIA-530, EIA-422, or EIA-425 configurable serial ports via front panel and transition module	
	Optional SCSI interface with DMA via P2 or LCP2 adapter board	
	Optional Ethernet transceiver interface via DB15 connector on transition module	
	Four MVIP IndustryPack interface channels with DMA	



**Table 1-1. MVME1X2P4 Features Summary (Continued)**

Feature	MVME162P4	MVME172P4
VMEbus interface	VMEbus system controller functions	
	VMEbus-to-local-bus interface (A32/A24, D32/D16/D8)	
	Local-bus-to-VMEbus interface (A16/A24/A32, D8/D16/D32)	
	Programmable interrupter and interrupt handler	
	Global Control/Status register for interprocessor communications	
	DMA capability for fast local-memory/VMEbus transfers (A16/A24/A32, D16/D32 (D16/D32/D64 BLT))	
Switches	Two pushbutton switches ( <b>ABORT</b> and <b>RESET</b> )	
Status Indicators	Eight LEDs: Board Fail ( <b>FAIL</b> ), CPU Status ( <b>STAT</b> ), CPU Activity ( <b>RUN</b> ), System Controller ( <b>SCON</b> ), LAN Activity ( <b>LAN</b> ), Fuse Status ( <b>FUSE</b> ), SCSI Activity ( <b>SCSI</b> ), VME Activity ( <b>VME</b> )	

## Applicable Industry Standards

These boards conform to the requirements of the following documents:

- ❑ VMEbus Specification (IEEE 1014-87)
- ❑ EIA-232-D Serial Interface Specification, EIA
- ❑ SCSI Specification, ANSI
- ❑ IndustryPack Specification, GreenSpring

## Block Diagram

Figure 1-2 is a general block diagram of the MVME1X2P4.

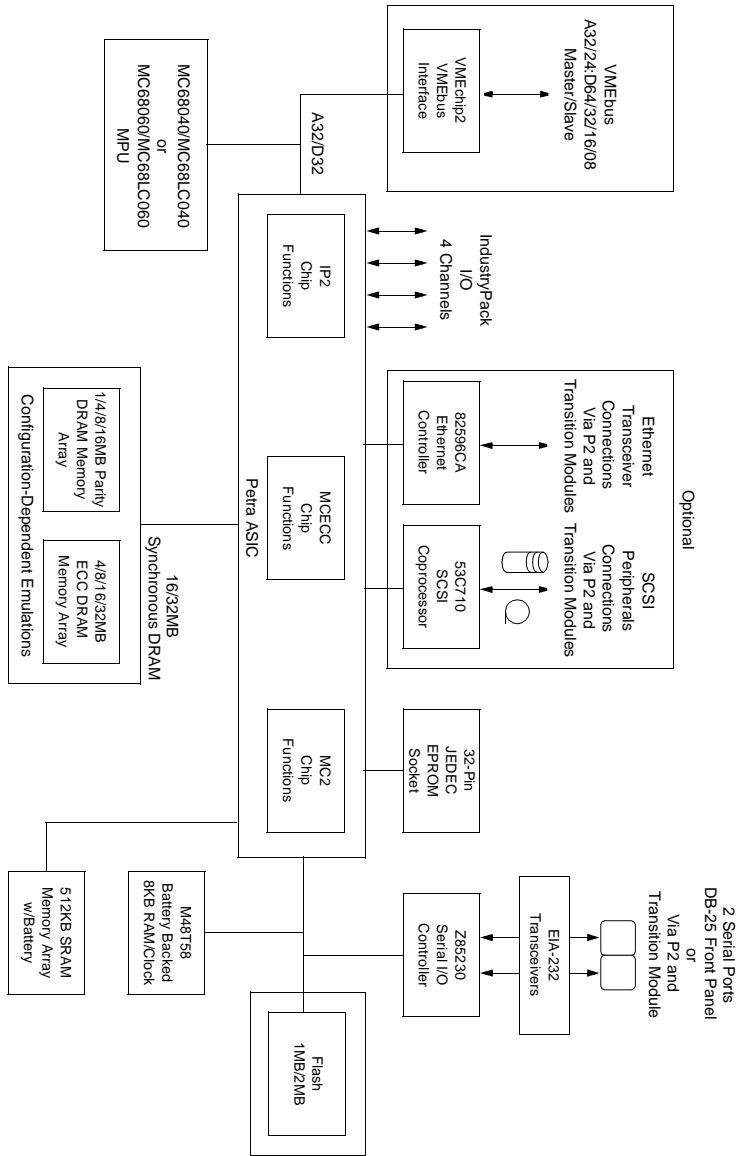


Figure 1-2. MVME1X2P4 Block Diagram

---

## Functional Description

This section highlights a few specific features of the MVME1X2P4 Embedded Controller. For a complete functional description of the major blocks of the MVME1X2P4, refer to the *Installation and Use* manual.

### VMEbus Interface and VMEchip2

The local-bus-to-VMEbus interface and the VMEbus-to-local-bus interface are provided by the VMEchip2 ASIC. The VMEchip2 can also provide the VMEbus system controller functions. Refer to the VMEchip2 description in Chapter 2 for detailed programming information.

In support of possible future configurations in which the MVME1X2P4 might be offered as an embedded controller without the VMEbus interface, certain logic in the VMEchip2 has been duplicated in the Petra chip. [Table 1-2](#) shows the location of the overlapping logic. As long as the VMEchip2 ASIC is present, the redundant logic is inhibited in the Petra chip.

Note that the **ABORT** switch logic in the VMEchip2 is not used. Likewise unused are the GPI inputs to the VMEchip2, which are located at \$FFF40088 bits 7-0. Instead, the **ABORT** switch interrupt is integrated into the Petra ASIC at location \$FFF42043. The GPI inputs are integrated into the Petra ASIC at location \$FFF4202C bits 23-16.

**Table 1-2. Functions Duplicated in VMEchip2 and Petra ASICs**

VMEchip2		Petra Chip		Notes
Address	Bit #	Address	Bit #	
\$FFF40060	28-24	\$FFF42044	28-24	1,5
\$FFF40060	22-19, 17,16	\$FFF42044	22-19, 17,16	2,5
\$FFF4004C	13-8	\$FFF42044	13-8	3,5
\$FFF40048	7	\$FFF42048	8	4
\$FFF40048	9	\$FFF42048	9	4,5
\$FFF40048	10	\$FFF42048	10	4,5
\$FFF40048	11	\$FFF42048	11	4,5
\$FFF40064	31-0	\$FFF4204C	3-0	8
		\$FFF42040	6- 0	6
\$FF800000-\$FFBFFFFFF	31-0	\$FF800000-\$FFBFFFFFF	31-0	7
\$FFE00000-\$FFEFFFFFF	31-0	Programmable	31-0	7

**Notes**

1. **RESET** switch control.
2. Watchdog timer control.
3. Access and watchdog timer parameters.
4. MPU TEA (bus error) status
5. Bit numbering for the VMEchip2 and Petra ASICs has a one-to-one correspondence.
6. **ABORT** switch interrupt control. Implemented also in the VMEchip2, but with a different bit organization (refer to the VMEchip2 description in Chapter 2). In the MVME1X2P4, the **ABORT** switch is wired to the Petra chip, not the VMEchip2.
7. The SRAM and EPROM decoder in the VMEchip2 (version 2) must be disabled by software before any accesses are made to these address spaces.

8. 32-bit prescaler. The prescaler can also be accessed at \$FFF40064 when the optional VMEbus is not enabled.

## Memory Maps

There are two points of view for memory maps:

1. The mapping of all resources as viewed by local bus masters (local bus memory map)
2. The mapping of onboard resources as viewed by VMEbus masters (VMEbus memory map)

The memory maps and I/O maps described in the following tables are correct for all local bus masters. Some address translation capability exists in the VMEchip2. This capability makes it possible to have multiple MVME1X2P4 modules on the same VMEbus with different virtual local bus maps as viewed by different VMEbus masters.

### Local Bus Memory Map

The local bus memory map is split into different address spaces by the transfer type (TT) signals. The local resources respond to the normal access and interrupt acknowledge codes.

### Normal Address Range

The following tables show the memory maps of devices that respond to the normal address range. The normal address range is defined by the Transfer Type (TT) signals on the local bus. On the MVME1X2P4, Transfer Types 0, 1, and 2 define the normal address range.

**Table 1-3** is the entire map from \$00000000 to \$FFFFFFFF. Many areas of the map are user-programmable, and suggested uses are shown in the table. The cache inhibit function is programmable in the MC68xx0x0 MMU. The onboard I/O space must be marked cache inhibit and serialized in its page table. **Table 1-4 on page 1-14** further defines the map for the local I/O devices on the MVME1X2P4.

**Table 1-3. MVME1X2P4 Local Bus Memory Map**

Address Range	Devices Accessed	Port Width	Size	Software Cache Inhibit	Note(s)
Programmable	DRAM on board	D32	4MB-16MB	N	2
Programmable	SRAM	D32	128KB-2MB	N	2
Programmable	VMEbus A32/A24	D32/D16	--	?	4
Programmable	IP_a Memory	D32-D8	64KB-8MB	?	2, 4
Programmable	IP_b Memory	D32-D8	64KB-8MB	?	2, 4
Programmable	IP_c Memory	D32-D8	64KB-8MB	?	2, 4
Programmable	IP_d Memory	D32-D8	64KB-8MB	?	2, 4
\$FF800000-\$FF9FFFFFFF	Flash/ EPROM	D32	2MB	N	1, 5
\$FFA00000-\$FFBFFFFFFF	EPROM/Flash	D32	2MB	N	6
\$FFC00000-\$FFCFFFFFFF	Not decoded	--	1MB	N	7
\$FFD00000-\$FFDFFFFFFF	Not decoded	--	1MB	N	7
\$FFE00000-\$FE7FFFFF	SRAM default	D32	512KB	N	--
\$FFE80000-\$FFEFFFFFFF	Not decoded	--	512KB	N	7
\$FFF00000-\$FFFFFFFFFF	Local I/O	D32-D8	878KB	Y	3
\$FFFF0000-\$FFFFFFFFFF	VMEbus A16	D32/D16	64KB	?	2, 4

## Notes

1. Reset enables the decoder for this space of the memory map so that it will decode address spaces \$FF800000-\$FF9FFFFFF and \$00000000-\$003FFFFFF. The decode at 0 must be disabled in the MC2 chip before DRAM is enabled. DRAM is enabled with the DRAM Control register at address \$FFF42048, bit 24. EPROM/Flash is disabled at the low address space with the EPROM Control register at address \$FFF42040, bit 20.
2. This area is user-programmable. The DRAM and SRAM decoder is programmed in the Petra chip, the local-to-VMEbus decoders are programmed in the VMEchip2, and the IP memory space is also programmed in the Petra chip.
3. Size is approximate.
4. Cache inhibit depends on devices in area mapped.
5. The EPROM and Flash are sized by the Petra chip from an 8-bit private bus to the 32-bit MPU local bus. Because the device size is less than the allocated memory map size for some entries, the device contents repeat for those entries.  
*If bit MC2\_GPI3 is set (bit 20 in the MC2 General-Purpose Inputs register, or pin 5 on switch S4), the Flash device is accessed. If bit MC2\_GPI3 is not set, the EPROM is accessed.*
6. The Flash and EPROM are sized by the Petra chip from an 8-bit private bus to the 32-bit MPU local bus. Because the device size is less than the allocated memory map size for some entries, the device contents repeat for those entries.  
*If bit MC2\_GPI3 is set (bit 20 in the MC2 General-Purpose Inputs register, or pin 5 on switch S4), the EPROM is accessed. If bit MC2\_GPI3 is not set, the Flash device is accessed.*
7. These areas are not decoded unless one of the programmable decoders is initialized to decode this space. If they are not decoded, an access to this address range will generate a local bus time-out. The local bus timer must be enabled.

Table 1-4 below describes the "Local I/O Devices" portion of the local bus main memory map for the MVME1X2P4.

**Table 1-4. MVME1X2P4 Local I/O Devices Memory Map**

Address Range	Device	Port Width	Size	Note(s)
\$FFF00000 - \$FFF3FFFF	Reserved	--	256KB	4
\$FFF40000 - \$FFF400FF	VMEchip2 (LCSR)	D32	256B	1, 3
\$FFF40100 - \$FFF401FF	VMEchip2 (GCSR)	D32-D8	256B	1, 3
\$FFF40200 - \$FFF40FFF	Reserved	--	3.5KB	4, 5
\$FFF41000 - \$FFF41FFF	Reserved	--	4KB	4
\$FFF42000 - \$FFF42FFF	MC2 chip	D32-D8	4KB	1
\$FFF43000 - \$FFF430FF	MCECC #1	D8	256B	1
\$FFF43100 - \$FFF431FF	MCECC #2	D8	256B	1
\$FFF43200 - \$FFF43FFF	MCECCs (repeated)		3.5KB	1, 5
\$FFF44000 - \$FFF44FFF	Reserved		8KB	4
\$FFF45000 - \$FFF45FFF	SCC (Z85230)	D8	4KB	1, 2
\$FFF46000 - \$FFF46FFF	LAN (82596CA)	D32	4KB	1, 6
\$FFF47000 - \$FFF47FFF	SCSI (53C710)	D32-D8	4KB	1
\$FFF48000 - \$FFF57FFF	Reserved	--	64KB	4
\$FFF58000 - \$FFF5807F	IP2 chip IP_a I/O	D16	128B	1
\$FFF58080 - \$FFF580FF	IP2 chip IP_a ID	D16	128B	1
\$FFF58100 - \$FFF5817F	IP2 chip IP_b I/O	D16	128B	1
\$FFF58180 - \$FFF581FF	IP2 chip IP_b ID Read	D16	128B	1
\$FFF58200 - \$FFF5827F	IP2 chip IP_c I/O	D16	128B	1
\$FFF58280 - \$FFF582FF	IP2 chip IP_c ID	D16	128B	1
\$FFF58300 - \$FFF5837F	IP2 chip IP_d I/O	D16	128B	1
\$FFF58380 - \$FFF583FF	IP2 chip IP_d ID Read	D16	128B	1
\$FFF58400 - \$FFF584FF	IP2 chip IP_ab I/O	D32-D16	256B	1
\$FFF58500 - \$FFF585FF	IP2 chip IP_cd I/O	D32-D16	256B	1
\$FFF58600 - \$FFF586FF	IP2 chip IP_ab I/O repeated	D32-D16	256B	1
\$FFF58700 - \$FFF587FF	IP2 chip IP_cd I/O repeated	D32-D16	256B	1



**Table 1-4. MVME1X2P4 Local I/O Devices Memory Map (Continued)**

Address Range	Device	Port Width	Size	Note(s)
\$FFF58800 - \$FFF5887F	Reserved	--	128B	1
\$FFF58880 - \$FFF588FF	Reserved	--	128B	1
\$FFF58900 - \$FFF5897F	Reserved	--	128B	1
\$FFF58980 - \$FFF589FF	Reserved	--	128B	1
\$FFF58A00 - \$FFF58A7F	Reserved	--	128B	1
\$FFF58A80 - \$FFF58AFF	Reserved	--	128B	1
\$FFF58B00 - \$FFF58B7F	Reserved	--	128B	1
\$FFF58B80 - \$FFF58BFF	Reserved	--	128B	1
\$FFF58C00 - \$FFF58CFF	Reserved	--	256B	1
\$FFF58D00 - \$FFF58DFF	Reserved	--	256B	1
\$FFF58E00 - \$FFF58EFF	Reserved	--	256B	1
\$FFF58F00 - \$FFF58FFF	Reserved	--	256B	1
\$FFFBC000 - \$FFFBC01F	IP2 chip registers	D32-D8	2KB	1
\$FFFBC800 - \$FFFBC81F	Reserved	--	2KB	1
\$FFFBD000 - \$FFFBFFFF	Reserved	--	12KB	4
\$FFFC0000 - \$FFFC7FFF	M48T58 (BBRAM, TOD clock)	D32-D8	32KB	1
\$FFFC8000 - \$FFFCBFFF	M48T58 (Clear FWEN if MC1)	D32-D8	16KB	1, 7
\$FFFC0000 - \$FFFCFFFF	M48T58 (Set FWEN if MC1)	D32-D8	16KB	1, 7
\$FFFD0000 - \$FFFEFFFF	Reserved	--	128KB	4

**Notes**

1. For a complete description of the register bits, refer to the data sheet for the specific chip. For a more detailed memory map, refer to the following detailed peripheral device memory maps.
2. The SCC is an 8-bit device located on a Petra chip private data bus. Byte access is required.
3. Writes to the LCSR in the VMEchip2 ASIC must be 32 bits. LCSR writes of 8 or 16 bits terminate with a TEA signal. Writes to the GCSR may be 8, 16, or 32 bits. Reads to the LCSR and GCSR may be 8, 16, or 32 bits. Byte reads should be used to read the interrupt vector.
4. This area returns no acknowledge signal. If the local bus timer is enabled, the access times out and is terminated by a TEA signal.
5. Size is approximate.
6. Port commands to the 82596CA must be written as two 16-bit writes: upper word first and lower word second.
7. *If the MCI Flash write-enable mode is set, a memory map reference to these areas has the added functionality of “enable/disable Flash write protect”. A board configuration switch (S5, pin 4) sets the MCI mode.*

## Detailed I/O Memory Maps

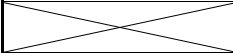
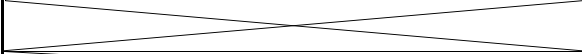
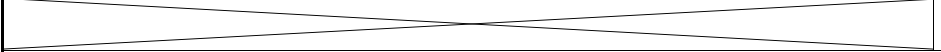

Tables 1-5 through 1-14 give the detailed memory maps for:

VMEchip2	<a href="#">Table 1-5</a>
MC2 functions (Petra chip)	<a href="#">Table 1-6</a>
IP2 functions (Petra chip)	<a href="#">Table 1-7</a>
IP2 Control and Status registers (Petra chip)	<a href="#">Table 1-8</a>
Z85230 SCC register addresses	<a href="#">Table 1-9</a>
82596CA Ethernet LAN chip	<a href="#">Table 1-10</a>
53C710 SCSI chip	<a href="#">Table 1-11</a>
M48T58 BBRAM/TOD clock	<a href="#">Table 1-12</a>
BBRAM configuration area	<a href="#">Table 1-13</a>
TOD clock	<a href="#">Table 1-14</a>

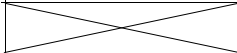
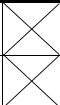

You can obtain manufacturers' errata sheets for the various chips listed above by contacting your local Motorola sales representative. A non-disclosure agreement may be necessary.

**Table 1-5. VMEchip2 Memory Map (Sheet 1 of 3)**

VMEchip2 LCSR Base Address = \$FFF40000  
 OFFSET:

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
0	SLAVE ENDING ADDRESS 1																
4	SLAVE ENDING ADDRESS 2																
8	SLAVE ADDRESS TRANSLATION ADDRESS 1																
C	SLAVE ADDRESS TRANSLATION ADDRESS 2																
10					ADDER 2	SNP 2	WP 2	SUP 2	USR 2	A32 2	A24 2	BLK D64 2	BLK 2	PRGM 2	DATA 2		
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
14	MASTER ENDING ADDRESS 1																
18	MASTER ENDING ADDRESS 2																
1C	MASTER ENDING ADDRESS 3																
20	MASTER ENDING ADDRESS 4																
24	MASTER ADDRESS TRANSLATION ADDRESS 4																
28	MAST D16 EN	MAST WP EN	MASTER AM 4						MAST D16 EN	MAST WP EN	MASTER AM 3						
2C	GCSR GROUP SELECT								GCSR BOARD SELECT				MAST 4 EN	MAST 3 EN	MAST 2 EN	MAST 1 EN	
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
30											WAIT RMW	ROM ZERO	DMA TB SNP MODE	SRAM SPEED			
34																	
38	DMA CONTROLLER																
3C	DMA CONTROLLER																
40	DMA CONTROLLER																
44	DMA CONTROLLER																
48		TICK 2/1	TICK IRQ 1 EN	CLR IRQ	IRQ STAT	VMEBUS INTERRUPT LEVEL			VMEBUS INTERRUPT VECTOR								

This sheet continues on facing page. →

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
SLAVE STARTING ADDRESS 1																
SLAVE STARTING ADDRESS 2																
SLAVE ADDRESS TRANSLATION SELECT 1																
SLAVE ADDRESS TRANSLATION SELECT 2																
				ADDER 1	SNP 1	WP 1	SUP 1	USR 1	A32 1	A24 1	BLK D64 1	BLK 1	PRGM 1	DATA 1		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
MASTER STARTING ADDRESS 1																
MASTER STARTING ADDRESS 2																
MASTER STARTING ADDRESS 3																
MASTER STARTING ADDRESS 4																
MASTER ADDRESS TRANSLATION SELECT 4																
MAST D16 EN	MAST WP EN	MASTER AM 2						MAST D16 EN	MAST WP EN	MASTER AM 1						
IO2 EN	IO2 WP EN	IO2 S/U	IO2 P/D	IO1 EN	IO1 D16 EN	IO1 WP EN	IO1 S/U	ROM SIZE	ROM BANK B SPEED			ROM BANK A SPEED				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
ARB ROBN	MAST DHB	MAST DWB		MST FAIR	MST RWD	MASTER VMEBUS		DMA HALT	DMA EN	DMA TBL	DMA FAIR	DM RELM		DMA VMEBUS		
DMA TBL INT	DMA LB SNP MODE			DMA INC VME	DMA INC LB	DMA WRT	DMA D16	DMA D64 BLK	DMA BLK	DMA AM 5	DMA AM 4	DMA AM 3	DMA AM 2	DMA AM 1	DMA AM 0	
LOCAL BUS ADDRESS COUNTER																
VMEBUS ADDRESS COUNTER																
BYTE COUNTER																
TABLE ADDRESS COUNTER																
DMA TABLE INTERRUPT COUNT				MPU CLR STAT	MPU LBE ERR	MPU LPE ERR	MPU LOB ERR	MPU LTO ERR	DMA LBE ERR	DMA LPE ERR	DMA LOB ERR	DMA LTO ERR	DMA TBL ERR	DMA VME ERR	DMA DONE	

1360 9403

← This sheet begins on facing page.

**Table 1-5. VMEchip2 Memory Map (Sheet 2 of 3)**

**VMEchip2 LCSR Base Address = \$FFF40000  
OFFSET:**

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
4C	X							ARB BGTO EN	DMA TIME OFF			DMA TIME ON			VME GLOBAL TIMER	
50	TICK TIMER 1															
54	TICK TIMER 1															
58	TICK TIMER 2															
5C	TICK TIMER 2															
60	X	SCON	SYS FAIL	BRD FAIL STAT	PURS STAT	CLR PURS STAT	BRD FAIL OUT	RST SW EN	SYS RST	WD CLR TO	WD CLR CNT	WD TO STAT	TO BF EN	WD SRST LRST	WD RST EN	WD EN
64	PRE															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
68	AC FAIL IRQ	AB IRQ	SYS FAIL IRQ	MWP BERR IRQ	PE IRQ	IRQ1E IRQ	TIC2 IRQ	TIC1 IRQ	VME IACK IRQ	DMA IRQ	SIG3 IRQ	SIG2 IRQ	SIG1 IRQ	SIG0 IRQ	LM1 IRQ	LM0 IRQ
6C	EN IRQ 31	EN IRQ 30	EN IRQ 29	EN IRQ 28	EN IRQ 27	EN IRQ 26	EN IRQ 25	EN IRQ 24	EN IRQ 23	EN IRQ 22	EN IRQ 21	EN IRQ 20	EN IRQ 19	EN IRQ 18	EN IRQ 17	EN IRQ 16
70	X															
74	CLR IRQ 31	CLR IRQ 30	CLR IRQ 29	CLR IRQ 28	CLR IRQ 27	CLR IRQ 26	CLR IRQ 25	CLR IRQ 24	CLR IRQ 23	CLR IRQ 22	CLR IRQ 21	CLR IRQ 20	CLR IRQ 19	CLR IRQ 18	CLR IRQ 17	CLR IRQ 16
78	X	AC FAIL IRQ LEVEL			X	ABORT IRQ LEVEL			X	SYS FAIL IRQ LEVEL			X	MST WP ERROR IRQ LEVEL		
7C	X	VME IACK IRQ LEVEL			X	DMA IRQ LEVEL			X	SIG 3 IRQ LEVEL			X	SIG 2 IRQ LEVEL		
80	X	SW7 IRQ LEVEL			X	SW6 IRQ LEVEL			X	SW5 IRQ LEVEL			X	SW4 IRQ LEVEL		
84	X	SPARE IRQ LEVEL			X	VME IRQ 7 IRQ LEVEL			X	VME IRQ 6 IRQ LEVEL			X	VME IRQ 5 IRQ LEVEL		
88	VECTOR BASE REGISTER 0				VECTOR BASE REGISTER 1				MST IRQ EN	SYS FAIL LEVEL	AC FAIL LEVEL	ABORT LEVEL	GPIOEN			
8C	X															

This sheet continues on facing page. →

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
VME ACCESS TIMER		LOCAL BUS TIMER		WD TIME OUT SELECT				PRESCALER CLOCK ADJUST									
COMPARE REGISTER																	
COUNTER																	
COMPARE REGISTER																	
COUNTER																	
OVERFLOW COUNTER 2				X		CLR OVF 2	COC EN 2	TIC EN 2	OVERFLOW COUNTER 1				X		CLR OVF 1	COC EN 1	TIC EN 1
SCALER																	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
SW7 IRQ	SW6 IRQ	SW5 IRQ	SW4 IRQ	SW3 IRQ	SW2 IRQ	SW1 IRQ	SW0 IRQ	SPARE	VME IRQ7	VME IRQ6	VME IRQ5	VME IRQ4	VME IRQ3	VME IRQ2	VME IRQ1		
EN IRQ 15	EN IRQ 14	EN IRQ 13	EN IRQ 12	EN IRQ 11	EN IRQ 10	EN IRQ 9	EN IRQ 8	EN IRQ 7	EN IRQ 6	EN IRQ 5	EN IRQ 4	EN IRQ 3	EN IRQ 2	EN IRQ 1	EN IRQ 0		
SET IRQ 15	SET IRQ 14	SET IRQ 13	SET IRQ 12	SET IRQ 11	SET IRQ 10	SET IRQ 9	SET IRQ 8	X									
CLR IRQ 15	CLR IRQ 14	CLR IRQ 13	CLR IRQ 12	CLR IRQ 11	CLR IRQ 10	CLR IRQ 9	CLR IRQ 8										
X		P ERROR IRQ LEVEL		X		IRQ1E IRQ LEVEL		X		TIC TIMER 2 IRQ LEVEL		X		TIC TIMER 1 IRQ LEVEL			
X		SIG 1 IRQ LEVEL		X		SIG 0 IRQ LEVEL		X		LM 1 IRQ LEVEL		X		LM 0 IRQ LEVEL			
X		SW3 IRQ LEVEL		X		SW2 IRQ LEVEL		X		SW1 IRQ LEVEL		X		SW0 IRQ LEVEL			
X		VME IRQ 4 IRQ LEVEL		X		VMEB IRQ 3 IRQ LEVEL		X		VME IRQ 2 IRQ LEVEL		X		VME IRQ 1 IRQ LEVEL			
GPIOO				GPIOI				GPI									
MP IRQ EN		REV EROM		DIS SRAM		DIS MST		NO EL BBSY		DIS BSYT		EN INT		DIS BGN			

1361 9403

← This sheet begins on facing page.

**Table 1-5. VMEchip2 Memory Map (Sheet 3 of 3)**

VMEchip2 GCSR Base Address = \$FFF40100

Offsets		Bit Numbers															
VME -bus	Local Bus	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	Chip Revision								Chip ID							
2	4	L M 3	L M 2	L M 1	L M 0	S I G 3	S I G 2	S I G 1	S I G 0	R S T	I S F	BF	S C O N	SYS FL	X	X	X
4	8	General Purpose Control and Status register 0															
6	C	General Purpose Control and Status register 1															
8	10	General Purpose Control and Status register 2															
A	14	General Purpose Control and Status register 3															
C	18	General Purpose Control and Status register 4															
E	1C	General Purpose Control and Status register 5															



**Table 1-6. Petra MC2 Functions: Register Map**

MC2 Sector Base Address = \$FFF42000

Offset	D31-D24	D23-D16	D15-D8	D7-D0
\$00	MC2 chip ID	MC2 chip Revision	General Control	Interrupt Vector Base Register
\$04	Tick Timer 1 Compare Register			
\$08	Tick Timer 1 Counter Register			
\$0C	Tick Timer 2 Compare Register			
\$10	Tick Timer 2 Counter Register			
\$14	LSB Prescaler Count Register	Prescaler Clock Adjust	Tick Timer 2 Control	Tick Timer 1 Control
\$18	Tick Timer 4 Interrupt Control	Tick Timer 3 Interrupt Control	Tick Timer 2 Interrupt Control	Tick Timer 1 Interrupt Control
\$1C	DRAM Parity Error Interrupt Control	SCC Interrupt Control	Tick Timer 4 Control	Tick Timer 3 Control
\$20	DRAM Space Base Address Register		SRAM Space Base Address Register	
\$24	DRAM Space Size	DRAM/SRAM Options	SRAM Space Size	Reserved
\$28	LANC Error Status	Reserved	LANC Interrupt Control	LANC Bus Error Interrupt Control
\$2C	SCSI Error Status	General Purpose Inputs	<i>Board</i> Version	SCSI Interrupt Control
\$30	Tick Timer 3 Compare Register			
\$34	Tick Timer 3 Counter Register			
\$38	Tick Timer 4 Compare Register			
\$3C	Tick Timer 4 Counter Register			
\$40	Bus Clock	EPROM Access Time Control	Flash Parameter Control	<b>ABORT</b> Switch Interrupt Control
\$44	<b>RESET</b> Switch Control	Watchdog Timer Control	Access/Watchdog Time Base Select	Reserved
\$48	DRAM Control	Reserved	MPU Status	Reserved
\$4C	32-bit Prescaler Count Register			

The following memory map table includes all devices selected by the Petra IP2 sector map decoder.

**Table 1-7. Petra IP2 Functions: Overall Memory Map**

Address Range	Selected Device	Port Width	Size
Programmable	IP_a/IP_ab Memory Space	D32-D8	64KB-16MB
Programmable	IP_b Memory Space	D16-D8	64KB-8MB
Programmable	IP_c/IP_cd Memory Space	D32-D8	64KB-16MB
Programmable	IP_d Memory Space	D16-D8	64KB-8MB
FFFF58000-FFFF5807F	IP_a I/O Space	D16	128B
FFFF58080-FFFF580BF	IP_a ID Space	D16	64B
FFFF580C0-FFFF580FF	IP_a ID Space Repeated	D16	64B
FFFF58100-FFFF5817F	IP_b I/O Space	D16	128B
FFFF58180-FFFF581BF	IP_b ID Space	D16	64B
FFFF581C0-FFFF581FF	IP_b ID Space Repeated	D16	64B
FFFF58200-FFFF5827F	IP_c I/O Space	D16	128B
FFFF58280-FFFF582BF	IP_c ID Space	D16	64B
FFFF582C0-FFFF582FF	IP_c ID Space Repeated	D16	64B
FFFF58300-FFFF5837F	IP_d I/O Space	D16	128B
FFFF58380-FFFF583BF	IP_d ID Space	D16	64B
FFFF583C0-FFFF583FF	IP_d ID Space Repeated	D16	64B
FFFF58400-FFFF584FF	IP_ab I/O Space	D32-D16	256B
FFFF58500-FFFF585FF	IP_cd I/O Space	D32-D16	256B
FFFF58600-FFFF586FF	IP_ab I/O Space Repeated	D32-D16	256B
FFFF58700-FFFF587FF	IP_cd I/O Space Repeated	D32-D16	256B
FFFFBC00-FFFFBC083	Control/Status Registers	D32-D8	32B

A summary of the IP2 sector CSR registers appears in [Table 1-8](#). The CSR registers can be accessed as bytes, words, or longwords. They should not be accessed as lines. They are shown in the table as bytes.

**Table 1-8. Petra IP2 Memory Map — Control/Status Registers**

IP2 Sector Base Address = \$FFFBC000

Register Offset	Register Name	Register Bit Names							
		D7	D6	D5	D4	D3	D2	D1	D0
\$00	CHIP ID	0	0	1	0	0	0	1	1
\$01	CHIP REVISION	0	0	0	0	0	0	1	1
\$02	RESERVED	0	0	0	0	0	0	0	0
\$03	VECTOR BASE	IV7	IV6	IV5	IV4	IV3	IV2	IV1	IV0
\$04	IP_a MEM BASE UPPER	a_BASE31	a_BASE30	a_BASE29	a_BASE28	a_BASE27	a_BASE26	a_BASE25	a_BASE24
\$05	IP_a MEM BASE LOWER	a_BASE23	a_BASE22	a_BASE21	a_BASE20	a_BASE19	a_BASE18	a_BASE17	a_BASE16
\$06	IP_b MEM BASE UPPER	b_BASE31	b_BASE30	b_BASE29	b_BASE28	b_BASE27	b_BASE26	b_BASE25	b_BASE24
\$07	IP_b MEM BASE LOWER	b_BASE23	b_BASE22	b_BASE21	b_BASE20	b_BASE19	b_BASE18	b_BASE17	b_BASE16
\$08	IP_c MEM BASE UPPER	c_BASE31	c_BASE30	c_BASE29	c_BASE28	c_BASE27	c_BASE26	c_BASE25	c_BASE24
\$09	IP_c MEM BASE LOWER	c_BASE23	c_BASE22	c_BASE21	c_BASE20	c_BASE19	c_BASE18	c_BASE17	c_BASE16
\$0A	IP_d MEM BASE UPPER	d_BASE31	d_BASE30	d_BASE29	d_BASE28	d_BASE27	d_BASE26	d_BASE25	d_BASE24
\$0B	IP_d MEM BASE LOWER	d_BASE23	d_BASE22	d_BASE21	d_BASE20	d_BASE19	d_BASE18	d_BASE17	d_BASE16
\$0C	IP_a MEM SIZE	a_SIZE23	a_SIZE22	a_SIZE21	a_SIZE20	a_SIZE19	a_SIZE18	a_SIZE17	a_SIZE16
\$0D	IP_b MEM SIZE	b_SIZE23	b_SIZE22	b_SIZE21	b_SIZE20	b_SIZE19	b_SIZE18	b_SIZE17	b_SIZE16
\$0E	IP_c MEM SIZE	c_SIZE23	c_SIZE22	c_SIZE21	c_SIZE20	c_SIZE19	c_SIZE18	c_SIZE17	c_SIZE16
\$0F	IP_d MEM SIZE	d_SIZE23	d_SIZE22	d_SIZE21	d_SIZE20	d_SIZE19	d_SIZE18	d_SIZE17	d_SIZE16
\$10	IP_a INT0 CONTROL	a0_PLTY	a0_E/L*	a0_INT	a0_IEN	a0_ICLR	a0_IL2	a0_IL1	a0_IL0
\$11	IP_a INT1 CONTROL	a1_PLTY	a1_E/L*	a1_INT	a1_IEN	a1_ICLR	a1_IL2	a1_IL1	a1_IL0
\$12	IP_b INT0 CONTROL	b0_PLTY	b0_E/L*	b0_INT	b0_IEN	b0_ICLR	b0_IL2	b0_IL1	b0_IL0
\$13	IP_b INT1 CONTROL	b1_PLTY	b1_E/L*	b1_INT	b1_IEN	b1_ICLR	b1_IL2	b1_IL1	b1_IL0
\$14	IP_c INT0 CONTROL	c0_PLTY	c0_E/L*	c0_INT	c0_IEN	c0_ICLR	c0_IL2	c0_IL1	c0_IL0
\$15	IP_c INT1 CONTROL	c1_PLTY	c1_E/L*	c1_INT	c1_IEN	c1_ICLR	c1_IL2	c1_IL1	c1_IL0
\$16	IP_d INT0 CONTROL	d0_PLTY	d0_E/L*	d0_INT	d0_IEN	d0_ICLR	d0_IL2	d0_IL1	d0_IL0
\$17	IP_d INT1 CONTROL	d1_PLTY	d1_E/L*	d1_INT	d1_IEN	d1_ICLR	d1_IL2	d1_IL1	d1_IL0
\$18	IP_a GENERAL CONTROL	a_ERR	0	a_RT1	a_RT0	a_WIDTH1	a_WIDTH0	a_BTD	a_MEN
\$19	IP_b GENERAL CONTROL	b_ERR	0	b_RT1	b_RT0	b_WIDTH1	b_WIDTH0	b_BTD	b_MEN
\$1A	IP_c GENERAL CONTROL	c_ERR	0	c_RT1	c_RT0	c_WIDTH1	c_WIDTH0	c_BTD	c_MEN
\$1B	IP_d GENERAL CONTROL	d_ERR	0	d_RT1	d_RT0	d_WIDTH1	d_WIDTH0	d_BTD	d_MEN
\$1C	RESERVED	0	0	0	0	0	0	0	0
\$1D	IP_CLOCK	0	0	0	0	0	0	0	IP32
\$1E	DMA ARBITRATION CONTROL	0	0	0	0	0	ROTAT	PR11	PR10
\$1F	IP_RESET	0	0	0	0	0	0	<i>IPI_RES</i>	RES

2592 9908 (1-6)

**Table 1-8. Petra IP2 Memory Map — Control/Status Registers (Cont'd)**

IP2 Sector Base Address = \$FFFBC000

Register Offset	Register Name	Register Bit Names							
		D7	D6	D5	D4	D3	D2	D1	D0
DMAC for IndustryPack a, request 0. This register set is referred to as DMACa in the text.									
\$20	DMA_a STATUS	0	DLBE	0	IPEND	CHANI	TBL	IPTO	DONE
\$21	DMA_a INT CTRL	0	0	DINT	DIEN	DICLR	DIL2	DIL1	DIL0
\$22	DMA ENABLE	0	0	0	0	0	0	0	DEN
\$23	RESERVED	0	0	0	0	0	0	0	0
\$24	DMA_a CONTROL 1	DHALT	0	DTBL	ADMA	WIDTH1	WIDTH0	0	XXX
\$25	DMA_a CONTROL 2	INTE	0	DMAEI	DMAEO	ENTO	TOIP	0	0
\$26	RESERVED	0	0	0	0	0	0	0	0
\$27	RESERVED	0	0	0	0	0	0	0	0
\$28	DMA_a LB ADDR	LBA31	LBA30	LBA29	LBA28	LBA27	LBA26	LBA25	LBA24
\$29	DMA_a LB ADDR	LBA23	LBA22	LBA21	LBA20	LBA19	LBA18	LBA17	LBA16
\$2A	DMA_a LB ADDR	LBA15	LBA14	LBA13	LBA12	LBA11	LBA10	LBA9	LBA8
\$2B	DMA_a LB ADDR	LBA7	LBA6	LBA5	LBA4	LBA3	LBA2	LBA1	LBA0
\$2C	DMA_a IP ADDR	0	0	0	0	0	0	0	0
\$2D	DMA_a IP ADDR	IPA23	IPA22	IPA21	IPA20	IPA19	IPA18	IPA17	IPA16
\$2E	DMA_a IP ADDR	IPA15	IPA14	IPA13	IPA12	IPA11	IPA10	IPA9	IPA8
\$2F	DMA_a IP ADDR	IPA7	IPA6	IPA5	IPA4	IPA3	IPA2	IPA1	IPA0
\$30	DMA_a BYTE CNT	0	0	0	0	0	0	0	0
\$31	DMA_a BYTE CNT	BCNT23	BCNT22	BCNT21	BCNT20	BCNT19	BCNT18	BCNT17	BCNT16
\$32	DMA_a BYTE CNT	BCNT15	BCNT14	BCNT13	BCNT12	BCNT11	BCNT10	BCNT9	BCNT8
\$33	DMA_a BYTE CNT	BCNT7	BCNT6	BCNT5	BCNT4	BCNT3	BCNT2	BCNT1	BCNT0
\$34	DMA_a TBL ADDR	TA31	TA30	TA29	TA28	TA27	TA26	TA25	TA24
\$35	DMA_a TBL ADDR	TA23	TA22	TA21	TA20	TA19	TA18	TA17	TA16
\$36	DMA_a TBL ADDR	TA15	TA14	TA13	TA12	TA11	TA10	TA9	TA8
\$37	DMA_a TBL ADDR	TA7	TA6	TA5	TA4	TA3	TA2	TA1	TA0

2592 9908 (2-6)

**Table 1-8. Petra IP2 Memory Map — Control/Status Registers (Cont'd)**

IP2 Sector Base Address = \$FFFBC000

Register Offset	Register Name	Register Bit Names							
		D7	D6	D5	D4	D3	D2	D1	D0
DMAC for IndustryPack b, request 0 or for IndustryPack a, request 1. This register set is referred to as DMACb in the text.									
\$38	DMA_b STATUS	0	DLBE	0	IPEND	CHANI	TBL	IPTO	DONE
\$39	DMA_b INT CTRL	0	0	DINT	DIEN	DICLR	DIL2	DIL1	DIL0
\$3A	DMA ENABLE	0	0	0	0	0	0	0	DEN
\$3B	RESERVED	0	0	0	0	0	0	0	0
\$3C	DMA_b CONTROL 1	DHALT	0	DTBL	ADMA	WIDTH1	WIDTH0	A_CH1	XXX
\$3D	DMA_b CONTROL 2	INTE	0	DMAEI	DMAEO	ENTO	TOIP	0	0
\$3E	RESERVED	0	0	0	0	0	0	0	0
\$3F	RESERVED	0	0	0	0	0	0	0	0
\$40	DMA_b LB ADDR	LBA31	LBA30	LBA29	LBA28	LBA27	LBA26	LBA25	LBA24
\$41	DMA_b LB ADDR	LBA23	LBA22	LBA21	LBA20	LBA19	LBA18	LBA17	LBA16
\$42	DMA_b LB ADDR	LBA15	LBA14	LBA13	LBA12	LBA11	LBA10	LBA9	LBA8
\$43	DMA_b LB ADDR	LBA7	LBA6	LBA5	LBA4	LBA3	LBA2	LBA1	LBA0
\$44	DMA_b IP ADDR	0	0	0	0	0	0	0	0
\$45	DMA_b IP ADDR	IPA23	IPA22	IPA21	IPA20	IPA19	IPA18	IPA17	IPA16
\$46	DMA_b IP ADDR	IPA15	IPA14	IPA13	IPA12	IPA11	IPA10	IPA9	IPA8
\$47	DMA_b IP ADDR	IPA7	IPA6	IPA5	IPA4	IPA3	IPA2	IPA1	IPA0
\$48	DMA_b BYTE CNT	0	0	0	0	0	0	0	0
\$49	DMA_b BYTE CNT	BCNT23	BCNT22	BCNT21	BCNT20	BCNT19	BCNT18	BCNT17	BCNT16
\$4A	DMA_b BYTE CNT	BCNT15	BCNT14	BCNT13	BCNT12	BCNT11	BCNT10	BCNT9	BCNT8
\$4B	DMA_b BYTE CNT	BCNT7	BCNT6	BCNT5	BCNT4	BCNT3	BCNT2	BCNT1	BCNT0
\$4C	DMA_b TBL ADDR	TA31	TA30	TA29	TA28	TA27	TA26	TA25	TA24
\$4D	DMA_b TBL ADDR	TA23	TA22	TA21	TA20	TA19	TA18	TA17	TA16
\$4E	DMA_b TBL ADDR	TA15	TA14	TA13	TA12	TA11	TA10	TA9	TA8
\$4F	DMA_b TBL ADDR	TA7	TA6	TA5	TA4	TA3	TA2	TA1	TA0

2592 9908(3-6)

**Table 1-8. Petra IP2 Memory Map — Control/Status Registers (Cont'd)**

IP2 Sector Base Address = \$FFFBC000

Register Offset	Register Name	Register Bit Names							
		D7	D6	D5	D4	D3	D2	D1	D0
DMAC for IndustryPack c, request 0. This register set is referred to as DMACc in the text.									
\$50	DMA_c STATUS	0	DLBE	0	IPEND	CHANI	TBL	IPTO	DONE
\$51	DMA_c INT CTRL	0	0	DINT	DIEN	DICLR	DIL2	DIL1	DIL0
\$52	DMA ENABLE	0	0	0	0	0	0	0	DEN
\$53	RESERVED	0	0	0	0	0	0	0	0
\$54	DMA_c CONTROL 1	DHALT	0	DTBL	ADMA	WIDTH1	WIDTH0	0	XXX
\$55	DMA_c CONTROL 2	INTE	0	DMAEI	DMAEO	ENTO	TOIP	0	0
\$56	RESERVED	0	0	0	0	0	0	0	0
\$57	RESERVED	0	0	0	0	0	0	0	0
\$58	DMA_c LB ADDR	LBA31	LBA30	LBA29	LBA28	LBA27	LBA26	LBA25	LBA24
\$59	DMA_c LB ADDR	LBA23	LBA22	LBA21	LBA20	LBA19	LBA18	LBA17	LBA16
\$5A	DMA_c LB ADDR	LBA15	LBA14	LBA13	LBA12	LBA11	LBA10	LBA9	LBA8
\$5B	DMA_c LB ADDR	LBA7	LBA6	LBA5	LBA4	LBA3	LBA2	LBA1	LBA0
\$5C	DMA_c IP ADDR	0	0	0	0	0	0	0	0
\$5D	DMA_c IP ADDR	IPA23	IPA22	IPA21	IPA20	IPA19	IPA18	IPA17	IPA16
\$5E	DMA_c IP ADDR	IPA15	IPA14	IPA13	IPA12	IPA11	IPA10	IPA9	IPA8
\$5F	DMA_c IP ADDR	IPA7	IPA6	IPA5	IPA4	IPA3	IPA2	IPA1	IPA0
\$60	DMA_c BYTE CNT	0	0	0	0	0	0	0	0
\$61	DMA_c BYTE CNT	BCNT23	BCNT22	BCNT21	BCNT20	BCNT19	BCNT18	BCNT17	BCNT16
\$62	DMA_c BYTE CNT	BCNT15	BCNT14	BCNT13	BCNT12	BCNT11	BCNT10	BCNT9	BCNT8
\$63	DMA_c BYTE CNT	BCNT7	BCNT6	BCNT5	BCNT4	BCNT3	BCNT2	BCNT1	BCNT0
\$64	DMA_c TBL ADDR	TA31	TA30	TA29	TA28	TA27	TA26	TA25	TA24
\$65	DMA_c TBL ADDR	TA23	TA22	TA21	TA20	TA19	TA18	TA17	TA16
\$66	DMA_c TBL ADDR	TA15	TA14	TA13	TA12	TA11	TA10	TA9	TA8
\$67	DMA_c TBL ADDR	TA7	TA6	TA5	TA4	TA3	TA2	TA1	TA0

2592 9908 (4-6)

**Table 1-8. Petra IP2 Memory Map — Control/Status Registers (Cont'd)**

IP2 Sector Base Address = \$FFFBC000

Register Offset	Register Name	Register Bit Names							
		D7	D6	D5	D4	D3	D2	D1	D0
DMAC for IndustryPack d, request 0 or for IndustryPack c, request 1, and for PACER CLOCK. This register set, not including the Pacer Clock, is referred to as DMACd in the text.									
\$68	DMA_d STATUS	0	DLBE	0	IPEND	CHANI	TBL	IPTO	DONE
\$69	DMA_d INT CTRL	0	0	DINT	DIEN	DICLR	DIL2	DIL1	DIL0
\$6A	DMA ENABLE	0	0	0	0	0	0	0	DEN
\$6B	RESERVED	0	0	0	0	0	0	0	0
\$6C	DMA_d CONTROL 1	DHALT	0	DTBL	ADMA	WIDTH1	WIDTH0	C_CH1	XXX
\$6D	DMA_d CONTROL 2	INTE	0	DMAEI	DMAEO	ENTO	TOIP	0	0
\$6E	RESERVED	0	0	0	0	0	0	0	0
\$6F	RESERVED	0	0	0	0	0	0	0	0
\$70	DMA_d LB ADDR	LBA31	LBA30	LBA29	LBA28	LBA27	LBA26	LBA25	LBA24
\$71	DMA_d LB ADDR	LBA23	LBA22	LBA21	LBA20	LBA19	LBA18	LBA17	LBA16
\$72	DMA_d LB ADDR	LBA15	LBA14	LBA13	LBA12	LBA11	LBA10	LBA9	LBA8
\$73	DMA_d LB ADDR	LBA7	LBA6	LBA5	LBA4	LBA3	LBA2	LBA1	LBA0
\$74	DMA_d IP ADDR	0	0	0	0	0	0	0	0
\$75	DMA_d IP ADDR	IPA23	IPA22	IPA21	IPA20	IPA19	IPA18	IPA17	IPA16
\$76	DMA_d IP ADDR	IPA15	IPA14	IPA13	IPA12	IPA11	IPA10	IPA9	IPA8
\$77	DMA_d IP ADDR	IPA7	IPA6	IPA5	IPA4	IPA3	IPA2	IPA1	IPA0
\$78	DMA_d BYTE CNT	0	0	0	0	0	0	0	0
\$79	DMA_d BYTE CNT	BCNT23	BCNT22	BCNT21	BCNT20	BCNT19	BCNT18	BCNT17	BCNT16
\$7A	DMA_d BYTE CNT	BCNT15	BCNT14	BCNT13	BCNT12	BCNT11	BCNT10	BCNT9	BCNT8
\$7B	DMA_d BYTE CNT	BCNT7	BCNT6	BCNT5	BCNT4	BCNT3	BCNT2	BCNT1	BCNT0
\$7C	DMA_d TBL ADDR	TA31	TA30	TA29	TA28	TA27	TA26	TA25	TA24
\$7D	DMA_d TBL ADDR	TA23	TA22	TA21	TA20	TA19	TA18	TA17	TA16
\$7E	DMA_d TBL ADDR	TA15	TA14	TA13	TA12	TA11	TA10	TA9	TA8
\$7F	DMA_d TBL ADDR	TA7	TA6	TA5	TA4	TA3	TA2	TA1	TA0
\$80	PACER INT CONTROL	0	IRE	INT	IEN	ICLR	IL2	IL1	IL0
\$81	PACER GEN CONTROL	PLTY	PLS	0	EN	CLR	PS2	PS1	PS0
\$82	PACER TIMER	T15	T14	T13	T12	TA11	TA10	T9	T8
\$83	PACER TIMER	TA7	TA6	TA5	TA4	TA3	TA2	T1	T0

2592 9908 (5-6)

**Table 1-8. Petra IP2 Memory Map — Control/Status Registers (Cont'd)**

IP2 Sector Base Address = \$FFFBC000

Register Offset	Register Name	Register Bit Names							
		D7	D6	D5	D4	D3	D2	D1	D0
\$90	<i>ALTERNATE DMA_a BYTE COUNT</i>	0	0	0	0	0	0	0	0
\$91		<i>A_BCNT23</i>	<i>A_BCNT22</i>	<i>A_BCNT21</i>	<i>A_BCNT20</i>	<i>A_BCNT19</i>	<i>A_BCNT18</i>	<i>A_BCNT17</i>	<i>A_BCNT16</i>
\$92		<i>A_BCNT15</i>	<i>A_BCNT14</i>	<i>A_BCNT13</i>	<i>A_BCNT12</i>	<i>A_BCNT11</i>	<i>A_BCNT10</i>	<i>A_BCNT9</i>	<i>A_BCNT8</i>
\$93		<i>A_BCNT7</i>	<i>A_BCNT6</i>	<i>A_BCNT5</i>	<i>A_BCNT4</i>	<i>A_BCNT3</i>	<i>A_BCNT2</i>	<i>A_BCNT1</i>	<i>A_BCNT0</i>
\$94	<i>ALTERNATE DMA_b BYTE COUNT</i>	0	0	0	0	0	0	0	0
\$95		<i>A_BCNT23</i>	<i>A_BCNT22</i>	<i>A_BCNT21</i>	<i>A_BCNT20</i>	<i>A_BCNT19</i>	<i>A_BCNT18</i>	<i>A_BCNT17</i>	<i>A_BCNT16</i>
\$96		<i>A_BCNT15</i>	<i>A_BCNT14</i>	<i>A_BCNT13</i>	<i>A_BCNT12</i>	<i>A_BCNT11</i>	<i>A_BCNT10</i>	<i>A_BCNT9</i>	<i>A_BCNT8</i>
\$97		<i>A_BCNT7</i>	<i>A_BCNT6</i>	<i>A_BCNT5</i>	<i>A_BCNT4</i>	<i>A_BCNT3</i>	<i>A_BCNT2</i>	<i>A_BCNT1</i>	<i>A_BCNT0</i>
\$98	<i>ALTERNATE DMA_c BYTE COUNT</i>	0	0	0	0	0	0	0	0
\$99		<i>A_BCNT23</i>	<i>A_BCNT22</i>	<i>A_BCNT21</i>	<i>A_BCNT20</i>	<i>A_BCNT19</i>	<i>A_BCNT18</i>	<i>A_BCNT17</i>	<i>A_BCNT16</i>
\$9A		<i>A_BCNT15</i>	<i>A_BCNT14</i>	<i>A_BCNT13</i>	<i>A_BCNT12</i>	<i>A_BCNT11</i>	<i>A_BCNT10</i>	<i>A_BCNT9</i>	<i>A_BCNT8</i>
\$9B		<i>A_BCNT7</i>	<i>A_BCNT6</i>	<i>A_BCNT5</i>	<i>A_BCNT4</i>	<i>A_BCNT3</i>	<i>A_BCNT2</i>	<i>A_BCNT1</i>	<i>A_BCNT0</i>
\$9C	<i>ALTERNATE DMA_d BYTE COUNT</i>	0	0	0	0	0	0	0	0
\$9D		<i>A_BCNT23</i>	<i>A_BCNT22</i>	<i>A_BCNT21</i>	<i>A_BCNT20</i>	<i>A_BCNT19</i>	<i>A_BCNT18</i>	<i>A_BCNT17</i>	<i>A_BCNT16</i>
\$9E		<i>A_BCNT15</i>	<i>A_BCNT14</i>	<i>A_BCNT13</i>	<i>A_BCNT12</i>	<i>A_BCNT11</i>	<i>A_BCNT10</i>	<i>A_BCNT9</i>	<i>A_BCNT8</i>
\$9F		<i>A_BCNT7</i>	<i>A_BCNT6</i>	<i>A_BCNT5</i>	<i>A_BCNT4</i>	<i>A_BCNT3</i>	<i>A_BCNT2</i>	<i>A_BCNT1</i>	<i>A_BCNT0</i>
\$A0	<i>ALTERNATE DMA_a CTL 2</i>	<i>EN_SC</i>	<i>DMAEII</i>	<i>CHAIN_</i> <i>DMAEII</i>				<i>SCI</i>	<i>SC0</i>
\$A/1	<i>ALTERNATE DMA_b CTL 2</i>	<i>EN_SC</i>	<i>DMAEII</i>	<i>CHAIN_</i> <i>DMAEII</i>				<i>SCI</i>	<i>SC0</i>
\$A/2	<i>ALTERNATE DMA_c CTL 2</i>	<i>EN_SC</i>	<i>DMAEII</i>	<i>CHAIN_</i> <i>DMAEII</i>				<i>SCI</i>	<i>SC0</i>
\$A/3	<i>ALTERNATE DMA_d CTL 2</i>	<i>EN_SC</i>	<i>DMAEII</i>	<i>CHAIN_</i> <i>DMAEII</i>				<i>SCI</i>	<i>SC0</i>

2592 9908 (6-6)



**Table 1-9. Z85230 SCC Register Addresses**

SCC	Z85230 SCC Register	Address
SCC #1	Port B Control	\$FFF45001
	Port B Data	\$FFF45003
	Port A Control	\$FFF45005
	Port A Data	\$FFF45007
SCC #2	Port B Control	\$FFF45801
	Port B Data	\$FFF45803
	Port A Control	\$FFF45805
	Port A Data	\$FFF45807

**Table 1-10. 82596CA Ethernet LAN Memory Map**

**82596CA Ethernet LAN  
Directly Accessible Registers**

Address	Data Bits					
	D31	...	D16	D15	...	D0
\$FFF46000	Upper Command Word			Lower Command Word		
\$FFF46004	MPU Channel Attention (CA)					

**Notes**

1. Refer to the MPU Port and MPU Channel Attention registers in Chapter 3.
2. After reset you must write the System Configuration Pointer to the command registers prior to writing to the MPU Channel Attention register. Writes to the System Configuration Pointer must be upper word first, lower word second.

**Table 1-11. 53C710 SCSI Memory Map**

Base Address is \$FFF47000

Big Endian Mode	53C710 Register Address Map				SCRIPTs Mode and Little Endian Mode
00	SIEN	SDID	SCNTL1	SCNTL0	00
04	SOCL	SODL	SXFER	SCID	04
08	SBCL	SBDL	SIDL	SFBR	08
0C	SSTAT2	SSTAT1	SSTAT0	DSTAT	0C
10	DSA				10
14	CTEST3	CTEST2	CTEST1	CTEST0	14
18	CTEST7	CTEST6	CTEST5	CTEST4	18
1C	TEMP				1C
20	LCRC	CTEST8	ISTAT	DFIFO	20
24	DCMD	DBC			24
28	DNAD				28
2C	DSP				2C
30	DSPS				30
34	SCRATCH				34
38	DCNTL	DWT	DIEN	DMODE	38
3C	ADDER				3C

**Note** Accesses may be 8-bit or 32-bit, but not 16-bit.

## BBRAM/TOD Clock Memory Map

The M48T58 BBRAM (also called Non-Volatile RAM or NVRAM) is divided into six areas as shown in Table 1-12. The first five areas are defined by software, while the sixth area, the time-of-day (TOD) clock, is defined by the chip hardware. The first area is reserved for user data. The second area is used by Motorola networking software. The third area may be used by an operating system. The fourth area is used by the MVME1X2P4 board debugger (MVME1X2Bug). The fifth area, detailed in Table 1-13, is the configuration area. The sixth area, the TOD clock, detailed in Table 1-14, is defined by the chip hardware.

**Table 1-12. M48T58 BBRAM/TOD Clock Memory Map**

Address Range	Description	Size (Bytes)
\$FFFC0000 - \$FFFC0FFF	User Area	4096
\$FFFC1000 - \$FFFC10FF	Networking Area	256
\$FFFC1100 - \$FFFC16F7	Operating System Area	1528
\$FFFC16F8 - \$FFFC1EF7	Debugger Area	2048
\$FFFC1EF8 - \$FFFC1FF7	Configuration Area	256
\$FFFC1FF8 - \$FFFC1FFF	TOD Clock	8

**Table 1-13. BBRAM Configuration Area Memory Map**

Address Range	Description	Size (Bytes)
\$FFFC1EF8 - \$FFFC1EFB	Version	4
\$FFFC1EFC - \$FFFC1F07	Serial Number	12
\$FFFC1F08 - \$FFFC1F17	Board ID	16
\$FFFC1F18 - \$FFFC1F27	PWA	16
\$FFFC1F28 - \$FFFC1F2B	Speed	4
\$FFFC1F2C - \$FFFC1F31	Ethernet Address	6
\$FFFC1F32 - \$FFFC1F33	Reserved	2
\$FFFC1F34 - \$FFFC1F35	Local SCSI ID	2
\$FFFC1F36 - \$FFFC1F3D	Memory Mezz. PWB	8
\$FFFC1F3E - \$FFFC1F45	Memory Mezz. Serial Number	8
\$FFFC1F46 - \$FFFC1F4D	Static Mezz. PWB	8
\$FFFC1F4E - \$FFFC1F4D	Static Mezz. Serial	8
\$FFFC1F56 - \$FFFC1F5D	ECC1 Mezz. PWB	8
\$FFFC1F5E - \$FFFC1F5D	ECC1 Mezz Serial	8
\$FFFC1F66 - \$FFFC1F65	ECC2 Mezz. PWB	8
\$FFFC1F6E - \$FFFC1F75	ECC2 Mezz. Serial	8
\$FFFC1F76 - \$FFFC1F7D	Ser. Port 2 Pers. PWB	8
\$FFFC1F7E - \$FFFC1F85	Ser. Port 2 Pers. Serial No.	8
\$FFFC1F86 - \$FFFC1F8D	IP_a Board ID	8
\$FFFC1F8E - \$FFFC1F95	IP_a Board Serial Number	8
\$FFFC1F96 - \$FFFC1F9D	IP_a Board PWB	8
\$FFFC1F9E - \$FFFC1FA5	IP_b Board ID	8
\$FFFC1FA6 - \$FFFC1FAD	IP_b Board Serial Number	8
\$FFFC1FAE - \$FFFC1FB5	IP_b Board PWB	8
\$FFFC1FB6 - \$FFFC1FBD	IP_c Board ID	8
\$FFFC1FBE - \$FFFC1FC5	IP_c Board Serial Number	8
\$FFFC1FC6 - \$FFFC1FCD	IP_c Board PWB	8
4FFFC1FCE - \$FFFC1FD5	IP_d Board ID	8

**Table 1-13. BBRAM Configuration Area Memory Map (Continued)**

Address Range	Description	Size (Bytes)
\$FFFC1FD6 - \$FFFC1FDD	IP_d Board Serial Number	8
\$FFFC1FDE - \$FFFC1FE5	IP_d Board PWB	8
\$FFFC1FE6 - \$FFFC1FF6	Reserved	65
\$FFFC1FF7	Checksum	1

**Table 1-14. TOD Clock Memory Map**

Address	Data Bits								Function	
	D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0		
\$FFFC1FF8	W	R	S	Calibration					Control	
\$FFFC1FF9	ST	--	--	--	--	--	--	--	Seconds	00
\$FFFC1FFA	x	--	--	--	--	--	--	--	Minutes	00
\$FFFC1FFB	x	x	--	--	--	--	--	--	Hour	00
\$FFFC1FFC	x	FT	x	x	x	--	--	--	Day	01
\$FFFC1FFD	x	x	--	--	--	--	--	--	Date	01
\$FFFC1FFE	x	x	x	--	--	--	--	--	Month	01
\$FFFC1FFF	--	--	--	--	--	--	--	--	Year	00

**Notes** W = Write Bit  
R = Read Bit  
S = Sign Bit  
ST = Stop Bit  
FT = Frequency Test  
x = Must be set to 0

The data structure of the configuration bytes starts at \$FFFC1EF8. It is organized as shown below.

```
struct brdi_cnfg {
    char          version[4];
    char          serial[12];
    char          id[16];
    char          pwa[16];
    char          speed[4];
    char          ethernet[6];
    char          fill[2];
    char          lscsiid[2];
    char          pmem_pwb[8];
    char          pmem_serial[8];
    char          smem_pwb [8] ;
    char          smem_serial [8];
    char          ecc1mem_pwb [8];
    char          ecc1mem_serial [8];
    char          ecc2mem_pwb [8] ;
    char          ecc2mem_srial [8];
    char          port2_pwb[8];
    char          port2_serial[8];
    char          ipa_brdid[8];
    char          ipa_serial[8];
    char          ipa_pwb[8];
    char          ipb_brdid[8];
    char          ipb_serial[8];
    char          ipb_pwb[8];
    char          ipc_brdid[8];
    char          ipc_serial[8];
    char          ipc_pwb[8];
    char          ipd_brdid[8];
    char          ipd_serial[8];
    char          ipd_pwb[8];
    char          reserved[17];
    char          cksum[1];
}
```

The fields are defined as follows:

1. Four bytes are reserved for the revision or version of this structure. This revision is stored in ASCII format, with the first two bytes being the major version numbers and the last two bytes being the

minor version numbers. For example, if the version of this structure is 1.0, this field contains:

0100

2. Twelve bytes are reserved for the serial number of the board in ASCII format. For example, this field could contain:

000000470476

3. Sixteen bytes are reserved for the board ID in ASCII format. For example, for an MVME162P4 board with MC68040, SCSI, Ethernet, 16MB SDRAM, and 512KB SRAM, this field might contain:

MVME162P-344SE

(The 14 characters are followed by two blanks.)

4. Sixteen bytes are reserved for the printed wiring assembly (PWA) number assigned to this board in ASCII format. This includes the 01-w prefix. This is for the main logic board if more than one board is required for a set. Additional boards in a set are defined by a structure for that set. For example, for an MVME162P4 board with MC68040, SCSI, Ethernet, 16MB SDRAM, and 512KB SRAM, at revision D, the PWA field might contain:

01-W3528F50D

(The 12 characters are followed by four blanks.)

5. Four bytes contain the speed of the board in MHz. The first two bytes are the whole number of MHz and the second two bytes are fractions of MHz. For example, for a 60.00 MHz board, this field contains:

6000

6. Six bytes are reserved for the Ethernet address. The address is stored in hexadecimal format. (Refer to the detailed description earlier in this chapter.) If the board does not support Ethernet, this field is filled with zeros.
7. These two bytes are reserved.

8. Two bytes are reserved for the local SCSI ID. The SCSI ID is stored in ASCII format.
9. Eight bytes are reserved for the printed wiring board (PWB) number assigned to the memory mezzanine board in ASCII format.

**Note** Although no memory mezzanine is present on MVME1X2P4 series boards, the on-board memory is modeled as such for backward compatibility. The firmware fills this and other fields relating to memory mezzanines with an “n/a”.

10. Eight bytes are reserved for the serial number assigned to the memory mezzanine board in ASCII format.
11. Eight bytes are reserved for the printed wiring board (PWB) number assigned to the serial port 2 personality board in ASCII format.
  - Static Memory Mezzanine PWB identifier in ASCII
  - Static Memory Mezzanine serial number in ASCII
  - ECC1 Memory Mezzanine PWB identifier in ASCII
  - ECC1 Memory Mezzanine serial number in ASCII
  - ECC2 Memory Mezzanine PWB identifier in ASCII
  - ECC2 Memory Mezzanine serial number in ASCII
12. Eight bytes are reserved for the serial number assigned to the serial port 2 personality board in ASCII format.
13. Eight bytes are reserved for the board identifier, in ASCII, assigned to the optional first IndustryPack module, IP a.
14. Eight bytes are reserved for the serial number, in ASCII, assigned to the optional first IndustryPack module, IP a.
15. Eight bytes are reserved for the printed wiring board (PWB) number assigned to the optional first IndustryPack module, IP a.
16. Eight bytes are reserved for the board identifier, in ASCII, assigned to the optional second IndustryPack module, IP b.
17. Eight bytes are reserved for the serial number, in ASCII, assigned to the optional second IndustryPack module, IP b.



18. Eight bytes are reserved for the printed wiring board (PWB) number assigned to the optional second IndustryPack module, IP b.
19. Eight bytes are reserved for the board identifier, in ASCII, assigned to the optional third IndustryPack module, IP c.
20. Eight bytes are reserved for the serial number, in ASCII, assigned to the optional third IndustryPack module, IP c.
21. Eight bytes are reserved for the printed wiring board (PWB) number assigned to the optional third IndustryPack module, IP c.
22. Eight bytes are reserved for the board identifier, in ASCII, assigned to the optional fourth IndustryPack module, IP d.
23. Eight bytes are reserved for the serial number, in ASCII, assigned to the optional fourth IndustryPack module, IP d.
24. Eight bytes are reserved for the printed wiring board (PWB) number assigned to the optional fourth IndustryPack module, IP d.
25. Growth space (65 bytes) is reserved. This pads the structure to an even 256 bytes.
26. The final byte of the area is reserved for a checksum (as defined in the *Debugging Package for Motorola 68K CISC CPUs User's Manual*) for security and data integrity of the configuration area of the NVRAM. This data is stored in hexadecimal format.

### Interrupt Acknowledge Map

The local bus distinguishes interrupt acknowledge cycles from other cycles by placing the binary value %11 on TT1-TT0. It also specifies the level that is being acknowledged using TM2-TM0. The interrupt handler selects which device within that level is being acknowledged.

### VMEbus Memory Map

This section describes the mapping of local resources as viewed by VMEbus masters. Default addresses for the slave, master, and GCSR address decoders are provided by the **ENV** command.

## VMEbus Accesses to the Local Bus

The VMEchip2 includes a user-programmable map decoder for the VMEbus-to-local-bus interface. The map decoder allows you to program the starting and ending address and the modifiers to which the MVME1X2P4 responds.

## VMEbus Short I/O Memory Map

The VMEchip2 includes a user-programmable map decoder for the GCSR. The GCSR map decoder allows you to program the starting address of the GCSR in the VMEbus short I/O space.

# Software Support Considerations

The MVME1X2P4 is a complex board that interfaces to both the VMEbus and the SCSI bus. These multiple bus interfaces raise the issue of cache coherency and support of indivisible cycles. There are also many sources of bus error. First, let us consider how interrupts are handled.

## Interrupts

The MC680x0 uses hardware-vectored interrupts.

Most interrupt sources are level and base vector programmable. Interrupt vectors from the Petra MC2 sector and the VMEchip2 have two sections: a base value which can be set by the processor, usually the upper four bits, and the lower bits which are set according to the particular interrupt source. There is an onboard daisy chain of interrupt sources, with interrupts from the Petra MC2 sector having the highest priority, those from the Petra IP2 sector having the next highest priority, and interrupt sources from the VMEchip2 having the lowest priority. Refer to Appendix B for an example of interrupt usage.

The VMEchip2 ASIC and the MC2 and IP2 sectors of the Petra chip are used to implement the multilevel MC680x0 interrupt architecture. A PLD (programmable logic device) is used to combine the individual IPLx signals from each ASIC.

## Cache Coherency

The MC680x0 has the ability to monitor local bus cycles executed by other local bus masters such as the SCSI DMA controller, the LAN, the VMEchip2 DMA controller, the VMEbus-to-local-bus controller, and the Petra's IP sector DMA controller.

When snooping is enabled, the MPU can invalidate cache entries as required by the current cycle. (In addition, boards equipped with an MC68040 or MC68LC040 processor can also source data as required by the current cycle.)

The MPU cannot monitor VMEbus cycles that do not access the local bus on the MVME1X2P4. Software must ensure that data shared by multiple processors is kept in memory that is not cached. The software must also mark all onboard and off-board I/O areas as cache inhibited and serialized.

## Sources of Local BERR\*

A TEA\* signal (indicating a bus error) is returned to the local bus master when a local bus time-out occurs, a DRAM parity error occurs and parity checking is enabled, or a VME bus error occurs during a VMEbus access.

**Note** Although the MVME1X2P4 series contains synchronous DRAM (SDRAM) rather than parity DRAM, it can be made to emulate parity DRAM for compatibility with the MC2 programming model.

The MVME1X2P4 devices that are capable of asserting a local bus error are described below.

## Local Bus Time-Out

A Local Bus time-out occurs whenever a local bus cycle does not complete within the programmed time (VMEbus-bound cycles are not timed by the local bus timer). As long as the system is configured properly, this should only happen if software accesses a nonexistent location within the onboard address range.

### VMEbus Access Time-Out

A VMEbus Access time-out occurs whenever a VMEbus-bound transfer does not receive a VMEbus bus grant within the programmed time. This is usually caused by another bus master holding the bus for an excessive period of time.

### VMEbus BERR\*

A VMEbus BERR\* occurs when the BERR\* signal line is asserted on the VMEbus while a local bus master is accessing the VMEbus. VMEbus BERR\* should occur only if one of the following occurs:

- ❑ An initialization routine samples to see if a device is present on the VMEbus and if it is not, software accesses a nonexistent device within the VMEbus range.
- ❑ Incorrect configuration information causes the VMEchip2 to incorrectly access a device on the VMEbus (such as driving LWORD\* low to a 16-bit board).
- ❑ A hardware error occurs on the VMEbus.
- ❑ A VMEbus slave reports an access error (such as parity error).

### Local DRAM Parity Error

**Note** Although the MVME1X2P4 series contains synchronous DRAM (SDRAM) rather than parity DRAM, it can be made to emulate parity DRAM for compatibility with the MC2 programming model.

When parity checking is enabled, the current bus master receives a bus error and a parity error occurs while it is accessing the local DRAM.

### VMEchip2

An 8- or 16-bit write to the LCSR in the VMEchip2 causes a local BERR\*.

## Bus Error Processing

Because different conditions can cause bus error exceptions, the software must be able to distinguish the source. To aid in this, status registers are provided for every local bus master. The next section describes the various causes of bus errors and the associated status registers.

Generally, the bus error handler can interrogate the status bits and proceed with the result. However, an interrupt can happen during the execution of the bus error handler (before an instruction can write to the status register to raise the interrupt mask). If the interrupt service routine causes a second bus error, the status that indicates the source of the first bus error may be lost. Software must be written to deal with such cases.

## Description of Error Conditions on the MVME1X2P4

This section lists the various error conditions that may be reported by the MVME1X2P4 hardware. A subsection heading identifies each type of error condition. A standard format gives a description of the error, indicates how notification of the error condition is made, indicates which status register(s) have information about the error, and concludes with some comments pertaining to each particular error.

### MPU Parity Error

**Note** Although the MVME1X2P4 series contains synchronous DRAM (SDRAM) rather than parity DRAM, it can be made to emulate parity DRAM for compatibility with the MC2 programming model.

Description:	A DRAM parity error.
MPU Notification:	TEA is asserted during an MPU DRAM access.
Status:	Bit 9 of the MPU Status and DMA Interrupt Count register in the VMEchip2. Address \$FFF40048.
Comments:	After memory has been initialized, this error normally indicates a hardware problem.

### MPU Offboard Error

Description:	An error occurred while the MPU was attempting to access an offboard resource.
MPU Notification:	TEA is asserted during offboard access.
Status:	Bit 8 of the MPU Status and DMA Interrupt Count register. Address \$FFF40048.
Comments:	This can be caused by a VMEbus timeout, a VMEbus BERR*, or an MVME1X2P4 VMEbus access timeout. The latter is the time from when the VMEbus has been requested to when it is granted.

### MPU TEA - Cause Unidentified

Description:	An error occurred while the MPU was attempting an access.
MPU Notification:	TEA is asserted during an MPU access.
Status:	Bit 10 of the MPU Status and DMA Interrupt Count register. Address \$FFF40048 in the VMEchip2.
Comments:	No status was given as to the cause of the TEA assertion.

### MPU Local Bus Timeout

Description:	An error occurred while the MPU was attempting to access a local resource.
MPU Notification:	TEA is asserted during the MPU access.
Status:	Bit 7 of the MPU Status and DMA Interrupt Count register (actually in the DMAC Status register). Address \$FFF40048.
Comments:	The Local Bus timer timed out. This usually indicates the MPU tried to read or write an address at which there was no resource. Otherwise, it indicates a hardware problem.

## DMAC VMEbus Error

Description:	The DMAC experienced a VMEbus error during an attempted transfer.
MPU Notification:	DMAC interrupt (when enabled)
Status:	The VME bit is set in the DMAC Status register. Address \$FFF40048, bit 1.
Comments:	This indicates the DMAC attempted to access a VMEbus address at which there was no resource or the VMEbus slave returned a BERR* signal.

## DMAC Parity Error

**Note** Although the MVME1X2P4 series contains synchronous DRAM (SDRAM) rather than parity DRAM, it can be made to emulate parity DRAM for compatibility with the MC2 programming model.

Description:	Parity error while the DMAC was reading DRAM.
MPU Notification:	DMAC interrupt (when enabled)
Status:	The DLPE bit is set in the DMAC Status register. Address \$FFF40048, bit 5.
Comments:	If the TBL bit is set (address \$FFF40048 bit 2), the error occurred during a command table access; otherwise the error occurred during a data access.

### DMAC Offboard Error

Description:	Error encountered while the Local Bus side of the DMAC was attempting to go to the VMEbus.
MPU Notification:	DMAC interrupt (when enabled)
Status:	The DLOB bit is set in the DMAC Status register. Address \$FFF40048, bit 4.
Comments:	This is normally caused by a programming error. The Local Bus address of the DMAC should not be programmed with a Local Bus address that maps to the VMEbus. If the TBL bit is set (address \$FFF40048 bit 2), the error occurred during a command table access; otherwise the error occurred during a data access.

### DMAC LTO Error

Description:	A Local Bus timeout (LTO) occurred while the DMAC was Local Bus master.
MPU Notification:	DMAC interrupt (when enabled)
Status:	The DLTO bit is set in the DMAC Status register. Address \$FFF40048, bit 3.
Comments:	This indicates the DMAC attempted to access a Local Bus address at which there was no resource. If the TBL bit is set (address \$FFF40048 bit 2), the error occurred during a command table access; otherwise the error occurred during a data access.



## DMAC TEA - Cause Unidentified

Description:	An error occurred while the DMAC was Local Bus master and additional status was not provided.
MPU Notification:	DMAC interrupt (when enabled)
Status:	The DLBE bit is set in the DMAC Status register. Address \$FFF40048, bit 6.
Comments:	An 8- or 16-bit write to the LCSR in the VMEchip2 causes this error. If the TBL bit is set (address \$FFF40048 bit 2), the error occurred during a command table access; otherwise the error occurred during a data access.

## LAN Parity Error

**Note** Although the MVME1X2P4 series contains synchronous DRAM (SDRAM) rather than parity DRAM, it can be made to emulate parity DRAM for compatibility with the MC2 programming model.

Description:	Parity error while the LANCE was reading DRAM.
MPU Notification:	Petra MC2 Sector Interrupt (LAN ERROR IRQ)
Status:	Petra MC2 Sector LAN Error Status register. Address \$FFF42028.
Comments:	The LANCE has no ability to respond to TEA, so the error interrupt and status are provided in the Petra MC2 sector. Control for the interrupt is in the MC2 sector's LAN Error Interrupt Control register (\$FFF4202B).

**LAN Offboard Error**

Description:	Error encountered while the LANCE was attempting to go to the VMEbus.
MPU Notification:	Petra MC2 Sector Interrupt (LAN ERROR IRQ)
Status:	Petra MC2 Sector LAN Error Status register. Address \$FFF42028.
Comments:	The LANCE has no ability to respond to TEA, so the error interrupt and status are provided in the Petra MC2 sector. Control for the interrupt is in the MC2 sector's LAN Error Interrupt Control register (\$FFF4202B).

**LAN LTO Error**

Description:	Local Bus Time-out occurred while the LANCE was Local Bus master.
MPU Notification:	Petra MC2 Sector Interrupt (LAN ERROR IRQ)
Status:	Petra MC2 Sector LAN Error Status register. Address \$FFF42028.
Comments:	The LANCE has no ability to respond to TEA, so the error interrupt and status are provided in the Petra MC2 sector. Control for the interrupt is in the MC2 sector's LAN Error Interrupt Control register (\$FFF4202B).

## SCSI Parity Error

**Note** Although the MVME1X2P4 series contains synchronous DRAM (SDRAM) rather than parity DRAM, it can be made to emulate parity DRAM for compatibility with the MC2 programming model.

Description: Parity error detected while the 53C710 was reading DRAM.

MPU Notification: 53C710 Interrupt

Status: 53C710 DMA Status register.  
53C710 DMA Interrupt Status register.  
Petra MC2 Sector SCSI Error Status register.  
Address \$FFF4202C.

Comments: 53C710 interrupt enables are controlled in the 53C710 and in the Petra MC2 sector's SCSI Interrupt Control register (\$FFF4202F).

## SCSI Offboard Error

Description: Error encountered while the 53C710 was attempting to go to the VMEbus.

MPU Notification: 53C710 Interrupt.

Status: 53C710 DMA Status register.  
53C710 DMA Interrupt Status register.  
Petra MC2 Sector SCSI Error Status register.  
Address \$FFF4202C.

Comments: 53C710 interrupt enables are controlled in the 53C710 and in the Petra MC2 sector's SCSI Interrupt Control register (\$FFF4202F).

## SCSI LTO Error

Description:	Local Bus Timeout occurred while the 53C710 was Local Bus master.
MPU Notification:	53C710 Interrupt
Status:	53C710 DMA Status register. 53C710 DMA Interrupt Status register. Petra MC2 Sector SCSI Error Status register. Address \$FFF4202C.
Comments:	53C710 interrupt enables are controlled in the 53C710 and in the Petra MC2 sector's SCSI Interrupt Control register (\$FFF4202F).

## Proper Use of Bus Timers

As an example of how to use the bus timers, consider the sequence of events that occur when the MPU on one MVME1X2P4 accesses the local bus memory on another MVME1X2P4 using the VMEbus. This scenario would involve three bus timers:

- ❑ The Local Bus timer
- ❑ The VMEbus Access timer
- ❑ The Global VMEbus timer

The local bus timer measures the time an access to an onboard resource takes. The VMEbus timer measures the time from when the VMEbus request has been initiated to when a VMEbus grant has been obtained. The global bus timer measures the time from when a VMEbus cycle begins to when it completes. These timers are normally set to quite different values.

When the processor or another local bus master initiates an access to the VMEbus, it first waits until any other local bus masters get off the bus. Then the master begins its cycle and the local bus timer starts counting. It continues to count until an address decode of the VMEbus address space is detected and then terminates. This is normally a very short period of

time. In fact, all local bus non-error bus accesses (such as the time to access onboard memory) are normally very short. Therefore, it is advisable to set this timer to a small value, such as 256  $\mu$ sec.

The next timer to take over when one MVME1X2P4 accesses another is the VMEbus access timer. This measures the time between when the VMEbus has been address-decoded (and hence a VMEbus request has been made) and the time when VMEbus mastership has been granted. Because we have found in the past that some VME systems can become very busy, we recommend that this time-out be set to a large value, such as 32 msec.

Once the VMEbus has been granted, a third timer takes over. This is the global VMEbus timer. This timer starts when a transfer actually begins (DS0 or DS1 goes active) and ends when that transfer completes (DS0 or DS1 goes inactive). This time should be longer than any expected legitimate transfer time on the bus. We normally set it to 256  $\mu$ sec.

This timer can also be disabled for debug purposes. Before an MVME1X2P4 access to another MVME1X2P4 can complete, however, the VMEchip2 on the accessed MVME1X2P4 must decode a slave access and request the local bus of the second MVME1X2P4. When the local bus is granted (any in-process onboard transfers having completed), then the local bus timer of the accessed MVME1X2P4 starts. Normally, this is also set to 256  $\mu$ sec. When the memory has the data available, a transfer acknowledge signal (TA) is given. This translates into a DTACK signal on the VMEbus which is then translated into a TA signal to the first requesting processor, and the transfer is complete. If the VMEbus global timer expires on a legitimate transfer, the VMEbus-to-local-bus controller in the VMEchip2 may become confused and the VMEchip2 may misbehave; therefore, the bus timers' values must be set correctly. The correct settings depend on the system configuration.

## MVME1X2P4 MC680x0 Indivisible Cycles

The MC68040 and MC68060 perform operations that require indivisible read-modify-write (RMW) memory accesses. These RMW sequences occur when the MMU modifies table entries or when the MPU executes a TAS, CAS, or CAS2 instruction. TAS cycles are always single-address

RMW operations, while the CAS, CAS2, and MMU operations can be multiple-address RMW cycles. The VMEbus does not support multiple-address RMW cycles and there is no defined protocol for supporting multiple-address RMW cycles which start onboard and then access off-board resources. The MVME1X2P4 does not fully support all RMW operations in all possible cases.

The MVME1X2P4 supports a limited subset of RMW instructions and makes the following assumptions:

- ❑ The MVME1X2P4 supports single-address RMW cycles caused by TAS and CAS instructions.
- ❑ Because it is not possible to tell if the MC680x0 is executing a single- or multiple-address read-modify-write cycle, software should only execute single-address RMW instructions. Multiple-address RMW cycles caused by CAS or CAS2 instructions are not guaranteed indivisible and may cause illegal VMEbus cycles.
- ❑ Lock cycles caused by MMU table walks do not cause illegal VMEbus cycles, and they are not guaranteed indivisible.

## **Illegal Access to IP Modules from External VMEbus Masters**

When a device other than the local MVME1X2P4 is operating as VMEbus master, access by that device to the local IP modules is subject to restrictions.

Access to the IndustryPack memory space is supported in all cases. Due to the difference in data width between the VMEbus and the IP modules (D32 versus D16), however, access to the IndustryPack I/O, ID, and Interrupt Acknowledge space is *not* supported for single IP modules. This applies to IndustryPacks A, B, C, and D.

## Introduction

This chapter describes the VMEchip2 ASIC, the local-bus/VMEbus interface chip.

The VMEchip2 interfaces the local bus to the VMEbus. In addition to its VMEbus-defined functions, the VMEchip2 includes a local-bus-to-VMEbus DMA controller, VME board support features, and Global Control and Status Registers (GCSRs) for interprocessor communications.

The following table summarizes the characteristics of the VMEchip2 ASIC.

**Table 2-1. Features of the VMEchip2 ASIC**

Function	Features
Local-Bus-to-VMEbus Interface	Programmable local bus map decoder
	Programmable short, standard, and extended VMEbus addressing
	Programmable AM codes
	Programmable 16-bit and 32-bit VMEbus data width
	Software-enabled write posting mode
	Write post buffer (one cache line or one four-byte)
	Automatically performs dynamic bus sizing for VMEbus cycles
	Software-configured VMEbus access timers
	Local-bus-to-VMEbus Requester with: <ul style="list-style-type: none"> <li data-bbox="417 1222 820 1263">– Software-enabled fair request mode</li> <li data-bbox="417 1263 824 1347">– Software-configured release modes: Release-When-Done (RWD) and Release-On-Request (ROR)</li> <li data-bbox="417 1347 954 1381">– Software-configured BR0*-BR3* request levels</li> </ul>

**Table 2-1. Features of the VMEchip2 ASIC (Continued)**

<b>Function</b>	<b>Features</b>
VMEbus-to-Local-Bus Interface	Programmable VMEbus map decoder
	Programmable AM decoder
	Programmable local bus snoop enable
	Simple VMEbus-to-local-bus address translation
	8-bit, 16-bit and 32-bit VMEbus data width
	8-bit, 16-bit and 32-bit block transfer
	Standard and extended VMEbus addressing
	Software-enabled write posting mode
	Write post buffer (17 four-bytes in BLT mode, two four-bytes in non-BLT mode)
	An eight four-byte read ahead buffer (BLT mode only)
32-bit Local-Bus-to-VMEbus DMA Controller	Programmable 16-bit, 32-bit, and 64-bit VMEbus data width
	Programmable short, standard, and extended VMEbus addressing
	Programmable AM code
	Programmable local bus snoop enable
	16 four-byte FIFO data buffer
	Up to 4 GB of data per DMA request
	Automatically adjustment of transfer size to optimize bus utilization
	DMA complete interrupt
	DMAC command chaining supported by a singly-linked list of DMA commands
	VMEbus DMA controller requester with: <ul style="list-style-type: none"> <li>– Software-enabled fair request modes;</li> <li>– Software-configured release modes: Release-On-Request (ROR), and Release-On-End-Of-Data (ROEOD);</li> <li>– Software-configured BR0-BR3 request levels; and</li> <li>– Software enabled bus-tenure timer</li> </ul>
VMEbus Interrupter	Software-configured IRQ1-IRQ7 interrupt request level
	8-bit software-programmed status/ID register



**Table 2-1. Features of the VMEchip2 ASIC (Continued)**

Function	Features
VMEbus System Controller	Arbiter with software-configured arbitration modes: – Priority (PRI), – Round-Robin-Select (RRS) – Single-level (SGL)
	Programmable arbitration timer
	IACK daisy-chain driver
	Programmable bus timer
	SYSRESET logic
Global Control Status Register Set	Four location monitors
	Global control of locally detected failures
	Global control of local reset
	Four global attention interrupt bits
	A chip ID and revision register
Interrupt Handler	Four 16-bit dual-ported general purpose registers
	All interrupts level-programmable
	All interrupts maskable
	All interrupts providing a unique vector
	Software and external interrupts
Watchdog timer	Control and status bits, 4-bit counter
Two tick timers	Control and status bits, 32-bit counter

## Functional Blocks

The following sections provide an overview of the functions implemented by the VMEchip2 ASIC. See Figure 2-1 for a block diagram of the VMEchip2. Detailed programming models for the local control and status registers (LCSRs) and the global control and status registers (GCSRs) appear in subsequent sections.

### Local-Bus-to-VMEbus Interface

The local-bus-to-VMEbus interface allows local bus masters access to global resources on the VMEbus. This interface includes a *local bus slave*, a *write post buffer*, and a *VMEbus master*.

Using programmable map decoders with programmable attribute bits, the local-bus-to-VMEbus interface can be configured to provide the following VMEbus capabilities:

Addressing capabilities: A16, A24, A32

Data transfer capabilities: D08, D16, D32

The *local bus slave* includes six local bus map decoders for accessing the VMEbus. The first four map decoders are general purpose programmable decoders, while the other two are fixed and are dedicated for I/O decoding.

The first four map decoders compare local bus address lines A31 through A16 with a 16-bit start address and a 16-bit end address. When an address in the selected range is detected, a VMEbus select is generated to the VMEbus master. Each map decoder also has eight attribute bits and an enable bit. The attribute bits are for VMEbus AM (address modifier) codes, D16 enable, and write post (WP) enable.

The fourth map decoder also includes a 16-bit alternate address register and a 16-bit alternate address select register. This allows any or all of the upper 16 address bits from the local bus to be replaced by bits from the alternate address register. The feature allows the local bus master to access any VMEbus address.

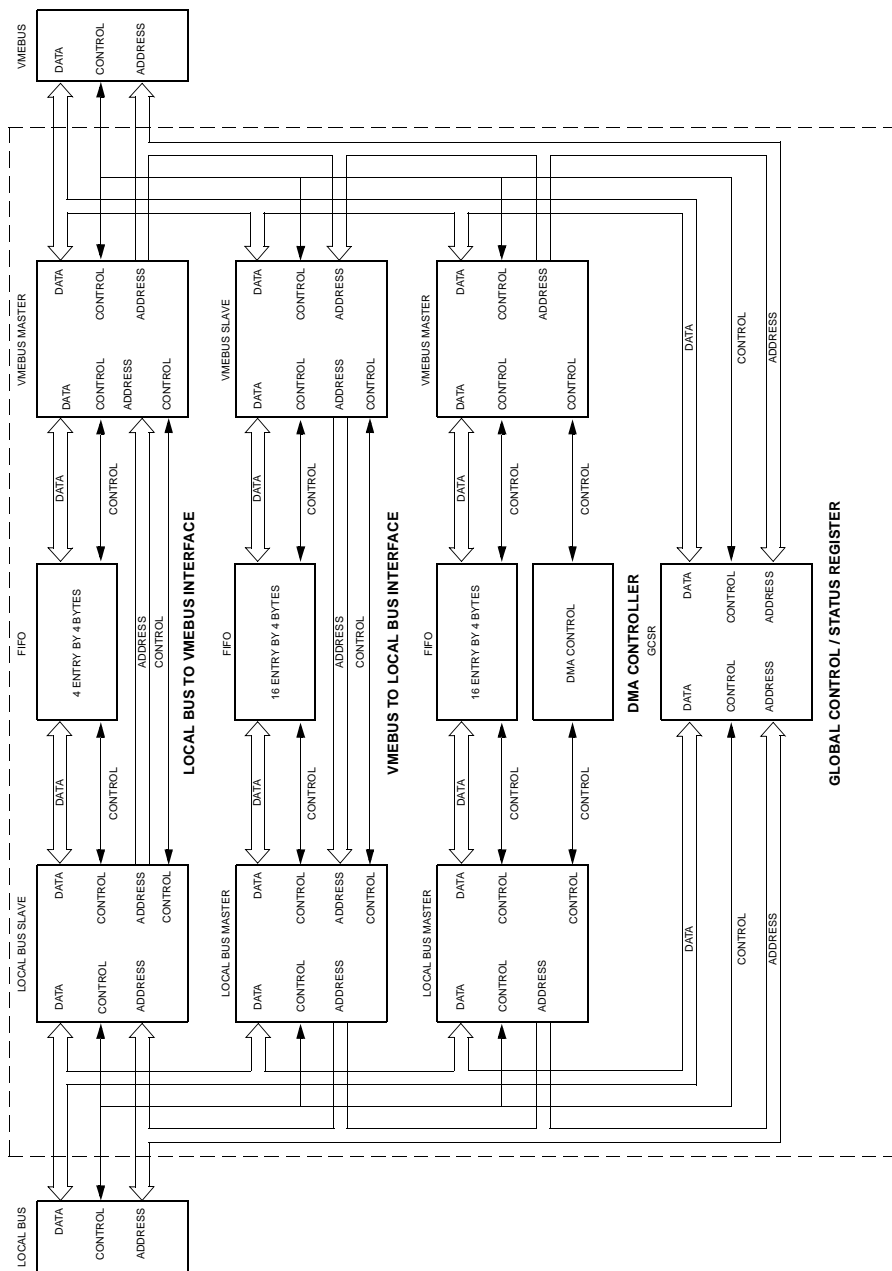


Figure 2-1. VMEchip2 Block Diagram

Using the four programmable map decoders, separate VMEbus maps can be created, each with its own attributes. For example, one map can be configured as A32, D32 with write posting enabled while a second map can be A24, D16 with write posting disabled.

The first I/O map decoder decodes local bus addresses \$FFFF0000 through \$FFFFFFFF as the short I/O A16/D16 or A16/D32 area. The other provides an A24/D16 space at \$F0000000 to \$F0FFFFFF and an A32/D16 space at \$F1000000 to \$FF7FFFFF.

Supervisor/non-privileged and program/data space is determined by attribute bits. Write posting may be enabled or disabled for each decoder I/O space and this map decoder may be enabled or disabled.

When *write posting* is enabled, the VMEchip2 stores the local bus address and data and then acknowledges the local bus master. The local bus is then free to perform other operations while the VMEbus master requests the VMEbus and performs the requested operation.

The write post buffer stores data in single-byte, double-byte, quad-byte, or one-cache-line (four quad-bytes) form. Write posting should only be enabled when bus errors are not expected. If a bus error is returned on a write posted cycle and the interrupt is enabled, the local processor is interrupted. The address of the error is not saved. Normal memory never returns a bus error on a write cycle. However, some VMEbus ECC memory cards perform a read-modify-write operation and therefore may return a bus error if there is an error on the read portion of a read-modify-write. Write posting should not be enabled when this type of memory card is used. Also, memory should not be sized using write operations if write posting is enabled. I/O areas that have holes should not be write posted if software may access non-existent memory. Using the programmable map decoders, write posting can be enabled for “safe” areas and disabled for areas which are not “safe”.

Block transfer is not supported because the MC680x0 block transfer capability is not compatible with the VMEbus.

The *VMEbus master* supports dynamic bus sizing. When a local device initiates a quad-byte access to a VMEbus slave that only has the D16 data transfer capability, the chip executes two double-byte cycles on the VMEbus, acknowledging the local device after all requested four-bytes

have been accessed. This enhances the portability of software because it allows software to run on the system regardless of the physical organization of global memory.

Using the local bus map decoder attribute register, the AM code that the master places on the VMEbus can be programmed under software control.

The VMEchip2 includes a software-controlled VMEbus access timer. It starts ticking when the chip is requested to do a VMEbus data transfer or an interrupt acknowledge cycle. The timer stops ticking once the chip has started the data transfer on the VMEbus. If the data transfer does not begin before the timer times out, the timer drives the local bus error signal, and sets the appropriate status bit in the Local Control and Status Register (LCSR). Using control bits in the LCSR, the timer can be disabled, or it can be enabled to drive the local bus error signal after 64  $\mu$ s, 1 ms, or 32 ms.

The VMEchip2 includes a software-controlled VMEbus write post timer. It starts ticking when a data transfer to the VMEbus is write posted. The timer stops ticking once the chip has started the data transfer on the VMEbus. If this does not happen before the timer times out, the chip aborts the write posted cycle and sends an interrupt to the local bus interrupter. If the write post bus error interrupt is enabled in the local bus interrupter, the local processor is interrupted to indicate a write post time-out has occurred. The write post timer has the same timing as the VMEbus access timer.

### **Local-Bus-to-VMEbus Requester**

The requester provides all the signals necessary to allow the local-bus-to-VMEbus master to request and be granted use of the VMEbus. The chip connects to all signals that a VMEbus requester is required to drive and monitor.

Requiring no external jumpers, the chip provides the means for software to program the requester to request the bus on any one of the four bus request levels, automatically establishing the bus grant daisy-chains for the three inactive levels.

The requester requests the bus if any of the following conditions occur:

1. The local bus master initiates either a data transfer cycle or an interrupt acknowledge cycle to the VMEbus.
2. The chip is requested to acquire control of the VMEbus as signaled by the DWB input signal pin.
3. The chip is requested to acquire control of the VMEbus as signaled by the DWB control bit in the LCSR.

The local-bus-to-VMEbus requester in the VMEchip2 implements a fair mode. By setting the LVFAIR bit, the requester refrains from requesting the VMEbus until it detects its assigned request line in its negated state.

The local-bus-to-VMEbus requester attempts to release the VMEbus when the requested data transfer operation is complete, the DWB pin is negated, the DWB bit in the LCSR is negated and the bus is not being held by a lock cycle. The requester releases the bus as follows:

1. When the chip is configured in release-when-done (RWD) mode, the requester releases the bus when the above conditions are satisfied.
2. When the chip is configured in release-on-request (ROR) mode, the requester releases the bus when the above conditions are satisfied and there is a bus request pending on one of the VMEbus request lines.

To minimize the timing overhead of the arbitration process, the local-bus-to-VMEbus requester in the VMEchip2 executes an early release of the VMEbus. If it is about to release the bus and it is executing a VMEbus cycle, the requester releases BBSY before its associated master completes the cycle. This allows the arbiter to arbitrate any pending requests, and grant the bus to the next requester, at the same time that the active master completes its cycle.

## VMEbus-to-Local-Bus Interface

The VMEbus-to-local-bus interface allows an off-board VMEbus master access to onboard resources. The VMEbus-to-local-bus interface includes the *VMEbus slave*, *write post buffer*, and *local bus master*.

Adhering to the IEEE 1014-87 VMEbus standard, the *slave* can withstand address-only cycles, as well as address pipelining, and respond to unaligned transfers. Using programmable map decoders, it can be configured to provide the following VMEbus capabilities:

Addressing capabilities: A24, A32

Data transfer capabilities: D08(EO), D16, D32, D8/BLT,  
D16/BLT, D32/BLT, D64/BLT  
(BLT = block transfer)

The slave can be programmed to perform *write posting* operations. When in this mode, the chip latches incoming data and addressing information into a staging FIFO and then acknowledges the VMEbus write transfer by asserting DTACK\*. The chip then requests control of the local bus and independently accesses the local resource after it has been granted the local bus. The write-posting pipeline is two deep in non-block transfer mode and 16 deep in block transfer mode.

To significantly improve the access time of the slave when it responds to a VMEbus block read cycle, the VMEchip2 contains a 16 four-byte deep read-ahead pipeline. When responding to a block read cycle, the chip performs block read cycles on the local bus to keep the FIFO buffer full. Data for subsequent transfers is then retrieved from the on-chip buffer, significantly improving the response time of the slave in block transfer mode.

The VMEchip2 includes an on-chip map decoder that allows software to configure the global addressing range of onboard resources. The decoder allows the local address range to be partitioned into two separate banks, each with its own start and end address (in increments of 64KB), as well as setting each bank's address modifier codes, write post enable, and snoop enable.

Each map decoder includes an alternate address register and an alternate address select register. These registers allow any or all of the upper 16 VMEbus address lines to be replaced by signals from the alternate address register. This allows the address of local resources to differ from their VMEbus address.

The alternate address register also provides the upper eight bits of the local address when the VMEbus slave cycle is A24.

The *local bus master* requests the local bus and executes cycles as required. To reduce local bus loading and improve performance it always attempts to transfer data using a burst transfer as defined by the MC680x0.

When snooping is enabled, the local bus master requests the cache controller in the MC680x0 to monitor the local bus addresses.

## Local-Bus-to-VMEbus DMA Controller

The DMA Controller (DMAC) operates in conjunction with the local bus master, the VMEbus master, and a 16 four-byte FIFO buffer. The DMA controller has a 32-bit local address counter, 32-bit table address counter, a 32-bit VMEbus address counter, a 32-bit byte counter, and control and status registers. The Local Control and Status register (LCSR) provides software with the ability to control the operational modes of the DMAC. Software can program the DMAC to transfer up to 4GB of data in the course of a single DMA operation. The DMAC supports transfers from any local bus address to any VMEbus address. The transfers may be from 1 byte to 4GB in length.

To optimize local bus use, the DMAC automatically adjusts the size of individual data transfers until 32-bit transfers can be executed. Based on the address of the first byte, the DMAC transfers a single-byte, a double-byte, or a mixture of both, and then continues to execute quad-byte block transfer cycles. When the DMAC is set for 64-bit transfers, the octal-byte transfers takes place. Based on the address of the last byte, the DMAC transfers a single byte, a double byte, or a mixture of both to end the transfer.



Using control register bits in the LCSR, the DMAC can be configured to provide the following VMEbus capabilities:

Addressing capabilities: A16, A24, A32

Data transfer capabilities: D16, D32, D16/BLT, D32/BLT,  
D64/BLT (BLT = block transfer)

Using the DMA AM (Address Modifier) control register, the address modifier code that the VMEbus DMA controller places on the VMEbus can be programmed under software control. In addition, the DMAC can be programmed to execute block-transfer cycles over the VMEbus.

Complying with the VMEbus specification, the DMAC automatically terminates block-transfer cycles whenever a 256-byte (D32/BLT) or 2-KB (D64/BLT) boundary is crossed. It does so by momentarily releasing AS\* (Address Strobe) and then, in accordance with its bus release/bus request configuration, initiating a new block-transfer cycle.

To optimize VMEbus use, the DMAC automatically adjusts the size of individual data transfers until 64-bit transfers (D64/BLT mode), 32-bit transfers (D32 mode), or 16-bit transfers (D16 mode) can be executed. Based on the address of the first byte, the DMAC transfers single bytes, double bytes, or a mixture of both, and then continues to execute transfer cycles based on the programmed data width. Based on the address of the last byte, the DMAC transfers single bytes, double bytes, or a mixture of both to end the transfer.

To optimize local bus use when the VMEbus is operating in D16 mode, the data FIFO converts D16 VMEbus transfers to D32 local bus transfers. The FIFO also aligns data if the source and destination addresses are not aligned so the local bus and VMEbus can operate at their maximum data transfer sizes.

To allow other boards access to the VMEbus, the DMAC has bus tenure timers to limit the time the DMAC spends on the VMEbus and to ensure a minimum time off the VMEbus. Since the local bus is generally faster than the VMEbus, other local bus masters may use the local bus while the DMAC is waiting for the VMEbus.

The DMAC also supports command chaining through the use of a singly-linked list built in local memory. Each entry in the list includes a VMEbus address, a local bus address, a byte count, a control word, and a pointer to the next entry. When the command chaining mode is enabled, the DMAC reads and executes commands from the list in local memory until all commands are executed.

The DMAC can be programmed to send an interrupt request to the local bus interrupter when any specific table entry has completed. In addition the DMAC always sends an interrupt request at the normal completion of a request or when an error is detected. If the DMAC interrupt is enabled in the DMAC, the local bus is interrupted.

For increased flexibility in managing the bus tenure to optimize bus usage as required by the system configuration, the chip contains control bits that allow the DMAC time on and off the bus to be programmed. Using these control bits, software can instruct the DMA controller to acquire the bus, maintain mastership for a specific amount of time, and then, after relinquishing it, refrain from requesting it for another specific amount of time.

### **No-Address-Increment DMA Transfers**

During normal memory-to-memory DMA transfers, the DMA controller is programmed to increment the local bus and VMEbus address. This allows a block of data to be transferred between VMEbus memory and local bus memory. In some applications, it may be desirable to transfer a block of data from local bus memory to a single VMEbus address. This single VMEbus address may be a FIFO or similar type device which can accept a large amount of data but only appears at single VMEbus address. The DMA controller provides support for these devices by allowing transfers without incrementing the VMEbus address. The DMA controller also allows DMA transfers without incrementing the local bus address, although the MVME1X2P4 has no onboard devices that benefit from not incrementing the local bus address.

The transfer mode on the VMEbus may be D16, D16/BLT, D32, D32/BLT or D64/BLT. When the no-increment address mode is selected, some of the VMEbus address lines and local bus address lines continue to increment in some modes. This is required to support the various port sizes

and to allow transfers which are not an even byte count or which start at an odd address, with respect to the port size. A 16-bit device should respond with VA<1> high or low. Devices on the local bus should respond to any combination of LA<3..2>. This is required to support the burst mode on the MC680x0 bus.

Normally when the non-increment mode is used, the starting address and byte count would be aligned to the port size. For example, a DMA transfer to a 16-bit FIFO would start on a 16-bit boundary and would have an even number of 16-bit transfers. If the starting address is not aligned or the byte count is odd, the DMA controller will increment the lower address lines. This is required because the lower order address lines are used to define the size of the transfer and the byte lanes.

The VMEbus uses VA<2..1>, LWORD\*, and DS<1..0>\* to define the transfer size and byte lanes. If the VMEbus port size is D32, then VA<1>, LWORD\* and DS<1..0>\* are used to define the transfer size and byte lanes. During D16 transfers, the VMEbus address line VA<1> toggles. If the VMEbus port size is D64, then VA<2..1>, LWORD\* and DS<1..0>\* are used to define the transfer size and byte lanes. Local bus address LA<3..0> and SIZ<1..0> are used to define the transfer size and byte lanes on local bus. During local bus transfers, LA<3..2> count.

The DMA controller internally increments the VMEbus address counter and if the transfer mode is BLT, the DMA controller generates a new address strobe (AS\*) when a block boundary is crossed.

## DMAC VMEbus Requester

The chip contains an independent VMEbus requester associated with the DMA Controller. This allows flexibility in instituting different bus tenure policies for the single-transfer-oriented master and the block-transfer-oriented DMA controller. The DMAC requester provides all the signals necessary to allow the on-chip DMA Controller to request and be granted use of the VMEbus.

Requiring no external jumpers, the chip provides the means for software to program the DMAC requester to request the bus on any one of the four bus request levels, automatically establishing the bus grant daisy-chains for the three inactive levels.

The DMAC requester requests the bus as required to transfer data to or from the FIFO buffer.

The requester implements a fair mode. By setting the DFAIR bit, the requester refrains from requesting the bus until it detects its assigned request line in its negated state.

The requester releases the bus when requested to by the DMA controller. The DMAC always releases the VMEbus when the FIFO is full (VMEbus to local bus) or empty (local bus to VMEbus). The DMAC can also be programmed to release the VMEbus when another VMEbus master requests the bus, when the time on timer has expired, or when the time on timer has expired and another VMEbus master is requesting the bus. To minimize the timing overhead of the arbitration process, the DMAC requester executes an early release of the bus. If it is about to release the bus and it is executing a VMEbus cycle, the requester releases BBSY before its associated VMEbus master completes the cycle. This allows the arbiter to arbitrate any pending requests, and grant the bus to the next requester, at the same time that the DMAC completes its cycle.

## Tick and Watchdog Timers

The VMEchip2 has two 32-bit tick timers and a watchdog timer. The tick timers run on a 1MHz clock which is derived from the local bus clock by the prescaler.

### Prescaler

The prescaler is used to derive the various clocks required by the tick timers, VME access timers, reset timer, bus arbitration timer, local bus timer, and VMEbus timer. The prescaler divides the local bus clock to produce the constant-frequency clocks required. Software is required to load the appropriate constant, depending upon the local bus clock, following reset to ensure proper operation of the prescaler.

## Tick Timers

The VMEchip2 includes two general-purpose tick timers. These timers can be used to generate interrupts at various rates or the counters can be read at various times for interval timing. The timers have a resolution of 1  $\mu$ s. When free running, they roll over every 71.6 minutes.

Each tick timer has a 32-bit counter, a 32-bit compare register, a 4-bit overflow register, an enable bit, an overflow clear bit, and a clear-on-compare enable bit. The counter is readable and writable at any time. When enabled in free-run mode, it increments every 1 $\mu$ s. When the counter is enabled in clear-on-compare mode, it increments every 1 $\mu$ s until the counter value matches the value in the compare register. When a match occurs, the counter is cleared. When a match occurs, in either mode, an interrupt is sent to the local bus interrupter and the overflow counter is incremented. An interrupt to the local bus is only generated if the tick timer interrupt is enabled by the local bus interrupter. The overflow counter can be cleared by writing a 1 to the overflow clear bit.

Tick timer 1 or 2 can be programmed to generate a pulse on the VMEbus IRQ1 interrupt line at the tick timer period. This provides a broadcast interrupt function which allows several VME boards to receive an interrupt at the same time. In certain applications, this interrupt can be used to synchronize multiple processors. This interrupt is not acknowledged on the VMEbus. This mode is intended for specific applications and is not defined in the VMEbus specification.

## Watchdog Timer

The watchdog timer has a 4-bit counter, four clock select bits, an enable bit, a local reset enable bit, a SYSRESET enable bit, a board fail enable bit, counter reset bit, WDTO status bit, and WDTO status reset bit.

When enabled, the counter increments at a rate determined by the clock select bits. If the counter is not reset by software, the counter reaches its terminal count. When this occurs, the WDTO status bit is set; and if the local or SYSRESET function is enabled, the selected reset is generated; if the board fail function is enabled, the board fail signal is generated.

## VMEbus Interrupter

The interrupter provides all the signals necessary to allow software to request interrupt service from a VMEbus interrupt handler. The chip connects to all signals that a VMEbus interrupter is required to drive and monitor.

Requiring no external jumpers, the chip provides the means for software to program the interrupter to request an interrupt on any one of the seven interrupt request lines. In addition, the chip controls the propagation of the acknowledge on the IACK daisy-chain.

The interrupter operates in release-on-acknowledge (ROAK) mode. An 8-bit control register provides software with the means to dynamically program the status/ID information. Upon reset, this register is initialized to a status/ID of \$0F (the uninitialized vector in the 68K-based environment).

The VMEbus interrupter has an additional feature not defined in the VMEbus specification. The VMEchip2 supports a broadcast mode on the IRQ1 signal line. When this feature is used, the normal IRQ1 interrupt to the local bus interrupter should be disabled and the edge-sensitive IRQ1 interrupt to the local bus interrupter should be enabled. All boards in the system which are not participating in the broadcast interrupt function should not drive or respond to any signals on the IRQ1 signal line.

There are two ways to broadcast an IRQ1 interrupt. The VMEbus interrupter in the VMEchip2 may be programmed to generate a level 1 interrupt. This interrupt must be cleared using the interrupt clear bit in the control register because the interrupt is never acknowledged on the VMEbus. The VMEchip2 allows the output of one of the tick timers to be connected to the IRQ1 interrupt signal line on the VMEbus. When this function is enabled, a pulse appears on the IRQ1 signal line at the programmed interrupt rate of the tick timer.

## VMEbus System Controller

With the exception of the optional SERCLK driver and the Power Monitor, the chip includes all the functions that a VMEbus system controller must provide. The system controller is enabled/disabled with the aid of an external jumper (J1), the only jumper required in a VMEchip2-based VMEbus interface.

### Arbiter

The arbitration algorithm used by the chip arbiter is selected by software. All three arbitration modes defined in the VMEbus Specification are supported: Priority (PRI), Round-Robin-Select (RRS), as well as Single (SGL). When operating in the PRI mode, the arbiter asserts the BCLR line whenever it detects a request for the bus whose level is higher than the one being serviced.

The chip includes an arbitration timer, preventing a bus lockup when no requester assumes control of the bus after the arbiter has issued a grant. Using a control bit, this timer can be enabled or disabled. When enabled, it assumes control of the bus by driving the BBSY signal after 256  $\mu$ secs, releasing it after satisfying the requirements of the VMEbus specification, and then re-arbitrating any pending bus requests.

### IACK Daisy-Chain Driver

Complying with the latest revision of the VMEbus specification, the System Controller includes an IACK Daisy-Chain Driver, ensuring that the timing requirements of the IACK daisy-chain are satisfied.

### Bus Timer

The Bus Timer is enabled/disabled by software to terminate a VMEbus cycle by asserting BERR\* if any of the VMEbus data strobes is maintained in its asserted state for longer than the programmed time-out period. The time-out period can be set to 8, 64, or 256 secs. The bus timer terminates an unresponded VMEbus cycle only if both it and the system controller are enabled.

In addition to the VMEbus timer, the chip contains a local bus timer. This timer asserts the local TEA when the local bus cycle maintained in its asserted state for longer than the programmed time-out period. This timer can be enabled or disabled under software control. The time-out period can be programmed for 8, 64, or 256 seconds.

## Reset Driver

The chip includes both a global and a local reset driver. When the chip operates as the VMEbus system controller, the reset driver provides a global system reset by asserting the VMEbus signal **SYSRESET\***. A **SYSRESET\*** may be generated by the **RESET** switch, a power-up reset, a watch dog timeout, or by a control bit in the **LCSR**. **SYSRESET\*** remains asserted for at least 200 msec, as required by the VMEbus specification.

Similarly, the chip provides an input signal and a control bit to initiate a local reset operation.

The local reset driver is enabled even when the chip is not the system controller. A local reset may be generated by the **RESET** switch, a power up reset, a watch dog time-out, a VMEbus **SYSRESET\*** signal, or a control bit in the **GCSR**.

## Local Bus Interrupter and Interrupt Handler

There are 31 interrupt sources in the VMEchip2 ASIC:

VMEbus ACFAIL interrupter	Tick timer 2-1
<b>ABORT</b> switch	DMAC done
VMEbus SYSFAIL interrupter	GCSR SIG3-0
Write post bus error	GCSR location monitor 1-0
External input	Software interrupts 7-0
VMEbus IRQ1 edge-sensitive interrupter	VMEbus IRQ7-1 interrupts
VMEchip2 VMEbus interrupter acknowledge	



Each of the 31 interrupts can be enabled to generate a local bus interrupt at any level. For example, VMEbus IRQ5 can be programmed to generate a level 2 local bus interrupt.

The VMEbus AC fail interrupter is an edge-sensitive interrupter connected to the VMEbus ACFAIL\* signal line. This interrupter is filtered to remove the ACFAIL\* glitch which is related to the BBSY\* glitch.

The SYS fail interrupter is an edge-sensitive interrupter connected to the VMEbus SYSFAIL\* signal line.

The write post bus error interrupter is an edge-sensitive interrupter connected to the local-bus-to-VMEbus write post bus error signal line.

The VMEbus IRQ1 edge-sensitive interrupter is an edge-sensitive interrupter connected to the VMEbus IRQ1\* signal line. This interrupter is used when one of the tick timers is connected to the IRQ1\* signal line. When this interrupt is acknowledged, the vector is provided by the VMEchip2 and a VMEbus interrupt acknowledge is not generated. When this interrupt is enabled, the VMEbus IRQ1 level-sensitive interrupter should be disabled.

The VMEchip2 VMEbus interrupter acknowledge interrupter is an edge-sensitive interrupter connected to the acknowledge output of the VMEbus interrupter. An interrupt is generated when an interrupt on the VMEbus from VMEchip2 is acknowledged by a VMEbus interrupt handler.

The tick timer interrupters are edge-sensitive interrupters connected to the output of the tick timers.

The DMAC interrupter is an edge-sensitive interrupter connected to the DMAC.

The GCSR SIG3-0 interrupters are edge-sensitive interrupters connected to the output of the signal bits in the GCSR.

The location monitor interrupters are edge-sensitive interrupters connected to the location monitor bits in the GCSR.

The software 7-0 interrupters can be set by software to generate interrupts.

The VMEbus IRQ7-1 interrupters are level-sensitive interrupters connected to the VMEbus IRQ7\*-IRQ1\* signal lines.

The interrupt handler provides all logic necessary to identify and handle all local interrupts as well as VMEbus interrupts. When a local interrupt is acknowledged, a unique vector is provided by the chip. Edge-sensitive interrupters are not cleared during the interrupt acknowledge cycle and must be reset by software as required. If the interrupt source is the VMEbus, the interrupt handler instructs the VMEbus master to execute a VMEbus IACK cycle to obtain the vector from the VMEbus interrupter. The chip connects to all signals that a VMEbus handler is required to drive and monitor. On the local bus, the interrupt handler is designed to comply with the interrupt handling signaling protocol of the MC680x0 microprocessor.

## Global Control and Status Registers

The VMEchip2 ASIC includes a set of registers that are accessible from both the VMEbus and the local bus. These registers are provided to aid in interprocessor communications over the VMEbus. These registers are fully described in a later section.

## LCSR Programming Model

This section defines the programming model for the Local Control and Status registers (LCSR) in the VMEchip2 ASIC. The local bus map decoder for the LCSR is included in the VMEchip2. The base address of the LCSR is \$FFF40000 and the registers are 32 bits wide. Single-byte, double-byte, and quad-byte *read* operations are permitted, but single-byte and double-byte *write* operations are not. Single- and double-byte write operations return a TEA signal to the local bus. Read-modify-write operations should be used to modify a byte or a two-byte of a register.

Each register definition includes a table with five lines:

1. The base address of the register and the number of bits defined in the table.
2. The bits defined by this table.
3. The name of the register or the name of the bits in the register.
4. The operations possible on the register bits, defined as follows:

<b>R</b>	This bit is a read-only status bit.
<b>R/W</b>	This bit is readable and writable.
<b>W/AC</b>	This bit can be set and it is automatically cleared. This bit can also be read.
<b>C</b>	Writing a 1 to this bit clears this bit or another bit. This bit reads 0.
<b>S</b>	Writing a 1 to this bit sets this bit or another bit. This bit reads 0.

5. The state of the bit following a reset, defined as follows:

<b>P</b>	The bit is affected by powerup reset.
<b>S</b>	The bit is affected by SYSRESET.
<b>L</b>	The bit is affected by local reset.
<b>X</b>	The bit is not affected by reset.

Table 2-2 shows a summary of the LCSRs.



15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
SLAVE STARTING ADDRESS 1																
SLAVE STARTING ADDRESS 2																
SLAVE ADDRESS TRANSLATION SELECT 1																
SLAVE ADDRESS TRANSLATION SELECT 2																
X				ADDER 1	SNP 1	WP 1	SUP 1	USR 1	A32 1	A24 1	BLK D64 1	BLK 1	PRGM 1	DATA 1		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
MASTER STARTING ADDRESS 1																
MASTER STARTING ADDRESS 2																
MASTER STARTING ADDRESS 3																
MASTER STARTING ADDRESS 4																
MASTER ADDRESS TRANSLATION SELECT 4																
MAST D16 EN	MAST WP EN	MASTER AM 2						MAST D16 EN	MAST WP EN	MASTER AM 1						
IO2 EN	IO2 WP EN	IO2 S/U	IO2 P/D	IO1 EN	IO1 D16 EN	IO1 WP EN	IO1 S/U	ROM SIZE	ROM BANK B SPEED			ROM BANK A SPEED				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
ARB ROBN	MAST DHB	MAST DWB	X	MST FAIR	MST RWD	MASTER VMEBUS		DMA HALT	DMA EN	DMA TBL	DMA FAIR	DM RELM		DMA VMEBUS		
DMA TBL INT	DMA LB SNP MODE		X	DMA INC VME	DMA INC LB	DMA WRT	DMA D16	DMA D64 BLK	DMA BLK	DMA AM 5	DMA AM 4	DMA AM 3	DMA AM 2	DMA AM 1	DMA AM 0	
LOCAL BUS ADDRESS COUNTER																
VMEBUS ADDRESS COUNTER																
BYTE COUNTER																
TABLE ADDRESS COUNTER																
DMA TABLE INTERRUPT COUNT				MPU CLR STAT	MPU LBE ERR	MPU LPE ERR	MPU LOB ERR	MPU LTO ERR	DMA LBE ERR	DMA LPE ERR	DMA LOB ERR	DMA LTO ERR	DMA TBL ERR	DMA VME ERR	DMA DONE	

1360 9403

← This sheet begins on facing page.

**Table 2-2. VMEchip2 Memory Map—LCSR Summary (Sheet 2 of 2)**

**VMEchip2 LCSR Base Address = \$FFF40000  
OFFSET:**

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
4C	X							ARB BGTO EN	DMA TIME OFF			DMA TIME ON			VME GLOBAL TIMER	
50	TICK TIMER 1															
54	TICK TIMER 1															
58	TICK TIMER 2															
5C	TICK TIMER 2															
60	X	SCON	SYS FAIL	BRD FAIL STAT	PURS STAT	CLR PURS STAT	BRD FAIL OUT	RST SW EN	SYS RST	WD CLR TO	WD CLR CNT	WD TO STAT	TO BF EN	WD SRST LRST	WD RST EN	WD EN
64	PRE															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
68	AC FAIL IRQ	AB IRQ	SYS FAIL IRQ	MWP BERR IRQ	PE IRQ	IRQ1E IRQ	TIC2 IRQ	TIC1 IRQ	VME IACK IRQ	DMA IRQ	SIG3 IRQ	SIG2 IRQ	SIG1 IRQ	SIG0 IRQ	LM1 IRQ	LM0 IRQ
6C	EN IRQ 31	EN IRQ 30	EN IRQ 29	EN IRQ 28	EN IRQ 27	EN IRQ 26	EN IRQ 25	EN IRQ 24	EN IRQ 23	EN IRQ 22	EN IRQ 21	EN IRQ 20	EN IRQ 19	EN IRQ 18	EN IRQ 17	EN IRQ 16
70	X															
74	CLR IRQ 31	CLR IRQ 30	CLR IRQ 29	CLR IRQ 28	CLR IRQ 27	CLR IRQ 26	CLR IRQ 25	CLR IRQ 24	CLR IRQ 23	CLR IRQ 22	CLR IRQ 21	CLR IRQ 20	CLR IRQ 19	CLR IRQ 18	CLR IRQ 17	CLR IRQ 16
78	X	AC FAIL IRQ LEVEL			X	ABORT IRQ LEVEL			X	SYS FAIL IRQ LEVEL			X	MST WP ERROR IRQ LEVEL		
7C	X	VME IACK IRQ LEVEL			X	DMA IRQ LEVEL			X	SIG 3 IRQ LEVEL			X	SIG 2 IRQ LEVEL		
80	X	SW7 IRQ LEVEL			X	SW6 IRQ LEVEL			X	SW5 IRQ LEVEL			X	SW4 IRQ LEVEL		
84	X	SPARE IRQ LEVEL			X	VME IRQ 7 IRQ LEVEL			X	VME IRQ 6 IRQ LEVEL			X	VME IRQ 5 IRQ LEVEL		
88	VECTOR BASE REGISTER 0				VECTOR BASE REGISTER 1				MST IRQ EN	SYS FAIL LEVEL	AC FAIL LEVEL	ABORT LEVEL	GPIOEN			
8C	X															

This sheet continues on facing page. →

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
VME ACCESS TIMER		LOCAL BUS TIMER		WD TIME OUT SELECT				PRESCALER CLOCK ADJUST									
COMPARE REGISTER																	
COUNTER																	
COMPARE REGISTER																	
COUNTER																	
OVERFLOW COUNTER 2				X		CLR OVF 2	COC EN 2	TIC EN 2	OVERFLOW COUNTER 1				X		CLR OVF 1	COC EN 1	TIC EN 1
SCALER																	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
SW7 IRQ	SW6 IRQ	SW5 IRQ	SW4 IRQ	SW3 IRQ	SW2 IRQ	SW1 IRQ	SW0 IRQ	SPARE	VME IRQ7	VME IRQ6	VME IRQ5	VME IRQ4	VME IRQ3	VME IRQ2	VME IRQ1		
EN IRQ 15	EN IRQ 14	EN IRQ 13	EN IRQ 12	EN IRQ 11	EN IRQ 10	EN IRQ 9	EN IRQ 8	EN IRQ 7	EN IRQ 6	EN IRQ 5	EN IRQ 4	EN IRQ 3	EN IRQ 2	EN IRQ 1	EN IRQ 0		
SET IRQ 15	SET IRQ 14	SET IRQ 13	SET IRQ 12	SET IRQ 11	SET IRQ 10	SET IRQ 9	SET IRQ 8	X									
CLR IRQ 15	CLR IRQ 14	CLR IRQ 13	CLR IRQ 12	CLR IRQ 11	CLR IRQ 10	CLR IRQ 9	CLR IRQ 8										
X		P ERROR IRQ LEVEL		X		IRQ1E IRQ LEVEL		X		TIC TIMER 2 IRQ LEVEL		X		TIC TIMER 1 IRQ LEVEL			
X		SIG 1 IRQ LEVEL		X		SIG 0 IRQ LEVEL		X		LM 1 IRQ LEVEL		X		LM 0 IRQ LEVEL			
X		SW3 IRQ LEVEL		X		SW2 IRQ LEVEL		X		SW1 IRQ LEVEL		X		SW0 IRQ LEVEL			
X		VME IRQ 4 IRQ LEVEL		X		VMEB IRQ 3 IRQ LEVEL		X		VME IRQ 2 IRQ LEVEL		X		VME IRQ 1 IRQ LEVEL			
GPIOO				GPIOI				GPI									
MP IRQ EN		REV EROM		DIS SRAM		DIS MST		NO EL BBSY		DIS BSYT		EN INT		DIS BGN			

1361 9403

← This sheet begins on facing page.

## Programming the VMEbus Slave Map Decoders

This section includes programming information for the VMEbus-to-local-bus map decoders.

The VMEbus-to-local-bus interface allows off-board VMEbus masters access to local on-board resources. The address of the local resources as viewed from the VMEbus is controlled by the VMEbus slave map decoders, which are part of the VMEbus-to-local-bus interface. Two VMEbus slave map decoders in the VMEchip2 allow two segments of the VMEbus to be mapped to the local bus. A segment may vary in size from 64KB to 4GB in increments of 64KB. Address translation is provided by the address translation registers. The upper 16 bits of the local bus address come from the address translation address register rather than from the upper 16 bits of the VMEbus address.

Each VMEbus slave map decoder has the following registers: *address translation address register*, *address translation select register*, *starting address register*, *ending address register*, *address modifier select register*, and *attribute register*. The tables on the following pages list the addresses and bit definitions of these registers.

The VMEbus slave map decoders described in this section are disabled by local reset, SYSRESET\*, or power-up reset. Use caution when enabling the map decoders or when modifying their registers after they are enabled. The safest time to enable or modify the map decoder registers is when the VMEchip2 is VMEbus master.

To modify the map decoder registers, follow this procedure: Set the DWB bit in the LCSR and then wait for the DHB bit in the LCSR to be set, indicating that VMEbus mastership has been acquired. The map decoder registers can then be modified and the VMEbus released by clearing the DWB bit in the LCSR. Because the VMEbus is held during this programming operation, the registers should be programmed quickly with interrupts disabled.

The VMEbus slave map decoders can be programmed, without obtaining VMEbus mastership, if they are disabled and the following procedure is followed: The address translation registers and starting and ending address registers should be programmed first, and then the map decoders should be enabled by programming the address modifier select registers.



You program a VMEbus slave map decoder by loading the starting address of the segment into the *starting address register* and the ending address of the segment into the *ending address register*. If the VMEbus address modifier codes indicate an A24 VMEbus address cycle, then the upper eight bits of the VMEbus address are forced to 0 before the compare. The address modifier select register should be programmed for the required address modifier codes. A VMEbus slave map decoder is disabled when the address modifier select register is cleared.

The *address translation registers* allow local resources to have different VMEbus and local bus addresses. Only address bits A31 through A16 may be modified.

The *address translation registers* also provide the upper eight local bus address lines when an A24 VMEbus cycle is used to access a local resource. The address translation register should be programmed with the translated address and the address translation select register should be programmed to enable the translated address. If address translation is not desired, then the address translation registers should be programmed to 0.

The *address translation address register* and the *address translation select register* operate in the following way: If you set a bit in the address translation select register, then the corresponding bit in the address translation address register drives the appropriate local bus address line. If you clear the bit in the address translation select register, then the corresponding VMEbus address line drives the appropriate local bus address line. The most significant bit of the address translation select register corresponds to the most significant bit of address translation register and to A32 of the local bus and A32 of the VMEbus.

In addition to the address translation method previously described, the VMEchip2 as used on the MVME1X2P4 includes an adder which can be used for address translation. When the adder is enabled, the local bus address is generated by adding the offset value to the VMEbus address lines VA<31..16>. The offset is the value in the address translation/offset register. If the VMEbus transfer is A24, then the VMEbus address lines VA<31..24> are forced to 0 before the addition. The adders are enabled by setting bit 11 for map decoder 1 and bit 27 for map decoder 2 in register

\$FFF40010. The adders allow any size board to be mapped on any 64KB boundary. The adders are disabled and the address replacement method is used following reset.

Write posting is enabled for the segment by setting the write post enable bit in the *attribute register*. Local bus snooping for the segment is enabled by setting the snoop bits in the attribute register. The snoop bits in the attribute register are driven on to the local bus when the VMEbus-to-local-bus interface is local bus master.

### VMEbus Slave Ending Address Register 1

ADR/SIZ	\$FFF40000 (16 bits of 32)		
BIT	31	...	16
NAME	Ending Address Register 1		
OPER	R/W		
RESET	0 PS		

This register is the ending address register for the first VMEbus-to-local-bus map decoder.

### VMEbus Slave Starting Address Register 1

ADR/SIZ	\$FFF40000 (16 bits of 32)		
BIT	15	...	0
NAME	Starting Address Register 1		
OPER	R/W		
RESET	0 PS		

This register is the starting address register for the first VMEbus-to-local-bus map decoder.

**VMEbus Slave Ending Address Register 2**

<b>ADR/SIZ</b>	<b>\$FFF40004 (16 bits of 32)</b>		
<b>BIT</b>	31	...	16
<b>NAME</b>	Ending Address Register 2		
<b>OPER</b>	R/W		
<b>RESET</b>	0 PS		

This register is the ending address register for the second VMEbus-to-local-bus map decoder.

**VMEbus Slave Starting Address Register 2**

<b>ADR/SIZ</b>	<b>\$FFF40004 (16 bits of 32)</b>		
<b>BIT</b>	15	...	0
<b>NAME</b>	Starting Address Register 2		
<b>OPER</b>	R/W		
<b>RESET</b>	0 PS		

This register is the starting address register for the second VMEbus-to-local-bus map decoder.

**VMEbus Slave Address Translation Address Offset Register 1**

<b>ADR/SIZ</b>	<b>\$FFF40008 (16 bits of 32)</b>		
<b>BIT</b>	31	...	16
<b>NAME</b>	Address Translation Address Offset Register 1		
<b>OPER</b>	R/W		
<b>RESET</b>	0 PS		

This register is the address translation address register for the first VMEbus-to-local-bus map decoder. It should be programmed to the local bus starting address. When the adder is engaged, this register is the offset value.

## 2 VMEbus Slave Address Translation Select Register 1

ADR/SIZ	\$FFF40008 (16 bits of 32)		
BIT	15	...	0
NAME	Address Translation Select Register 1		
OPER	R/W		
RESET	0 PS		

This register is the address translation select register for the first VMEbus-to-local-bus map decoder. The address translation select register value is based on the segment size (the difference between the VMEbus starting and ending addresses).

If the segment size is between the sizes shown in the table below, assume the larger size.

Segment Size	Address Translation Select Value
64KB	FFFF
128KB	FFFE
256KB	FFFC
512KB	FFF8
1MB	FFF0
2MB	FFE0
4MB	FFC0
8MB	FF80
16MB	FF00

Segment Size	Address Translation Select Value
32MB	FE00
64MB	FC00
128MB	F800
256MB	F000
512MB	E000
1GB	C000
2GB	8000
4GB	0000

**VMEbus Slave Address Translation Address Offset Register 2**

<b>ADR/SIZ</b>	<b>\$FFF4000C (16 bits of 32)</b>		
<b>BIT</b>	31	...	16
<b>NAME</b>	Address Translation Address Offset Register 2		
<b>OPER</b>	R/W		
<b>RESET</b>	0 PS		

This register is the address translation address register for the second VMEbus-to-local-bus map decoder. It should be programmed to the local bus starting address. When the adder is enabled, this register is the offset value.

**VMEbus Slave Address Translation Select Register 2**

<b>ADR/SIZ</b>	<b>\$FFF4000C (16 bits of 32)</b>		
<b>BIT</b>	15	...	0
<b>NAME</b>	Address Translation Select Register 2		
<b>OPER</b>	R/W		
<b>RESET</b>	0 PS		

This register is the address translation select register for the second VMEbus-to-local-bus map decoder. The address translation select register value is based on the segment size (the difference between the VMEbus starting and ending addresses). If the segment size is between the sizes shown in the table below, assume the larger size.

Segment Size	Address Translation Select Value
64KB	FFFF
128KB	FFFE
256KB	FFFC
512KB	FFF8
1MB	FFF0
2MB	FFE0
4MB	FFC0
8MB	FF80
16MB	FF00

Segment Size	Address Translation Select Value
32MB	FE00
64MB	FC00
128MB	F800
256MB	F000
512MB	E000
1GB	C000
2GB	8000
4GB	0000

## 2 VMEbus Slave Write Post / Snoop Control Register 2

ADR/SIZ	\$FFF40010 (8 bits [4 used] of 32)							
BIT	31	30	29	28	27	26	25	24
NAME					ADDER 2	SNP2		WP2
OPER					R/W	R/W		R/W
RESET					0 PS	0 PS		0 PS

This register is the slave write post and snoop control register for the second VMEbus-to-local-bus map decoder.

**WP2** When this bit is high, write posting is enabled for the address range defined by the second VMEbus slave map decoder. When this bit is low, write posting is disabled for the address range defined by the second VMEbus slave map decoder.

**SNP2** These bits control the snoop enable lines to the local bus for the address range defined by the second VMEbus slave map decoder. The snooping functions differ according to processor type, as shown:

SNP2		Requested Snoop Operation	
26	25	MC68040	MC68060
0	0	Snoop disabled	Snoop enabled
0	1	Source dirty, sink byte/word/longword	Snoop disabled
1	0	Source dirty, invalidate line	Snoop enabled
1	1	Snoop disabled (Reserved)	Snoop disabled

**ADDER2** When this bit is high, the adder is used for address translation. When this bit is low, the adder is not used for address translation.

## VMEbus Slave Address Modifier Select Register 2

ADR/SIZ	\$FFF40010 (8 bits of 32)							
<b>BIT</b>	23	22	21	20	19	18	17	16
<b>NAME</b>	SUP	USR	A32	A24	D64	BLK	PGM	DAT
<b>OPER</b>	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
<b>RESET</b>	0 PSL	0 PSL	0 PSL	0 PSL	0 PSL	0 PSL	0 PSL	0 PSL

This register is the address modifier select register for the second VMEbus-to-local-bus map decoder. There are three groups of address modifier select bits: DAT, PGM, BLK and D64; A24 and A32; and USR and SUP. At least one bit must be set from each group to enable the map decoder.

**DAT** When this bit is high, the second map decoder responds to VMEbus data access cycles. When this bit is low, the second map decoder does not respond to VMEbus data access cycles.

**PGM** When this bit is high, the second map decoder responds to VMEbus program access cycles. When this bit is low, the second map decoder does not respond to VMEbus program access cycles.

**BLK** When this bit is high, the second map decoder responds to VMEbus block access cycles. When this bit is low, the second map decoder does not respond to VMEbus block access cycles.

**D64** When this bit is high, the second map decoder responds to VMEbus D64 block access cycles. When this bit is low, the second map decoder does not respond to VMEbus D64 block access cycles.

- A24** When this bit is high, the second map decoder responds to VMEbus A24 (standard) access cycles. When this bit is low, the second map decoder does not respond to VMEbus A24 access cycles.
- A32** When this bit is high, the second map decoder responds to VMEbus A32 (extended) access cycles. When this bit is low, the second map decoder does not respond to VMEbus A32 access cycles.
- USR** When this bit is high, the second map decoder responds to VMEbus user (non-privileged) access cycles. When this bit is low, the second map decoder does not responded to VMEbus user access cycles.
- SUP** When this bit is high, the second map decoder responds to VMEbus supervisory access cycles. When this bit is low, the second map decoder does not respond to VMEbus supervisory access cycles.



## VMEbus Slave Write Post and Snoop Control Register 1

ADR/SIZ	\$FFF40010 (8 bits [4 used] of 32)							
BIT	15	14	13	12	11	10	9	8
NAME					ADDER 1	SNP1		WP1
OPER					R/W	R/W		R/W
RESET					0 PS	0 PS		0 PS

This register is the slave write post and snoop control register for the first VMEbus-to-local-bus map decoder.

**WP1** When this bit is high, write posting is enabled for the address range defined by the first VMEbus slave map decoder. When this bit is low, write posting is disabled for the address range defined by the first VMEbus slave map decoder.

**SNP1** These bits control the snoop enable lines to the local bus for the address range defined by the first VMEbus slave map decoder. The snooping functions differ according to processor type, as shown:

SNP1		Requested Snoop Operation	
10	9	MC68040	MC68060
0	0	Snoop disabled	Snoop enabled
0	1	Source dirty, sink byte/word/longword	Snoop disabled
1	0	Source dirty, invalidate line	Snoop enabled
1	1	Snoop disabled (Reserved)	Snoop disabled

**ADDER1** When this bit is high, the adder is used for address translation. When this bit is low, the adder is not used for address translation.

## VMEbus Slave Address Modifier Select Register 1

ADR/SIZ	\$FFF40010 (8 bits of 32)							
BIT	7	6	5	4	3	2	1	0
NAME	SUP	USR	A32	A24	D64	BLK	PGM	DAT
OPER	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RESET	0 PSL	0 PSL	0 PSL	0 PSL	0 PSL	0 PSL	0 PSL	0 PSL

This register is the address modifier select register for the first VMEbus-to-local-bus map decoder. There are three groups of address modifier select bits: DAT, PGM, BLK and D64; A24 and A32; and USR and SUP. At least one bit must be set from each group to enable the first map decoder.

**DAT** When this bit is high, the first map decoder responds to VMEbus data access cycles. When this bit is low, the first map decoder does not respond to VMEbus data access cycles.

**PGM** When this bit is high, the first map decoder responds to VMEbus program access cycles. When this bit is low, the first map decoder does not respond to VMEbus program access cycles.

**BLK** When this bit is high, the first map decoder responds to VMEbus block access cycles. When this bit is low, the first map decoder does not respond to VMEbus block access cycles.

**D64** When this bit is high, the first map decoder responds to VMEbus D64 block access cycles. When this bit is low, the first map decoder does not respond to VMEbus D64 block access cycles.

**A24** When this bit is high, the first map decoder responds to VMEbus A24 (standard) access cycles. When this bit is low, the first map decoder does not respond to VMEbus A24 access cycles.

- A32** When this bit is high, the first map decoder responds to VMEbus A32 (extended) access cycles. When this bit is low, the first map decoder does not respond to VMEbus A32 access cycles.
- USR** When this bit is high, the first map decoder responds to VMEbus user (non-privileged) access cycles. When this bit is low, the first map decoder does not respond to VMEbus user access cycles.
- SUP** When this bit is high, the first map decoder responds to VMEbus supervisory access cycles. When this bit is low, the first map decoder does not respond to VMEbus supervisory access cycles.

## Programming the Local-Bus-to-VMEbus Map Decoders

This section includes programming information on the local-bus-to-VMEbus map decoders and the GCSR base address registers.

The local-bus-to-VMEbus interface allows onboard local bus masters access to off-board VMEbus resources. The address of the VMEbus resources as viewed from the local bus is controlled by the local bus slave map decoders, which are part of the local-bus-to-VMEbus interface. Four of the six local-bus-to-VMEbus map decoders are programmable, while the two I/O map decoders are fixed. The first I/O map decoder provides an A16/D16 or A16/D32 space at \$FFFF0000 to \$FFFFFFF which is the VMEbus short I/O space. The second I/O map decoder provides an A24/D16 space at \$F000000 to \$F0FFFFFF and an A32/D16 space at \$F1000000 to \$FF7FFFFF.

A programmable segment may vary in size from 64KB to 4GB in increments of 64KB. Address translation for the fourth segment is provided by the address translation registers which allow the upper 16 bits of the VMEbus address to be provided by the address translation address register rather than the upper 16 bits of the local bus.

Each of the four programmable local bus map decoders has a *starting address*, an *ending address*, an *address modifier register* with attribute bits, and an enable bit. The fourth decoder also has *address translation registers*. The addresses and bit definitions for these registers are in the tables below.

A local bus slave map decoder is programmed by loading the starting address of the segment into the *starting address register* and the ending address of the segment into the *ending address register*. The address modifier code is programmed into the *address modifier register*. Because the local-bus-to-VMEbus interface does not support VMEbus block transfers, block transfer address modifier codes should not be programmed.

The *address translation register* allows a local bus master to view a portion of the VMEbus that may be hidden by onboard resources or an area of the VMEbus may be mapped to two local address. For example, some devices in the I/O map may support write posting while others do not. The VMEbus area in question may be mapped to two local bus addresses, one with write posting enabled and one with write posting disabled. The address translation registers allow local bus address bits A31 through A16 to be modified. The address translation register should be programmed with the translated address, and the address translation select register should be programmed to enable the translated address. If address translation is not desired, then the address translation registers should be programmed to 0.

The *address translation address register* and the *address translation select register* operate in the following way: If you set a bit in the address translation select register, then the corresponding bit in the address translation address register drives the appropriate VMEbus address line. If you clear the bit in the address translation select register, then the corresponding local bus address line drives the appropriate VMEbus address line. The most significant bit of the address translation select register corresponds to the most significant bit of the address translation address register and to A32 of the local bus and A32 of the VMEbus.

Write posting is enabled for the segment by setting the write post enable bit in the address modifier register. D16 transfers are forced by setting the D16 bit in the address modifier register. A segment is enabled by setting the enable bit. Segments should not be programmed to overlap.

The first I/O map decoder maps the local bus address range \$FFFF0000 to \$FFFFFFFF to the A16 (short I/O) map of the VMEbus. This segment may be enabled using the enable bit. Write posting may be enabled for this segment using the write post enable bit. The transfer size may be D16 or D32 as defined by the D16 bit in the control register.

The second I/O map decoder provides support for the other I/O map of the VMEbus. This decoder maps the local bus address range \$F0000000 to \$F0FFFFFF to the A24 map of the VMEbus and the address range \$F1000000 to \$FF7FFFFFF to the A32 map of the VMEbus. The transfer size is always D16. This segment may be enabled using the enable bit. Write posting may be enabled using the write post enable bit.

The local bus map decoders should not be programmed such that more than one map decoder responds to the same local bus address or a map decoder conflicts with on-board resources. You may, however, program the map decoders to allow a VMEbus address to be accessed from more than one local bus address.

**Local Bus Slave (VMEbus Master) Ending Address Register 1**

<b>ADR/SIZ</b>	<b>\$FFF40014 (16 bits of 32)</b>		
<b>BIT</b>	31	...	16
<b>NAME</b>	Ending Address Register 1		
<b>OPER</b>	R/W		
<b>RESET</b>	0 PS		

This register is the ending address register for the first local-bus-to-VMEbus map decoder.

**Local Bus Slave (VMEbus Master) Starting Address Register 1**

<b>ADR/SIZ</b>	<b>\$FFF40014 (16 bits of 32)</b>		
<b>BIT</b>	15	...	0
<b>NAME</b>	Starting Address Register 1		
<b>OPER</b>	R/W		
<b>RESET</b>	0 PS		

This register is the starting address register for the first local-bus-to-VMEbus map decoder.

**Local Bus Slave (VMEbus Master) Ending Address Register 2**

<b>ADR/SIZ</b>	<b>\$FFF40018 (16 bits of 32)</b>		
<b>BIT</b>	31	...	16
<b>NAME</b>	Ending Address Register 2		
<b>OPER</b>	R/W		
<b>RESET</b>	0 PS		

This register is the ending address register for the second local-bus-to-VMEbus map decoder.

**Local Bus Slave (VMEbus Master) Starting Address Register 2**

<b>ADR/SIZ</b>	<b>\$FFF40018 (16 bits of 32)</b>		
<b>BIT</b>	15	...	0
<b>NAME</b>	Starting Address Register 2		
<b>OPER</b>	R/W		
<b>RESET</b>	0 PS		

This register is the starting address register for the second local-bus-to-VMEbus map decoder.

**Local Bus Slave (VMEbus Master) Ending Address Register 3**

<b>ADR/SIZ</b>	<b>\$FFF4001C (16 bits of 32)</b>		
<b>BIT</b>	31	...	16
<b>NAME</b>	Ending Address Register 3		
<b>OPER</b>	R/W		
<b>RESET</b>	0 PS		

This register is the ending address register for the third local-bus-to-VMEbus map decoder.

**Local Bus Slave (VMEbus Master) Starting Address Register 3**

<b>ADR/SIZ</b>	<b>\$FFF4001C (16 bits of 32)</b>		
<b>BIT</b>	15	...	0
<b>NAME</b>	Starting Address Register 3		
<b>OPER</b>	R/W		
<b>RESET</b>	0 PS		

This register is the starting address register for the third local-bus-to-VMEbus map decoder.

**Local Bus Slave (VMEbus Master) Ending Address Register 4**

<b>ADR/SIZ</b>	<b>\$FFF40020 (16 bits of 32)</b>		
<b>BIT</b>	31	...	16
<b>NAME</b>	Ending Address Register 4		
<b>OPER</b>	R/W		
<b>RESET</b>	0 PS		

This register is the ending address register for the fourth local-bus-to-VMEbus map decoder.

**Local Bus Slave (VMEbus Master) Starting Address Register 4**

<b>ADR/SIZ</b>	<b>\$FFF40020 (16 bits of 32)</b>		
<b>BIT</b>	15	...	0
<b>NAME</b>	Starting Address Register 4		
<b>OPER</b>	R/W		
<b>RESET</b>	0 PS		

This register is the starting address register for the fourth local-bus-to-VMEbus map decoder.

**Local Bus Slave (VMEbus Master) Address Translation Address Register 4**

<b>ADR/SIZ</b>	<b>\$FFF40024 (16 bits of 32)</b>		
<b>BIT</b>	31	...	16
<b>NAME</b>	Address Translation Address Register 4		
<b>OPER</b>	R/W		
<b>RESET</b>	0 PS		

This register is the address translation address register for the fourth local-bus-to-VMEbus bus map decoder.



**Local Bus Slave (VMEbus Master) Address Translation Select Register 4**

ADR/SIZ	\$FFF40024 (16 bits of 32)		
BIT	15	...	0
NAME	Address Translation Select Register 4		
OPER	R/W		
RESET	0 PS		

This register is the address translation select register for the fourth local-bus-to-VMEbus bus map decoder.

**Local Bus Slave (VMEbus Master) Attribute Register 4**

ADR/SIZ	\$FFF40028 (8 bits of 32)							
BIT	31	30	29	28	27	26	25	24
NAME	D16	WP	AM					
OPER	R/W	R/W	R/W					
RESET	0 PS	0 PS	0 PS					

This register is the attribute register for the fourth local-bus-to-VMEbus bus map decoder.

- AM** These bits define the VMEbus address modifier codes that the VMEbus master uses for the segment defined by map decoder 4. Because the local-bus-to-VMEbus interface does not support block transfers, the block transfer address modifier codes should not be used.
- WP** When this bit is high, write posting is enabled to the segment defined by map decoder 4. When this bit is low, write posting is disabled to the segment defined by map decoder 4.
- D16** When this bit is high, D16 data transfers are performed to the segment defined by map decoder 4. When this bit is low, D32 data transfers are performed to the segment defined by map decoder 4.

**Local Bus Slave (VMEbus Master) Attribute Register 3**

<b>ADR/SIZ</b>	<b>\$FFF40028 (8 bits of 32)</b>							
<b>BIT</b>	23	22	21	20	19	18	17	16
<b>NAME</b>	D16	WP	AM					
<b>OPER</b>	R/W	R/W	R/W					
<b>RESET</b>	0 PS	0 PS	0 PS					

This register is the attribute register for the third local-bus-to-VMEbus bus map decoder.

- AM** These bits define the VMEbus address modifier codes that the VMEbus master uses for the segment defined by map decoder 3. Because the local-bus-to-VMEbus interface does not support block transfers, the block transfer address modifier codes should not be used.
- WP** When this bit is high, write posting is enabled to the segment defined by map decoder 3. When this bit is low, write posting is disabled to the segment defined by map decoder 3.
- D16** When this bit is high, D16 data transfers are performed to the segment defined by map decoder 3. When this bit is low, D32 data transfers are performed to the segment defined by map decoder 3.

**Local Bus Slave (VMEbus Master) Attribute Register 2**

ADR/SIZ	\$FFF40028 (8 bits of 32)							
<b>BIT</b>	15	14	13	12	11	10	9	8
<b>NAME</b>	D16	WP	AM					
<b>OPER</b>	R/W	R/W	R/W					
<b>RESET</b>	0 PS	0 PS	0 PS					

This register is the attribute register for the second local-bus-to-VMEbus bus map decoder.

- AM** These bits define the VMEbus address modifier codes that the VMEbus master uses for the segment defined by map decoder 2. Since the local-bus-to-VMEbus interface does not support block transfers, the block transfer address modifier codes should not be used.
- WP** When this bit is high, write posting is enabled to the segment defined by map decoder 2. When this bit is low, write posting is disabled to the segment defined by map decoder 2.
- D16** When this bit is high, D16 data transfers are performed to the segment defined by map decoder 2. When this bit is low, D32 data transfers are performed to the segment defined by map decoder 2.

## Local Bus Slave (VMEbus Master) Attribute Register 1

ADR/SIZ	\$FFF40028 (8 bits of 32)							
BIT	7	6	5	4	3	2	1	0
NAME	D16	WP	AM					
OPER	R/W	R/W	R/W					
RESET	0 PS	0 PS	0 PS					

This register is the attribute register for the first local-bus-to-VMEbus bus map decoder.

- AM** These bits define the VMEbus address modifier codes that the VMEbus master uses for the segment defined by map decoder 1. Because the local-bus-to-VMEbus interface does not support block transfers, the block transfer address modifier codes should not be used.
- WP** When this bit is high, write posting is enabled to the segment defined by map decoder 1. When this bit is low, write posting is disabled to the segment defined by map decoder 1.
- D16** When this bit is high, D16 data transfers are performed to the segment defined by map decoder 1. When this bit is low, D32 data transfers are performed to the segment defined by map decoder 1.

## VMEbus Slave GCSR Group Address Register

<b>ADR/SIZ</b>	<b>\$FFF4002C (8 bits of 32)</b>		
<b>BIT</b>	31	...	24
<b>NAME</b>	GCSR Group Address Register 4		
<b>OPER</b>	R/W		
<b>RESET</b>	\$00 PS		

This register defines the group address of the GCSR as viewed from the VMEbus. The GCSR address is defined by the group address and the board address. Once enabled, the GCSR register should not be reprogrammed unless the VMEchip2 ASIC is VMEbus master.

**GCSR Group** These bits define the group portion of the GCSR address. These bits are compared with VMEbus address lines A8 through A15. The recommended group address for the MVME1X2P4 is \$D2.

## VMEbus Slave GCSR Board Address Register

ADR/SIZ	\$FFF4002C (4 bits of 32)						
BIT	23	...	20				
NAME	GCSR Board Address						
OPER	R/W						
RESET	\$F PS						

This register defines the board address of the GCSR as viewed from the VMEbus. The GCSR address is defined by the group address and the board address. Once enabled, the GCSR register should not be reprogrammed unless the VMEchip2 is VMEbus master. The value \$F in the GCSR board address register disables the map decoder. The map decoder is enabled when the board address is not \$F.

**GCSR Board** These bits define the board number portion of the GCSR address. These bits are compared with VMEbus address lines A4 through A7. The GCSR is enabled by values \$0 through \$E. The address \$XXFY in the VMEbus A16 space is reserved for the location monitors LM0 through LM3.

Note that *XX* is the group address and *Y* is the location monitor (1,LM0; 3,LM1; 5,LM2; 7,LM3).

## Local-Bus-to-VMEbus Enable Control Register

ADR/SIZ	\$FFF4002C (4 bits of 32)							
BIT					19	18	17	16
NAME					EN4	EN3	EN2	EN1
OPER					R/W	R/W	R/W	R/W
RESET					0 PSL	0 PSL	0 PSL	0 PSL

This register is the map decoder enable register for the four programmable local-bus-to-VMEbus map decoders.

- EN1** When this bit is high, the first local-bus-to-VMEbus map decoder is enabled. When this bit is low, the first local-bus-to-VMEbus map decoder is disabled.
- EN2** When this bit is high, the second local-bus-to-VMEbus map decoder is enabled. When this bit is low, the second local-bus-to-VMEbus map decoder is disabled.
- EN3** When this bit is high, the third local-bus-to-VMEbus map decoder is enabled. When this bit is low, the third local-bus-to-VMEbus map decoder is disabled.
- EN4** When this bit is high, the fourth local-bus-to-VMEbus map decoder is enabled. When this bit is low, the fourth local-bus-to-VMEbus map decoder is disabled.

## Local-Bus-to-VMEbus I/O Control Register

ADR/SIZ	\$FFF4002C (8 bits of 32)							
BIT	15	14	13	12	11	10	9	8
NAME	I2EN	I2WP	I2SU	I2PD	I1EN	I1D16	I1WP	I1SU
OPER	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RESET	0 PSL	0 PS	0 PS	0 PS	0 PS	0 PS	0 PS	0 PS

This register controls the VMEbus short I/O map and the F page (\$F0000000 through \$FF7FFFFFFF) I/O map.

- I1SU** When this bit is high, the VMEchip2 drives a supervisor address modifier code when the short I/O space is accessed. When this bit is low, the VMEchip2 drives a user address modifier code when the short I/O space is accessed.
- I1WP** When this bit is high, write posting is enabled to the VMEbus short I/O segment. When this bit is low, write posting is disabled to the VMEbus short I/O segment.
- I1D16** When this bit is high, D16 data transfers are performed to the VMEbus short I/O segment. When this bit is low, D32 data transfers are performed to the VMEbus short I/O segment.
- I1EN** When this bit is high, the VMEbus short I/O map decoder is enabled. When this bit is low, the VMEbus short I/O map decoder is disabled.
- I2PD** When this bit is high, the VMEchip2 drives a program address modifier code when the F page is accessed. When this bit is low, the VMEchip2 drives a data address modifier code when the F page is accessed.
- I2SU** When this bit is high, the VMEchip2 drives a supervisor address modifier code when the F page is accessed. When this bit is low, the VMEchip2 drives a user address modifier code when the F page is accessed.



- I2WP** When this bit is high, write posting is enabled to the local bus F page. When this bit is low, write posting is disabled to the local bus F page.
- I2EN** When this bit is high, the F page (\$F0000000 through \$FF7FFFFFFF) map decoder is enabled. The F0 page is defined as A24/D16 on the VMEbus while the F1-FE pages are defined as A32/D16. When this bit is low, the F page is disabled.

### ROM Control Register

ADR/SIZ	\$FFF4002C							
BIT	7	6	5	4	3	2	1	0
NAME	SIZE		BSSPD			ASPD		
OPER	R/W		R/W			R/W		
RESET	0 PS		0 PS			0 PS		

This function is not used on the MVME1X2P4.

## Programming the VMEchip2 DMA Controller

This section includes programming information on the DMA controller, VMEbus interrupter, MPU status register, and local-bus-to-VMEbus requester register.

The VMEchip2 features a local-bus -to-VMEbus DMA controller (DMAC). The DMAC has two modes of operation: command chaining, and direct. In direct mode, the local bus address, the VMEbus address, the byte count, and the control register of the DMAC are programmed and the DMAC is enabled. The DMAC transfers data, as programmed, until the byte count is zero or an error is detected. When the DMAC stops, the status bits in the DMAC status register are set and an interrupt is sent to the local bus interrupter. If the DMAC interrupt is enabled in the local bus interrupter, the local bus is interrupted. The time on and time off timers should be programmed to control the VMEbus bandwidth used by the DMAC.

A maximum of 4GB of data may be transferred with one DMAC command. Larger transfers can be accomplished using the command chaining mode. In command chaining mode, a singly-linked list of commands is built in local memory and the table address register in the DMAC is programmed with the starting address of the list of commands. The DMAC control register is programmed and the DMAC is enabled. The DMAC executes commands from the list until all commands are executed or an error is detected. When the DMAC stops, the status bits are set in the DMAC status register and an interrupt is sent to the local bus interrupter. If the DMAC interrupt is enabled in the local bus interrupter, the local bus is interrupted. When the DMAC finishes processing a command in the list, and interrupts are enabled for that command, the DMAC sends an interrupt to the local bus interrupter. If the DMAC interrupt is enabled in the local bus interrupter, the local bus is interrupted.

The DMAC control is divided into two registers. The first register is only accessible by the processor. The second register can be loaded by the processor in direct mode and by the DMAC in command chaining mode.

Once the DMAC is enabled, the counter and control registers should not be modified by software. When you use the command chaining mode, the list of commands must be in local 32-bit memory and the entries must be quad-byte aligned.

A DMAC command list includes one or more DMAC command packets. A DMAC command packet includes a control word that defines the VMEbus AM code, the VMEbus transfer size, the VMEbus transfer method, the DMA transfer direction, the VMEbus and local bus address counter operation, and the local bus snoop operation. The format of the control word is the same as the lower 16 bits of the control register. The command packet also includes a local bus address, a VMEbus address, a byte count, and a pointer to the next command packet in the list. The end of a command is indicated by setting bit 0 or 1 of the next command address. [Table 2-3](#) shows the command packet format.

**Table 2-3. DMAC Command Packet Format**

Entry	Function	
0 (bits 0-15)	--	Control Word
1 (bits 0-31)	Local Bus Address	
2 (bits 0-31)	VMEbus Address	
3 (bits 0-31)	Byte Count	
4 (bits 0-31)	Address of Next Command Packet	

## DMAC Registers

This section provides addresses and bit level descriptions of the DMAC counters, control registers, and status registers. Other control functions are also included in this section.

### EPROM Decoder, SRAM and DMA Control Register

ADR/SIZ	\$FFF40030 (8 bits [6 used] of 32)							
BIT	23	22	21	20	19	18	17	16
NAME			WAIT RMW	ROM0	TBLSC		SRAMS	
OPER			R/W	R/W	R/W		R/W	
RESET			0 PSL	1 PSL	0 PS		0 PS	

This register controls the snoop control bits used by the DMAC when it is accessing table entries.

**SRAMS**      These VMEchip2 bits are not used on the MVME1X2P4.

**TBLSC** These bits control the snoop signal lines on the local bus when the DMAC is table walking. The snooping functions differ according to processor type, as shown:

TBLSC		Requested Snoop Operation	
19	18	MC68040	MC68060
0	0	Snoop disabled	Snoop enabled
0	1	Source dirty, sink byte/word/longword	Snoop disabled
1	0	Source dirty, invalidate line	Snoop enabled
1	1	Snoop disabled (Reserved)	Snoop disabled

**ROM0** This bit is not used on the MVME1X2P4. Its function is performed by the ROM0 bit in the Petra/MC2 PROM Access Time Control register. Refer to Chapter 3.

**WAIT RMW** This function is not used on the MVME1X2P4.

### Local-Bus-to-VMEbus Requester Control Register

ADR/SIZ	\$FFF40030 (8 bits [7 used] OF 32)							
BIT	15	14	13	12	11	10	9	8
NAME	ROBN	DHB	DWB		LVFAIR	LVRW D	LVREQ	
OPER	R/W	R	R/W		R/W	R/W	R/W	
RESET	0 PS	0 PS	0 PSL		0 PS	0 PS	0 PS	

This register controls the VMEbus request level, the request mode, and release mode for the local-bus-to-VMEbus interface.

**LVREQ** These bits define the VMEbus request level. The request level can only change while the VMEchip2 is bus master. The VMEchip2 always requests at the old level until it becomes bus master and the new level takes effect. If the VMEchip2 is bus master when the level is changed, the new level does not take effect until the bus has been released and re-requested at the old level. The requester always requests the VMEbus at level 3 the first time following a SYSRESET.

- 0** The request level is 0.
- 1** The request level is 1.
- 2** The request level is 2.
- 3** The request level is 3.

- LVRWD** When this bit is high, the requester operates in release-when-done mode. When this bit is low, the requester operates in release-on-request mode.
- LVFAIR** When this bit is high, the requester operates in fair mode. When this bit is low, the requester does not operate in fair mode. In fair mode, the requester waits until the request signal line for the selected level is inactive before requesting the VMEbus.
- DWB** When this bit is high, the VMEchip2 requests the VMEbus and does not release it. When this bit is low, the VMEchip2 releases the VMEbus according to the release mode programmed in the LVRWD bit. When the VMEbus has been acquired, the DHB bit is set.
- DHB** When this bit is high, the VMEbus has been acquired in response to the DWB bit being set. When the DWB bit is cleared, this bit is cleared.
- ROBN** When this bit is high, the VMEbus arbiter operates in round-robin mode. When this bit is low, the arbiter operates in priority mode.

### DMAC Control Register 1 (bits 0-7)

ADR/SIZ	\$FFF40030 (8 bits of 32)							
BIT	7	6	5	4	3	2	1	0
NAME	DHALT	DEN	DTBL	DFAIR	DRELM		DREQL	
OPER	S	S	R/W	R/W	R/W		R/W	
RESET	0 PS	0 PS	0 PS	0 PS	0 PS		0 PS	

This control register is loaded by the processor; it is not modified when the DMAC loads new values from the command packet.

- DREQL** These bits define the VMEbus request level for the DMAC requester. The request level can only change while the VMEchip2 is bus master. The VMEchip2

always requests at the old level until it becomes bus master and the new level takes effect. If the VMEchip2 is bus master when the level is changed, the new level does not take effect until the bus has been released and re-requested at the old level. The requester always requests the VMEbus at level 3 the first time following a SYSRESET.

- 0** VMEbus request level 0
- 1** VMEbus request level 1
- 2** VMEbus request level 2
- 3** VMEbus request level 3

**DRELM** These bits define the VMEbus release mode for the DMAC requester. The DMAC always releases the bus when the FIFO is full (VMEbus to local bus) or empty (local bus to VMEbus).

- 0** Release when the time on timer has expired and a BRx\* signal is active on the VMEbus.
- 1** Release when the time on timer has expired.
- 2** Release when a BRx\* signal is active on the VMEbus.
- 3** Release when a BRx\* signal is active on the VMEbus or the time on timer has expired.

**DFAIR** When this bit is high, the DMAC requester operates in fair mode. It waits until its request level is inactive before requesting the VMEbus. When this bit is low, the DMAC requester does not operate in fair mode.

**DTBL** The DMAC operates in direct mode when this bit is low, and it operates in command chaining mode when this bit is high.

**DEN** The DMAC is enabled when this bit is set high. This bit always reads 0.

**DHALT** When this bit is high, the DMAC halts at the end of a command when the DMAC is operating in command chaining mode. When this bit is low, the DMAC executes the next command in the list.

**DMAC Control Register 2 (bits 8-15)**

ADR/SIZ	\$FFF40034 (8 bits [7 USED] of 32)							
BIT	15	14	13	12	11	10	9	8
NAME	INTE	SNP			VINC	LINC	TVME	D16
OPER	R/W	R/W			R/W	R/W	R/W	R/W
RESET	0 PS	0 PS			0 PS	0 PS	0 PS	0 PS

This portion of the control register is loaded by the processor or by the DMAC when it loads the command word from the command packet. Because this register is loaded from the command packet in command chaining mode, the descriptions here also apply to the control word in the command packet.

**D16** When this bit is high, the DMAC executes D16 cycles on the VMEbus. When this bit is low, the DMAC executes D32/D64 cycles on the VMEbus.

**TVME** This bit defines the direction in which the DMAC transfers data. When this bit is high, data is transferred to the VMEbus. When it is low, data is transferred to the local bus.

**LINC** When this bit is high, the local bus address counter is incremented during DMA transfers. When this bit is low, the counter is not incremented. This bit should normally be set high. In special situations such as transferring data to or from a FIFO, it may be desirable to not increment the counter.

**VINC** When this bit is high, the VMEbus address counter is incremented during DMA transfers. When this bit is low, the counter is not incremented. This bit should normally be set high. In special situations such as transferring data to or from a FIFO, it may be desirable to not increment the counter.

**SNP** These bits control the snoop signal lines on the local bus when the DMAC is local bus master and it is not accessing the command table. The snooping functions differ according to processor type, as shown:

SNP		Requested Snoop Operation	
14	13	MC68040	MC68060
0	0	Snoop disabled	Snoop enabled
0	1	Source dirty, sink byte/word/longword	Snoop disabled
1	0	Source dirty, invalidate line	Snoop enabled
1	1	Snoop disabled (Reserved)	Snoop disabled

**INTE** This bit is used only in command chaining mode. It is only modified when the DMAC loads the control register from the control word in the command packet. When this bit in the command packet is set, an interrupt is sent to the local bus interrupter when the command in the packet has been executed. The local bus is interrupted if the DMAC interrupt is enabled.

### DMAC Control Register 2 (bits 0-7)

ADR/SIZ	\$FFF40034 (8 bits of 32)							
BIT	7	6	5	4	3	2	1	0
NAME	BLK		VME AM					
OPER	R/W		R/W					
RESET	0 PS		0 PS					

This portion of the control register is loaded by the processor or the DMAC when it loads the command word from the command packet. Because this byte is loaded from the command packet in command chaining mode, the descriptions here also apply to the control word in the command packet.

**VME AM** These bits define the address modifier codes the DMAC drives on the VMEbus when it is bus master. During non-block transfer cycles, bits 0-5 define the VMEbus address modifiers. During block transfers, bits 2-5 define



VMEbus address modifier bits 2-5, and address modifier bits 0 and 1 are provided by the DMAC to indicate a block transfer. Block transfer mode should not be set in the address modifier codes. The special block transfer bits should be set to enable block transfers. If non-block cycles are required to reach a 32- or 64-bit boundary, bits 0 and 1 are used during these cycles.

**BLK**

These bits control the block transfer modes of the DMAC:

- 0** Block transfers disabled
- 1** The DMAC executes D32 block transfer cycles on the VMEbus. In block transfer mode, the DMAC may execute byte and two-byte cycles at the beginning and ending of a transfer in non-block transfer mode. If the D16 bit is set, the DMAC executes D16 block transfers.
- 2** Block transfers disabled
- 3** The DMAC executes D64 block transfer cycles on the VMEbus. In block transfer mode, the DMAC may execute byte, two-byte and four-byte cycles at the beginning and ending of a transfer in non-block transfer mode. If the D16 bit is set, the DMAC executes D16 block transfers.

**DMAC Local Bus Address Counter**

<b>ADR/SIZ</b>	<b>\$FFF40038 (32 bits)</b>		
<b>BIT</b>	31	...	0
<b>NAME</b>	DMAC Local Bus Address Counter		
<b>OPER</b>	R/W		
<b>RESET</b>	0 PS		

In direct mode, this counter is programmed with the starting address of the data in local bus memory.

**DMAC VMEbus Address Counter**

<b>ADR/SIZ</b>	<b>\$FFF4003C (32 bits)</b>		
<b>BIT</b>	31	...	0
<b>NAME</b>	DMAC VMEbus Address Counter		
<b>OPER</b>	R/W		
<b>RESET</b>	0 PS		

In direct mode, this counter is programmed with the starting address of the data in VMEbus memory.

**DMAC Byte Counter**

<b>ADR/SIZ</b>	<b>\$FFF40040 (32 bits)</b>		
<b>BIT</b>	31	...	0
<b>NAME</b>	DMAC Byte Counter		
<b>OPER</b>	R/W		
<b>RESET</b>	0 PS		

In direct mode, this counter is programmed with the number of bytes of data to be transferred.

**Table Address Counter**

<b>ADR/SIZ</b>	<b>\$FFF40044 (32 bits)</b>		
<b>BIT</b>	31	...	0
<b>NAME</b>	Table Address Counter		
<b>OPER</b>	R/W		
<b>RESET</b>	0 PS		

In command chaining mode, this counter should be loaded by the processor with the starting address of the list of commands. This register gets reloaded by the DMAC with the starting address of the current command. The last command in a list should have bits 0 and 1 set in the next command pointer.

## VMEbus Interrupter Control Register

ADR/SIZ	\$FFF40048 (8 bits [7 used] of 32)							
BIT	31	30	29	28	27	26	25	24
NAME		IRQ1S		IRQC	IRQS	IRQL		
OPER		R/W		S	R	S		
RESET		0 PS		0 PS	0 PS	0 PS		

This register controls the VMEbus interrupter.

**IRQL** These bits define the level of the VMEbus interrupt generated by the VMEchip2. A VMEbus interrupt is generated by writing the desired level to these bits. These bits always read 0 and writing 0 to these bits has no effect.

**IRQS** This bit is the IRQ status bit. When this bit is high, the VMEbus interrupt has not been acknowledged. When this bit is low, the VMEbus interrupt has been acknowledged. This is a read-only status bit.

**IRQC** This bit is the VMEbus interrupt clear bit. When this bit is set high, the VMEbus interrupt is removed. This feature is only used when the IRQ1 broadcast mode is used. Normal VMEbus interrupts should never be cleared. This bit always reads 0; writing a 0 to it has no effect.

**IRQ1S** These bits control the function of the IRQ1 signal line on the VMEbus:

- 0** The IRQ1 signal from the interrupter is connected to the VMEbus IRQ1 signal line.
- 1** The output from tick timer 1 is connected to the VMEbus IRQ1 signal line.
- 2** The IRQ1 signal from the interrupter is connected to the VMEbus IRQ1 signal line.
- 3** The output from tick timer 2 is connected to the VMEbus IRQ1 signal line.

## VMEbus Interrupter Vector Register

<b>ADR/SIZ</b>	<b>\$FFF40048 (8 bits of 32)</b>		
<b>BIT</b>	23	...	16
<b>NAME</b>	Interrupter Vector		
<b>OPER</b>	R/W		
<b>RESET</b>	\$0F PS		

This register controls the VMEbus interrupter vector.

## MPU Status and DMA Interrupt Count Register

<b>ADR/SIZ</b>	<b>\$FFF40048 (8 bits of 32)</b>							
<b>BIT</b>	15	14	13	12	11	10	9	8
<b>NAME</b>	DMAIC				MCLR	MLBE	MLPE	MLOB
<b>OPER</b>	R				C	R	R	R
<b>RESET</b>	0 PS				0 PS	0 PS	0 PS	0 PS

This is the MPU status register and DMAC interrupt counter.

- MLOB** When this bit is set, the MPU received a TEA and the status indicated off-board. This bit is cleared by writing a 1 to the MCLR bit in this register.
- MLPE** When this bit is set, the MPU received a TEA and the status indicated a parity error during a DRAM data transfer. This bit is cleared by writing a 1 to the MCLR bit in this register. This bit is not defined for MVME1X2P4 implementation.
- MLBE** When this bit is set, the MPU received a TEA and no additional status was provided. This bit is cleared by writing a 1 to the MCLR bit in this register.
- MCLR** Writing a 1 to this bit clears the MPU status bits 7, 8, 9 and 10 (MLTO, MLOB, MLPE, and MLBE) in this register.

**DMAIC** The DMAC interrupt counter is incremented when an interrupt is sent to the local bus interrupter. The value in this counter indicates the number of commands processed when the DMAC is operated in command chaining mode. If the interrupt count exceeds 15, the counter rolls over. This counter operates regardless of whether the DMAC interrupts are enabled. This counter is cleared when the DMAC is enabled.

### DMAC Status Register

ADR/SIZ	\$FFF40048 (8 bits of 32)							
BIT	7	6	5	4	3	2	1	0
NAME	MLTO	DLBE	DLPE	DLOB	DLTO	TBL	VME	DONE
OPER	R	R	R	R	R	R	R	R
RESET	0 PS	0 PS	0 PS	0 PS	0 PS	0 PS	0 PS	0 PS

This is the DMAC status register.

**DONE** This bit is set when the DMAC has finished executing commands, either without errors or because the halt bit was set. This bit is cleared when the DMAC is enabled.

**VME** If this bit is set, the DMAC has received a VMEbus BERR during a data transfer. This bit is cleared when the DMAC is enabled.

**TBL** If this bit is set, the DMAC has received an error on the local bus while it was reading commands from the command packet. Additional information is provided in bits 3 - 6 (DLTO, DLOB, DLPE, and DLBE). This bit is cleared when the DMAC is enabled.

**DLTO** If this bit is set, the DMAC has received a TEA and the status indicated a local bus time-out. This bit is cleared when the DMAC is enabled.

**DLOB** If this bit is set, the DMAC has received a TEA and the status indicated off-board. This bit is cleared when the DMAC is enabled.

- DLPE** If this bit is set, the DMAC has received a TEA and the status indicated a parity error during a DRAM data transfer. This bit is cleared when the DMAC is enabled. This bit is not defined for MVME1X2P4 implementation.
- DLBE** If this bit is set, the DMAC has received a TEA and no additional status was provided. This bit is cleared when the DMAC is enabled.
- MLTO** If this bit is set, the MPU has received a TEA and the status indicated a local bus time-out. This bit is cleared by writing a 1 to the MCLR bit in this register.

## Programming the Tick and Watchdog Timers

The VMEchip2 has two 32-bit tick timers and one watchdog timer. This section provides addresses and bit level descriptions of the prescaler, tick timer, watchdog timer registers, and various other timer registers.

### VMEbus Arbiter Time-Out Control Register

ADR/SIZ	\$FFF4004C (8 bits [1 used] of 32)							
BIT	31	30	29	28	27	26	25	24
NAME								ARBTO
OPER								R/W
RESET								0 PS

This register controls the VMEbus arbiter time-out timer.

- ARBTO** When this bit is high, the VMEbus grant time-out timer is enabled. When this bit is low, the VMEbus grant timer is disabled. When the timer is enabled and the arbiter does not receive a BBSY signal within 256  $\mu$ s after a grant is issued, the arbiter asserts BBSY and removes the grant. The arbiter then rearbitrates any pending requests.

## DMAC Ton/Toff Timers and VMEbus Global Time-out Control Register

ADR/SIZ	\$FFF4004C (8 bits of 32)							
BIT	23	22	21	20	19	18	17	16
NAME	TIME OFF			TIME ON			VGTO	
OPER	R/W			R/W			R/W	
RESET	0 PS			0 PS			0 PS	

This register controls the DMAC time off timer, the DMAC time on timer, and the VMEbus global time-out timer.

**VGTO** These bits define the VMEbus global time-out value. When DS0 or DS1 is asserted on the VMEbus, the timer begins timing. If the timer times out before the data strobes are removed, a BERR signal is sent to the VMEbus. The global time-out timer is disabled when the VMEchip2 is not system controller.

- 0 8  $\mu$ s
- 1 64  $\mu$ s
- 2 256  $\mu$ s
- 3 The timer is disabled

**TIME ON** These bits define the maximum time the DMAC spends on the VMEbus:

- 0 16  $\mu$ s
- 1 32  $\mu$ s
- 2 64  $\mu$ s
- 3 128  $\mu$ s
- 4 256  $\mu$ s
- 5 512  $\mu$ s
- 6 1024  $\mu$ s
- 7 When done (or no data)

**TIME OFF** These bits define the minimum time the DMAC spends off the VMEbus:

- 0 0  $\mu$ s
- 1 16  $\mu$ s
- 2 32  $\mu$ s
- 3 64  $\mu$ s
- 4 128  $\mu$ s
- 5 256  $\mu$ s
- 6 512  $\mu$ s
- 7 1024  $\mu$ s

## VME Access, Local Bus, and Watchdog Time-out Control Register

ADR/SIZ	\$FFF4004C (8 bits of 32)							
BIT	15	14	13	12	11	10	9	8
NAME	VATO		LBTO		WDTO			
OPER	R/W		R/W		R/W			
RESET	0 PS		0 PS		0 PS			

**WDTO** These bits define the watchdog time-out period:

Bit Encoding	Time-out
0	512 $\mu$ s
1	1 ms
2	2 ms
3	4 ms
4	8 ms
5	16 ms
6	32 ms
7	64 ms

Bit Encoding	Time-out
8	128 ms
9	256 ms
10	512 ms
11	1 s
12	4 s
13	16 s
14	32 s
15	64 s

**LBTO** These bits define the local bus time-out value. The timer begins timing when TS is asserted on the local bus. If TA or TAE is not asserted before the timer times out, a TEA signal is sent to the local bus. The timer is disabled if the transfer is bound for the VMEbus.

0	8 $\mu$ s	2	256 $\mu$ s
1	64 $\mu$ s	3	The timer is disabled

**VATO** These bits define the VMEbus access time-out value. When a transaction is headed to the VMEbus and the VMEchip2 is not the current VMEbus master, the access timer begins timing. If the VMEchip2 has not received bus mastership before the timer times out and the transaction is not write posted, a TEA signal is sent to the local bus. If the transaction is write posted, a write post error interrupt is sent to the local bus interrupter.

0	64 $\mu$ s	2	32 ms
1	1 ms	3	The timer is disabled



## Prescaler Control Register

<b>ADR/SIZ</b>	<b>\$FFF4004C (8 bits of 32)</b>		
<b>BIT</b>	7	...	0
<b>NAME</b>	Prescaler Adjust		
<b>OPER</b>	R/W		
<b>RESET</b>	\$DF P		

The prescaler provides the various clocks required by the counters and timers in the VMEchip2. In order to specify absolute times from these counters and timers, the prescaler must be adjusted for different local bus clocks. The prescaler register should be programmed based on the following equation. This provides a 1MHz clock to the Tick timers.

$$\text{prescaler register} = 256 - \text{Bclock (MHz)}$$

For example, for operation at 25MHz the prescaler value is \$E7, and at 32MHz it is \$E0.

Non-integer local bus clocks introduce an error into the specified times for the various counters and timers. This is most notable in the tick timers. The tick timer clock can be derived by the following equation.

$$\text{tick timer clock} = \text{Bclock} / (256 - \text{prescaler value})$$

If the prescaler is not correctly programmed, the bus timers do not generate their specified values and the VMEbus reset time may be violated. The maximum clock frequency for the tick timers is the B clock divided by two. The prescaler register control logic does not allow the value 255 (\$FF) to be programmed.

## Tick Timer 1 Compare Register

<b>ADR/SIZ</b>	<b>\$FFF40050 (32 bits)</b>		
<b>BIT</b>	31	...	0
<b>NAME</b>	Tick timer 1 Compare Register		
<b>OPER</b>	R/W		
<b>RESET</b>	0 P		

The tick timer 1 counter is compared to this register. When they are equal, an interrupt is sent to the local bus interrupter and the overflow counter is incremented. If the clear-on-compare mode is enabled, the counter is also cleared. For periodic interrupts, the following equation should be used to calculate the compare register value for a specific period (T).

$$\text{compare register value} = T (\mu\text{s})$$

When programming the tick timer for periodic interrupts, the counter should be cleared to zero by software and then enabled. If the counter does not initially start at 0, the time to the first interrupt may be longer or shorter than expected. Remember the rollover time for the counter is 71.6 minutes.

## Tick Timer 1 Counter

<b>ADR/SIZ</b>	<b>\$FFF40054 (32 bits)</b>		
<b>BIT</b>	31	...	0
<b>NAME</b>	Tick timer 1 Counter		
<b>OPER</b>	R/W		
<b>RESET</b>	0 P		

This is the tick timer 1 counter. When enabled, it increments every microsecond. Software may read or write the counter at any time.

## Tick Timer 2 Compare Register

<b>ADR/SIZ</b>	<b>\$FFF40058 (32 bits)</b>		
<b>BIT</b>	31	...	0
<b>NAME</b>	Tick timer 2 Compare Register		
<b>OPER</b>	R/W		
<b>RESET</b>	0 P		

The tick timer 2 counter is compared to this register. When they are equal, an interrupt is sent to the local bus interrupter and the overflow counter is incremented. If the clear-on-compare mode is enabled, the counter is also cleared. For periodic interrupts, the following equation should be used to determine the compare register value for a specific period.

$$\text{compare register value} = T (\mu\text{s})$$

When programming the tick timer for periodic interrupts, the counter should be cleared to zero by software and then enabled. If the counter does not initially start at 0, the time to the first interrupt may be longer or shorter than expected. Remember the rollover time for the counter is 71.6 minutes.

## Tick Timer 2 Counter

<b>ADR/SIZ</b>	<b>\$FFF4005C (32 bits)</b>		
<b>BIT</b>	31	...	0
<b>NAME</b>	Tick timer 2 Counter		
<b>OPER</b>	R/W		
<b>RESET</b>	0 P		

This is the tick timer 2 counter. When enabled, it increments every microsecond. Software may read or write the counter at any time.

## Board Control Register

ADR/SIZ	\$FFF40060 (8 bits [7 used] of 32)							
BIT	31	30	29	28	27	26	25	24
NAME		SCON	SFFL	BRFLI	PURS	CPURS	BDFLO	RSWE
OPER		R	R	R	R	C	R/W	R/W
RESET		X	X	1 PSL	1 P	0 PS	1 PSL	1 P

- RSWE** The **RESET** switch enable bit is used with the “no VMEbus interface” option. This bit is duplicated at the same bit position in the MC2 chip at location \$FFF42044. When this bit or the duplicate bit in the MC2 chip is high, the **RESET** switch is enabled. When both bits are low, the **RESET** switch is disabled.
- BDFLO** When this bit is high, the VMEchip2 asserts the BRDFAIL signal pin. When this bit is low, this bit does not contribute to the BRDFAIL signal on the VMEchip2.
- CPURS** When this bit is set high, the powerup reset status bit is cleared. This bit is always read 0.
- PURS** This bit is set by a powerup reset. It is cleared by a write to the CPURS bit.
- BRFLI** When this status bit is high, the BRDFAIL signal pin on the VMEchip2 is asserted. When this status bit is low, the BRDFAIL signal pin on the VMEchip2 is not asserted. The BRDFAIL pin may be asserted by an external device, the BDFLO bit in this register, or a watchdog time-out.
- SFFL** When this status bit is high, the SYSFAIL signal line on the VMEbus is asserted. When this status bit is low, the SYSFAIL signal line on the VMEbus is not asserted.
- SCON** When this status bit is high, the VMEchip2 is configured as system controller. When this status bit is low, the VMEchip2 is not configured as system controller.

**Watchdog Timer Control Register**

ADR/SIZ	\$FFF40060 (8 bits of 32)							
BIT	23	22	21	20	19	18	17	16
NAME	SRST	WDCS	WDCC	WDTO	WDBFE	WDS/L	WDRSE	WDEN
OPER	S	C	C	R	R/W	R/W	R/W	R/W
RESET	0 P S	0	0	0 P	0 P S L	0 P S L	1 P S L	0 P S L

- WDEN** When this bit is high, the watchdog timer is enabled. When this bit is low, the watchdog timer is not enabled.
- WDRSE** When this bit is high, and a watchdog time-out occurs, a SYSRESET or LRESET is generated. The WDS/L bit in this register selects the reset. When this bit is low, a watchdog time-out does not cause a reset.
- WDS/L** When this bit is high and the watchdog timer has timed out and the watchdog reset enable (WDRSE bit in this register) is high, a SYSRESET signal is generated on the VMEbus which in turn causes LRESET to be asserted. When this bit is low and the watchdog timer has timed out and the watchdog reset enable (WDRSE bit in this register) is high, an LRESET signal is generated on the local bus.
- WDBFE** When this bit is high and the watchdog timer has timed out, the VMEchip2 asserts the BRDFAIL signal pin. When this bit is low, the watchdog timer does not contribute to the BRDFAIL signal on the VMEchip2.
- WDTO** When this status bit is high, a watchdog time-out has occurred. When this status bit is low, a watchdog time-out has not occurred. This bit is cleared by writing a 1 to the WDCS bit in this register.
- WDCC** When this bit is set high, the watchdog counter is reset. The counter must be reset within the time-out period or a watchdog time-out occurs.

- WDCS** When this bit is set high, the watchdog time-out status bit (WDTO bit in this register) is cleared.
- SRST** When this bit is set high, a SYSRESET signal is generated on the VMEbus. SYSRESET resets the VMEchip2 and clears this bit.

### Tick Timer 2 Control Register

ADR/SIZ	\$FFF40060 (8 bits [7 used] of 32)							
BIT	15	14	13	12	11	10	9	8
<b>NAME</b>	OVF					COVF	COC	EN
<b>OPER</b>	R					C	R/W	R/W
<b>RESET</b>	0 PS					0 PS	0 PS	0 PS

- EN** When this bit is high, the counter increments. When this bit is low, the counter does not increment.
- COC** When this bit is high, the counter is reset to 0 when it compares with the compare register. When this bit is low, the counter is not reset.
- COVF** The overflow counter is cleared when a 1 is written to this bit.
- OVF** These bits are the output of the overflow counter. The overflow counter is incremented each time the tick timer sends an interrupt to the local bus interrupter. The overflow counter can be cleared by writing a 1 to the COVF bit.

## Tick Timer 1 Control Register

ADR/SIZ	\$FFF40060 (8 bits of 32)							
BIT	7	6	5	4	3	2	1	0
NAME	OVF					COVF	COC	EN
OPER	R					C	R/W	R/W
RESET	0 PS					0 PS	0 PS	0 PS

- EN** When this bit is high, the counter increments. When this bit is low, the counter does not increment.
- COC** When this bit is high, the counter is reset to 0 when it compares with the compare register. When this bit is low, the counter is not reset.
- COVF** The overflow counter is cleared when a 1 is written to this bit.
- OVF** These bits are the output of the overflow counter. The overflow counter is incremented each time the tick timer sends an interrupt to the local bus interrupter. The overflow counter can be cleared by writing a 1 to the COVF bit.

## Prescaler Counter

ADR/SIZ	\$FFF40064 (32 bits)		
BIT	31	...	0
NAME	Prescaler Counter		
OPER	R/W		
RESET	0 P		

The VMEchip2 has a 32-bit prescaler that provides the clocks required by the various timers in the chip. Access to the prescaler is provided for test purposes. The counter is described here because it may be useful in other applications. The lower 8 bits of the prescaler counter increment to \$FF at the local bus clock rate and then they are loaded from the prescaler adjust register. When the load occurs, the upper 24 bits are incremented. When the prescaler adjust register is correctly programmed, the lower 8 bits increment at the local bus clock rate and the upper 24 bits increment every microsecond. The counter may be read at any time.


## 2 Programming the Local Bus Interrupter

The local bus interrupter is used by devices that need to interrupt the local bus. There are 31 devices that can interrupt the local bus through the VMEchip2. In the general case, each interrupter has a level select register, an enable bit, a status bit, a clear bit, and a set bit for the software interrupts. Each interrupter also provides a unique interrupt vector to the processor. The upper four bits of the vector are programmable in the vector base registers. The lower four bits are unique for each interrupter. There are two base registers, one for the first 16 interrupters, and one for the next 8 interrupters. The VMEbus interrupters provide their own vectors. A summary of the interrupts appears in [Table 2-4](#).

The status bit of an interrupter is affected by the enable bit. If the enable bit is low, the status bit is also low. Interrupts may be polled by setting the enable bit and programming the level to 0. This enables the status bit and prevents the local bus from being interrupted. The enable bit does not clear edge-sensitive interrupts. If necessary, edge-sensitive interrupts should be cleared, in order to remove any old interrupts, and then re-enabled. The master interrupt enable (MIEN) bit must be set before the VMEchip2 can generate any interrupts. The MIEN bit is in I/O Control Register 1.



**Table 2-4. Local Bus Interrupter Summary**

<b>Interrupt</b>	<b>Vector</b>	<b>Priority for Simultaneous Interrupts</b>
VMEbus IRQ1	External	Lowest 
VMEbus IRQ2	External	
VMEbus IRQ3	External	
VMEbus IRQ4	External	
VMEbus IRQ5	External	
VMEbus IRQ6	External	
VMEbus IRQ7	External	
Spare	\$Y7	
Software 0	\$Y8	
Software 1	\$Y9	
Software 2	\$YA	
Software 3	\$YB	
Software 4	\$YC	
Software 5	\$YD	
Software 6	\$YE	
Software 7	\$YF	
GCSR LM0	\$X0	
GCSR LM1	\$X1	
GCSR SIG0	\$X2	
GCSR SIG1	\$X3	
GCSR SIG2	\$X4	
GCSR SIG3	\$X5	

**Table 2-4. Local Bus Interrupter Summary (Continued)**

Interrupt	Vector	Priority for Simultaneous Interrupts
DMAC	\$X6	: : ↓ Highest
VMEbus Interrupter Acknowledge	\$X7	
Tick Timer 1	\$X8	
Tick Timer 2	\$X9	
VMEbus IRQ1 Edge-Sensitive	\$XA	
(Not used on MVME1X2P4)	\$XB	
VMEbus Master Write Post Error	\$XC	
VMEbus SYSFAIL	\$XD	
(Not used on MVME1X2P4)	\$XE	
VMEbus ACFAIL	\$XF	

**Notes**

1. X = The contents of vector base register 0.
2. Y = The contents of vector base register 1.
3. Refer to the Vector Base register description later in this chapter for recommended Vector Base register values.

**Local Bus Interrupter Status Register (bits 24-31)**

<b>ADR/SIZ</b>	<b>\$FFF40068 (8 bits of 32)</b>							
<b>BIT</b>	31	30	29	28	27	26	25	24
<b>NAME</b>	ACF	AB	SYSF	MWP	PE	VIIE	TIC2	TIC1
<b>OPER</b>	R	R	R	R	R	R	R	R
<b>RESET</b>	0 PSL	0 PSL	0 PSL	0 PSL	0 PSL	0 PSL	0 PSL	0 PSL

This register is the local bus interrupter status register. When an interrupt status bit is high, a local bus interrupt is being generated. When an interrupt status bit is low, a local interrupt is not being generated. The interrupt status bits are:

<b>TIC1</b>	Tick timer 1 interrupt.
<b>TIC2</b>	Tick timer 2 interrupt
<b>VIIE</b>	VMEbus IRQ1 edge-sensitive interrupt.
<b>PE</b>	Not used on MVME1X2P4.
<b>MWP</b>	VMEbus master write post error interrupt.
<b>SYSF</b>	VMEbus SYSFAIL interrupt.
<b>AB</b>	Not used on MVME1X2P4.
<b>ACF</b>	VMEbus ACFAIL interrupt.

**Local Bus Interrupter Status Register (bits 16-23)**

<b>ADR/SIZ</b>	<b>\$FFF40068 (8 bits of 32)</b>							
<b>BIT</b>	23	22	21	20	19	18	17	16
<b>NAME</b>	VIA	DMA	SIG3	SIG2	SIG1	SIG0	LM1	LM0
<b>OPER</b>	R	R	R	R	R	R	R	R
<b>RESET</b>	0 PSL	0 PSL	0 PSL	0 PSL	0 PSL	0 PSL	0 PSL	0 PSL

This register is the local bus interrupter status register. When an interrupt status bit is high, a local bus interrupt is being generated. When an interrupt status bit is low, a local interrupt is not being generated. The interrupt status bits are:

<b>LM0</b>	GCSR LM0 interrupt.
<b>LM1</b>	GCSR LM1 interrupt.
<b>SIG0</b>	GCSR SIG0 interrupt.
<b>SIG1</b>	GCSR SIG1 interrupt.
<b>SIG2</b>	GCSR SIG2 interrupt.
<b>SIG3</b>	GCSR SIG3 interrupt.
<b>DMA</b>	DMAC interrupt.
<b>VIA</b>	VMEbus interrupter acknowledge interrupt.

**Local Bus Interrupter Status Register (bits 8-15)**

<b>ADR/SIZ</b>	<b>\$FFF40068 (8 bits of 32)</b>							
<b>BIT</b>	15	14	13	12	11	10	9	8
<b>NAME</b>	SW7	SW6	SW5	SW4	SW3	SW2	SW1	SW0
<b>OPER</b>	R	R	R	R	R	R	R	R
<b>RESET</b>	0 PSL	0 PSL	0 PSL	0 PSL	0 PSL	0 PSL	0 PSL	0 PSL

This register is the local bus interrupter status register. When an interrupt status bit is high, a local bus interrupt is being generated. When an interrupt status bit is low, a local interrupt is not being generated. The interrupt status bits are:

- SW0**            Software 0 interrupt.
- SW1**            Software 1 interrupt.
- SW2**            Software 2 interrupt.
- SW3**            Software 3 interrupt.
- SW4**            Software 4 interrupt.
- SW5**            Software 5 interrupt.
- SW6**            Software 6 interrupt.
- SW7**            Software 7 interrupt.

**Local Bus Interrupter Status Register (bits 0-7)**

<b>ADR/SIZ</b>	<b>\$FFF40068 (8 bits of 32)</b>							
<b>BIT</b>	7	6	5	4	3	2	1	0
<b>NAME</b>	SPARE	VME7	VME6	VME5	VME4	VME3	VME2	VME1
<b>OPER</b>	R	R	R	R	R	R	R	R
<b>RESET</b>	0 PSL	0 PSL	0 PSL	0 PSL	0 PSL	0 PSL	0 PSL	0 PSL

This register is the local bus interrupter status register. When an interrupt status bit is high, a local bus interrupt is being generated. When an interrupt status bit is low, a local interrupt is not being generated. The interrupt status bits are:

**VME1** VMEbus IRQ1 Interrupt.

**VME2** VMEbus IRQ2 Interrupt.

**VME3** VMEbus IRQ3 Interrupt.

**VME4** VMEbus IRQ4 Interrupt.

**VME5** VMEbus IRQ5 Interrupt.

**VME6** VMEbus IRQ6 Interrupt.

**VME7** VMEbus IRQ7 Interrupt.

**SPARE** Not used.

**Local Bus Interrupter Enable Register (bits 24-31)**

<b>ADR/SIZ</b>	<b>\$FFF4006C (8 bits of 32)</b>							
<b>BIT</b>	31	30	29	28	27	26	25	24
<b>NAME</b>	EACF	EAB	ESYSF	EMWP	EPE	EVIIE	ETIC2	ETIC1
<b>OPER</b>	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
<b>RESET</b>	0 PSL	0 PSL	0 PSL	0 PSL	0 PSL	0 PSL	0 PSL	0 PSL

This register is the local bus interrupter enable register. When an enable bit is high, the corresponding interrupt is enabled. When an enable bit is low, the corresponding interrupt is disabled. The enable bit does not clear edge-sensitive interrupts or prevent the flip-flop from being set. If necessary, edge-sensitive interrupters should be cleared to remove any old interrupts and then re-enabled.

<b>ETIC1</b>	Enable tick timer 1 interrupt.
<b>ETIC2</b>	Enable tick timer 2 interrupt.
<b>EVIIE</b>	Enable VMEbus IRQ1 edge-sensitive interrupt.
<b>EPE</b>	Not used on MVME1X2P4.
<b>EMWP</b>	Enable VMEbus master write post error interrupt.
<b>ESYSF</b>	Enable VMEbus SYSFAIL interrupt.
<b>EAB</b>	Not used on MVME1X2P4.
<b>EACF</b>	Enable VMEbus ACFAIL interrupt.

**Local Bus Interrupter Enable Register (bits 16-23)**

<b>ADR/SIZ</b>	<b>\$FFF4006C (8 bits of 32)</b>							
<b>BIT</b>	23	22	21	20	19	18	17	16
<b>NAME</b>	EVIA	EDMA	ESIG3	ESIG2	ESIG1	ESIG0	ELM1	ELM0
<b>OPER</b>	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
<b>RESET</b>	0 PSL	0 PSL	0 PSL	0 PSL	0 PSL	0 PSL	0 PSL	0 PSL

This register is the local bus interrupter enable register. When an enable bit is high, the corresponding interrupt is enabled. When an enable bit is low, the corresponding interrupt is disabled. The enable bit does not clear edge-sensitive interrupts or prevent the flip-flop from being set. If necessary, edge-sensitive interrupters should be cleared to remove any old interrupts and then re-enabled.

<b>ELM0</b>	Enable GCSR LM0 interrupt.
<b>ELM1</b>	Enable GCSR LM1 interrupt.
<b>ESIG0</b>	Enable GCSR SIG0 interrupt.
<b>ESIG1</b>	Enable GCSR SIG1 interrupt.
<b>ESIG2</b>	Enable GCSR SIG2 interrupt.
<b>ESIG3</b>	Enable GCSR SIG3 interrupt.
<b>EDMA</b>	Enable DMAC interrupt.
<b>EVIA</b>	VMEbus interrupter acknowledge interrupt.



**Local Bus Interrupter Enable Register (bits 8-15)**

<b>ADR/SIZ</b>	<b>\$FFF4006C (8 bits of 32)</b>							
<b>BIT</b>	15	14	13	12	11	10	9	8
<b>NAME</b>	ESW7	ESW6	ESW5	ESW4	ESW3	ESW2	ESW1	ESW0
<b>OPER</b>	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
<b>RESET</b>	0 PSL	0 PSL	0 PSL	0 PSL	0 PSL	0 PSL	0 PSL	0 PSL

This is the local bus interrupter enable register. When an enable bit is high, the corresponding interrupt is enabled. When an enable bit is low, the corresponding interrupt is disabled. The enable bit does not clear edge-sensitive interrupts or prevent the flip-flop from being set. If necessary, edge-sensitive interrupters should be cleared to remove any old interrupts and then re-enabled.

- ESW0**            Enable software 0 interrupt.
- ESW1**            Enable software 1 interrupt.
- ESW2**            Enable software 2 interrupt.
- ESW3**            Enable software 3 interrupt.
- ESW4**            Enable software 4 interrupt.
- ESW5**            Enable software 5 interrupt.
- ESW6**            Enable software 6 interrupt.
- ESW7**            Enable software 7 interrupt.

**Local Bus Interrupter Enable Register (bits 0-7)**

<b>ADR/SIZ</b>	<b>\$FFF4006C (8 bits of 32)</b>							
<b>BIT</b>	7	6	5	4	3	2	1	0
<b>NAME</b>	SPARE	EIRQ7	EIRQ6	EIRQ5	EIRIQ4	EIRQ3	EIRQ2	EIRQ1
<b>OPER</b>	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
<b>RESET</b>	0 PSL	0 PSL	0 PSL	0 PSL	0 PSL	0 PSL	0 PSL	0 PSL

This is the local bus interrupter enable register. When an enable bit is high, the corresponding interrupt is enabled. When an enable bit is low, the corresponding interrupt is disabled. The enable bit does not clear edge-sensitive interrupts or prevent the flip-flop from being set. If necessary, edge-sensitive interrupters should be cleared to remove any old interrupts and then re-enabled.

<b>EIRQ1</b>	Enable VMEbus IRQ1 interrupt.
<b>EIRQ2</b>	Enable VMEbus IRQ2 interrupt.
<b>EIRQ3</b>	Enable VMEbus IRQ3 interrupt.
<b>EIRQ4</b>	Enable VMEbus IRQ4 interrupt.
<b>EIRQ5</b>	Enable VMEbus IRQ5 interrupt.
<b>EIRQ6</b>	Enable VMEbus IRQ6 interrupt.
<b>EIRQ7</b>	Enable VMEbus IRQ7 interrupt.
<b>SPARE</b>	SPARE.

**Software Interrupt Set Register (bits 8-15)**

ADR/SIZ	\$FFF40070 (8 bits of 32)							
<b>BIT</b>	15	14	13	12	11	10	9	8
<b>NAME</b>	SSW7	SSW6	SSW5	SSW4	SSW3	SSW2	SSW1	SSW0
<b>OPER</b>	S	S	S	S	S	S	S	S
<b>RESET</b>	0 PSL	0 PSL	0 PSL	0 PSL	0 PSL	0 PSL	0 PSL	0 PSL

This register is used to set the software interrupts. An interrupt is set by writing a 1 to it. The software interrupt set bits are:

**SSW0** Set software 0 interrupt.

**SSW1** Set software 1 interrupt.

**SSW2** Set software 2 interrupt.

**SSW3** Set software 3 interrupt.

**SSW4** Set software 4 interrupt.

**SSW5** Set software 5 interrupt.

**SSW6** Set software 6 interrupt.

**SSW7** Set software 7 interrupt.

**Interrupt Clear Register (bits 24-31)**

ADR/SIZ	\$FFF40074 (8 bits of 32)							
<b>BIT</b>	31	30	29	28	27	26	25	24
<b>NAME</b>	CACF	CAB	CSYSF	CMWP	CPE	CVIIE	CTIC2	CTIC1
<b>OPER</b>	C	C	C	C	C	C	C	C
<b>RESET</b>	0 PSL	0 PSL	0 PSL	0 PSL	0 PSL	0 PSL	0 PSL	0 PSL

This register is used to clear the edge-sensitive interrupts. An interrupt is cleared by writing a 1 to its clear bit. The clear bits are defined below.

**CTIC1** Clear tick timer 1 interrupt.

**CTIC2** Clear tick timer 2 interrupt.

<b>CVIIE</b>	Clear VMEbus IRQ1 edge-sensitive interrupt.
<b>CPE</b>	Not used on MVME1X2P4.
<b>CMWP</b>	Clear VMEbus master write post error interrupt.
<b>CSYSF</b>	Clear VMEbus SYSFAIL interrupt.
<b>CAB</b>	Not used on MVME1X2P4.
<b>CACF</b>	Clear VMEbus ACFAIL interrupt.

### Interrupt Clear Register (bits 16-23)

<b>ADR/SIZ</b>	<b>\$FFF40074 (8 bits of 32)</b>							
<b>BIT</b>	23	22	21	20	19	18	17	16
<b>NAME</b>	CVIA	CDMA	CSIG3	CSIG2	CSIG1	CSIG0	CLM1	CLM0
<b>OPER</b>	C	C	C	C	C	C	C	C
<b>RESET</b>	X	X	X	X	X	X	X	X

This register is used to clear the edge-sensitive interrupts. An interrupt is cleared by writing a 1 to its clear bit. The clear bits are defined below.

<b>CLM0</b>	Clear GCSR LM0 interrupt.
<b>CLM1</b>	Clear GCSR LM1 interrupt.
<b>CSIG0</b>	Clear GCSR SIG0 interrupt.
<b>CSIG1</b>	Clear GCSR SIG1 interrupt.
<b>CSIG2</b>	Clear GCSR SIG2 interrupt.
<b>CSIG3</b>	Clear GCSR SIG3 interrupt.
<b>CDMA</b>	Clear DMA controller interrupt.
<b>CVIA</b>	Clear VMEbus interrupter acknowledge interrupt.

**Interrupt Clear Register (bits 8-15)**

ADR/SIZ	\$FFF40074 (8 bits of 32)							
<b>BIT</b>	15	14	13	12	11	10	9	8
<b>NAME</b>	CSW7	CSW6	CSW5	CSW4	CSW3	CSW2	CSW1	CSW0
<b>OPER</b>	C	C	C	C	C	C	C	C
<b>RESET</b>	X	X	X	X	X	X	X	X

This register is used to clear the edge software interrupts. An interrupt is cleared by writing a 1 to its clear bit. The clear bits are:

**CSW0** Clear software 0 interrupt.

**CSW1** Clear software 1 interrupt.

**CSW2** Clear software 2 interrupt.

**CSW3** Clear software 3 interrupt.

**CSW4** Clear software 4 interrupt.

**CSW5** Clear software 5 interrupt.

**CSW6** Clear software 6 interrupt.

**CSW7** Clear software 7 interrupt.

**Interrupt Level Register 1 (bits 24-31)**

ADR/SIZ	\$FFF40078 (8 bits [6 used] of 32)							
<b>BIT</b>	31	30	29	28	27	26	25	24
<b>NAME</b>	ACF LEVEL				AB LEVEL			
<b>OPER</b>	R/W				R/W			
<b>RESET</b>	0 PSL				0 PSL			

This register is used to define the level of the abort interrupt and the ACFAIL interrupt.

**AB LEVEL** Not used on MVME1X2P4.

**ACF LEVEL** These bits define the level of the ACFAIL interrupt.

**Interrupt Level Register 1 (bits 16-23)**

ADR/SIZ	\$FFF40078 (8 bits [6 used] of 32)							
BIT	23	22	21	20	19	18	17	16
NAME	SYSF LEVEL				WPE LEVEL			
OPER	R/W				R/W			
RESET	0 PSL				0 PSL			

This register is used to define the level of the SYSFAIL interrupt and the master write post bus error interrupt.

**WPE LEVEL** These bits define the level of the master write post bus error interrupt.

**SYSF LEVEL** These bits define the level of the SYSFAIL interrupt.

**Interrupt Level Register 1 (bits 8-15)**

ADR/SIZ	\$FFF40078 (8 bits [6 used] of 32)							
BIT	15	14	13	12	11	10	9	8
NAME	PE LEVEL				IRQ1E LEVEL			
OPER	R/W				R/W			
RESET	0 PSL				0 PSL			

This register is used to define the level of the VMEbus IRQ1 edge-sensitive interrupt and the level of the external (parity error) interrupt.

**IRQ1E LEVEL**

These bits define the level of the VMEbus IRQ1 edge-sensitive interrupt.

**PE LEVEL** Used if the ECC memory model is enabled. Not used in the "parity" model. Refer to [Chapter 5, MCECC Functions](#) for interrupt setup details.

**Interrupt Level Register 1 (bits 0-7)**

ADR/SIZ	\$FFF40078 (8 bits [6 used] of 32)							
BIT	7	6	5	4	3	2	1	0
NAME	TICK2 LEVEL				TICK1 LEVEL			
OPER	R/W				R/W			
RESET	0 PSL				0 PSL			

This register is used to define the level of the tick timer 1 interrupt and the tick timer 2 interrupt.

**TICK1 LEVEL**

These bits define the level of the tick timer 1 interrupt.

**TICK2 LEVEL**

These bits define the level of the tick timer 2 interrupt.

**Interrupt Level Register 2 (bits 24-31)**

ADR/SIZ	\$FFF4007C (8 bits [6 used] of 32)							
BIT	31	30	29	28	27	26	25	24
NAME	VIA LEVEL				DMA LEVEL			
OPER	R/W				R/W			
RESET	0 PSL				0 PSL			

This register is used to define the level of the DMA controller interrupt and the VMEbus acknowledge interrupt.

**DMA LEVEL** These bits define the level of the DMA controller interrupt.

**VIA LEVEL** These bits define the level of the VMEbus interrupter acknowledge interrupt.

**Interrupt Level Register 2 (bits 16-23)**

ADR/SIZ	\$FFF4007C (8 bits [6 used] of 32)							
BIT	23	22	21	20	19	18	17	16
NAME	SIG3 LEVEL				SIG2 LEVEL			
OPER	R/W				R/W			
RESET	0 PSL				0 PSL			

This register is used to define the level of the GCSR SIG2 interrupt and the GCSR SIG3 interrupt.

**SIG2 LEVEL** These bits define the level of the GCSR SIG2 interrupt.

**SIG3 LEVEL** These bits define the level of the GCSR SIG3 interrupt.

**Interrupt Level Register 2 (bits 8-15)**

ADR/SIZ	\$FFF4007C (8 bits [6 used] of 32)							
BIT	15	14	13	12	11	10	9	8
NAME	SIG1 LEVEL				SIG0 LEVEL			
OPER	R/W				R/W			
RESET	0 PSL				0 PSL			

This register is used to define the level of the GCSR SIG0 interrupt and the GCSR SIG1 interrupt.

**SIG0 LEVEL** These bits define the level of the GCSR SIG0 interrupt.

**SIG1 LEVEL** These bits define the level of the GCSR SIG1 interrupt.



**Interrupt Level Register 2 (bits 0-7)**

ADR/SIZ	\$FFF4007C (8 bits [6 used] of 32)							
BIT	7	6	5	4	3	2	1	0
NAME	LM1 LEVEL				LM0 LEVEL			
OPER	R/W				R/W			
RESET	0 PSL				0 PSL			

This register is used to define the level of the GCSR LM0 interrupt and the GCSR LM1 interrupt.

**LM0 LEVEL** These bits define the level of the GCSR LM0 interrupt.

**LM1 LEVEL** These bits define the level of the GCSR LM1 interrupt.

**Interrupt Level Register 3 (bits 24-31)**

ADR/SIZ	\$FFF40080 (8 bits [6 used] of 32)							
BIT	31	30	29	28	27	26	25	24
NAME	SW7 LEVEL				SW6 LEVEL			
OPER	R/W				R/W			
RESET	0 PSL				0 PSL			

This register is used to define the level of the software 6 interrupt and the software 7 interrupt.

**SW6 LEVEL** These bits define the level of the software 6 interrupt.

**SW7 LEVEL** These bits define the level of the software 7 interrupt.

**Interrupt Level Register 3 (bits 16-23)**

ADR/SIZ	\$FFF40080 (8 bits [6 used] of 32)							
BIT	23	22	21	20	19	18	17	16
NAME	SW5 LEVEL				SW4 LEVEL			
OPER	R/W				R/W			
RESET	0 PSL				0 PSL			

This register is used to define the level of the software 4 interrupt and the software 5 interrupt.

**SW4 LEVEL** These bits define the level of the software 4 interrupt.

**SW5 LEVEL** These bits define the level of the software 5 interrupt.

**Interrupt Level Register 3 (bits 8-15)**

ADR/SIZ	\$FFF40080 (8 bits [6 used] of 32)							
BIT	15	14	13	12	11	10	9	8
NAME	SW3 LEVEL				SW2 LEVEL			
OPER	R/W				R/W			
RESET	0 PSL				0 PSL			

This register is used to define the level of the software 2 interrupt and the software 3 interrupt.

**SW2 LEVEL** These bits define the level of the software 2 interrupt.

**SW3 LEVEL** These bits define the level of the software 3 interrupt.

**Interrupt Level Register 3 (bits 0-7)**

ADR/SIZ	\$FFF40080 (8 bits [6 used] of 32)							
BIT	7	6	5	4	3	2	1	0
NAME	SW1 LEVEL				SW0 LEVEL			
OPER	R/W				R/W			
RESET	0 PSL				0 PSL			

This register is used to define the level of the software 0 interrupt and the software 1 interrupt.

**SW0 LEVEL** These bits define the level of the software 0 interrupt.

**SW1 LEVEL** These bits define the level of the software 1 interrupt.

**Interrupt Level Register 4 (bits 24-31)**

ADR/SIZ	\$FFF40084 (8 bits [6 used] of 32)							
BIT	31	30	29	28	27	26	25	24
NAME	SPARE LEVEL				VIRQ7 LEVEL			
OPER	R/W				R/W			
RESET	0 PSL				0 PSL			

This register is used to define the level of the VMEbus IRQ7 interrupt and the spare interrupt. The VMEbus level 7 (IRQ7) interrupt may be mapped to any local bus interrupt level.

**VIRQ7 LEVEL**

These bits define the level of the VMEbus IRQ7 interrupt.

**SPARE LEVEL**

Not used on the MVME1X2P4.

**Interrupt Level Register 4 (bits 16-23)**

ADR/SIZ	\$FFF40084 (8 bits [6 used] of 32)							
BIT	23	22	21	20	19	18	17	16
NAME	VIRQ6				VIRQ5 LEVEL			
OPER	R/W				R/W			
RESET	0 PSL				0 PSL			

This register is used to define the level of the VMEbus level 5 (IRQ5) interrupt and the VMEbus level 6 (IRQ6) interrupt. The IRQ5 and IRQ6 interrupts may be mapped to any local bus interrupt level.

**VIRQ5 LEVEL**

These bits define the level of the VMEbus IRQ5 interrupt.

**VIRQ6 LEVEL**

These bits define the level of the VMEbus IRQ6 interrupt.

**Interrupt Level Register 4 (bits 8-15)**

ADR/SIZ	\$FFF40084 (8 bits [6 used] of 32)							
BIT	15	14	13	12	11	10	9	8
NAME	VIRQ4				VIRQ3 LEVEL			
OPER	R/W				R/W			
RESET	0 PSL				0 PSL			

This register is used to define the level of the VMEbus level 3 (IRQ3) interrupt and the VMEbus level 4 (IRQ4) interrupt. The IRQ3 and IRQ4 interrupts may be mapped to any local bus interrupt level.

**VIRQ3 LEVEL**

These bits define the level of the VMEbus IRQ3 interrupt.

**VIRQ4 LEVEL**

These bits define the level of the VMEbus IRQ4 interrupt.

**Interrupt Level Register 4 (bits 0-7)**

ADR/SIZ	\$FFF40084 (8 bits [6 used] of 32)							
BIT	7	6	5	4	3	2	1	0
NAME	VIRQ2				VIRQ1 LEVEL			
OPER	R/W				R/W			
RESET	0 PSL				0 PSL			

This register is used to define the level of the VMEbus level 1 (IRQ1) interrupt and the VMEbus level 2 (IRQ2) interrupt. The IRQ1 and IRQ2 interrupts may be mapped to any local bus interrupt level.

**VIRQ1 LEVEL**

These bits define the level of the VMEbus IRQ1 interrupt.

**VIRQ2 LEVEL**

These bits define the level of the VMEbus IRQ2 interrupt.

**Vector Base Register**

ADR/SIZ	\$FFF40088 (8 bits of 32)							
BIT	31	30	29	28	27	26	25	24
NAME	VBR 0				VBR 1			
OPER	R/W				R/W			
RESET	0 PSL				0 PSL			

This register is used to define the interrupt base vectors.

**VBR 1** These bits define the interrupt base vector 1.

**VBR 0** These bits define the interrupt base vector 0.

**Note** Refer to [Table 2-4, Local Bus Interrupter Summary](#), for further information.

A suggested setting for the VMEchip2 Vector Base register is: VBR0 = 6, VBR1 = 7 (i.e., setting the Vector Base register at address \$FFF40088 to \$67xxxxxx). This produces a Vector Base0 of \$60 corresponding to the “X” in [Table 2-4](#), and a Vector Base1 of \$70 corresponding to the “Y” in [Table 2-4](#).

## I/O Control Register 1

ADR/SIZ	\$FFF40088 (8 bits of 32)							
BIT	23	22	21	20	19	18	17	16
NAME	MIEN	SYSFL	ACFL	ABRTL	GPOEN3	GPOEN2	GPOEN1	GPOEN0
OPER	R/W	R	R	R	R/W	R/W	R/W	R/W
RESET	0 PSL	X	X	X	0 PS	0 PS	0 PS	0 PS

This register is a general purpose I/O control register. Bits 16-19 control the direction of the four General Purpose I/O pins (GPIO0-3).

- GPOEN0** When this bit is low, the GPIO0 pin is an input.  
When this bit is high, the GPIO0 pin is an output.
- GPOEN1** When this bit is low, the GPIO1 pin is an input.  
When this bit is high, the GPIO1 pin is an output.
- GPOEN2** When this bit is low, the GPIO2 pin is an input.  
When this bit is high, the GPIO2 pin is an output.
- GPOEN3** When this bit is low, the GPIO3 pin is an input.  
When this bit is high, the GPIO3 pin is an output.
- ABRTL** This bit indicates the status of the **ABORT** switch.  
When this bit is high, the **ABORT** switch is depressed.  
When this bit is low, the **ABORT** switch is not depressed.
- ACFL** This bit indicates the status of the ACFAIL signal line on the VMEbus. When this bit is high, the ACFAIL signal line is active. When this bit is low, the ACFAIL signal line is not active.
- SYSFL** This bit indicates the status of the SYSFAIL signal line on the VMEbus. When this bit is high, the SYSFAIL signal line is active. When this bit is low, the SYSFAIL signal line is not active.
- MIEN** When this bit is low, all interrupts controlled by the VMEchip2 are masked. When this bit is high, all interrupts controlled by the VMEchip2 are not masked.

**I/O Control Register 2**

ADR/SIZ	\$FFF40088 (8 bits of 32)							
BIT	15	14	13	12	11	10	9	8
NAME	GPIOO3	GPIOO2	GPIOO1	GPIOO0	GPIOI3	GPIOI2	GPIOI1	GPIOI0
OPER	R/W	R/W	R/W	R/W	R	R	R	R
RESET	0 PSL	0 PS	0 PS	0 PS	X	X	X	X

GPIOO1 Connects to pin 16 of the Remote Status and Control register.

GPIOO2 Connects to pin 17 of the Remote Status and Control register.

GPIOO3 Connects to pin 18 of the Remote Status and Control register.

GPIOI1 Not used.

GPIOI2 Not used.

GPIOI3 Not used.

**I/O Control Register 3**

ADR/SIZ	\$FFF40088 (8 bits of 32)							
BIT	7	6	5	4	3	2	1	0
NAME	GPI7	GPI6	GPI5	GPI4	GPI3	GPI2	GPI1	GPI0
OPER	R	R	R	R	R	R	R	R
RESET	X	X	X	X	X	X	X	X

This function is not used on the MVME1X2P4.

## Miscellaneous Control Register

ADR/SIZ	\$FFF4008C (8 bits of 32)							
BIT	7	6	5	4	3	2	1	0
NAME	MPIRQEN	REVEROM	DISSRAM	DISMST	NOELBBSY	DISBSYT	ENINT	DISBGN
OPER	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RESET	0 PSL	0 PSL	0 PSL	0 PS	0 PS	0 PS	0 PS	0 PS

**DISBGN** When this bit is high, the VMEbus BGIN filters are disabled. When this bit is low, the VMEbus BGIN filters are enabled. This bit should not be set.

**ENINT** When this bit is high, the local bus interrupt filters are enabled. When this bit is low, the local bus interrupt filters are disabled. This bit should not be set.

**DISBSYT** When this bit is low, the minimum VMEbus BBSY\* time when the local bus master has been retried off the local bus is 32 local bus clocks. When this bit is high, the minimum VMEbus BBSY\* time when the local bus master has been retried off the local bus is 3 local bus clocks.

When a local bus master attempts to access the VMEbus and a VMEbus master attempts to access the local bus, a deadlock is created. The VMEchip2 detects this condition and requests the local bus master to give up the local bus and retry the cycle. This allows the VMEbus master to complete the cycle to the local bus. If the VMEchip2 receives VMEbus mastership, the local master has not returned from the retry, and this bit is high, VMEchip2 drives VMEbus BBSY\* for the minimum time (about 90 ns) and then releases the VMEbus. If the local master does not return from the retry within this 90 ns window, the board loses its turn on the VMEbus. If the VMEchip2 receives VMEbus mastership, the local master has not returned from the retry, and this bit is low, VMEchip2 drives VMEbus BBSY\* for a minimum of 32 local bus clocks, which allows the local bus master time to return



from the retry and the board does not lose its turn on the VMEbus. For this reason, it is recommended that this bit remain low.

- NOELBBSY** When this bit is high, the early release feature of bus busy feature on the VMEbus is disabled. The VMEchip2 drives BBSY\* low whenever VMEbus AS\* is low. When this bit is low, the early release feature of bus busy feature on the VMEbus is not disabled.
- DISMST** When this bit is high, the VME LED on the MVME1X2P4 illuminates on assertion of Local Bus Reset or when the VMEchip2 ASIC is driving Local Bus Busy. When this bit is low, the VME LED on the MVME1X2P4 illuminates on assertion of Local Bus Reset, when the VMEchip2 is driving Local Bus Busy, or when the VMEchip2 is driving the VMEbus address strobe.
- DISSRAM** When this bit is high, the SRAM decoder in the VMEchip2 is disabled. When this bit is low, the SRAM decoder in the VMEchip2 is enabled. Because the SRAM decoder in the VMEchip2 is not used on the MVME1X2P4, this bit must be set.
- REVEROM** This function is not used on the MVME1X2P4. This bit must not be set.
- MPIRQEN** This function is not used on the MVME1X2P4. This bit must not be set.

## GCSR Programming Model

This section describes the programming model for the Global Control and Status Registers (GCSR) in the VMEchip2. The local bus map decoder for the GCSR registers is included in the VMEchip2. The local bus base address for the GCSR is \$FFF40100. The registers in the GCSR are 16 bits wide and they are byte accessible from both the VMEbus and the local bus. The GCSR is located in the 16-bit VMEbus short I/O space and it responds to address modifier codes \$29 or \$2D. The address of the GCSR as viewed from the VMEbus depends upon the GCSR group select value *XX* and GCSR board select value *Y* programmed in the LCSR. The board value *Y* may be \$0 through \$E, allowing 15 boards in one group. The value \$F is reserved for the location monitors.

The VMEchip2 includes four location monitors (LM0-LM3). The location monitors provide a broadcast signaling capability on the VMEbus. When a location monitor address is generated on the VMEbus, all location monitors in the group are cleared. The signal interrupts SIG0-SIG3 should be used to signal individual boards. The location monitors are located in the VMEbus short I/O space and the specific address is determined by the VMEchip2 group address. The location monitors LM0-LM3 are located at addresses \$XXF1, \$XXF3, \$XXF5, and \$XXF7 respectively. A location monitor cycle on the VMEbus is generated by a read or write to VMEbus short I/O address \$XXFN, where *XX* is the group address and *N* is the specific location monitor address. When the VMEchip2 generates a location monitor cycle to the VMEbus, within its own group, the VMEchip2 DTACKs itself. A VMEchip2 cannot DTACK location monitor cycles to other groups.

The GCSR section of the VMEchip2 contains the following registers: a *Chip ID register*, a *Chip Revision register*, a *Location Monitor Status register*, an *Interrupt Control register*, a *Board Control register*, and six *General Purpose registers*.

The *Chip ID* and *Revision registers* are provided to allow software to determine the ID of the chip and its revision level. The VMEchip2 has a chip ID of ten. ID codes 0 and 1 are used by the old VMEchip. The initial revision of the VMEchip2 is 0. If mask changes are required, the revision level is incremented.

The *Location Monitor Status register* provides the status of the location monitors. A location monitor bit is cleared when the VMEchip2 detects a VMEbus cycle to the corresponding location monitor address. When the LM0 or LM1 bits are cleared, an interrupt is set to the local bus interrupter. If the LM0 or LM1 interrupt is enabled in the local bus interrupter, then a local bus interrupt is generated. The location monitor bits are set by writing a 1 to the corresponding bit in the location monitor register. LM0 and LM1 can also be set by writing a 1 to the corresponding clear bits in the local interrupt clear register.

The *Interrupt Control register* provides four bits that allow the VMEbus to interrupt the local bus. An interrupt is sent to the local bus interrupter when one of the bits is set. If the interrupt is enabled in the local bus interrupter, then a local bus interrupt is generated. The interrupt bits are cleared by writing a 1 to the corresponding bit in the interrupt clear register.

The *Board Control register* allows a VMEbus master to reset the local bus, prevent the VMEchip2 from driving the SYSFAIL signal line, and detect if the VMEchip2 wants to drive the SYSFAIL signal line.

The six *General Purpose registers* can be read and written from both the local bus and the VMEbus. These registers are provided to allow local bus masters to communicate with VMEbus masters. The function of these registers is not defined by this specification. The GCSR supports read-modify-write cycles such as TAS.



The GCSR allows a VMEbus master to reset the local bus. This feature is very dangerous and should be used with caution.

The local reset feature is a partial system reset, not a complete system reset such as powerup reset or SYSRESET. When the local bus reset signal is asserted, a local bus cycle may be aborted. The VMEchip2 is connected to both the local bus and the VMEbus and if the aborted cycle is bound for the VMEbus, erratic operation may result.

Communications between the local processor and a VMEbus master should use interrupts or mailbox locations; reset should not be used in normal communications. Reset should be used only when the local processor is halted or the local bus is hung and reset is the last resort.

## Programming the GCSR

A complete description of the GCSR appears in the following tables. Each register definition includes a table with five lines.

1. The base address of the register and the number of bits defined in the table.
2. The bits defined by this table.
3. The name of the register or the name of the bits in the register.
4. The operations possible on the register bits, defined as follows:

**R**      This bit is a read-only status bit.  
**R/W**    This bit is readable and writable.  
**S/R**    Writing a 1 to this bit sets it. Reading it returns its current status.

5. The state of the bit following a reset, defined as follows:

**P**      This bit is affected by power-up reset.  
**S**      The bit is affected by SYSRESET.  
**L**      The bit is affected by local bus reset.  
**X**      The bit is not affected by reset.

Table 2-5 shows a summary of the GCSR.

**Table 2-5. VMEchip2 Memory Map (GCSR Summary)**

VMEchip2 GCSR Base Address = \$FFF40100

Offsets		Bit Numbers															
VM E bus	Local Bus	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	Chip Revision								Chip ID							
2	4	LM 3	LM 2	LM 1	LM 0	SI G3	SI G2	SI G1	SI G0	RS T	IS F	B F	SCO N	SYSF L	X	X	X
4	8	General Purpose Control and Status Register 0															
6	C	General Purpose Control and Status Register 1															
8	10	General Purpose Control and Status Register 2															
A	14	General Purpose Control and Status Register 3															
C	18	General Purpose Control and Status Register 4															
E	1C	General Purpose Control and Status Register 5															

### VMEchip2 Revision Register

ADR/SIZ	Local Bus: \$FFF40100/VMEbus: \$XXY0 (8 bits)		
BIT	15	...	8
NAME	VMEchip2 Revision Register		
OPER	R		
RESET	01 PS		

This register is the VMEchip2 revision register. The revision level for the VMEchip2 starts at 0 and is incremented if mask changes are required. The VMEchip2 used on the MVME1X2P4 is revision \$01 or greater.

**VMEchip2 ID Register**

ADR/SIZ	Local Bus: \$FFF40100/VMEbus: \$XXY0 (8 bits)		
BIT	7	...	0
NAME	VMEchip2 ID Register		
OPER	R		
RESET	10 PS		

This register is the VMEchip2 ID register. The ID for the VMEchip2 is 10.

**VMEchip2 LM/SIG Register**

ADR/SIZ	Local Bus: \$FFF40104/VMEbus: \$XXY2 (8 bits)							
BIT	15	14	13	12	11	10	9	8
NAME	LM3	LM2	LM1	LM0	SIG3	SIG2	SIG1	SIG0
OPER	R	R	R	R	S/R	S/R	S/R	S/R
RESET	1 PS	1 PS	1 PS	1 PS	0 PS	0 PS	0 PS	0 PS

This register is the VMEchip2 location monitor register and the interrupt register.

**SIG0**            The SIG0 bit is set when a VMEbus master writes a 1 to it. When the SIG0 bit is set, an interrupt is sent to the local bus interrupter. The SIG0 bit is cleared when the local processor writes a 1 to the SIG0 bit in this register or the CSIG0 bit in the local interrupt clear register.

**SIG1**            The SIG1 bit is set when a VMEbus master writes a 1 to it. When the SIG1 bit is set, an interrupt is sent to the local bus interrupter. The SIG1 bit is cleared when the local processor writes a 1 to the SIG1 bit in this register or the CSIG1 bit in the local interrupt clear register.

**SIG2**            The SIG2 bit is set when a VMEbus master writes a 1 to it. When the SIG2 bit is set, an interrupt is sent to the local bus interrupter. The SIG2 bit is cleared when the local processor writes a 1 to the SIG2 bit in this register or the CSIG2 bit in the local interrupt clear register.

- SIG3** The SIG3 bit is set when a VMEbus master writes a 1 to it. When the SIG3 bit is set, an interrupt is sent to the local bus interrupter. The SIG3 bit is cleared when the local processor writes a 1 to the SIG3 bit in this register or the CSIG3 bit in the local interrupt clear register.
- LM0** This bit is cleared by an LM0 cycle on the VMEbus. When this bit is cleared, an interrupt is set to the local bus interrupter. This bit is set when the local processor or a VMEbus master writes a 1 to the LM0 bit in this register or the CLM0 bit in local interrupt clear register.
- LM1** This bit is cleared by an LM1 cycle on the VMEbus. When this bit is cleared, an interrupt is set to the local bus interrupter. This bit is set when the local processor or a VMEbus master writes a 1 to the LM1 bit in this register or the CLM1 bit in local interrupt clear register.
- LM2** This bit is cleared by an LM2 cycle on the VMEbus. This bit is set when the local processor or a VMEbus master writes a 1 to the LM0 bit in this register.
- LM3** This bit is cleared by an LM3 cycle on the VMEbus. This bit is set when the local processor or a VMEbus master writes a 1 to the LM3 bit in this register.

## VMEchip2 Board Status/Control Register

ADR/SIZ	Local Bus: \$FFF40104/VMEbus: \$XXY2 (8 bits [5 used])							
BIT	7	6	5	4	3	2	1	0
NAME	RST	ISF	BF	SCON	SYSFL			
OPER	S/R	R/W	R	R	R			
RESET	0 PSL	0 PSL	1 PS	X	1 PSL			

This register is the VMEchip2 board status/control register.

- SYSFL** This bit is set when the VMEchip2 is driving the SYSFAIL signal.
- SCON** This bit is set if the VMEchip2 is system controller.
- BF** When this bit is high, the Board Fail signal is active. When this bit is low, the Board Fail signal is inactive. When this bit is set, the VMEchip2 drives SYSFAIL if the inhibit SYSFAIL bit is not set.
- ISF** When this bit is set, the VMEchip2 is prevented from driving the VMEbus SYSFAIL signal line. When this bit is cleared, the VMEchip2 is allowed to drive the VMEbus SYSFAIL signal line.
- RST** This bit allows a VMEbus master to reset the local bus. Refer to the note on local reset in the *GCSR Programming Model* section, earlier in this chapter. When this bit is set, a local bus reset is generated. This bit is cleared by the local bus reset.



**General Purpose Register 0**

ADR/SIZ	Local Bus: \$FFF40108/VMEbus: \$XXY4 (16 bits)		
BIT	15	...	0
NAME	General Purpose Register 0		
OPER	R/W		
RESET	0 PS		

This register is a general purpose register that allows a local bus master to communicate with a VMEbus master. The function of this register is not defined by the hardware specification.

**General Purpose Register 1**

ADR/SIZ	Local Bus: \$FFF4010C/VMEbus: \$XXY6 (16 bits)		
BIT	15	...	0
NAME	General Purpose Register 1		
OPER	R/W		
RESET	0 PS		

This register is a general purpose register that allows a local bus master to communicate with a VMEbus master. The function of this register is not defined by the hardware specification.

**General Purpose Register 2**

ADR/SIZ	Local Bus: \$FFF40110/VMEbus: \$XXY8 (16 bits)		
BIT	15	...	0
NAME	General Purpose Register 2		
OPER	R/W		
RESET	0 PS		

This register is a general purpose register that allows a local bus master to communicate with a VMEbus master. The function of this register is not defined by the hardware specification.

### General Purpose Register 3

ADR/SIZ	Local Bus: \$FFF40114/VMEbus: \$XXYA (16 bits)		
BIT	15	...	0
NAME	General Purpose Register 3		
OPER	R/W		
RESET	0 PS		

This register is a general purpose register that allows a local bus master to communicate with a VMEbus master. The function of this register is not defined by the hardware specification.

### General Purpose Register 4

ADR/SIZ	Local Bus: \$FFF40118/VMEbus: \$XXYC (16 bits)		
BIT	15	...	0
NAME	General Purpose Register 4		
OPER	R/W		
RESET	0 PS		

This register is a general purpose register that allows a local bus master to communicate with a VMEbus master. The function of this register is not defined by the hardware specification.

### General Purpose Register 5

ADR/SIZ	Local Bus: \$FFF4011C/VMEbus: \$XXYE (16 bits)		
BIT	15	...	0
NAME	General Purpose Register 5		
OPER	R/W		
RESET	0 PS		

This register is a general purpose register that allows a local bus master to communicate with a VMEbus master. The function of this register is not defined by the hardware specification.

# MC2 Functions 3

---

## Introduction

The Memory Controller ASIC (MC2 chip) is one of three ASICs (the MC2 chip, the IP2 chip, and the MCECC chip) used on earlier MVME162/172 models whose functions are now incorporated into a single ASIC, the Petra chip, on the MVME1X2P4.

For ease of use with programming models and documentation developed for earlier boards, however, the structure of this manual preserves the functional distinctions that formerly characterized those three ASICs. This chapter describes the MC2 functionality of the Petra chip.

## Summary of Major Features

MC2 functions now implemented on the Petra chip include:

**Table 3-1. MC2 Functions on the Petra ASIC**

Function	Features
Memory Control	BBRAM and time-of-day clock (M48T58) interface with bus sizing
	EPROM interface with bus sizing
	Flash interface with bus sizing
	SRAM controller supporting several configurations
	DRAM controller supporting several configurations
I/O Interfacing	Four Zilog serial interfaces implemented with Z85230 SCC device
	NCR 53C710 SCSI Coprocessor interface
	Intel 82596CA LAN Coprocessor interface
Timers	Four 32-bit tick timers
	Local bus access timer
	Watchdog timer
Interrupt Handling	Interrupt support for <b>ABORT</b> switch, LAN, SCSI, SCC, DRAM, and timers

## Functional Description

The following sections give an overview of the functions implemented by the Petra/MC2 chip. A detailed programming model for control and status registers in the Petra/MC2 chip appears in a later section.

### Petra/MC2 Chip Initialization

The Petra/MC2 chip accommodates several memory configurations and MVME1X2P4 population versions. Configuration registers are used to initialize the MVME1X2P4 Version register, General Purpose Inputs register, and DRAM/SRAM Options register (read only).

### Flash and EPROM Interface

The Petra/MC2 chip interfaces the MC680x0 local bus to one 1M x 8 or 2M x 8 Intel Flash device, and to one PLCC socket. *MVME162P4 models use an Intel 28F008SA device, while MVME172P4 models use an Intel 28F160S5. The Flash and EPROM memory map locations can be swapped based upon the setting of a configuration switch (S4, pin 5, line GPI3) input to the initialization PAL.* (The initialization device was discussed in the previous section.) This enables the MVME1X2P4 to execute reset code from either the EPROM or Flash.

The Petra/MC2 chip executes multiple cycles to the eight-bit Flash/EPROM devices so that byte, word, or longword accesses are allowed. Burst accesses to Flash/EPROM are inhibited by the interface so that they are broken into four longword accesses.

The Petra/MC2 chip supports write cycles to EPROM memory space with a normal cycle termination by asserting transfer acknowledge. Data is not changed. The Petra/MC2 chip allows the write cycle to time out.

The Flash memory has a number of write-protect features. A CSR bit in the Flash Parameter register (FWEN, bit 11) inhibits write cycles to Flash. There is also a jumper (J24) that will inhibit writes to Flash; refer to your MVME162P4 or MVME172P4 *Installation and Use* manual for details.

*The Petra ASIC also supports the MC1 ASIC's method of "Flash write enable". This method puts Flash memory in write enable mode by referencing the BBRAM at locations \$FFFCC000-\$FFFCFFFF. Conversely, the Flash can be write-protected by referencing the BBRAM at locations \$FFFC8000-\$FFFCBFFF. The MC1 mode is selected with a board-level configuration switch (S5 segment 4).*

## BBRAM Interface

The Petra/MC2 chip supports read/write access to the BBRAM by any bus master on the MC680x0 bus. The BBRAM interface operates identically to the Flash in that it performs dynamic sizing for accesses to the 8-bit BBRAM to make it appear contiguous. This feature allows code to be executable from the BBRAM. Burst accesses to Flash/EPROM are inhibited by the interface so that they are broken into four longword accesses. The BBRAM device access time must be no greater than 5 BCLK periods in fast mode or 9 BCLK periods in slow mode. The BBRAM speed option is controlled by control bit 8 in the General Control register at address \$FFF42000 in the Petra/MC2 chip.

## 82596CA LAN Interface

The LAN controller interface is described in the following sections.

### MPU Port and MPU Channel Attention

The Petra/MC2 chip allows the MC680x0 bus master to communicate directly with the Intel 82596CA LAN Coprocessor by providing a map decoder and required control and timing logic. Two types of direct access are feasible with the 82596CA: MPU Port and MPU Attention.

MPU Port access enables the MPU to write to an internal, 32-bit 82596CA command register. This allows the MPU to do four things:

1. Write an alternate System Configuration Pointer address.
2. Write an alternative dump area pointer and perform a dump.
3. Execute a software reset.
4. Execute a self-test.

Each MPU Port access must consist of two 16-bit writes: Upper Command Word (two bytes) and Lower Command Word (two bytes). The Upper Command Word (two bytes) is mapped at \$FFF46000 and the Lower Command Word (two bytes) is mapped at \$FFF46002.

The Petra/MC2 chip only supports (decodes) MPU Port writes. It does not decode MPU Port reads. (Nor does the 82596CA support MPU Port reads.)

MPU Channel Attention access is used to cause the 82596CA to begin executing memory-resident Command blocks. To execute an MPU Channel Attention, the MC680x0 processor bus master performs a simple read or write to address \$FFF46004.

### **MC680x0-Bus Master Support for 82596CA**

The 82596CA has DMA capability with an Intel i486-bus interface. When it is the local bus master, external hardware is needed to convert its bus cycles into MC680x0-bus cycles. When the 82596CA has local bus mastership, the Petra/MC2 chip drives the following MC680x0 signal lines:

- Snoop Control SC1-SC0 (with the value programmed into the LAN Interrupt Control register).
- Transfer Types TT1-TT0 (with the value of %00).
- Transfer Modifiers TM2-TM0 (with the value of %101).
- Transfer Start
- Read
- Size
- Transfer in progress

### **LANC Bus Error**

The 82596CA does not provide a way to terminate a bus cycle with an error indication. Bus errors are processed as follows:

The 82596CA interface logic monitors all bus cycles initiated by the 82596CA, and if a bus error is indicated ( $TAE^* = 0$  and  $TA^* = 1$ ), the Back Off signal ( $BOFF^*$ ) to the 82596CA is asserted to keep the 82596CA off the local bus and prevent it from transmitting bad data or corrupting local memory. The LANC Error Status register in the Petra/MC2 chip is updated and a LANC bus error interrupt is generated if it is enabled in the MC2 sector. The Back Off signal remains asserted until the 82596CA is reset via a port reset command. After the 82596CA is reset, pending operations must be restarted.

### LANC Interrupt

The Petra/MC2 chip provides an interrupt control register for normal LANC termination and another register for bus error termination of LANC operation. The Petra/MC2 chip requests an interrupt at the level programmed in the LANC interrupt control registers if the interrupt is enabled and a positive edge is detected on the 82596CA  $INT^*$  pin, or if the LANC bus error condition is detected.

### 53C710 SCSI Controller Interface

The Petra/MC2 chip provides a map decoder and an interrupt handler for the NCR 53C710 SCSI I/O Processor. The base address for the 53C710 is  $\$FFF47000$ . The Petra/MC2 chip requests an interrupt at the level programmed in the SCSI interrupt control register if the interrupt is enabled and a low level is detected on the 53C710  $IRQ^*$  pin.

### SRAM Memory Controller

The SRAM base address and size are programmable. The SRAM controller is designed to operate with 100 ns devices. The size of the SRAM is initialized in the DRAM/SRAM Options register when the MVME1X2P4 is reset. SRAM performance at 25MHz is 5,3,3,3 for read and write cycles. SRAM performance at 33MHz is 6,4,4,4 for read cycles and 6,3,3,3 for write cycles.

## SDRAM Memory Controller

*The Petra ASIC presents two interfaces to the SDRAM memory array. The two memory controllers modeled in Petra duplicate the functionality of the “parity memory controller” found in the MC2 ASICs as well as that of the “single-bit error correcting/double-bit error detecting” memory controller found in the MCECC ASICs.*

*The SDRAM memory array itself is always a single-bit error correcting and multi-bit error detection memory, irrespective of which interface model you use to access the SDRAM. When the MC2 memory controller interface is used to access the SDRAM, single-bit errors are undetectable to the user and multi-bit errors are defined to be parity errors. Memory scrubbing is not functional when the MC2 memory model is enabled.*

*Firmware will initialize the memory controller to maintain backward compatibility with MVME1X2LX (2IP) or -FX (4IP) products. If the Petra ASIC is supporting MVME1X2FX functionality, the parity memory controller model will be enabled. If the Petra ASIC is supporting MVME1X2LX functionality, firmware will enable either the parity or the ECC memory controller model, depending on board configuration. (The board configuration is a function of switch settings and resistor population options.)*

*User code can modify Petra register settings to operate in either mode. The benefit of operating with the ECC memory model is that single-bit errors will be scrubbed, so that they cannot degenerate into double-bit errors. User code can also modify map decoder/switch settings to enable the maximum amount of memory available. The minimum SDRAM configuration is 16MB.*

*The Petra memory controller maintains tags for each internal bank of SDRAM. Each bank may be in an active or idle state. Performance is then a function of the state of the memory bank being addressed. If that bank of memory is active, then the access time is also a function of the page of memory being referenced. If that page is open, access time is the fastest. If it is not open, then the open page must be closed and the new one opened. That would be the slowest access mode. Page size is determined by the SDRAM device. Pages range from 256 to 1024 locations.*



The following table summarizes the memory specifications. The table is of necessity incomplete, since it cannot account for the effects of the write posting operation. If the Petra memory controller is idle and a write cycle is initiated on the local bus, that cycle will be write posted and the local bus will be acknowledged in two ticks. If another bus cycle is initiated while the write post operation is in progress, that cycle will be stalled until write posting is complete.

Bear in mind that memory bandwidth on MVME1X2P4 boards is higher than before. For some memory architectures, it has increased significantly. Software will execute more quickly as a result.

**Table 3-2. SDRAM Performance**

Access Type	Memory States		
	Idle	Active Hit	Active Miss
Read Single	4	3	5
Read Burst	4-1-1-1	3-1-1-1	5-1-1-1
Write Burst	2-1-1-1	2-1-1-1	2-1-1-1
WR lword	2		
WR 1 or 2 bytes	9	8	10

## Z85230 SCC Interface

The Petra/MC2 chip supports as many as four Z85230 devices for serial communications control. (There is only one Z85230 on the MVME1X2P4. Refer to the *Board Level Hardware Description* in your *Installation and Use* manual for details.) The addresses for the devices are defined as shown in the following table. Note that the General Control register (at location \$FFF42000) contains CSR bits (SSCIT <1:0>) to control the delay time for the Z85230 IACK cycle.

The Petra/MC2 provides a map decoder and an interrupt handler for the Zilog Z85230. The base address is \$FFF45000. The Petra/MC2 requests an interrupt at the level programmed in the SCC interrupt control register if

the interrupt is enabled and a low level is detected on the SCC INT\* pin. The Z85230 provides the interrupt vector for the interrupt acknowledge cycle.

**Table 3-3. SCC Device Addressing**

Address Range	SCC Device Number	MVME1X2P4
\$FFF45000 - \$FFF453FF	0	Used
\$FFF45400 - \$FFF457FF	1	Not Used
\$FFF45800 - \$FFF45BFF	2	Not Used
\$FFF45C00 - \$FFF45FFF	3	Not Used

*The MC2 SCC interface had a bug that prevented direct access to the data register in the SCC device. This has been remedied in the Petra ASIC.*

## Tick Timers

The Petra/MC2 chip implements four 32-bit tick timers. These timers are identical to the timers in the VMEchip2. The timers run on a 1MHz clock which is derived from the processor clock by a prescaler. Each timer has a 32-bit counter, a 32-bit compare register, and a clear-on-compare enable bit. The counter is readable and writable at any time. The timers can be used to generate interrupts at various rates, or the counters can be read at various times for interval timing.

There are two modes of operation for these timers: free-running and clear-on-compare. In free-running mode, the timers have a resolution of 1  $\mu$ s and roll over after the count reaches the maximum value \$FFFFFFFF. The terminal count period for the timers is 71.6 minutes.

When the counter is enabled in the clear-on-compare mode, it increments every 1  $\mu$ s until the counter value matches the value in the compare register. When a match occurs, the counter is cleared.

When a match occurs, in either mode, an interrupt is sent to the local bus interrupter and the overflow counter is incremented. An interrupt to the local bus is only generated if the tick timer interrupt is enabled by the local bus interrupter. The overflow counter can be cleared by writing a 1 to the overflow clear bit.

## Watchdog Timer

A watchdog timer function is provided in the VMEchip2 as well as the Petra/MC2 chip. The watchdog timer implemented in the Petra/MC2 chip is used if the board is built in a "No VMEbus Interface" configuration. When the watchdog timer is enabled, it must be reset by software within the programmed time or it times out. The watchdog timer can be programmed to generate a board level reset signal or board fail signal if it times out. Note that, unlike the VMEchip2, the Petra/MC2 chip timer cannot generate a system reset because it is not connected to the VMEbus.

## Local Bus Timer

The MVME1X2P4 provides a time-out function for the local bus. When the timer is enabled and a local bus access times out, a Transfer Error Acknowledge (TEA) signal is sent to the local bus master. The time-out value is selectable by software for 8  $\mu$ sec, 64  $\mu$ sec, 256  $\mu$ sec, or infinity. The local bus timer does not operate during VMEbus-bound cycles. VMEbus-bound cycles are timed by the VMEbus access timer and the VMEbus global timer. Refer to the section on *Example of the Proper Use of Bus Timers* in Chapter 1 for more information.

The access timer logic is duplicated in the VMEchip2 and Petra/MC2 ASIC. Because the local bus timer in the VMEchip2 can detect an offboard access and the Petra/MC2 local bus timer cannot, the timer in the VMEchip2 is used in all cases except when a "No VMEbus Interface" configuration is implemented.

## Petra/MC2 Registers: Memory Map

The register map and address of the Petra ASIC's MC2 memory controller sector is documented in the following table. If the register is depicted as a 32-bit entity, it must be accessed as a longword. If it is accessed as a byte or word, the cycle is terminated with an error. If the register is depicted as an 8- or 16-bit entity, it can be accessed as a byte, word, or longword.

**Table 3-4. Petra MC2 Functions: Register Map**

MC2 Sector Base Address = \$FFF42000

Offset	D31-D24	D23-D16	D15-D8	D7-D0
\$00	MC2 Chip ID	MC2 Chip Revision	General Control	Interrupt Vector Base Register
\$04	Tick Timer 1 Compare Register			
\$08	Tick Timer 1 Counter Register			
\$0C	Tick Timer 2 Compare Register			
\$10	Tick Timer 2 Counter Register			
\$14	LSB Prescaler Count Register	Prescaler Clock Adjust	Tick Timer 2 Control	Tick Timer 1 Control
\$18	Tick Timer 4 Interrupt Control	Tick Timer 3 Interrupt Control	Tick Timer 2 Interrupt Control	Tick Timer 1 Interrupt Control
\$1C	DRAM Parity Error Interrupt Control	SCC Interrupt Control	Tick Timer 4 Control	Tick Timer 3 Control
\$20	DRAM Space Base Address Register		SRAM Space Base Address Register	
\$24	DRAM Space Size	DRAM/SRAM Options	SRAM Space Size	Reserved
\$28	LANC Error Status	Reserved	LANC Interrupt Control	LANC Bus Error Interrupt Control
\$2C	SCSI Error Status	General Purpose Inputs	<i>Board</i> Version	SCSI Interrupt Control
\$30	Tick Timer 3 Compare Register			
\$34	Tick Timer 3 Counter Register			
\$38	Tick Timer 4 Compare Register			
\$3C	Tick Timer 4 Counter Register			

**Table 3-4. Petra MC2 Functions: Register Map (Continued)**

MC2 Sector Base Address = \$FFF42000

Offset	D31-D24	D23-D16	D15-D8	D7-D0
\$40	Bus Clock	EPROM Access Time Control	Flash Parameter Control	<b>ABORT</b> Switch Interrupt Control
\$44	RESET Switch Control	Watchdog Timer Control	Access / Watchdog Time Base Select	Reserved
\$48	DRAM Control	Reserved	MPU Status	Reserved
\$4C	32-bit Prescaler Count Register			

3

## Programming Model

This section defines the programming model for the control and status registers (CSR) in the Petra/MC2 chip. The base address of the CSR is \$FFF42000. The operations possible for each bit in the CSR are as follows:

- R** This bit is a read-only status bit.
- R/W** This bit is readable and writable.
- C** Writing a 1 to this bit clears this bit or another bit. This bit reads 0.

The possible bit states after local and power-up reset are as defined below.

- P** The bit is affected by power-up reset.
- L** The bit is affected by local reset.
- X** The bit is not affected by reset.
- 0** The bit is always 0.
- 1** The bit is always 1.

## MC2 Sector ID Register

ADR/SIZ	\$FFF42000 (8 bits)							
BIT	31	30	29	28	27	26	25	24
NAME	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
OPER	R	R	R	R	R	R	R	R
RESET	1 PL	0 PL	0 PL	0 PL	0 PL	1 PL	0 PL	0 PL

**ID7-ID0** The chip ID number is \$84. This register is read only. It ignores a write but ends the cycle with TA\*, i.e., the cycle terminates without exceptions.

## MC2 Sector Revision Register

ADR/SIZ	\$FFF42000 (8 bits)							
BIT	23	22	21	20	19	18	17	16
NAME	RV7	RV6	RV5	RV4	RV3	RV2	RV1	RV0
OPER	R	R	R	R	R	R	R	R
RESET	0 PL	0 PL	0 PL	0 PL	0 PL	0 PL	1 PL	1 PL

**RV7-RV0** The current value of the chip revision is \$03. This register is read-only. It ignores a write but ends the cycle with TA\*, i.e., the cycle terminates without exceptions.

## General Control Register

ADR/SIZ	\$FFF42000 (8 bits)							
BIT	15	14	13	12	11	10	9	8
NAME				SCCIT1	SCCIT0		MIEN	
OPER	R	R	R	R/W	R/W	R	R/W	R
RESET	0 PL	0 PL	0 PL	0 PL	0 PL	0 PL	0 PL	0 PL

**MIEN** Master Interrupt Enable. When this bit is high, interrupts from and via the Petra/MC2 sector are allowed to reach the MPU. When it is low, all interrupts from the MC2

sector are disabled. Also, when the bit is low, all interrupt acknowledge cycles to the Petra/MC2 chip are passed on, via the IACKOUT\* pin. This bit is cleared by a reset.

### SCCIT<1:0>

These bits define the IACK daisy chain time for the SCC chips. They must be set based on the number of SCC devices.

SCCIT<1:0>	Number of Z85230s
00	1
01	2
10	3
11	4



**Caution**

Because MVME1X2P4 boards contain a single Z85230 device, the SCCIT bits must be initialized to 00 for these boards.

## Interrupt Vector Base Register

The interrupt vector base register is an 8-bit read/write register that is used to supply the vector to the MC68xx0x0 during interrupt acknowledge cycles. Only the most significant four bits are used. The least significant four bits encode the interrupt source during the acknowledge cycle.

The exception to this rule is that after reset occurs, the interrupt vector passed is \$0F, which remains in effect until a write is generated to the vector base register.

A normal read access to the vector base register yields the value \$0F if the read happens before it has been initialized. A normal read access yields all 0s on bits 0-3 and the value that was written on bits 4-7 if the read happens after the register has been initialized.

ADR/SIZ	\$FFF42000 (8 bits)							
BIT	7	6	5	4	3	2	1	0
NAME	IV7	IV6	IV5	IV4	IV3	IV2	IV1	IV0
OPER	R/W	R/W	R/W	R/W	R	R	R	R
RESET	0 PL	0 PL	0 PL	0 PL	1 PL	1 PL	1 PL	1 PL

The encoding for the interrupt sources is shown in the next table, where IV3-IV0 refer to bits 3-0 of the vector passed during the IACK cycle.

The priority referenced in the following table is established in the MC2 logic by implementing a daisy chain request/grant network. There is a similar request/grant daisy chain at the board level. At the board level, the Petra ASIC's MC2 sector is wired to have the highest priority, followed by the Petra's IndustryPack interface (IP2 sector) and the VMEchip2 ASIC.

**Table 3-5. Interrupt Vector Base Register Encoding and Priority**

Interrupt Source 0	IV3-IV0	Daisy Chain Priority
Unused	\$0 & \$1 & \$2	..
Timer 4	\$3	Lowest
Timer 3	\$4	
SCSI IRQ	\$5	
LANC ERR	\$6	
LANC IRQ	\$7	
Timer 2	\$8	
Timer 1	\$9	
Unused	\$A	
Parity Error	\$B	
Unused	\$C & \$D	
Serial I/O (Z85230)	Note 1	Next Highest
<b>ABORT</b> Switch	\$E	Highest
Unused	\$F	..



**Note** The Z85230 controller has an integrated interrupt vector register which is separate from the vector generation found in the Petra/MC2 chip. The Z85230 also supports a scheme where the base register value is changed based upon the interrupt requested.

## Programming the Tick Timers

There are four programmable tick timers in the Petra/MC2 chip. These timers are identical in function to the timers implemented in the PCCchip2 and the VMEchip2.

### Tick Timer 1 and 2 Compare and Counter Registers

The Tick Timer Counter is compared to the Compare register. When they are equal, an interrupt is sent to the local bus interrupter and the overflow counter is incremented. If the clear on compare mode is enabled, the counter is also cleared. For periodic interrupts, the following equation should be used to calculate the compare register value for a specific period (T).

$$T (\mu\text{s}) = \text{Compare Register}$$

When programming the tick timer for periodic interrupts, the counter should be cleared to zero by software and then enabled. If the counter does not initially start at 0, the time to the first interrupt may be longer or shorter than expected. The rollover time for the counter is 71.6 minutes.

The Tick Timer Counter, when enabled, increments every microsecond. Software may read or write the counter at any time.

**Tick Timer 1 Compare Register**

<b>ADR/SIZ</b>	<b>\$FFF42004 (32 bits)</b>		
<b>BIT</b>	31	...	0
<b>NAME</b>	Tick Timer 1 Compare Register		
<b>OPER</b>	R/W		
<b>RESET</b>	0 P		

**Tick Timer 1 Counter**

<b>ADR/SIZ</b>	<b>\$FFF42008 (32 bits)</b>		
<b>BIT</b>	31	...	0
<b>NAME</b>	Tick Timer 1 Counter		
<b>OPER</b>	R/W		
<b>RESET</b>	X		

**Tick Timer 2 Compare Register**

<b>ADR/SIZ</b>	<b>\$FFF4200C (32 bits)</b>		
<b>BIT</b>	31	...	0
<b>NAME</b>	Tick Timer 2 Compare Register		
<b>OPER</b>	R/W		
<b>RESET</b>	0 P		

**Tick Timer 2 Counter**

<b>ADR/SIZ</b>	<b>\$FFF42010 (32 bits)</b>		
<b>BIT</b>	31	...	0
<b>NAME</b>	Tick Timer 2 Counter		
<b>OPER</b>	R/W		
<b>RESET</b>	X		

## LSB Prescaler Count Register

This register is used to generate the 1MHz clock for the four tick timers. This register is read-only. It increments to \$FF at the processor frequency, then it is loaded from the Prescaler Clock Adjust register.

<b>ADR/SIZ</b>	<b>\$FFF42014 (8 bits)</b>	
<b>BIT</b>	31	24
<b>NAME</b>	LSB Prescaler Count	
<b>OPER</b>	R	
<b>RESET</b>	X	

## Prescaler Clock Adjust Register

This register adjusts the prescaler so that it maintains a 1MHz clock source for the tick timers. To provide a 1MHz clock, the prescaler adjust register should be programmed based on the following equation:

$$\text{Prescaler Clock Adjust Register} = 256 - \text{processor clock (MHz)}$$

As an example: For operation at 20MHz, the prescaler value is \$EC; at 25MHz it is \$E7; and at 33MHz it is \$DF.

Non-integer processor clocks introduce an error into the specified times for the tick timers. The tick timer clock can be derived by the following equation:

$$\text{Tick clock} = \text{processor clock} / (256 - \text{Prescaler Value})$$

The maximum clock frequency for the tick timers is the processor clock divided by two. The value \$FF is not allowed to be programmed into this register. If a write with the value of \$FF occurs to this register, the cycle terminates correctly but the register remains unchanged.

<b>ADR/SIZ</b>	<b>\$FFF42014 (8 bits)</b>	
<b>BIT</b>	23	16
<b>NAME</b>	Prescaler Clock Adjust	
<b>OPER</b>	R/W	
<b>RESET</b>	\$DF P	

## Tick Timer 1 and 2 Control Registers

Each tick timer has a control register. The control registers for timers 1 and 2 are defined in this section. Control registers for timers 3 and 4 are described in a later section.

### Tick Timer 2 Control Register

ADR/SIZ	\$FFF42014 (8 bits)							
BIT	15	14	13	12	11	10	9	8
NAME	OVF3	OVF2	OVF1	OVF0		COVF	COC	CEN
OPER	R	R	R	R	R	C	R/W	R/W
RESET	0 PL	0 PL	0 PL	0 PL	0	0 PL	0 PL	0 PL

### Tick Timer 1 Control Register

ADR/SIZ	\$FFF42014 (8 bits)							
BIT	7	6	5	4	3	2	1	0
NAME	OVF3	OVF2	OVF1	OVF0		COVF	COC	CEN
OPER	R	R	R	R	R	C	R/W	R/W
RESET	0 PL	0 PL	0 PL	0 PL	0	0 PL	0 PL	0 PL

**CEN** When this bit is high, the counter increments. When this bit is low, the counter does not increment.

**COC** When this bit is high, the counter is reset to 0 when it compares with the compare register. When this bit is low, the counter is not reset.

**COVF** The overflow counter is cleared when a 1 is written to this bit.

#### OVF3-OVF0

These bits are the output of the overflow counter. The overflow counter is incremented each time the tick timer sends an interrupt to the local bus interrupter. The overflow counter can be cleared by writing a 1 to COVF.

## Tick Timer Interrupt Control Registers

There are four tick timer interrupt control registers. The register format is the same for all four registers.

### Tick Timer 4 Interrupt Control Register

ADR/SIZ	\$FFF42018 (8 bits)							
BIT	31	30	29	28	27	26	25	24
NAME			INT	IEN	ICLR	IL2	IL1	IL0
OPER	R	R	R	R/W	C	R/W	R/W	R/W
RESET	0	0	0 PL	0 PL	0 PL	0 PL	0 PL	0 PL

### Tick Timer 3 Interrupt Control Register

ADR/SIZ	\$FFF42018 (8 bits)							
BIT	23	22	21	20	19	18	17	16
NAME			INT	IEN	ICLR	IL2	IL1	IL0
OPER	R	R	R	R/W	C	R/W	R/W	R/W
RESET	0	0	0 PL	0 PL	0 PL	0 PL	0 PL	0 PL

### Tick Timer 2 Interrupt Control Register

ADR/SIZ	\$FFF4201A (8 bits)							
BIT	15	14	13	12	11	10	9	8
NAME			INT	IEN	ICLR	IL2	IL1	IL0
OPER	R	R	R	R/W	C	R/W	R/W	R/W
RESET	0	0	0 PL	0 PL	0 PL	0 PL	0 PL	0 PL

### Tick Timer 1 Interrupt Control Register

ADR/SIZ	\$FFF4201B (8 bits)							
BIT	7	6	5	4	3	2	1	0
NAME			INT	IEN	ICLR	IL2	IL1	IL0
OPER	R	R	R	R/W	C	R/W	R/W	R/W
RESET	0	0	0 PL	0 PL	0 PL	0 PL	0 PL	0 PL

**IL2-IL0** These three bits select the interrupt level for the tick timers. Level 0 does not generate an interrupt.

**ICLR** Writing a logic 1 to this bit clears the tick timer interrupt (i.e., INT bit in this register). This bit is always read as 0.

**IEN** When this bit is set high, the interrupt is enabled. The interrupt is disabled when this bit is low.

**INT** When this bit is high, a Tick Timer interrupt is being generated at the level programmed in IL2-IL0 (if nonzero). This bit is edge-sensitive and can be cleared by writing a logic 1 into the ICLR control bit.

## DRAM Parity Error Interrupt Control Register

The DRAM Parity Error Interrupt Control register controls the interrupt logic for parity error interrupts. In the MVME1X2P4, the parity control and interrupt logic is contained in the DRAM Parity Error Interrupt Control register and the DRAM Control register, located at \$FFF4201C and \$FFF42048 respectively.

*This Interrupt Control register can also be used to generate interrupts if the MCECC memory controller detects conditions that require an interrupt. In the first-generation architectures, this interrupt was generated from the VMEchip2. If no VMEbus interface was present, no MCECC interrupt was possible. Since this is added capability, its use is optional.*

ADR/SIZ	\$FFF4201C (8 bits)							
BIT	31	30	29	28	27	26	25	24
NAME			INT	IEN	ICLR	IL2	IL1	IL0
OPER	R	R	R	R/W	C	R/W	R/W	R/W
RESET	0	0	0 PL	0 PL	0 PL	0 PL	0 PL	0 PL

- IL2-IL0** These three bits select the interrupt level for the DRAM parity error detection. Level 0 does not generate an interrupt.
- ICLR** Writing a logic 1 to this bit clears the DRAM parity error detection interrupt. This clears the INT bit in this register. This bit is always read as 0.
- IEN** This bit set to a 1 enables the parity error interrupt. If this bit is set to a 1, and the PAREN and PARINT bits are set to 01 or 11, and a parity error occurs, an interrupt is generated at the level programmed in the IL2-IL0 bits. The PAREN and PARINT bits are located at \$FFF42048 at bit 26 and 25.
- INT** When this bit is high, a interrupt is being generated due to a DRAM parity error. The interrupt is at the level programmed in IL2-IL0.

## SCC Interrupt Control Register

ADR/SIZ	\$FFF4201C (8 bits)							
BIT	23	22	21	20	19	18	17	16
NAME			INT	IEN		IL2	IL1	IL0
OPER	R	R	R	R/W	R	R/W	R/W	R/W
RESET	0	0	0 PL	0 PL	0	0 PL	0 PL	0 PL

**IL2-IL0** These three bits select the interrupt level for the SCC controller. Level 0 does not generate an interrupt.

**IEN** When this bit is set high, the interrupt is enabled. The interrupt is disabled when this bit is low.

**INT** This bit reflects the state of the INT pin from the Z85230 controller (qualified by the IEN bit). When this bit is high, an SCC controller interrupt is being generated at the level programmed in IL2-IL0. When the interrupt is cleared in the Z85230, INT returns to 0.



## Tick Timer 3 and 4 Control Registers

### Tick Timer 4 Control Register

ADR/SIZ	\$FFF4201C (8 bits)							
BIT	15	14	13	12	11	10	9	8
NAME	OVF3	OVF2	OVF1	OVF0		COVF	COC	CEN
OPER	R	R	R	R	R	C	R/W	R/W
RESET	0 PL	0 PL	0 PL	0 PL	0	0 PL	0 PL	0 PL

### Tick Timer 3 Control Register

ADR/SIZ	\$FFF4201C (8 bits)							
BIT	7	6	5	4	3	2	1	0
NAME	OVF3	OVF2	OVF1	OVF0		COVF	COC	CEN
OPER	R	R	R	R	R	C	R/W	R/W
RESET	0 PL	0 PL	0 PL	0 PL	0	0 PL	0 PL	0 PL

**CEN** When this bit is high, the counter increments. When this bit is low, the counter does not increment.

**COC** When this bit is high, the counter is reset to 0 when it compares with the compare register. When this bit is low, the counter is not reset.

**COVF** The overflow counter is cleared when a 1 is written to this bit.

#### **OVF3-OVF0**

These bits are the output of the overflow counter. The overflow counter is incremented each time the tick timer sends an interrupt to the local bus interrupter. The overflow counter can be cleared by writing a 1 to COVF.

## DRAM and SRAM Memory Controller Registers

The DRAM decode logic consists of a base register, a size register, and an options register. The SRAM decode logic consists of a similar set of registers.

The reset logic initializes the DRAM and SRAM Base registers so that DRAM space starts at address 0 and SRAM space starts at \$FFE00000. DRAM and SRAM are inhibited by reset. Software can examine the MVME1X2P4 DRAM/SRAM Options register at address \$FFF42024 bits 20-16 to determine the size of the SRAM and DRAM.

### DRAM Space Base Address Register

ADR/SIZ	\$FFF42020 (16 bits)					
BIT	31	..	20	19	..	16
NAME	B31-B20					
OPER	R/W			R		
RESET	0 PL			0		

**B31-B20** B31-B20 are compared to local bus address signals A31-A20 for memory reference cycles. If they compare, a DRAM cycle is initiated. Note that there is linkage between the Base Address register and its associated Size register. The Size register masks the least significant address signals for the comparison. Therefore, the Base Address register contents must be set to a multiple of the Size register. For example, if the size is set for 4096KB, the Base register must be set to 0, or 4096KB, or 8192KB, or 12288KB, etc.

## SRAM Space Base Address Register

ADR/SIZ	\$FFF42020 (16 bits)	
BIT	15-1	0
NAME	B31-B17	
OPER	R/W	R
RESET	\$FFE0 PL	

**B31-B17** B31-B17 are compared to local bus address signals A31-A17 for memory reference cycles. If they compare, an SRAM cycle is initiated. Note that the same linkage that exists between the DRAM Base and Size registers also exists for the SRAM decode logic. Refer to the DRAM Space Base register description.

## DRAM Space Size Register

*The setting of \$6 for DZ[2:0] signifies “DRAM mezzanine not present”. In the Petra context, it has the additional functionality of enabling the decoding of the control and status registers (CSRs) in the MCECC memory model, which is also implemented by the Petra ASIC. The CSRs are located in address range \$FFF43000 through \$FFF430FF for the first MCECC model and in the range \$FFF43100 through \$FFF431FF for the second MCECC model.*

*The use of the code \$6 allows backward compatibility with old software. It is also a method of selecting between the parity or ECC memory controller models.*

ADR/SIZ	\$FFF42024 (8 bits)							
BIT	31	30	29	28	27	26	25	24
NAME						DZ2	DZ1	DZ0
OPER	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RESET	0 PL	0 PL	0 PL	0 PL	0 PL	0 PL	0 PL	0 PL

**DZ2-DZ0** The size bits configure the non-ECC DRAM decoder for a particular memory size. The following table defines their encoding. Note that the table specifies the allowed bit combinations for DZ2-DZ0. Any other combinations generate unpredictable results.

DZ2-DZ0 are set equal to the DZ2-DZ0 bits of the DRAM/SRAM Options register. Note that changing DZ2-DZ0 so that the DRAM architecture changes between interleaved and non-interleaved relocates the data. DZ2-DZ0 are programmable to facilitate diagnostic software.

**Table 3-6. DRAM Size Control Bit Encoding**

<b>DZ2 - DZ0</b>	<b>Memory Size</b>
\$0	1MB
\$1	Not defined for MVME1X2P4
\$2	Not defined for MVME1X2P4
\$3	Not defined for MVME1X2P4
\$4	4MB
\$5	8MB
\$6	DRAM not present
\$7	16MB

## DRAM/SRAM Options Register

This is a read-only register. It is initialized at reset.

ADR/SIZ	\$FFF42024 (8 bits)							
BIT	23	22	21	20	19	18	17	16
NAME			F0	SZ1	SZ0	DZ2	DZ1	DZ0
OPER	R	R	R	R	R	R	R	R
RESET	Application Specific							

**DZ2-DZ0** Bits DZ2-DZ0 indicate the size and architecture of the non-ECC DRAM array. Software must initialize the DRAM Space Size register (\$FFF42024 bits 26-24) based on the value of DZ2-DZ0. DZ2-DZ0 are initialized at reset to a value which is determined by the contents of a factory-programmed resident device.

**Table 3-7. DRAM Size Control Bit Encoding**

DZ2 - DZ0	DRAM Configuration
\$0	1MB
\$1	Not defined for MVME1X2P4
\$2	Not defined for MVME1X2P4
\$3	Not defined for MVME1X2P4
\$4	4MB
\$5	8MB
\$6	DRAM is not present. (Also enables MCECC functionality. Refer to Chapter 5.)
\$7	16MB

**SZ1-SZ0** SZx bits indicate the size of the SRAM array. Software must initialize the SRAM Space Size register (\$FFF42024 bits 9-8) based on the value of SZ1-SZ0. SZ1-SZ0 are initialized at reset to a value which is determined by the contents of a factory-programmed resident device

**Table 3-8. SRAM Size Control Bit Encoding**

SZ1 - SZ0	SRAM Configuration
\$0	128KB
\$1	512KB
\$2	1MB
\$3	2MB

**F0** F0 set to a 0 indicates that one 28F016SA 2M x 8 Flash memory device is used. F0 set to a 1 indicates that four 28F020 256K x 8 Flash memory devices are used.

### SRAM Space Size Register

ADR/SIZ	\$FFF42024 (8 bits)							
BIT	15	14	13	12	11	10	9	8
NAME						SEN	SZ1	SZ0
OPER	R							
RESET	0 PL							

**SEN** SRAM ENABLE must be set to a 1 before the SRAM can be accessed.

**SZ1-SZ0** The size bits configure the SRAM decoder for a particular memory size. The following table defines their use. Note that the table specifies the allowed bit combinations for SZ1-SZ0. Any other combinations generate unpredictable results.

SZ1-SZ0 are set equal to the SZ1-SZ0 bits of the DRAM/SRAM Options register. SZ1-SZ0 are programmable to facilitate diagnostic software.

**Table 3-9. SRAM Size Control Bit Encoding**

SZ1 - SZ0	Memory Size
\$0	Reserved (do not use)
\$1	512KB (or 128KB)
\$2	1MB
\$3	2MB

**Note** For an MVME1X2P4 with 128KB of SRAM, the software must program SZ1-SZ0 = \$1 (512KB). Therefore, the SRAM contents will repeat in the memory map.

## LANC Error Status Register

ADR/SIZ	\$FFF42028 (8 bits)							
BIT	31	30	29	28	27	26	25	24
NAME					PRTY	EXT	LTO	SCLR
OPER	R	R	R	R	R	R	R	C
RESET	0	0	0	0	0 PL	0 PL	0 PL	0 PL

**SCLR** Writing a 1 to this bit clears bits LTO, EXT, and PRTY. Reading this bit always yields 0.

### LTO, EXT, PRTY

These bits indicate the status of the last local bus error condition encountered by the LANC while performing DMA accesses to the local bus. A local bus error condition is flagged by the assertion of TEA\*. When the LANC receives TEA\* if the source of the error is local time-out, then LTO is set and EXT and PRTY are cleared. If the source of the TEA\* is due to an error in going to the VMEbus, then EXT is set and the other two status bits are

cleared. If the source of the error is DRAM parity check error, then PRTY is set and the other two status bits are cleared. If the source of the error is none of the above conditions, then all three bits are cleared. Writing a 1 to bit 24 (SCLR) also clears all three bits.

## 82596CA LANC Interrupt Control Register

ADR/SIZ	\$FFF42028 (8 bits)							
BIT	15	14	13	12	11	10	9	8
NAME	PLTY	E/L*	INT	IEN	ICLR	IL2	IL1	IL0
OPER	R/W	R/W	R	R/W	C	R/W	R/W	R/W
RESET	0 PL	0 PL	0 PL	0 PL	0	0 PL	0 PL	0 PL

**IL2-IL0** Interrupt Request Level. These three bits select the interrupt level for the 82596CA LANC. Level 0 does not generate an interrupt.

**ICLR** In edge-sensitive mode, writing a logic 1 to this bit clears the INT status bit. This bit has no function in level-sensitive mode. This bit is always read as 0.

**IEN** Interrupt Enable. When this bit is set high, the interrupt is enabled. The interrupt is disabled when this bit is low.

**INT** This status bit reflects the state of the INT pin from the LANC (qualified by the IEN bit). When this bit is high, a LANC INT interrupt is being generated at the level programmed in IL2-IL0.

**E/L\*** Edge or Level. When this bit is high, the interrupt is edge-sensitive. The interrupt is level-sensitive when this bit is low.

**PLTY** Polarity. When this bit is low, interrupt is activated by a rising edge/high level of the LANC INT pin. When this bit is high, interrupt is activated by a falling edge/low level of the LANC INT pin. Note that if this bit is changed while



the E/L\* bit is set (or is being set), a LANC interrupt may be generated. This can be avoided by setting the ICLR bit during write cycles that change the E/L\* bit.

## LANC Bus Error Interrupt Control Register

ADR/SIZ	\$FFF42028 (8 bits)							
BIT	7	6	5	4	3	2	1	0
NAME	SC1	SC0	INT	IEN	ICLR	IL2	IL1	IL0
OPER	R/W	R/W	R	R/W	C	R/W	R/W	R/W
RESET	0 PL	0 PL	0 PL	0 PL	0	0 PL	0 PL	0 PL

- IL2-IL0** Interrupt Request Level. These three bits select the interrupt level for the 82596CA LANC bus error condition. Level 0 does not generate an interrupt.
- ICLR** Writing a logic 1 into this bit clears the INT status bit. This bit is always read as 0.
- IEN** Interrupt Enable. When this bit set high, the interrupt is enabled. The interrupt is disabled when this bit is low.
- INT** Interrupt Status. When this bit is high, a LANC Bus Error interrupt is being generated at the level programmed in IL2-IL0.

**SC1-SC0** Snoop Control. These control bits determine the value that the Petra/MC2 chip drives onto the local processor bus SC1 and SC0 pins when the 82596CA LANC performs DMA accesses. The snooping functions differ according to processor type, as shown:

SC1	SC0	Requested Snoop Operation	
		MC68040	MC68060
0	0	Snoop disabled	Snoop enabled
0	1	Source dirty, sink byte/word/longword	Snoop disabled
1	0	Source dirty, invalidate line	Snoop enabled
1	1	Snoop disabled (Reserved)	Snoop disabled

## SCSI Error Status Register

ADR/SIZ	\$FFF4202C (8 bits)							
BIT	31	30	29	28	27	26	25	24
NAME					PRTY	EXT	LTO	SCLR
OPER	R	R	R	R	R	R	R	R
RESET	0	0	0	0	0 PL	0 PL	0 PL	0 PL

**SCLR** Writing a 1 to this bit clears bits LTO, EXT, and PRTY. Reading this bit always yields 0.

### LTO, EXT, PRTY

These bits indicate the status of the last local bus error condition encountered by the SCSI processor while performing DMA accesses to the local bus. A local bus error condition is flagged by the assertion of TEA\*. When the SCSI processor receives TEA\*, if the source of the error is local time-out, then LTO is set and EXT and PRTY are cleared. If the source of the TEA\* is due to an error in going to the VMEbus, then EXT is set and the other two status bits are cleared. If the source of the error is DRAM parity check error, then PRTY is set and the other two status bits are cleared. If the source of the error

is none of the above conditions, then all three bits are cleared. Writing a 1 to bit 24 (SCLR) also clears all three bits.

## General-Purpose Inputs Register

The contents of a PAL, in conjunction with board-level configuration resistors and an array of software switches, produce the bit settings of the General Purpose Inputs register, Version register and DRAM/SRAM Options register when the Petra ASIC's MC2 sector is reset. These are read-only registers. Writes to these registers are terminated without exception, but do not change their contents.

ADR/SIZ	\$FFF4202C (8 bits)		
BIT	23	22 - 17	16
NAME	V15	V14 - V9	V8
OPER	R	R	R
RESET	Application Specific		

**V10-V8** V10-V8 are general-purpose inputs derived from three configuration bits in the Petra ASIC. The configuration bits for V10-V8 are bits GPIO, GPI1, and GPI2 respectively in the MC2 General-Purpose Inputs register.

**V11** Refer to the notes following the *MVME1X2P4 Local Bus Memory Map* in Chapter 1. The input for V11 is derived from configuration bit GPI3) in the MC2 General-Purpose Inputs register in the Petra ASIC. *GPI3 is controlled by board configuration switch S4, segment 5.*



Setting bit GPI3 to 0 will cause the reset code to execute from EPROM instead of Flash, as described in the *MVME1X2P4 Installation and Use* manual.

**V15-V12** V15-V12 are four additional general-purpose inputs derived from configuration bits in the Petra ASIC (refer also to the description for V10-V8). The configuration bits for V15-V12 are bits GPI4, GPI5, GPI6, and GPI7 respectively) in the MC2 General-Purpose Inputs register.

## MVME1X2P4 Version Register

The contents of a PAL, in conjunction with board-level configuration resistors and an array of software switches, produce the bit settings of the General Purpose Inputs register, Version register, and DRAM/SRAM Options register when the MC2 sector is reset. These are read-only registers. Writes to these registers are terminated without exception, but do not change their contents.

ADR/SIZ	\$FFF4202C (8 bits)		
BIT	15	14 - 9	8
NAME	V7	V6 - V1	V0
OPER	R	R	R
RESET	Application Specific		

**V0** V0 and V4 indicate the speed of the processor and local bus. Refer to the following table for the bit definitions.

Board	V0	Processor/Bus Frequency	V4	Processor Type
MVME162P4	0	25/25MHz	0	MC68LC040
	1	32/32MHz*	1	MC68040
MVME172P4	1	64/32MHz**	0	MC68LC060
	1	60/30MHz	1	MC68060

\* 32MHz not applicable to MC68LC040.

\*\* V0 always set to 1 on MVME172P4 boards.

- V1** V1 set to a 1 indicates that the VMEchip2 ASIC is not present. V1 set to a 0 indicates that a VMEbus interface is present.
- If V1 = 0, the MC2 sector reset logic and local bus access timer are inhibited.
- V2** V2 set to a 1 indicates that the SCSI interface is not present. V2 set to a 0 indicates that an SCSI interface is present.
- V3** V3 set to a 1 indicates that the Ethernet interface is not present. V3 set to a 0 indicates that an Ethernet interface is present.
- V4** V4 set to a 1 indicates that the MC680x0 is present. V4 set to a 0 indicates that an MC68LC0x0 is present.
- V5** Reserved for internal use only.
- V6** V6 = 0 indicates the board is an MVME1X2P4 model (P2 I/O and 4 IndustryPack connector pairs).  
V6 = 1 indicates the board is an MVME1X2P2 model (front panel I/O and 2 IndustryPack connector pairs).
- V7** Reserved for internal use only.

## SCSI Interrupt Control Register

ADR/SIZ	\$FFF4202C (8 bits)							
BIT	7	6	5	4	3	2	1	0
NAME			INT	IEN		IL2	IL1	IL0
OPER	R	R	R	R/W	R	R/W	R/W	R/W
RESET	0	0	R	0 PL	0	0 PL	0 PL	0 PL

- IL2-IL0** Interrupt Level. These three bits select the interrupt level for the SCSI processor. Level 0 does not generate an interrupt.

- IEN** Interrupt Enable. When this bit is set high, the interrupt is enabled. The interrupt is disabled when this bit is low.
- INT** Interrupt Status. This status bit reflects the state of the INT pin from the SCSI processor (qualified by the IEN bit). When this bit is high, a SCSI processor interrupt is being generated at the level programmed in IL2-IL0. This status bit does not need to be cleared, because it is level sensitive.

## Tick Timer 3 and 4 Compare and Counter Registers

Tick timers 3 and 4 are defined here because they maintain this relative position in the memory map. Refer to the sections on tick timers 1 and 2 in this chapter for an overall description of the tick timers.

### Tick Timer 3 Compare Register

<b>ADR/SIZ</b>	<b>\$FFF42030 (32 bits)</b>		
<b>BIT</b>	31	...	0
<b>NAME</b>	Tick Timer 3 Compare Register		
<b>OPER</b>	R/W		
<b>RESET</b>	0 P		

### Tick Timer 3 Counter

<b>ADR/SIZ</b>	<b>\$FFF42034 (32 bits)</b>		
<b>BIT</b>	31	...	0
<b>NAME</b>	Tick Timer 3 Counter		
<b>OPER</b>	R/W		
<b>RESET</b>	X		

### Tick Timer 4 Compare Register

ADR/SIZ	\$FFF42038 (32 bits)		
BIT	31	...	0
NAME	Tick Timer 4 Compare Register		
OPER	R/W		
RESET	0 P		

### Tick Timer 4 Counter

ADR/SIZ	\$FFF4203C (32 bits)		
BIT	31	...	0
NAME	Tick Timer 4 Counter		
OPER	R/W		
RESET	X		

## Bus Clock Register

The Bus Clock register should be programmed with the hexadecimal value of the operating clock frequency in MHz (i.e., \$21 for 33MHz). The MC2 sector uses the value programmed in this register to control the refresh timer so that the DRAMs are refreshed every 15.6 microseconds. After power-up, this register is initialized to \$10 (for 16MHz).

ADR/SIZ	\$FFF42040 (8 bits)							
BIT	31	30	29	28	27	26	25	24
NAME			BCK5	BCK4	BCK3	BCK2	BCK1	BCK0
OPER	R/W	R/W	R/W					
RESET	0 P	0 P	0 P	1 P	0 P	0 P	0 P	0 P

**BCK5-BCK0** The refresh rate is defined by the following equation:

$$\text{Refresh Rate} = \text{BCK}/\text{BUS CLOCK} \times 16$$

where *BCK* is the value programmed in the Bus Clock register, and *BUS CLOCK* is the MC68xx0x0 bus clock frequency.

## EPROM Access Time Control Register

The MVME1X2P4 is populated with a 150ns EPROM memory device. Due to the wide range of EPROM speeds, the contents can be changed by software to adjust for a specific speed.

ADR/SIZ	\$FFF42040 (8 bits)							
BIT	23	22	21	20	19	18	17	16
NAME				ROM0		ET2	ET1	ET0
OPER	R	R	R	R/W	R	R/W	R/W	R/W
RESET	0	0	0	1 PL	0	1 PL	1 PL	1 PL

**ET2-ET0** EPROM access time is controlled by the state of ET2-ET0. The following table defines the ET2-ET0 encoding (note that for the MVME1X2P4, whose bus frequency is  $\frac{1}{2}$  the processor frequency, only the 33MHz column applies).

ET2-ET0	EPROM Access $\leq N$ at 25MHz where $N =$	EPROM Access $\leq N$ at 33MHz where $N =$
\$0	60 ns	40 ns
\$1	100 ns	70 ns
\$2	140 ns	100 ns
\$3	180 ns	130 ns
\$4	220 ns	160 ns
\$5	260 ns	190 ns
\$6	300 ns	210 ns
\$7	340 ns	240 ns

**ROM0** Refer to the table on the Local Bus Memory Map, Note 1, in Chapter 1.



## Flash Access Time Control Register

The MVME1X2P4 is populated with a 120ns Flash memory device. Due to the wide range of Flash speeds, the contents can be changed by software to adjust for a specific speed.

ADR/SIZ	\$FFF42040 (8 bits)							
BIT	15	14	13	12	11	10	9	8
NAME					FWEN	FT2	FT1	FT0
OPER	R	R	R	R	R/W	R/W	R/W	R/W
RESET	0	0	0	0	0	1 PL	1 PL	1 PL

**FWEN** A Flash write enable function is internal to the ASIC for the Petra MC2 function. FWEN set to a 1 enables writes to the Flash memory space. FWEN set to a 0 inhibits writes to the Flash memory but the cycle completes without exception.

*The Petra ASIC supports both MC1 ASIC and MC2 ASIC modes of Flash write protection. The mode is controlled by a board-level configuration jumper (J24). J24 enables or inhibits all Flash writes. The associated switch (S5, pin 4) must be in the “MC2 Flash write enable mode” to enable FWEN bit functionality. The following table shows the correspondence between the settings of J24 and S5.*

<i>Flash Write Mode</i>	<i>J24</i>	<i>S5 Pin 4</i>
<i>No write possible</i>	<i>Out</i>	<i>Don't care</i>
<i>MC1 mode</i>	<i>In</i>	<i>Open</i>
<i>MC2 mode</i>	<i>In</i>	<i>Closed</i>

*For a definition of the MC1 Flash write enable function, refer to the BBRAM memory map.*

**FT2-FT0** Flash memory access time is controlled by the state of FT2-FT0. The following table defines the FT2-FT0 encoding (for the MVME1X2P4, whose bus frequency is  $\frac{1}{2}$  the processor frequency, only the 33MHz column applies).

FT2-FT0	Flash Access $\leq N$ at 25MHz where $N =$	Flash Access $\leq N$ at 33MHz where $N =$
\$0	60 ns	40 ns
\$1	100 ns	70 ns
\$2	140 ns	100 ns
\$3	180 ns	130 ns
\$4	220 ns	160 ns
\$5	260 ns	190 ns
\$6	300 ns	210 ns
\$7	340 ns	240 ns

## ABORT Switch Interrupt Control Register

The following table describes the **ABORT** switch interrupt logic in the Petra MC2 sector.

ADR/SIZ	\$FFF42040 (8 bits)							
BIT	7	6	5	4	3	2	1	0
NAME		ABS	INT	IEN	ICLR	IL2	IL1	IL0
OPER	R	R	R	R/W	C	R/W	R/W	R/W
RESET	0	0 PL	0 PL	0 PL	0 PL	0 PL	0 PL	0 PL

**IL2-IL0** These three bits select the interrupt level for the **ABORT** switch. Level 0 does not generate an interrupt.

**ICLR** Writing a logic 1 to this bit clears the abort interrupt (i.e., the INT bit in this register). This bit is always read as 0.

**IEN** With this bit set high, the interrupt is enabled. The interrupt is disabled when this bit is low.

- INT** When this bit is high, an interrupt is being generated for the **ABORT** switch. Therefore the interrupt is level-sensitive to the presence of the INT bit. The interrupt is at the level programmed in IL2-IL0.
- ABS** The **ABORT** switch status bit set to a 1 indicates that the **ABORT** switch is pressed. When it is a 0, the switch is inactive.

## RESET Switch Control Register

The **RESET** switch on the MVME1X2P4 front panel and several status or control bits are defined by this register.

ADR/SIZ	\$FFF42044 (8 bits)							
BIT	31	30	29	28	27	26	25	24
NAME				BRFLI	PURS	CPURS	BDFLO	RSWE
OPER	R	R	R	R	R	C	R/W	R/W
RESET	0	0	0	1 PL	1 P	0	1 PL	1 P

- RSWE** The **RESET** switch enable bit is used if the board is built in a “no VMEbus interface” configuration. This bit is duplicated at the same bit position in the VMEchip2 at location \$FFF40060. When this bit is high, or the duplicate bit in the VMEchip2 is high, the **RESET** switch is enabled. When both bits are low, the **RESET** switch is disabled.
- BDFLO** When this bit is high, the MC2 sector asserts the BRDFAIL signal pin. This signal is wire-ORed to the VMEchip2 board fail pin. It controls the board fail (**FAIL**) LED on the MVME1X2P4.
- CPURS** When this bit is set high, the power-up reset status bit is cleared. This bit is always read as 0.
- PURS** This bit is set by a power-up reset. It is cleared by a write to the CPURS bit.

**BRFLI** When this status bit is high, the BRDFAIL signal pin on the Petra/MC2 chip is asserted. When this status bit is low, the BRDFAIL signal pin on the Petra/MC2 chip is not asserted. The BRDFAIL pin may be asserted by an external device, the BDFLO bit in this register, or a watchdog time-out.

## Watchdog Timer Control Register

The watchdog timer control logic in the MC2 sector is used if the board is built in a “no VMEbus interface” configuration. This function is duplicated at the same bit locations in the VMEchip2 at location \$FFF40060. The VMEchip2 has the additional option of selecting SYSRESET (i.e., VMEbus reset). It is permissible to enable the watchdog timer in both the VMEchip2 ASIC and the Petra MC2 sector.

ADR/SIZ	\$FFF42044 (8 bits)							
BIT	23	22	21	20	19	18	17	16
NAME		WDCS	WDCC	WDTO	WDBFE		WDRSE	WDEN
OPER	R	C	C	R	R/W	R	R/W	R/W
RESET	0	0 P	0 P	0 P	0 PL	0	0 PL	0 PL

**WDEN** When this bit is high, the watchdog timer is enabled. When this bit is low, the watchdog timer is not enabled.

**WDRSE** When this bit is high, and a watchdog time-out occurs, a LRESET is generated. When this bit is low, a watchdog time-out does not cause a reset.

**WDBFE** When this bit is high and the watchdog timer has timed out, the MC2 chip asserts the BRDFAIL signal pin. When this bit is low, the watchdog timer does not contribute to the BRDFAIL signal on the MC2 chip.

**WDTO** When this status bit is high, a watchdog time-out has occurred. When this status bit is low, a watchdog time-out has not occurred. This bit is cleared by writing a 1 to the WDCS bit in this register.

- WDCC** When this bit is set high, the watchdog counter is reset. The counter must be reset within the time-out period or a watchdog time-out occurs.
- WDCS** When this bit is set high, the watchdog time-out status bit (WDTO bit in this register) is cleared.

## Access and Watchdog Time Base Select Register

The watchdog timer control logic in the MC2 sector is used if the board is built in a “no VMEbus interface” configuration. This function is duplicated at the same bit locations in the VMEchip2 at location \$FFF4004C. It is permissible to enable the watchdog timer in both the VMEchip2 and the MC2 sector.

ADR/SIZ	\$FFF42044 (8 bits)							
BIT	15	14	13	12	11	10	9	8
NAME			LBTO		WDTO			
OPER	R/W		R/W		R/W			
RESET	0		0PL		0 PL			

**WDTO** These bits define the watchdog time-out period:

Bit Encoding	Time-out
0	512 $\mu$ s
1	1 ms
2	2 ms
3	4 ms
4	8 ms
5	6 ms
6	32 ms
7	64 ms

Bit Encoding	Time-out
8	128 ms
9	256 ms
10	512 ms
11	1 s
12	4 s
13	16 s
14	32 s
15	64 s

**LBTO** These bits define the local bus time-out value. The timer begins timing when TS is asserted on the local bus. If TA or TEA is not asserted before the timer times out, a TEA signal is sent to the local bus. Note that Version register bit V1 must be set to a 1 to enable the access timer in the MC2 sector (i.e., it must be a "No VMEbus Interface" option).

0	8 $\mu$ s
1	64 $\mu$ s
2	256 $\mu$ s
3	The timer is disabled.

## DRAM Control Register

*This register enables access to the SDRAM array in the MC2 memory model. Note that the SDRAM array is capable of single-bit error correction and multi-bit error detection. The error correction mode is enabled in the MC2 model by the added definitions of the RAMEN, PAREN, and PARINT control bits.*

ADR/SIZ	\$FFF42048 (8 bits)							
BIT	31	30	29	28	27	26	25	24
NAME					WWP	PARINT	PAREN	RAMEN
OPER	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RESET	0	0	0	0	0 PL	0 PL	0 PL	0 PL

**RAMEN** This bit enables the access of the DRAM. The DRAM should be enabled after the DRAM Space Base Address register is enabled and the ROM0 bit has been cleared. The DRAM Space Base Address register is located at \$FFF42020 bits 31-16 and the ROM0 bit is located at \$FFF42040 bit 20.

**PARINT-PAREN**

<b>PARINT</b>	<b>PAREN</b>	<b>MPU</b>	<b>Alternate</b>	<b><i>Error Correction</i></b>
0	0	None	None	<i>Disabled</i>
0	1	Interrupt	None	<i>Enabled</i>
1	0	Checked	Checked	<i>Enabled</i>
1	1	Interrupt	Checked	<i>Enabled</i>

“None” means no parity checking. Parity errors are not detected or reported. “Interrupt” means that the MPU receives a parity interrupt if a parity error occurs. The bus cycle is terminated with TA\*, and runs at the same speed as unchecked cycles. “Checked” means that the cycle is terminated by TEA\* if a parity error occurs. Note that “Checked” cycles lengthen the DRAM accesses by one **clock tick**.

**WWP**

*Setting WWP to a 1 causes check bits 6 and 7 to be inverted when writing to the SDRAM array. This will generate a multi-bit error when read cycles are executed. The MC2 DRAM memory controller architecture will interpret the error as a parity error.*

## MPU Status Register

This logic is duplicated in the VMEchip2 ASIC at location \$FFF40048, bits 11, 10, 9, and 7. The duplication is to enable "No VMEbus Interface" operation if necessary.

ADR/SIZ	\$FFF42048 (8 bits)							
BIT	15	14	13	12	11	10	9	8
NAME					MCLR	MLBE	MLPE	MLTO
OPER	R	R	R	R	C	R	R	R
RESET	0	0	0	0	0 PL	0 PL	0 PL	0 PL

**MLTO** When this bit is set, the MPU received a TEA and the status indicated a local bus time-out. This bit is cleared by writing a 1 to the MCLR bit in this register. The MLTO bit is used if the board is built in a "no VMEbus interface" configuration and is duplicated in the VMEchip2 at address \$FFF40048 bit 7.

**MLPE** When this bit is set, the MPU received a TEA and the status indicated a parity error during a DRAM data transfer. This bit is cleared by writing a 1 to the MCLR bit in this register. The MLPE bit is used if the board is built in a "no VMEbus interface" configuration and is duplicated in the VMEchip2 at address \$FFF40048 bit 9.

**MLBE** When this bit is set, the MPU received a TEA and additional status was not provided. This bit is cleared by writing a 1 to the MCLR bit in this register. The MLBE bit is used if the board is built in a "no VMEbus interface" configuration and is duplicated in the VMEchip2 at address \$FFF40048 bit 10.

**MCLR** Writing a 1 to this bit clears the MPU status bits 8, 9 and 10 (MLTO, MLPE, and MLBE) in this register.



## 32-bit Prescaler Count Register

The prescaler register is used to clock timing functions in the MC2 sector. The lower 8 bits of the prescaler is programmed to generate an output with a one-microsecond period. Refer to the section on the LSB Prescaler Count register under Programming the Tick Timers in this chapter. The upper 24 bits are used to clock the local bus access timer and watchdog timer. To facilitate testing, the upper 24 bits can be written. Writes to this register must be 32 bits.

ADR/SIZ	\$FFF4204C (32 bits)	
BIT	31 . . . 8	7-0
NAME	MSB	LSB
OPER	R/W	R
RESET	0 P	

**LSB7-0** The least significant bits of the 32-bit prescaler. These bits are read only. They are duplicated in the memory map in the Petra MC2 sector at \$FFF42014.

**MSB31-8** The most significant bits of the prescaler.

Note that if the board is built in a “no VMEbus interface” configuration, the 32-bit Prescaler Count register is located at \$FFF40064 in addition to \$FFF4204C. This means that this register is located at the same address (\$FFF40064) on an MVME1X2P4 with the VMEchip2 as well as an MVME1X2P4 without the VMEchip2. This feature is provided for those applications which require a Prescaler Count register to run on all MVME1X2P4 versions.



## Introduction

The IndustryPack Interface Controller (IP2 chip) ASIC is one of three ASICs used on earlier MVME162/172 models (the MC2 chip, the IP2 chip, and the MCECC chip) whose functions are now incorporated into a single ASIC, the Petra chip, on the MVME1X2P4.

For ease of use in conjunction with programming models and documentation developed for earlier boards, however, the structure of this manual preserves the functional distinctions that formerly characterized those three ASICs. This chapter describes the IP2 functionality of the Petra chip. The Petra/IP2 chip interfaces up to four IndustryPack modules (IPs) to the MC680x0 processor.

## Summary of Major Features

IP2 functions now implemented on the Petra chip include:

**Table 4-1. IP2 Functions on the Petra ASIC**

Function	Features
IndustryPack I/O Support	All logic required to interface the MC680x0 bus to four IndustryPacks
	Support for IndustryPack I/O, Memory, Interrupt Acknowledge, and ID interface cycles
	Support for 8-bit, 16-bit, and 32-bit (double-size) IndustryPack cycles
	Programmable clock for strobe generation to the IndustryPack interface
	Fixed base addressing for IndustryPack I/O and ID spaces

**Table 4-1. IP2 Functions on the Petra ASIC (Continued)**

<b>Function</b>	<b>Features</b>
Memory Control	Programmable base address/size for IndustryPack memory space
	Support for four DMA channels: one each per IndustryPack interface, or two channels on IP_a and IP_c
	Dynamic bus sizing for accesses to IndustryPack memory space
Timers	Recovery timer for each IndustryPack to provide dead time between back-to-back accesses
Interrupt Handling	Thirteen Interrupt Handler Control registers, two for each IndustryPack, one per DMA controller (DMAC) and programmable clock

## IP2/IP1 Bug Fixes

*The Petra ASIC incorporates fixes and workarounds for a number of bugs that existed in previous versions of the IP2 chip. This section describes those solutions.*

### Command Chain Table Update

*The IP2 chip would not update the DMA registers correctly if the IP bus was operating at 32Mhz and the local bus memory inserted wait states between the second or third beats of a burst operation. This problem has been corrected in the Petra implementation.*

### DMA End

*The IP2 implementation did not flush the FIFOs (for inbound DMA) when the DMA\_END signal was asserted by the Industry Pack device. The workaround for this problem was to have software set the Byte Count register to 0. This would flush the FIFO. This workaround was not effective if the application required the DMA command chaining operation. To maintain backward compatibility with the old programming model and fix the problem, an alternate Byte Count register and Control Register 2 have been added to the programming model on a per-DMA-channel basis.*

## IP Bus Reset Compatibility

*The IP1 and IP2 implementations for the IP bus reset are different. A configuration switch (S5, pin 3) determines the method Petra will emulate.*

<i>IP Bus Reset Mode</i>	<i>S5 Pin 3</i>
<i>IP1 mode</i>	<i>Off</i>
<i>IP2 mode</i>	<i>On</i>

*A new status bit in the IP\_RESET register at address \$FFFBC01F reflects the bus reset mode. At bit location 01, IP1\_RESET indicates that the IP reset will follow the programmatic interface established in the IP1 ASIC. If the bit is cleared, the IP2 interface is used.*

## Functional Description

The following sections provide an overview of the functions provided by the Petra IP2 sector. A detailed programming model for the Petra/IP2 chip control and status registers appears in a later section of this chapter.

### General Description

The IP2 sector converts IP-bound MC680x0 Read/Write/Interrupt Acknowledge cycles to IndustryPack cycles. Control registers within the IP2 sector control the assumed width of the IndustryPack that is being accessed. The IP2 sector interfaces to four 16-bit IndustryPack positions. The naming convention for single-size IndustryPack population of each of these positions is: IndustryPack-a (IP\_a), IndustryPack-b (IP\_b), IndustryPack-c (IP\_c), and IndustryPack-d (IP\_d). The naming convention for double-size IndustryPack population of these positions is IndustryPack-a/b (IP\_ab) and IndustryPack-c/d (IP\_cd). (A double-size IndustryPack can occupy positions A and B, or it can occupy positions C and D.)

## Cache Coherency

The Petra IP2 sector monitors the snoop control (SC1, SC0) and memory inhibit (MI\*) signals to maintain cache coherency. When SC1 and SC0 indicate that snooping is inhibited, the IP2 sector ignores the memory inhibit (MI\*) signal line. When SC1 and SC0 do not indicate that snooping is inhibited, the IP2 sector waits for the negation of MI\* before responding to a cycle. If TA\* or TEA\* is asserted by another local bus slave before MI\* is negated, then the IP2 sector assumes that the cycle is over and that it is not to participate.

## Local Bus to IndustryPack DMA Controllers

The IP2 supports two basic types of DMA cycles: “standard DMA” (sDMA) and “addressed DMA” (aDMA). sDMA cycles are requested by the IP. When the DMA controller (DMAC) detects a DMA request and if that DMA controller is enabled, it will acknowledge the request by transferring data between the local bus and the I/O space of the requesting IP device. Alternatively, aDMA transfers are not linked to a request/acknowledge protocol. aDMA cycles are initiated by the DMA controller as soon as its control registers have been initialized by the MC680x0. It will transfer data between the local bus and a selected IP module memory space. The IP2 sector implements four DMA controllers which can operate in the sDMA or aDMA mode.

The DMA controllers can be configured so one controller is attached to each of the four possible IndustryPack interfaces or so that DMA controllers a and b are attached to IP\_a and controllers c and d are attached to IP\_c. The DMA controllers support 8-, 16-, and 32-bit IndustryPack widths. The four DMA channels can operate concurrently.

Each DMA controller has a 32-bit local address counter, a 32-bit table address counter, a 24-bit byte counter, control registers, status registers, and a 24-bit IP address counter. The data path for each DMA controller passes through a FIFO which is eight locations deep and four bytes wide.

sDMA transfers and byte count parameters must accommodate the I/O port width. If the port width is 16 bits, then the byte count must be initialized to an even value; if the width is 32 bits, then the byte count must be set to a value which is a multiple of four. This implies that transfer to I/O space

under DMA control will always be the same size as the port width. The IP address register must be initialized to 0 before sDMA is enabled. This counter is used to align data in the IP2 sector.

The data has no alignment restrictions as it is moved to or from the memory on the local bus. This would typically be DRAM on the MC680x0 local bus, but it could also be memory on the VMEbus.

To optimize local bus usage when the IndustryPack is less than 32 bits wide, the FIFO converts 8-bit and 16-bit IP transfers to 32-bit local bus transfers. The FIFO data path logic also aligns data if the source and destination addresses are not aligned, so the local bus and the IndustryPack can operate at their maximum data transfer sizes. The FIFO also buffers enough data so that accesses to the local bus are in the burst mode.

Each DMAC also supports command chaining through the use of a singly-linked list built in local (not IP) memory. Each entry in the list includes an IP address, a local bus address, a byte count, a control word, and a pointer to the next entry. When the command chaining mode is enabled, the DMAC reads and executes commands from the list in local memory until all commands are executed.

Each DMAC can be programmed to generate an interrupt request when any specific table entry has completed, when the byte count reaches 0, when an error condition occurs, or when the DMAEND\* signal is asserted by the IndustryPack.

The DMA arbiter has two modes of operation. One mode is to implement a round robin type of arbitration, which guarantees equal access to the local bus. The other method is to set the arbitration priority to one of four states. In this case, the priority is constant with one DMA channel having the highest priority, and the other three having the second, third, and fourth highest priority.

Note that the IP specification supports a DMA burst where the DMA cycles can be executed back to back. The DMA arbiter logic will not release a DMA channel until a burst of IP cycles are completed, if the burst protocol is observed. However, if the burst protocol is not observed, the arbiter is released and the next DMA request/grant is resolved. This takes two clock cycles due to the asynchronous clocks controlling the MC680x0 local bus and the IP buses. The IP designer should take this into consideration if maximum DMA bandwidth availability is required.

Note also that when the DMA register context is updated for the next command packet, a DMA function is used. The state of the snoop control signals for this DMA function (as well as the state of the snoop control signals for all other DMA cycle types) depends on the status of configuration bits SC1 and SC0 in the Petra ASIC. *Bits SC1 and SC0 are under the control of configuration switch S5 (pins 1 and 2).*

<i>S5-1 (SC1)</i>	<i>S5-2 (SC0)</i>	<i>Requested Snoop Operation</i>	
		<i>MC68040</i>	<i>MC68060</i>
<i>On</i>	<i>On</i>	<i>Snoop disabled</i>	<i>Snoop disabled</i>
<i>On</i>	<i>Off</i>	<i>Source dirty, sink byte/word/longword</i>	<i>Snoop enabled</i>
<i>Off</i>	<i>On</i>	<i>Source dirty, invalidate line</i>	<i>Snoop disabled</i>
<i>Off</i>	<i>Off</i>	<i>Snoop disabled (Reserved)</i>	<i>Snoop enabled</i>

## Clocking Environments and Performance

The IP2 sector has two clock domains. The majority of the logic is controlled by the MC680x0 local bus clock (25MHz or 32MHz in the case of an MC68040/MC68LC040 processor, 60MHz for an MC68060, or 64MHz for an MC68LC060). The IndustryPack interface is controlled by the IndustryPack clock. The IndustryPack clock can be 8MHz or set equal to the local bus clock. When logic signals cross from one clock domain to another, they must be synchronized to the new clock frequency. The latency time due this synchronization is generally hidden due to the FIFOs in the data path. However, there are two functions where the latency time affects performance. One of them is when a local bus master such as the MC680x0 accesses an IndustryPack resource, such as reading back-to-back memory locations. One to two IP clock cycles of overhead is associated with this function. The other is when arbitration logic must resolve inputs from both clock domains to determine which IndustryPack will be granted DMA service. There are two IP clock cycles of overhead associated with this function.

The following table explains the effect of this latency for given clocking environments. The bandwidth specified in the table is the bandwidth available to the IndustryPack bus. It can be split between one, two, three, or four IP modules.



**Table 4-2. IP2 sector Clock Cycles**

Bus Frequency		Period and Bandwidth to 32-Bit IP Space		
MC68060	IP	Back to Back Examine (Note 1)	Four Cycle DMA Burst (Note 2)	Single Cycle DMA (Note 3)
25MHz	8MHz	4 IP clocks 8MB/sec	10 IP clocks 12.8MB/sec	4 IP clocks 8MB/sec
32MHz	8MHz	3 IP clocks 10.6MB/sec	10 IP clocks 12.8MB/sec	4 IP clocks 8MB/sec
32MHz	32MHz (Note 5)	6 IP clocks 21MB/sec	12 IP clocks 42MB/sec (Note 4)	6 IP clocks 21MB/sec

**Notes**

1. This column is a measure of IndustryPack bandwidth for back-to-back cycles for a local bus master which is accessing a memory or I/O space location on an IndustryPack. It assumes a zero-wait-state acknowledge reply from the IndustryPack.
2. This column is a measure of IndustryPack bandwidth for DMA burst cycles between a local bus slave and a memory or I/O space location on an IndustryPack. It assumes a zero-wait-state acknowledge reply from the IndustryPack.
3. This column is a measure of IndustryPack bandwidth for DMA single cycles between a local bus slave and a memory or I/O space location on an IndustryPack. It assumes a zero-wait-state acknowledge reply from the IndustryPack.
4. Burst mode sDMA is not supported when both bus frequencies are 32MHz.
5. Because the specified bandwidth assumes a zero-wait-state IndustryPack cycle, it would be difficult to achieve the stated bandwidths for an IP bus frequency of 32MHz.

## Programmable Clock

The IP2 sector implements a general-purpose programmable clock output for external connection to the IndustryPacks. This feature complies with the STROBE function defined in the IndustryPack specification. The programmable clock's clock source is the MC680x0 bus clock. This clock input is fed through an 8-bit programmable pre-scaling counter whose output is fed to a 16-bit counter. The 16-bit counter increments at rising edges of the output of the pre-scale logic and clears every time it reaches the value programmed into the 16-bit programmable timer register. Depending on its programmed mode, the programmable clock output either pulses or toggles each time the 16-bit counter matches and clears. Additional control bits in the programmable clock control register allow software to stop, start, clear, and reverse the polarity of the programmable clock output. The programmable clock output's programmable frequency range is from approximately 4Hz to 16MHz. The programmable clock logic also includes local bus interrupt control.

## Error Reporting

The following paragraphs describe the IP2 sector error reporting.

### Error Reporting as a Local Bus Slave

The IP2 sector does not have the ability to assert the TEA\* signal as a local bus slave. Because of this, the only bus error cycles that should ever be encountered when accessing to or through the IP2 sector are those that come from an external local bus timer due to no response from an IndustryPack. Note that any external local bus timer should be set for no less than 5 microseconds to allow for normal accesses to the slowest IndustryPack.

### Error Reporting as a Local Bus Master

The IP2 sector does not connect to the ST1 and ST0 signal lines. Consequently, when it receives a TEA\* termination to any cycle for which it is local bus master, no status will be available to indicate the source of the bus error. There is a status bit in each DMAC status register which

indicates that a local bus error did occur as a consequence of a DMA operation. The contents of the local bus address counter can be examined for the address that caused the bus error.

## IndustryPack Error Reporting

Each IndustryPack interface has an error pin. The error status from the four interfaces are available in the General Control registers.

4

## Interrupts

The IP2 sector can be programmed to interrupt the local bus master via the IPL\* signal pins when one or more of the eight IndustryPack interrupts are asserted. The interrupt control registers allow each interrupt source to be level/edge sensitive and high/low true.

When the local bus master acknowledges an interrupt, if the IP2 sector determines that it is the source of the interrupt being acknowledged, it waits for IACKIN\* to be asserted, then it performs an interrupt acknowledge cycle to the appropriate IndustryPack in order to obtain the vector number. It then passes the vector number on to the local bus master and asserts TA\* to terminate the cycle.

When the local bus master acknowledges an interrupt, if the IP2 sector determines that it is not the source of the interrupt being acknowledged, it waits for IACKIN\* to be asserted, then it passes the acknowledge on down the daisy-chain by asserting IACKOUT\*.

The interrupter also provides interrupt capability for the programmable clock and for each of the four DMA controllers. Additionally, interrupts from the programmable clock can be programmed for rising and/or falling edge sensitivity. The vector passed to the local bus master during an interrupt acknowledge for the programmable clock and DMAC interrupts is from the vector base register in the IP2 sector. Part of the vector is programmable; the other part encodes the source of the interrupt.

## Overall Memory Map

The following memory map table includes all devices selected by the IP2 sector map decoder.

**Table 4-3. IP2 Sector Overall Memory Map**

Address Range	Selected Device	Port Width	Size
Programmable	IP_a/IP_ab Memory Space	D32-D8	64KB-16MB
Programmable	IP_b Memory Space	D16-D8	64KB-8MB
Programmable	IP_c/IP_cd Memory Space	D32-D8	64KB-16MB
Programmable	IP_d Memory Space	D16-D8	64KB-8MB
\$FFF58000-\$FFF5807F	IP_a I/O Space	D16	128B
\$FFF58080-\$FFF580BF	IP_a ID Space	D16	64B
\$FFF580C0-\$FFF580FF	IP_a ID Space Repeated	D16	64B
\$FFF58100-\$FFF5817F	IP_b I/O Space	D16	128B
\$FFF58180-\$FFF581BF	IP_b ID Space	D16	64B
\$FFF581C0-\$FFF581FF	IP_b ID Space Repeated	D16	64B
\$FFF58200-\$FFF5827F	IP_c I/O Space	D16	128B
\$FFF58280-\$FFF582BF	IP_c ID Space	D16	64B
\$FFF582C0-\$FFF582FF	IP_c ID Space Repeated	D16	64B
\$FFF58300-\$FFF5837F	IP_d I/O Space	D16	128B
\$FFF58380-\$FFF583BF	IP_d ID Space	D16	64B
\$FFF583C0-\$FFF583FF	IP_d ID Space Repeated	D16	64B
\$FFF58400-\$FFF584FF	IP_ab I/O Space	D32-D16	256B
\$FFF58500-\$FFF585FF	IP_cd I/O Space	D32-D16	256B
\$FFF58600-\$FFF586FF	IP_ab I/O Space Repeated	D32-D16	256B
\$FFF58700-\$FFF587FF	IP_cd I/O Space Repeated	D32-D16	256B
\$FFFBC000-\$FFFBC083	Control/Status Registers	D32-D8	32B

## Programming Model

This section defines the programming model for the control and status registers (CSRs) in the IP2 sector. The base address of the CSRs is hardwired to \$FFFBC000.

The possible operations for each bit in the CSR are as follows:

- R** This bit is a read-only status bit.
- R/W** This bit is readable and writable.
- R/C** This status bit is cleared by writing a 1 to it.
- C** Writing a 0 to this bit clears this bit or another bit. This bit reads as 0.
- S** Writing a 1 to this bit sets this bit or another bit. This bit reads as 0.

The possible states of the bits after assertion of the RESET\* pin (powerup reset or any local reset) are as defined below.

- R** The bit is affected by reset.
- X** The bit is not affected by reset.

A summary of the IP2 sector CSR registers appears in [Table 4-4](#). The CSR registers can be accessed as bytes, words, or longwords. They should not be accessed as lines. They are shown in the table as bytes, and the bits in most of the following register descriptions are labeled as bits 7 through 0.

**Table 4-4. Petra IP2 Memory Map — Control/Status Registers**

IP2 Sector Base Address = \$FFFBC000

Register Offset	Register Name	Register Bit Names							
		D7	D6	D5	D4	D3	D2	D1	D0
\$00	CHIP ID	0	0	1	0	0	0	1	1
\$01	CHIP REVISION	0	0	0	0	0	0	/	/
\$02	RESERVED	0	0	0	0	0	0	0	0
\$03	VECTOR BASE	IV7	IV6	IV5	IV4	IV3	IV2	IV1	IV0
\$04	IP_a MEM BASE UPPER	a_BASE31	a_BASE30	a_BASE29	a_BASE28	a_BASE27	a_BASE26	a_BASE25	a_BASE24
\$05	IP_a MEM BASE LOWER	a_BASE23	a_BASE22	a_BASE21	a_BASE20	a_BASE19	a_BASE18	a_BASE17	a_BASE16
\$06	IP_b MEM BASE UPPER	b_BASE31	b_BASE30	b_BASE29	b_BASE28	b_BASE27	b_BASE26	b_BASE25	b_BASE24
\$07	IP_b MEM BASE LOWER	b_BASE23	b_BASE22	b_BASE21	b_BASE20	b_BASE19	b_BASE18	b_BASE17	b_BASE16
\$08	IP_c MEM BASE UPPER	c_BASE31	c_BASE30	c_BASE29	c_BASE28	c_BASE27	c_BASE26	c_BASE25	c_BASE24
\$09	IP_c MEM BASE LOWER	c_BASE23	c_BASE22	c_BASE21	c_BASE20	c_BASE19	c_BASE18	c_BASE17	c_BASE16
\$0A	IP_d MEM BASE UPPER	d_BASE31	d_BASE30	d_BASE29	d_BASE28	d_BASE27	d_BASE26	d_BASE25	d_BASE24
\$0B	IP_d MEM BASE LOWER	d_BASE23	d_BASE22	d_BASE21	d_BASE20	d_BASE19	d_BASE18	d_BASE17	d_BASE16
\$0C	IP_a MEM SIZE	a_SIZE23	a_SIZE22	a_SIZE21	a_SIZE20	a_SIZE19	a_SIZE18	a_SIZE17	a_SIZE16
\$0D	IP_b MEM SIZE	b_SIZE23	b_SIZE22	b_SIZE21	b_SIZE20	b_SIZE19	b_SIZE18	b_SIZE17	b_SIZE16
\$0E	IP_c MEM SIZE	c_SIZE23	c_SIZE22	c_SIZE21	c_SIZE20	c_SIZE19	c_SIZE18	c_SIZE17	c_SIZE16
\$0F	IP_d MEM SIZE	d_SIZE23	d_SIZE22	d_SIZE21	d_SIZE20	d_SIZE19	d_SIZE18	d_SIZE17	d_SIZE16
\$10	IP_a INTO CONTROL	a0_PLTY	a0_E/L*	a0_INT	a0_IEN	a0_ICLR	a0_IL2	a0_IL1	a0_ILO
\$11	IP_a INT1 CONTROL	a1_PLTY	a1_E/L*	a1_INT	a1_IEN	a1_ICLR	a1_IL2	a1_IL1	a1_ILO
\$12	IP_b INTO CONTROL	b0_PLTY	b0_E/L*	b0_INT	b0_IEN	b0_ICLR	b0_IL2	b0_IL1	b0_ILO
\$13	IP_b INT1 CONTROL	b1_PLTY	b1_E/L*	b1_INT	b1_IEN	b1_ICLR	b1_IL2	b1_IL1	b1_ILO
\$14	IP_c INTO CONTROL	c0_PLTY	c0_E/L*	c0_INT	c0_IEN	c0_ICLR	c0_IL2	c0_IL1	c0_ILO
\$15	IP_c INT1 CONTROL	c1_PLTY	c1_E/L*	c1_INT	c1_IEN	c1_ICLR	c1_IL2	c1_IL1	c1_ILO
\$16	IP_d INTO CONTROL	d0_PLTY	d0_E/L*	d0_INT	d0_IEN	d0_ICLR	d0_IL2	d0_IL1	d0_ILO
\$17	IP_d INT1 CONTROL	d1_PLTY	d1_E/L*	d1_INT	d1_IEN	d1_ICLR	d1_IL2	d1_IL1	d1_ILO
\$18	IP_a GENERAL CONTROL	a_ERR	0	a_RT1	a_RT0	a_WIDTH1	a_WIDTH0	a_BTD	a_MEN
\$19	IP_b GENERAL CONTROL	b_ERR	0	b_RT1	b_RT0	b_WIDTH1	b_WIDTH0	b_BTD	b_MEN
\$1A	IP_c GENERAL CONTROL	c_ERR	0	c_RT1	c_RT0	c_WIDTH1	c_WIDTH0	c_BTD	c_MEN
\$1B	IP_d GENERAL CONTROL	d_ERR	0	d_RT1	d_RT0	d_WIDTH1	d_WIDTH0	d_BTD	d_MEN
\$1C	RESERVED	0	0	0	0	0	0	0	0
\$1D	IP_CLOCK	0	0	0	0	0	0	0	IP32
\$1E	DMA ARBITRATION CONTROL	0	0	0	0	0	ROTAT	PRI1	PRI0
\$1F	IP_RESET	0	0	0	0	0	0	<i>IPI_RES</i>	RES

2592 9908 (1-6)

**Table 4-4. Petra IP2 Memory Map — Control/Status Registers (Cont'd)**

IP2 Sector Base Address = \$FFFBC000

Register Offset	Register Name	Register Bit Names							
		D7	D6	D5	D4	D3	D2	D1	D0
DMAC for IndustryPack a, request 0. This register set is referred to as DMACa in the text.									
\$20	DMA_a STATUS	0	DLBE	0	IPEND	CHANI	TBL	IPTO	DONE
\$21	DMA_a INT CTRL	0	0	DINT	DIEN	DICLR	DIL2	DIL1	DIL0
\$22	DMA ENABLE	0	0	0	0	0	0	0	DEN
\$23	RESERVED	0	0	0	0	0	0	0	0
\$24	DMA_a CONTROL 1	DHALT	0	DTBL	ADMA	WIDTH1	WIDTH0	0	XXX
\$25	DMA_a CONTROL 2	INTE	0	DMAEI	DMAEO	ENTO	TOIP	0	0
\$26	RESERVED	0	0	0	0	0	0	0	0
\$27	RESERVED	0	0	0	0	0	0	0	0
\$28	DMA_a LB ADDR	LBA31	LBA30	LBA29	LBA28	LBA27	LBA26	LBA25	LBA24
\$29	DMA_a LB ADDR	LBA23	LBA22	LBA21	LBA20	LBA19	LBA18	LBA17	LBA16
\$2A	DMA_a LB ADDR	LBA15	LBA14	LBA13	LBA12	LBA11	LBA10	LBA9	LBA8
\$2B	DMA_a LB ADDR	LBA7	LBA6	LBA5	LBA4	LBA3	LBA2	LBA1	LBA0
\$2C	DMA_a IP ADDR	0	0	0	0	0	0	0	0
\$2D	DMA_a IP ADDR	IPA23	IPA22	IPA21	IPA20	IPA19	IPA18	IPA17	IPA16
\$2E	DMA_a IP ADDR	IPA15	IPA14	IPA13	IPA12	IPA11	IPA10	IPA9	IPA8
\$2F	DMA_a IP ADDR	IPA7	IPA6	IPA5	IPA4	IPA3	IPA2	IPA1	IPA0
\$30	DMA_a BYTE CNT	0	0	0	0	0	0	0	0
\$31	DMA_a BYTE CNT	BCNT23	BCNT22	BCNT21	BCNT20	BCNT19	BCNT18	BCNT17	BCNT16
\$32	DMA_a BYTE CNT	BCNT15	BCNT14	BCNT13	BCNT12	BCNT11	BCNT10	BCNT9	BCNT8
\$33	DMA_a BYTE CNT	BCNT7	BCNT6	BCNT5	BCNT4	BCNT3	BCNT2	BCNT1	BCNT0
\$34	DMA_a TBL ADDR	TA31	TA30	TA29	TA28	TA27	TA26	TA25	TA24
\$35	DMA_a TBL ADDR	TA23	TA22	TA21	TA20	TA19	TA18	TA17	TA16
\$36	DMA_a TBL ADDR	TA15	TA14	TA13	TA12	TA11	TA10	TA9	TA8
\$37	DMA_a TBL ADDR	TA7	TA6	TA5	TA4	TA3	TA2	TA1	TA0

2592 9908 (2-6)

**Table 4-4. Petra IP2 Memory Map — Control/Status Registers (Cont'd)**

IP2 Sector Base Address = \$FFFBC000

Register Offset	Register Name	Register Bit Names							
		D7	D6	D5	D4	D3	D2	D1	D0
DMAc for IndustryPack b, request 0 or for IndustryPack a, request 1. This register set is referred to as DMAc in the text.									
\$38	DMA_b STATUS	0	DLBE	0	IPEND	CHANI	TBL	IPTO	DONE
\$39	DMA_b INT CTRL	0	0	DINT	DIEN	DICLR	DIL2	DIL1	DIL0
\$3A	DMA ENABLE	0	0	0	0	0	0	0	DEN
\$3B	RESERVED	0	0	0	0	0	0	0	0
\$3C	DMA_b CONTROL 1	DHALT	0	DTBL	ADMA	WIDTH1	WIDTH0	A_CH1	XXX
\$3D	DMA_b CONTROL 2	INTE	0	DMAEI	DMAEO	ENTO	TOIP	0	0
\$3E	RESERVED	0	0	0	0	0	0	0	0
\$3F	RESERVED	0	0	0	0	0	0	0	0
\$40	DMA_b LB ADDR	LBA31	LBA30	LBA29	LBA28	LBA27	LBA26	LBA25	LBA24
\$41	DMA_b LB ADDR	LBA23	LBA22	LBA21	LBA20	LBA19	LBA18	LBA17	LBA16
\$42	DMA_b LB ADDR	LBA15	LBA14	LBA13	LBA12	LBA11	LBA10	LBA9	LBA8
\$43	DMA_b LB ADDR	LBA7	LBA6	LBA5	LBA4	LBA3	LBA2	LBA1	LBA0
\$44	DMA_b IP ADDR	0	0	0	0	0	0	0	0
\$45	DMA_b IP ADDR	IPA23	IPA22	IPA21	IPA20	IPA19	IPA18	IPA17	IPA16
\$46	DMA_b IP ADDR	IPA15	IPA14	IPA13	IPA12	IPA11	IPA10	IPA9	IPA8
\$47	DMA_b IP ADDR	IPA7	IPA6	IPA5	IPA4	IPA3	IPA2	IPA1	IPA0
\$48	DMA_b BYTE CNT	0	0	0	0	0	0	0	0
\$49	DMA_b BYTE CNT	BCNT23	BCNT22	BCNT21	BCNT20	BCNT19	BCNT18	BCNT17	BCNT16
\$4A	DMA_b BYTE CNT	BCNT15	BCNT14	BCNT13	BCNT12	BCNT11	BCNT10	BCNT9	BCNT8
\$4B	DMA_b BYTE CNT	BCNT7	BCNT6	BCNT5	BCNT4	BCNT3	BCNT2	BCNT1	BCNT0
\$4C	DMA_b TBL ADDR	TA31	TA30	TA29	TA28	TA27	TA26	TA25	TA24
\$4D	DMA_b TBL ADDR	TA23	TA22	TA21	TA20	TA19	TA18	TA17	TA16
\$4E	DMA_b TBL ADDR	TA15	TA14	TA13	TA12	TA11	TA10	TA9	TA8
\$4F	DMA_b TBL ADDR	TA7	TA6	TA5	TA4	TA3	TA2	TA1	TA0

2592 9908(3-6)



**Table 4-4. Petra IP2 Memory Map — Control/Status Registers (Cont'd)**

IP2 Sector Base Address = \$FFFBC000

Register Offset	Register Name	Register Bit Names							
		D7	D6	D5	D4	D3	D2	D1	D0
DMAC for IndustryPack c, request 0. This register set is referred to as DMACc in the text.									
\$50	DMA_c STATUS	0	DLBE	0	IPEND	CHANI	TBL	IPTO	DONE
\$51	DMA_c INT CTRL	0	0	DINT	DIEN	DICLR	DIL2	DIL1	DIL0
\$52	DMA ENABLE	0	0	0	0	0	0	0	DEN
\$53	RESERVED	0	0	0	0	0	0	0	0
\$54	DMA_c CONTROL 1	DHALT	0	DTBL	ADMA	WIDTH1	WIDTH0	0	XXX
\$55	DMA_c CONTROL 2	INTE	0	DMAEI	DMAEO	ENTO	TOIP	0	0
\$56	RESERVED	0	0	0	0	0	0	0	0
\$57	RESERVED	0	0	0	0	0	0	0	0
\$58	DMA_c LB ADDR	LBA31	LBA30	LBA29	LBA28	LBA27	LBA26	LBA25	LBA24
\$59	DMA_c LB ADDR	LBA23	LBA22	LBA21	LBA20	LBA19	LBA18	LBA17	LBA16
\$5A	DMA_c LB ADDR	LBA15	LBA14	LBA13	LBA12	LBA11	LBA10	LBA9	LBA8
\$5B	DMA_c LB ADDR	LBA7	LBA6	LBA5	LBA4	LBA3	LBA2	LBA1	LBA0
\$5C	DMA_c IP ADDR	0	0	0	0	0	0	0	0
\$5D	DMA_c IP ADDR	IPA23	IPA22	IPA21	IPA20	IPA19	IPA18	IPA17	IPA16
\$5E	DMA_c IP ADDR	IPA15	IPA14	IPA13	IPA12	IPA11	IPA10	IPA9	IPA8
\$5F	DMA_c IP ADDR	IPA7	IPA6	IPA5	IPA4	IPA3	IPA2	IPA1	IPA0
\$60	DMA_c BYTE CNT	0	0	0	0	0	0	0	0
\$61	DMA_c BYTE CNT	BCNT23	BCNT22	BCNT21	BCNT20	BCNT19	BCNT18	BCNT17	BCNT16
\$62	DMA_c BYTE CNT	BCNT15	BCNT14	BCNT13	BCNT12	BCNT11	BCNT10	BCNT9	BCNT8
\$63	DMA_c BYTE CNT	BCNT7	BCNT6	BCNT5	BCNT4	BCNT3	BCNT2	BCNT1	BCNT0
\$64	DMA_c TBL ADDR	TA31	TA30	TA29	TA28	TA27	TA26	TA25	TA24
\$65	DMA_c TBL ADDR	TA23	TA22	TA21	TA20	TA19	TA18	TA17	TA16
\$66	DMA_c TBL ADDR	TA15	TA14	TA13	TA12	TA11	TA10	TA9	TA8
\$67	DMA_c TBL ADDR	TA7	TA6	TA5	TA4	TA3	TA2	TA1	TA0

2592 9908 (4-6)

**Table 4-4. Petra IP2 Memory Map — Control/Status Registers (Cont'd)**

IP2 Sector Base Address = \$FFFBC000

Register Offset	Register Name	Register Bit Names							
		D7	D6	D5	D4	D3	D2	D1	D0
DMAC for IndustryPack d, request 0 or for IndustryPack c, request 1, and for PACER CLOCK. This register set, not including the Pacer Clock, is referred to as DMACd in the text.									
\$68	DMA_d STATUS	0	DLBE	0	IPEND	CHANI	TBL	IPTO	DONE
\$69	DMA_d INT CTRL	0	0	DINT	DIEN	DICLR	DIL2	DIL1	DIL0
\$6A	DMA ENABLE	0	0	0	0	0	0	0	DEN
\$6B	RESERVED	0	0	0	0	0	0	0	0
\$6C	DMA_d CONTROL 1	DHALT	0	DTBL	ADMA	WIDTH1	WIDTH0	C_CH1	XXX
\$6D	DMA_d CONTROL 2	INTE	0	DMAEI	DMAEO	ENTO	TOIP	0	0
\$6E	RESERVED	0	0	0	0	0	0	0	0
\$6F	RESERVED	0	0	0	0	0	0	0	0
\$70	DMA_d LB ADDR	LBA31	LBA30	LBA29	LBA28	LBA27	LBA26	LBA25	LBA24
\$71	DMA_d LB ADDR	LBA23	LBA22	LBA21	LBA20	LBA19	LBA18	LBA17	LBA16
\$72	DMA_d LB ADDR	LBA15	LBA14	LBA13	LBA12	LBA11	LBA10	LBA9	LBA8
\$73	DMA_d LB ADDR	LBA7	LBA6	LBA5	LBA4	LBA3	LBA2	LBA1	LBA0
\$74	DMA_d IP ADDR	0	0	0	0	0	0	0	0
\$75	DMA_d IP ADDR	IPA23	IPA22	IPA21	IPA20	IPA19	IPA18	IPA17	IPA16
\$76	DMA_d IP ADDR	IPA15	IPA14	IPA13	IPA12	IPA11	IPA10	IPA9	IPA8
\$77	DMA_d IP ADDR	IPA7	IPA6	IPA5	IPA4	IPA3	IPA2	IPA1	IPA0
\$78	DMA_d BYTE CNT	0	0	0	0	0	0	0	0
\$79	DMA_d BYTE CNT	BCNT23	BCNT22	BCNT21	BCNT20	BCNT19	BCNT18	BCNT17	BCNT16
\$7A	DMA_d BYTE CNT	BCNT15	BCNT14	BCNT13	BCNT12	BCNT11	BCNT10	BCNT9	BCNT8
\$7B	DMA_d BYTE CNT	BCNT7	BCNT6	BCNT5	BCNT4	BCNT3	BCNT2	BCNT1	BCNT0
\$7C	DMA_d TBL ADDR	TA31	TA30	TA29	TA28	TA27	TA26	TA25	TA24
\$7D	DMA_d TBL ADDR	TA23	TA22	TA21	TA20	TA19	TA18	TA17	TA16
\$7E	DMA_d TBL ADDR	TA15	TA14	TA13	TA12	TA11	TA10	TA9	TA8
\$7F	DMA_d TBL ADDR	TA7	TA6	TA5	TA4	TA3	TA2	TA1	TA0
\$80	PACER INT CONTROL	0	IRE	INT	IEN	ICLR	IL2	IL1	IL0
\$81	PACER GEN CONTROL	PLTY	PLS	0	EN	CLR	PS2	PS1	PS0
\$82	PACER TIMER	T15	T14	T13	T12	TA11	TA10	T9	T8
\$83	PACER TIMER	TA7	TA6	TA5	TA4	TA3	TA2	T1	T0

2592 9908 (5-6)

**Table 4-4. Petra IP2 Memory Map — Control/Status Registers (Cont'd)**

IP2 Sector Base Address = \$FFFBC000

Register Offset	Register Name	Register Bit Names							
		D7	D6	D5	D4	D3	D2	D1	D0
\$90	ALTERNATE DMA_a BYTE COUNT	0	0	0	0	0	0	0	0
\$91		A_BCNT23	A_BCNT22	A_BCNT21	A_BCNT20	A_BCNT19	A_BCNT18	A_BCNT17	A_BCNT16
\$92		A_BCNT15	A_BCNT14	A_BCNT13	A_BCNT12	A_BCNT11	A_BCNT10	A_BCNT9	A_BCNT8
\$93		A_BCNT7	A_BCNT6	A_BCNT5	A_BCNT4	A_BCNT3	A_BCNT2	A_BCNT1	A_BCNT0
\$94	ALTERNATE DMA_b BYTE COUNT	0	0	0	0	0	0	0	0
\$95		A_BCNT23	A_BCNT22	A_BCNT21	A_BCNT20	A_BCNT19	A_BCNT18	A_BCNT17	A_BCNT16
\$96		A_BCNT15	A_BCNT14	A_BCNT13	A_BCNT12	A_BCNT11	A_BCNT10	A_BCNT9	A_BCNT8
\$97		A_BCNT7	A_BCNT6	A_BCNT5	A_BCNT4	A_BCNT3	A_BCNT2	A_BCNT1	A_BCNT0
\$98	ALTERNATE DMA_c BYTE COUNT	0	0	0	0	0	0	0	0
\$99		A_BCNT23	A_BCNT22	A_BCNT21	A_BCNT20	A_BCNT19	A_BCNT18	A_BCNT17	A_BCNT16
\$9A		A_BCNT15	A_BCNT14	A_BCNT13	A_BCNT12	A_BCNT11	A_BCNT10	A_BCNT9	A_BCNT8
\$9B		A_BCNT7	A_BCNT6	A_BCNT5	A_BCNT4	A_BCNT3	A_BCNT2	A_BCNT1	A_BCNT0
\$9C	ALTERNATE DMA_d BYTE COUNT	0	0	0	0	0	0	0	0
\$9D		A_BCNT23	A_BCNT22	A_BCNT21	A_BCNT20	A_BCNT19	A_BCNT18	A_BCNT17	A_BCNT16
\$9E		A_BCNT15	A_BCNT14	A_BCNT13	A_BCNT12	A_BCNT11	A_BCNT10	A_BCNT9	A_BCNT8
\$9F		A_BCNT7	A_BCNT6	A_BCNT5	A_BCNT4	A_BCNT3	A_BCNT2	A_BCNT1	A_BCNT0
\$A0	ALTERNATE DMA_a CTL 2	EN_SC	DMAE11	CHAIN_ DMAE11				SC1	SC0
\$A1	ALTERNATE DMA_b CTL 2	EN_SC	DMAE11	CHAIN_ DMAE11				SC1	SC0
\$A2	ALTERNATE DMA_c CTL 2	EN_SC	DMAE11	CHAIN_ DMAE11				SC1	SC0
\$A3	ALTERNATE DMA_d CTL 2	EN_SC	DMAE11	CHAIN_ DMAE11				SC1	SC0

2592 9908 (6-6)

## Chip ID Register

The read-only Chip ID register is hard-wired to a hexadecimal value of \$23. Although writes to this register do nothing, the Petra/IP2 chip terminates them normally with TA\*.

ADR/SIZ	\$FFFBC000 (8 bits)							
BIT	7	6	5	4	3	2	1	0
NAME	CID7	CID6	CID5	CID4	CID3	CID2	CID1	CID0
OPER	R	R	R	R	R	R	R	R
RESET	0	0	1	0	0	0	1	1

## Chip Revision Register

The read-only Chip Revision register is hard-wired to reflect the revision level of the Petra/IP2 ASIC. The current value of this register is \$03. Although writes to this register do nothing, the Petra/IP2 chip terminates them normally with TA\*.

ADR/SIZ	\$FFFBC000 (8 bits)							
BIT	7	6	5	4	3	2	1	0
NAME	REV7	REV6	REV5	REV4	REV3	REV2	REV1	REV0
OPER	R	R	R	R	R	R	R	R
RESET	0	0	1	0	0	0	1	1

## Vector Base Register

ADR/SIZ	\$FFFBC003 (8 bits)							
BIT	7	6	5	4	3	2	1	0
NAME	IV7	IV6	IV5	IV4	IV3	IV2	IV1	IV0
OPER	R/W	R/W	R/W	R/W	R/W	R	R	R
RESET	0	0	0	0	1	1	1	1

The interrupt Vector Base register is an 8-bit read/write register that is used to supply the vector to the CPU during an interrupt acknowledge cycle for the four DMA controller interrupts and for the programmable clock interrupt. Only the most significant five bits are used. The least significant three bits encode the interrupt source during the acknowledge cycle. The exception to this is that after reset occurs, the interrupt vector passed is \$07, which remains in effect until a write is generated to the Vector Base register.

**Note** Note that this register does not affect the vector supplied during an interrupt acknowledge cycle for any of the eight IndustryPack IRQ\* signals.

A normal read access to the Vector Base register yields the value \$0F if the read happens before it has been initialized. A normal read access yields all 0's on bits 0-2, and the value that was last written on bits 3-7, if the read happens after the Vector Base register was initialized.

The encoding for the interrupt sources is shown below, where IV2-IV0 refer to bits 2-0 of the vector passed during the IACK cycle:

IV2-0	Interrupt Source
0	DMA_a
\$1	DMA_b
\$2	DMA_c
\$3	DMA_d
\$4	Programmable Clock

## Memory Base Address Registers: IP\_a, IP\_b, IP\_c, IP\_d

The memory base address registers define the base address at which the IP2 sector initiates memory cycles for their corresponding IndustryPacks. If a 32-bit, double-size IndustryPack is used, then the memory base address and memory size registers for IP\_a control access for double-size ab and those for IP\_c control accesses for double-size cd.

For each of the four sets of registers, BASE31-BASE16 are compared to MC680x0 address signals 31-16 respectively. The IP2 sector can address the IndustryPacks only at even multiples of their size. Consequently, any bits that are set within SIZE23-SIZE16, mask the value programmed into BASE23-BASE16 respectively. (Masked bits always compare, regardless of the value of the corresponding address bit.) For example, if a\_SIZE16 were set, then the MC680x0 address signal, A16, would not affect comparisons for accesses to IP\_a memory space. This would allow the base address for IP\_a to be programmed for one of: \$00000000, \$00020000, \$00040000, \$00060000, etc. If both a\_SIZE16 and a\_SIZE17 were set, then the base address for IP\_a could be programmed for one of \$00000000, \$00040000, \$00080000, \$000C0000, etc.

**Note** The Memory Bases for any registers among IP\_a, IP\_b, IP\_c, or IP\_d that are enabled should be programmed not to overlap each other.

### IP\_a or Double-Size IP\_ab Memory Base Address Registers

ADR/SIZ	\$FFFBC004 and \$FFFBC005 (8 bits each)							
BIT	7	6	5	4	3	2	1	0
NAME(\$04)	a_BASE3 1	a_BASE3 0	a_BASE2 9	a_BASE2 8	a_BASE2 7	a_BASE2 6	a_BASE2 5	a_BASE2 4
NAME(\$05)	a_BASE2 3	a_BASE2 2	a_BASE2 1	a_BASE2 0	a_BASE1 9	a_BASE1 8	a_BASE1 7	a_BASE1 6
OPER	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RESET	0 R	0 R	0 R	0 R	0 R	0 R	0 R	0 R

**IP\_b Memory Base Address Registers**

ADR/SIZ	<b>\$FFFBC006 and \$FFFBC007 (8 bits each)</b>							
BIT	7	6	5	4	3	2	1	0
<b>NAME(\$06)</b>	b_BASE3 1	b_BASE3 0	b_BASE2 9	b_BASE2 8	b_BASE2 7	b_BASE2 6	b_BASE2 5	b_BASE2 4
<b>NAME(\$07)</b>	b_BASE2 3	b_BASE2 2	b_BASE2 1	b_BASE2 0	b_BASE1 9	b_BASE1 8	b_BASE1 7	b_BASE1 6
<b>OPER</b>	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
<b>RESET</b>	0 R	0 R	0 R	0 R	0 R	0 R	0 R	0 R

**IP\_c or Double-Size IP\_cd Memory Base Address Registers**

ADR/SIZ	<b>\$FFFBC008 and \$FFFBC009 (8 bits each)</b>							
BIT	7	6	5	4	3	2	1	0
<b>NAME(\$08)</b>	c_BASE3 1	c_BASE3 0	c_BASE2 9	c_BASE2 8	c_BASE2 7	c_BASE2 6	c_BASE2 5	c_BASE2 4
<b>NAME(\$09)</b>	c_BASE2 3	c_BASE2 2	c_BASE2 1	c_BASE2 0	c_BASE1 9	c_BASE1 8	c_BASE1 7	c_BASE1 6
<b>OPER</b>	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
<b>RESET</b>	0 R	0 R	0 R	0 R	0 R	0 R	0 R	0 R

**IP\_d Memory Base Address Registers**

ADR/SIZ	<b>\$FFFBC00A and \$FFFBC00B (8 bits each)</b>							
BIT	7	6	5	4	3	2	1	0
<b>NAME(\$0A)</b>	d_BASE3 1	d_BASE3 0	d_BASE2 9	d_BASE2 8	d_BASE2 7	d_BASE2 6	d_BASE2 5	d_BASE2 4
<b>NAME(\$0B)</b>	d_BASE2 3	d_BASE2 2	d_BASE2 1	d_BASE2 0	d_BASE1 9	d_BASE1 8	d_BASE1 7	d_BASE1 6
<b>OPER</b>	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
<b>RESET</b>	0 R	0 R	0 R	0 R	0 R	0 R	0 R	0 R

## Memory Size Registers: IP\_a, IP\_b, IP\_c, IP\_d

As with the memory base address registers, the IP\_a size register is also used to control accesses to double-size IP\_ab and the IP\_c size register is used to control accesses to double-size IP\_cd.

ADR/SIZ	\$FFFBC00C through \$FFFBC00F (8 bits each)							
BIT	7	6	5	4	3	2	1	0
NAME(\$0C)	a_SIZE23	a_SIZE22	a_SIZE21	a_SIZE20	a_SIZE19	a_SIZE18	a_SIZE17	a_SIZE16
NAME(\$0D)	b_SIZE23	b_SIZE22	b_SIZE21	b_SIZE20	b_SIZE19	b_SIZE18	b_SIZE17	b_SIZE16
NAME(\$0E)	c_SIZE23	c_SIZE22	c_SIZE21	c_SIZE20	c_SIZE19	c_SIZE18	c_SIZE17	c_SIZE16
NAME(\$0F)	d_SIZE23	d_SIZE22	d_SIZE21	d_SIZE20	d_SIZE19	d_SIZE18	d_SIZE17	d_SIZE16
OPER	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RESET	0 R	0 R	0 R	0 R	0 R	0 R	0 R	0 R

**SIZE23-16** A, B, C, D SIZE should be programmed to match the size of the corresponding IndustryPack memory space. The IP2 sector performs its IP memory sizing by masking any bit in BASE23-BASE16 whose corresponding SIZE23-SIZE16 bit is set. The following table shows this. Note that only certain combinations of the SIZE bits (those shown in the table) make sense. Any other combination of the SIZE bits yields unpredictable results.

Size Bits								Address Lines that Are Compared	Resulting Memory Size
23	22	21	20	19	18	17	16		
0	0	0	0	0	0	0	0	A31-A16	64KB
0	0	0	0	0	0	0	1	A31-A17	128KB
0	0	0	0	0	0	1	1	A31-A18	256KB
0	0	0	0	0	1	1	1	A31-A19	512KB
0	0	0	0	1	1	1	1	A31-A20	1MB
0	0	0	1	1	1	1	1	A31-A21	2MB
0	0	1	1	1	1	1	1	A31-A22	4MB
0	1	1	1	1	1	1	1	A31-A23	8MB
1	1	1	1	1	1	1	1	A31-A24	16MB

Note that 16MB is only possible using a double-size IP.



## IRQ0/IRQ1 Interrupt Control Registers: IP\_a, IP\_b, IP\_c, IP\_d

ADR/SIZ	\$FFFBC010 through \$FFFBC017 (8 bits each)							
BIT	7	6	5	4	3	2	1	0
<b>NAME(\$10)</b>	a0_PLTY	a0_EL*	a0_INT	a0_IEN	a0_ICLR	a0_IL2	a0_IL1	a0_IL0
<b>NAME(\$11)</b>	a1_PLTY	a1_EL*	a1_INT	a1_IEN	a1_ICLR	a1_IL2	a1_IL1	a1_IL0
<b>NAME(\$12)</b>	b0_PLTY	b0_EL*	b0_INT	b0_IEN	b0_ICLR	b0_IL2	b0_IL1	b0_IL0
<b>NAME(\$13)</b>	b1_PLTY	b1_EL*	b1_INT	b1_IEN	b1_ICLR	b1_IL2	b1_IL1	b1_IL0
<b>NAME(\$14)</b>	c0_PLTY	c0_EL*	c0_INT	c0_IEN	c0_ICLR	c0_IL2	c0_IL1	c0_IL0
<b>NAME(\$15)</b>	c1_PLTY	c1_EL*	c1_INT	c1_IEN	c1_ICLR	c1_IL2	c1_IL1	c1_IL0
<b>NAME(\$16)</b>	d0_PLTY	d0_EL*	d0_INT	d0_IEN	d0_ICLR	d0_IL2	d0_IL1	d0_IL0
<b>NAME(\$17)</b>	d1_PLTY	d1_EL*	d1_INT	d1_IEN	d1_ICLR	d1_IL2	d1_IL1	d1_IL0
<b>OPER</b>	R/W	R/W	R	R/W	C	R/W	R/W	R/W
<b>RESET</b>	0 R	0 R	0 R	0 R	0 R	0 R	0 R	0 R

- IL2-IL0** These three bits select the interrupt level for the corresponding IndustryPack interrupt request. Level 0 does not generate an interrupt.
- ICLR** In edge-sensitive mode, writing a logic 1 to this bit clears the corresponding INT status bit. In level-sensitive mode, this bit has no function. It always reads as 0.
- IEN** When IEN is set, the interrupt is enabled. When IEN is cleared, the interrupt is disabled.
- INT** When this bit is high, an interrupt is being generated for the corresponding IndustryPack IRQ. The interrupt is at the level programmed in IL2-IL0.
- E/L\*** When this bit is high, the interrupt is edge sensitive. When the bit is low, the interrupt is level sensitive.
- PLTY** When this bit is low, interrupt is activated by a falling edge/low level of the IndustryPack IRQ\*. When this bit is high, interrupt is activated by a rising edge/high level of the IndustryPack IRQ\*. Note that if this bit is changed while the E/L\* bit is set (or is being set), an interrupt may be generated. This can be avoided by setting the ICLR bit

during write cycles that change the PLTY bit. Because IndustryPack IRQ\* signals are active low, PLTY would normally be cleared.

## General Control Registers: IP\_a, IP\_b, IP\_c, IP\_d

4

ADR/SIZ	\$FFFBC018 through \$FFFBC01B (8 bits each)							
BIT	7	6	5	4	3	2	1	0
NAME(\$18)	a_ERR	0	a_RT1	a_RT0	a_WIDTH1	a_WIDTH0	a_BTD	a_MEN
NAME(\$19)	b_ERR	0	b_RT1	b_RT0	b_WIDTH1	b_WIDTH0	b_BTD	b_MEN
NAME(\$1A)	c_ERR	0	c_RT1	c_RT0	c_WIDTH1	c_WIDTH0	c_BTD0	c_MEN
NAME(\$1B)	d_ERR	0	d_RT1	d_RT0	d_WIDTH1	d_WIDTH0	d_BTD	d_MEN6
OPER	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RESET	? R	0 R	0 R	0 R	0 R	0 R	1 R	0 R

### MEN

a\_MEN/b\_MEN/c\_MEN/d\_MEN enable the local bus to perform read/write accesses to their corresponding IndustryPack memory space when set, and disable such accesses when cleared. When a double-size IndustryPack is used in ab, a\_MEN should be set and the WIDTH and MEN control bits in the IP\_b General Control register should be cleared. When a double-size IndustryPack is used in cd, c\_MEN should be set, and the WIDTH and MEN control bits in the IP\_d General Control register should be cleared.

### BTD

Setting BTD (bus turnaround delay) to a 1 will insert one inactive clock period following a read cycle on the IP bus. This idle cycle is to eliminate bus contention which would occur if the time to assert a valid address by the IP2 sector is less than the time required by the Industry Pack to put the bus in a high impedance state following the read cycle. The IP2 sector will drive the bus valid following the positive edge of the IP clock (typically in seven nanoseconds, worst case in 15). Note that IndustryPack modules which were designed to meet the 0.7 or earlier revision of the GreenSprings IndustryPack Specification

were allowed 40 ns to three-state the bus following a read cycle. The state of BTD affects IP bus cycles which are a result of the DMA function because they are the only cycles which can occur back to back. When BTD is set to a 0, the IndustryPack interface will start the next cycle as soon as possible.

**Note** The default BTD setting is to insert the additional one-clock-period delay between read cycles.

### WIDTH1, WIDTH0

The IP2 sector assumes the memory space data-bus width of each of IP\_a, IP\_b, IP\_c, and IP\_d to be the value decoded from its control bits WIDTH1 and WIDTH0. Note that the width bits control the assumed memory width for the load-stored (programmed I/O) data path. There is a similar set of bits for the DMA logic memory width control. The following table shows widths inferred by these bits. When a double-size IndustryPack is used in ab, then IP\_a should be programmed for 32 bit width, and the WIDTH and MEN control bits in the IP\_b General Control register should be cleared. When a double-size IndustryPack is used at cd, then IP\_c should both be programmed for 32 bit width, and the WIDTH and MEN control bits in the IP\_d General Control register should be cleared.

WIDTH1	WIDTH0	Memory Space Data Width
0	0	32 bits
0	1	8 bits
1	0	16 bits
1	1	Reserved

**Note** When programming b\_WIDTH1-b\_WIDTH0 for either 8 bits or 16 bits, a\_WIDTH1-a\_WIDTH0 must be programmed for a width of 8 bits or 16 bits. This applies whether or not a\_MEN is set. For example, if offset \$19 is set to the value \$09, then offset

\$18 can be set to \$04, \$05, \$08, or \$09, but not to \$00, or \$01. The same relationship also pertains to IP\_c and IP\_d, i.e., when programming d\_WIDTH1-d\_WIDTH0 for either 8-bits or 16-bits, c\_WIDTH1-c\_WIDTH0 must be programmed for a width of 8-bits or 16-bits. This applies whether or not c\_MEN is set.

**RT1, RT0**

The recovery timers determine the time that must expire from the acknowledgment of an IndustryPack I/O, ID, or Interrupt Acknowledge cycle until the IP2 sector asserts a new I/O, ID, or Int SEL\* to the same IndustryPack. This may help with some devices on IndustryPacks that require dead time between cycles. Each recovery timer's counter starts incrementing at the assertion of its IPACK\* signal and continues to increment until it matches the value encoded from its two recovery timer control bits. When it reaches that value, the recovery time has expired and a new cycle can be generated to the IndustryPack. The recovery timer counters are cleared at reset. The recovery times encoded by the recovery timer control bits are shown in the following table. When a double-size IndustryPack is used at ab and the I/O space for ab is accessed in the double-size address range, the RT bits for a and b should be programmed identically. The same pertains to the RT bits for c and d.

<b>RT1</b>	<b>RT0</b>	<b>Recovery Time with IP = 8MHz</b>	<b>Recovery Time with IP = 32MHz</b>
0	0	0 microseconds	0 microseconds
0	1	2 microseconds	0.5 microsecond
1	0	4 microseconds	1 microsecond
1	1	8 microseconds	2 microseconds

There are some restrictions on using recovery timers with double-size IndustryPacks. When using a double-size IndustryPack, programmed recovery times for back-to-back I/O and/or ID accesses are ensured if a single-size access is followed by a single-size access, or if a double-size, longword access is followed by a single- or double-size access. However, if a single-size (or byte or word) I/O or ID access is followed by a double-size I/O access, the double-size access may be allowed to happen before the recovery times for both a and b (or both c and d) have expired. This behavior is avoided if I/O accesses are restricted to single-size only, or if they are restricted to double-size, longword only and the double-size accesses are not interspersed with ID accesses. Note that memory accesses do not affect, nor are they affected by, this behavior.

- |              |   |
|--------------|---|
| <b>a_ERR</b> | This bit reflects the state of the ERROR* signal from the IP_a interface. |
| <b>b_ERR</b> | This bit reflects the state of the ERROR* signal from the IP_b interface. |
| <b>c_ERR</b> | This bit reflects the state of the ERROR* signal from the IP_c interface. |
| <b>d_ERR</b> | This bit reflects the state of the ERROR* signal from the IP_d interface. |

## IP Clock Register

ADR/SIZ	\$FFFBC01D (8 bits)							
BIT	7	6	5	4	3	2	1	0
NAME	0	0	0	0	0	0	0	IP32
OPER	R	R	R	R	R	R	R	S
RESET	0 R	0 R	0 R	0 R	0 R	0 R	0 R	0 R

### IP32

Setting IP32 to a 1 enables the IndustryPack bus to operate synchronously with the MC680x0 local bus clock. Setting it to a 0 will enable 8MHz operation. In this mode, the IndustryPack bus cycles and MC680x0 local bus cycles operate asynchronously.

The IP32 bit controls clock synchronization logic. It does not change the clock frequency on the bus. Jumper J14 controls the IP bus clock source. If J14 pins 1 and 2 are jumpered, then the IP clock source is set to 8MHz. For this setting, the IP32 control bit must be a 0.

If pins 3 and 2 are jumpered, then the IP clock source is set to be synchronous with the MC680x0 local bus clock. This clock may be 25MHz, 30MHz, or 32MHz depending on the model. For this setting, the IP32 control bit must be a 1.

## DMA Arbitration Control Register

The DMA arbitration control register contents determine whether a fixed or fair arbitration algorithm is used to determine how the MC680x0 local bus is attached to the internal DMA data paths.

ADR/SIZ	\$FFFBC01E (8 bits)							
BIT	7	6	5	4	3	2	1	0
NAME	0	0	0	0	0	ROTAT	PRI1	PRI0
OPER	R	R	R	R	R	R/W	R/W	R/W
RESET	0 R	0 R	0 R	0 R	0 R	0 R	0 R	0 R

**ROTAT** ROTAT set to a 0 enables a rotating arbitration method where each DMAC has equal access to the MC680x0 local bus. If ROTAT is set to a 1, the priority is fixed according to the following table

**PRI1,PRI0** Fixed priority assignment is defined by the following tables.

PRI1 - PRI0	Priority with one DMA channel at IP sockets a, b, c, & d			
	Highest	Next Highest	Next Lowest	Lowest
00	DMA_a	DMA_b	DMA_c	DMA_d
01	DMA_b	DMA_c	DMA_d	DMA_a
10	DMA_c	DMA_d	DMA_a	DMA_b
11	DMA_d	DMA_a	DMA_b	DMA_c

PRI1 - PRI0	Priority with two DMA channel at IP sockets a and c			
	Highest	Next Highest	Next Lowest	Lowest
00	DMA_a chan 0	DMA_a chan 1	DMA_c chan 0	DMA_c chan 1
01	DMA_a chan 1	DMA_c chan 0	DMA_c chan 1	DMA_a chan 0
10	DMA_c chan 0	DMA_c chan 1	DMA_a chan 0	DMA_a chan 1
11	DMA_c chan 1	DMA_a chan 0	DMA_a chan 1	DMA_c chan 0

## IP RESET Register

ADR/SIZ	\$FFFBC01F (8 bits)							
BIT	7	6	5	4	3	2	1	0
NAME	0	0	0	0	0	0	<i>IP1_RES</i>	RES
OPER	R	R	R	R	R	R	R	R
RESET	0 R	0 R	0 R	0 R	0 R	0 R	0 R	0 R

### RES

Setting RES to a 1 asserts the IP2 sector IPRESET\* signal. IPRESET\* is intended to be connected to the Reset\* signal on all four IndustryPacks. When software sets the RES bit, IPRESET\* stays asserted until software clears RES.

### *IP1\_RES*

*When IP1\_RES is initialized to a 1, the IPIC reset mode is enabled. The IPIC reset mode works as follows:*

*Setting IP1\_RES to a 1 asserts the IPRESET\* signal. It remains set for at least 1 millisecond and then clears.*

*When IP1\_RES is set to 0, the IP2 reset mode is enabled. IP1\_RES is initialized when Petra is reset. The IP2 method of IP reset is used on MVME1X2P4-series boards.*



## Programming the DMA Controllers

The Petra IP2 sector implements four DMA channels. They can operate in the standard or addressed mode. sDMA transfers must accommodate the I/O port width. If the port width is 16 bits, then the byte count must be even; if the width is 32 bits, then the byte count must be a multiple of four bytes. The IP address counter must be initialized to zero before an sDMA transfer is enabled. No other restrictions are placed on DMA operations.

Each DMAC has two modes of operation: *command chaining* and *direct*. In the direct mode, the local bus address, the IndustryPack address, the byte count, and the control register of a DMAC are programmed and the DMAC is enabled. The DMAC transfers data, as programmed, until the byte count is zero, DMAEND is detected true as an input, or an error is detected. When the DMAC stops, the status bits in the DMAC status register are set and an interrupt is sent to the local bus master (if the DMACs interrupts are enabled). Multiple DMAC commands can be automatically invoked using the command chaining mode.

In the command chaining mode, a singly-linked list of commands is built in local memory and the table address register in the DMAC is programmed with the starting address of the list of commands. DMAC Control Register 1 is programmed and the DMAC is enabled. The DMAC executes commands from the list until all commands are executed or until an error is detected. When the DMAC stops, the status bits are set in the DMAC status register and an interrupt is sent to the local bus master (if the DMAC interrupts are enabled). Additionally, when the DMAC finishes processing a command in the list, and interrupts are enabled for that command, the DMAC sends an interrupt to the local bus master if its interrupts are enabled.

Note that when the DMA register context is updated for the next command packet, a DMA function is used. The state of the snoop control signals for this DMA function (as well as the state of the snoop control signals for all other DMA cycle types) depends on the status of configuration bits SC0 and SC1 in the Petra ASIC. *Bits SC1 and SC0 are under the control of configuration switch S5 (pins 1 and 2).* Refer to the *Hardware Preparation* section of your MVME1X2P2 Installation and Use manual for details on S5.

Each DMAC's control is divided into two registers. The first register is only accessible by the processor. The second register can be loaded by the processor in the direct mode and by the DMAC in the command chaining mode.

There is a case when the byte count for a DMA (to local bus) operation is initially set larger than the actual received data. In this case, there is residual data in the internal data paths of the DMA controller. To flush the data, set the byte count register to 0. A normal termination of the DMA occurs after the byte count register has been initialized to zero.

*If an inbound DMA session is terminated before the byte count reaches zero, data may remain in the FIFO. You can purge the FIFO by having software write the byte count register to a zero. This will have the effect of creating a normal termination for the transaction: the FIFO will be emptied and a DMA termination interrupt will be generated.*

*An alternative method of terminating the DMA is available with the Petra ASIC. In this case, the IP module asserts a DMA end signal. The FIFO empties, a DMA termination interrupt is generated, and the DMA end status is set.*

Once a DMAC is enabled, its counter and control registers should not be modified by software. When the command chaining DMA mode is used, the list of commands must be in local (not IP), 32-bit memory and the entries must be aligned to a 16-byte boundary. That is, the address which is loaded into the DMA table address counter must have bits 3 through 0 set to a 0. This is true for the initial value which is loaded by the processor as well as for the subsequent values which are loaded by the command chaining logic. If they are not set to 0, the command chaining process will halt.

A DMAC command packet includes a control word that defines:

- ❑ Single command interrupt enable
- ❑ DMA transfer direction
- ❑ IndustryPack data width
- ❑ sDMA or aDMA selection
- ❑ DMAEND direction and usage

The format of the control word is the same as that of control register 2. The command packet also includes a local bus address, an IP address, a byte count, and a pointer to the next command packet in the list. The end of a command is indicated by setting bit 0 or 1 of the next command address. The command packet format is shown in the following table.

Entry	Function				
0	31	Address of Next Command Packet		0	
1	31	Local Bus Address		0	
2	31	Control Word	24 23	Byte Count	0
3	31	IndustryPack Address DMA		0	

## DMA Enable Function

There are certain DMA channel contexts which are illegal. If an attempt is made to program the DMA control register 1 for each channel (a and b or c and d) to an illegal state, the DEN (DMA enable control bit) will not set to a 1 when it is loaded via a processor store instruction. This condition can be tested by writing a 1 to the DEN bit and reading a 0. Refer to the description of the DMA Enable register for the required programming sequence of the control registers and enable bits.

The following are legal contexts for DMA channel configurations. Note that configuration rules for DMA controllers for IP\_a and IP\_b are defined. The same relationships exist for IP\_c and IP\_d.

- ❑ If the IP\_a data bus is 8 or 16 bits, no restrictions are placed on IP\_b.
- ❑ If the IP\_a data bus is 32 bits and ADMA mode is selected, the DMA controller associated with IP\_b cannot be used.
- ❑ If the A\_CH1 bit is set in the DMA controller register associated with IP\_b, and channels A and B both operate in the SDMA mode, the DMA channels associated with IP\_a and IP\_b can both be used if the data widths for channels A and B are set equal. This case allows the DMA channel that normally responds to the IP\_b-DMAreq\_0 pin to respond to the IP\_a-DMAreq\_1 pin. This enables full-duplex operation at IP\_a.

## DMA Control and Status Register Set Definition

The four sets of DMA controller CSRs are almost identical in functionality. Each register set is grouped as DMACa, DMACb, DMACc, and DMACd. These register sets are shown pictorially in the CSR register summary section. Only one register set is defined except that the offset is noted for the four possible values. Refer to the definitions of bit 1 in DMA Control Register 1 for a description of how the register sets are associated with the physical DMA request from the Industry Packs.

## DMA Status Register

ADR/SIZ	\$FFFBC020, \$38, \$50, \$68 (8 bits each)							
BIT	7	6	5	4	3	2	1	0
NAME	0	DLBE	<i>IPENDI</i>	IPEND	CHANI	TBL	IPTO	DONE
OPER	R	R	R/W	R	R	R	R	R
RESET	0 R	0 R	0 R	0 R	0 R	0 R	0 R	1 R

- DONE** This bit is set when DMAC has finished executing commands and there were no errors, or DMAC has finished executing commands because the DHALT bit was set. This bit is cleared when DMAC is enabled. A DMAC interrupt will be generated if interrupts are enabled.
- IPTO** When this bit is set, a DMAC access to an IndustryPack timed out. This bit is cleared when DMA is enabled. A DMAC interrupt will be generated if interrupts are enabled.
- TBL** When this bit is set, DMAC received an error on the local bus while it was reading commands from the command packet. Additional information is provided in bit 6 (DLBE). This bit is cleared when DMAC is enabled.
- CHANI** When this bit is set, the INTE bit in DMA Control Register 2 was detected. This bit is cleared when DMA is enabled or the interrupt status bit is cleared in the DMA interrupt control register or the DHALT bit was detected in DMA Control register 1. A DMAC interrupt will be generated if interrupts are enabled.
- IPEND** When this bit is set, the DMA process was terminated if the DMAEND signal was asserted by the Industry Pack and the DMAEI bit is set in DMA Control register 2. This bit is cleared when DMA is enabled. A DMAC interrupt will be generated if interrupts are enabled.

**IPEND1** *The IPEND1 bit is set if an early termination of DMA data is detected and the dmaei1 bit is set in Alternate Control Register 2. Setting this bit does not in itself cause an interrupt; the DMA interrupt associated with this activity occurs when the Done status, bit 0, is set. The IPEND1 bit is part of the fix to allow the use of the IP bus protocol which asserts the DMA\_DONE signal. The IPEND bit functionality remains unchanged, so that it is backward compatible with the IP2 design.*

**DLBE** When this bit is set, DMAC received a TEA. (TEA is a transfer error acknowledge signal on the MC680x0 local bus. It indicates that a time-out has occurred.) This bit is cleared when DMAC is enabled. A DMAC interrupt will be generated if interrupts are enabled.

### DMA Interrupt Control Register

ADR/SIZ	\$FFFBC021, \$39, \$51, \$69 (8 bits each)							
<b>BIT</b>	7	6	5	4	3	2	1	0
<b>NAME</b>	0	0	DINT	DIEN	DICLR	DIL2	DIL1	DIL0
<b>OPER</b>	R	R	R	R/W	C	R/W	R/W	R/W
<b>RESET</b>	0 R	0 R	1 R	0 R	0 R	0 R	0 R	0 R

**DIL2-DIL0** These three bits select the interrupt level for DMA. Level 0 does not generate an interrupt.

**DICLR** Writing a logic 1 to this bit clears the DINT status bit.

**DIEN** When DIEN is set, the interrupt is enabled. When DIEN is cleared, the interrupt is disabled.

**DINT** When this bit is high, an interrupt will be generated for a DMAC if the DIEN bit is set to a 1. The interrupt is at the level programmed in DL2-DL0. The DINT bit is set when one of the following bits are set in the Status register: DLBE, IPEND, CHANI, IPTO, and DONE.

## DMA Enable Register

ADR/SIZ	\$FFFBC022, \$3A, \$52, \$6A (8 bits each)							
BIT	7	6	5	4	3	2	1	0
NAME	0	0	0	0	0	0	0	DEN
OPER	R	R	R	R	R	R	R	S
RESET	0 R	0 R	0 R	0 R	0 R	0 R	0 R	0 R

**DEN**

Setting the DEN bit to a 1 will enable the DMA function. Software should not write to the DMA control registers between the time the DEN bit is set and the DMA process is completed. In general, this is true on a per-channel basis but there are interrelationships between the DMA channels/board architecture which require the initialization of certain bits for each pair of DMA channels before the DEN bits can be set. That is, if DMA channels a and b are going to be used concurrently, DMA Control Register 1 should be initialized for both channels before either channel is enabled. This is also true for DMA channels c and d. Refer to the section on the DMA Enable function for further details.

### DMA Control Register 1

ADR/SIZ	\$FFFBC024, \$3C, \$54, \$6C (8 bits each)							
BIT	7	6	5	4	3	2	1	0
NAME	DHALT	0	DTBL	ADMA	WIDTH1	WIDTH0	A_CH1 or C_CH1	XXX
OPER	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RESET	0 R	0 R	0 R	0 R	0 R	0 R	0 R	0R

**XXX** This bit must remain cleared. If it is set to a 1, the Petra/IP2 ASIC will not function correctly.

#### A\_CH1, C\_CH1

When A\_CH1 is set to a 0, DMA request 0 from Industry Pack b is associated with the DMACb register set. When it is set to a 1, DMA request 1 from Industry Pack a is associated with the DMACb register set. When C\_CH1 is set to a zero, DMA request 0 from Industry Pack d is associated with the DMACd register set. When it is set to a 1, DMA request 1 from Industry Pack c is associated with the DMACd register set. Note that DMACa register set is always associated with DMA request 0 from Industry Pack a and DMACc register set is always associated with DMA request 0 from Industry Pack c. Therefore these bit positions are not defined for these two register sets. Refer to the section on the Enable DMA Function for information and restrictions on the operation of A\_CH1 and C\_CH1.



**WIDTH1-WIDTH0**

WIDTH bits specify the width of the IndustryPack interface at position a or position a\_b. The following table defines the bit encoding. Note that these width control bits are independent of the width control bits in the General Control registers. Also note that unlike the width control bits in the General Control registers, these width control bits define the width of both the memory and I/O interface.

WIDTH1	WIDTH0	Assumed Data Bus Width
0	0	32 bits
0	1	8 bits
1	0	16 bits
1	1	Reserved

**ADMA** Setting ADMA to a 1 will enable the address mode DMA operation. Setting it to a 0 will enable the standard mode of DMA operation. Refer to the section on the DMA Enable Function for information and restrictions on the operation of the ADMA control bit.

**DTBL** DMAC operates in the direct mode when this bit is low, and it operates in the command chaining mode when this bit is high.

**DHALT** When this bit is high, DMA halts at the end of a command when DMA is operating in the command chaining mode. When this bit is low, DMA executes the next command in the list. Software must be careful not to change the state of bits 0 through 6 of this control register when the DHALT bit is set.

## DMA Control Register 2

This register is loaded by the processor or by DMA when it loads the command word from the command packet. Because this register is loaded from the command packet in the command chaining mode, the descriptions here also apply to the control word in the command packet.

ADR/SIZ	\$FFFBC025, \$3D, \$55, \$6D (8 bits each)							
BIT	7	6	5	4	3	2	1	0
NAME	INTE	0	DMAEI	DMAEO	ENTO	TOIP	0	0
OPER	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RESET	0 R	0 R	0 R	0 R	0 R	0 R	0 R	0 R

**TOIP** This bit defines the direction in which DMAC transfers data. When this bit is high, data is transferred to the IndustryPack. When it is low, data is transferred from the IndustryPack.

**ENTO** ENTO set to a 1 will enable the watchdog time-out function for DMA cycles on the IP bus. The time-out period is fixed at approximately 1 msec. If a time-out does occur, the IP bus cycle is terminated and the IPTO bit is set in the DMA Status register. Note that the IndustryPack interface in the Petra/IP2 ASIC will wait indefinitely if the ENTO bit is cleared and a DMA cycle on the IP bus is not acknowledged. The IP2 sector must be reset to clear this condition. It is recommended that ENTO be set to a 1.

**DMAEO** When DMAEO is set, DMA drives DMAEND and asserts it during the DMA IP cycle in which the byte count expires. When DMAEO is cleared, DMA's DMAEND driver is disabled.

**DMAEI**

When DMAEI is set, DMA terminates if the assertion of DMAEND is detected and the sDMA function is enabled. That is, the ADMA control bit in the DMA Control Register 1 must be set to a 0. If the assertion of DMAEND is not detected, DMA will terminate according to the byte count value and the command chaining mode.

*Note that in previous implementations, in cases where DMA was halted due to the DMAEI function, the DMA operation stopped immediately and did not flush the inbound data paths. In such instances there was valid data which had been transferred from an I/O device as far as the IP interface, but not on to the MVME162/172 host memory. The Petra implementation has preserved this functionality.*

*The Petra implementation also incorporates a DMA end function which does flush the inbound data path. For details, refer to the Alternate Control Register 2 description.*

**INTE**

This bit is used only in command chaining mode. It is modified only when DMA loads its control register from the control word in the command packet. When this bit in the command packets is set, an interrupt is sent to the local bus interrupter when the command in the packet has been executed. The local bus is interrupted if DMA's interrupt is enabled.

### DMA Local Bus Address Counter

In the direct mode, this counter is programmed with the starting address of the data in local bus memory.

<b>ADR/SIZ</b>	<b>\$FFFBC028, \$40, \$58, \$70 (32 bits each)</b>		
<b>BIT</b>	31	...	0
<b>NAME</b>	DMA Local bus Address Counter		
<b>OPER</b>	R/W		
<b>RESET</b>	0 R		

### DMA IndustryPack Address Counter

In direct mode, this counter is programmed with the starting address of the data buffer in IndustryPack memory. The value programmed in the IndustryPack address counter is the address that would be used when referencing the IndustryPack memory space from the local bus. Refer to the addressing mapping from the local bus to the IndustryPack bus for different IndustryPack memory widths, described later in this chapter.

**Note** For sDMA operations, the IndustryPack address counter must be cleared before the DMAC is enabled.

<b>ADR/SIZ</b>	<b>\$FFFBC02C, \$44, \$5C, \$74 (32 bits each)</b>		
<b>BIT</b>	31...23	22	0
<b>NAME</b>	0	DMA Industry Pack Address Counter	
<b>OPER</b>	R	R/W	
<b>RESET</b>	0 R		

### DMA Byte Counter

In direct mode, this counter is programmed with the number of bytes of data to be transferred.

For sDMA operations, if the port width is 16 bits, then the byte count must be initialized to an even value; if the width is 32 bits, then the byte count must be initialized to a multiple of four bytes.

If the port width is 8 bits, the byte count value has no restrictions.

ADR/SIZ	\$FFFBC030, \$48, \$60, \$78 (32 bits each)		
BIT	31...24	23	0
NAME	0	DMA_a Byte Counter	
OPER	R	R/W	
RESET	0 R		

### DMA Table Address Counter

In the command chaining mode, this counter should be loaded by the processor with the starting address of the list of commands. Note that the command packets in local bus memory must always be 16-byte aligned. That is, the starting address of any command packet must have the least significant nibble of the address set to a 0. If the Table Address counter is initialized with a value where the four least significant bits are not a 0, the logic will interpret it as a halt condition and the command chaining process will not start. Therefore the entry in the last command packet which is loaded into the Table Address counter should have one or more of these address bits set to a 1 to halt the command chaining process.

ADR/SIZ	\$FFFBC034, \$4C, \$64, \$7C (32 bits each)		
BIT	31	...	0
NAME	DMA Table Address Counter		
OPER	R/W		
RESET	0 R		

## Programming the Programmable Clock

Programmable clock registers are defined in the following paragraphs.

### Programmable Clock Interrupt Control Register

ADR/SIZ	\$FFFBC080 (8 bits)							
BIT	7	6	5	4	3	2	1	0
NAME	0	IRE	INT	IEN	ICLR	IL2	IL1	IL0
OPER	R	R/W	R	R/W	C	R/W	R/W	R/W
RESET	0 R	0 R	0 R	0 R	0 R	0 R	0 R	0 R

**IL2-0** These three bits select the interrupt level for the programmable clock interrupt. Level 0 does not generate an interrupt.

**ICLR** Writing a logic 1 to this bit clears the INT status bit. This bit always reads as 0.

**IEN** When IEN is set, the programmable clock interrupt is enabled. When IEN is cleared, the interrupt is disabled.

**INT** When this bit is high, an interrupt is being generated for the programmable clock at the level programmed in IL2-IL0.

**IRE** This bit controls which action of the programmable clock output causes interrupts.

IRE	Programmable Clock Action That Causes Interrupts
0	Rising edge
1	Falling edge

### Programmable Clock General Control Register

ADR/SIZ	\$FFFBC081 (8 bits)							
BIT	7	6	5	4	3	2	1	0
NAME	PLTY	PLS	0	EN	CLR	PS2	PS1	PS0
OPER	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W
RESET	0 R	0 R	0 R	0 R	0 R	0 R	0 R	0 R

**PS2-0** These three bits select the frequency of the pre-scale logic output. The MC680x0 bus clock (BCK) is used as the input to the pre-scale logic. BCK is either 25MHz or 32MHz. BCK frequency can be determined by examining the Version register in the MC2 sector.

PS2-PS0	Pre-scaler Output Frequency	
	PLS = 0	PLS = 1
0	BCK/2	No Output
1	BCK/4	BCK/2
2	BCK/8	BCK/4
3	BCK/16	BCK/8
4	BCK/32	BCK/16
5	BCK/64	BCK/32
6	BCK/128	BCK/64
7	BCK/256	BCK/128

**CLR** Setting this bit forces the programmable clock's internal registers (except for the interrupt and general control registers) to zero. These registers include the prescaler and timer counters. Note that these registers will remain cleared until the CLR bit is set to a 0.

**EN** When the EN bit is set, the programmable clock is enabled. When it is cleared, the programmable clock is suspended. EN performs its function by

enabling/disabling the prescaler's counter. Note that clearing EN does not clear any of the programmable clock's registers.

- PLS** When PLS is set, the programmable clock output is asserted for one BCK period. When PLS is cleared, the programmable clock output toggles creating a square wave.
- PLTY** PLTY controls the polarity of the programmable clock output. When PLTY is cleared, the negated (and cleared) state of the output is a logic 0, and the asserted state is a logic 1. When PLTY is set, the opposite is true.

### Programmable Clock Timer Register

<b>ADR/SIZ</b>	<b>\$FFFBC082 (16 bits)</b>		
<b>BIT</b>	15	...	0
<b>NAME</b>	Programmable Clock Timer Register		
<b>OPER</b>	R/W		
<b>RESET</b>	0 R		

When enabled, the programmable clock timer counter increments until it matches the value contained in this register, at which time it rolls over and resumes counting. The effect is that the frequency of the programmable clock output is the frequency of the (*pre-scaler output*)/(*the-value-in-this-register* + 1). For example, if the PLS bit is cleared, PLS2-0 are %000, and the timer register contains \$0001, then the programmable clock output frequency is  $BCK/4 = 8\text{MHz}$  if  $BCK = 32\text{MHz}$ . For the pulsed output, the formula for the period of the frequency of the recurring pulse is  $1/((\textit{pre-scaler output})/(\textit{the-value-in-this-register} + 1))$ . For example, if the PLS bit is set, PLS2-0 are %001, and the timer register contains \$0001, then the programmable clock frequency of the pulsed output is  $BCK/4 = 8\text{MHz}$  if  $BCK = 32\text{MHz}$ .



## Alternate DMA Byte Count a,b,c,d

ADR/SIZ	\$FFFBC090,94,98,9c							
BIT	23							0
NAME	abc23							abc0
OPER	R/W							R/W
RESET	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS

Address \$FFFBC090 selects Alternate DMA **Byte Count** register a.

Address \$FFFBC094 selects Alternate DMA **Byte Count** register b.

Address \$FFFBC098 selects Alternate DMA **Byte Count** register c.

Address \$FFFBC09C selects Alternate DMA **Byte Count** register d.

**abc[23:0]** Associated with each DMA Byte Count register is an Alternate DMA Byte Count register. These registers are part of an IP2 bug fix which allows flushing of the FIFOs when the DMA end signal is detected on IP read operations. This counter is set to 0 when the dmaend signal is detected. A count of zero will have the effect of flushing the FIFOs in a manner similar to the condition where the byte counter reaches a terminal count of zero. When the DMA end is detected, the contents of the DMA Byte Count register will equal the initial byte count minus the number of bytes that were transferred to the local bus memory. For all other cases, the Alternate DMA Byte counter and the DMA Byte counter contain the same value. These registers are presented in the register map for test purposes. It is not anticipated that they will serve any purpose to application software.

## Alternate DMA Control Register 2 a,b,c,d

ADR/SIZ	\$FFFBC0A0,A1.A2.A3							
BIT	7	6	5	4	3	2	1	0
NAME		<i>dmaei1</i>	<i>chain_dmaei1</i>					
OPER		R/W	R/W					
RESET	0 PLS	0 PLS	0 PLS	V PLS	V PLS	V PLS	V PLS	V PLS

*Address \$FFFBC0A0 selects Alternate DMA Control register d.*

*Address \$FFFBC0A1 selects Alternate DMA Control register c.*

*Address \$FFFBC0A2 selects Alternate DMA Control register b.*

*Address \$FFFBC0A3 selects Alternate DMA Control register a.*

***chain\_dmaei1*** Refer to the description for *dmaei1* control.

***dmaei1*** The *dmaei1* and *chain\_dmaei1* bits are part of an IP2 bug fix which allows flushing of the FIFOs when the *dmaend* signal is detected on IP read operations. This function enables the early termination of DMA if the data flow is to the host MVME1X2P4 board. When the *dmaend* signal is driven by the IP device, the DMA service is terminated and the data path is flushed. To enable this function, the *dmaei1* bit must be set in the Alternate DMA Control Register 2 and the *dmaei* bit must be set in the DMA Control Register 2. If the *chain\_dmaei1* bit is set, the *dmaei1* bit is updated by the Control Word during the DMA chained operation. The *chain\_dmaei1* bit is not updated by the DMA chained operation.

*If only the *dmaei1* bit is set in DMA Control Register 2, the DMA channel will behave in the manner of the IP2 ASIC. That is, the flawed operation is still available if a requirement for it exists.*

**DMA Status Register*****ipend1***

*The ipend1 status bit is added to the DMA Status register for each DMA channel. The ipend1 bit is set when an early termination of the DMA data is detected and the dmaei1 bit is set in the Alternate Control Register 2. The setting of this bit does not cause an interrupt. The DMA interrupt associated with this activity will be generated when the DONE bit in the DMA Status register is set. The ipend1 bit is cleared when the channel is enabled by setting the DEN bit.*

## Local Bus to IndustryPack Addressing

The following sections provide examples that illustrate local bus versus IndustryPack addressing for different IndustryPack spaces and programmed port widths. Throughout the examples, LBA refers to the local bus address defined by LA<23-0>, and IPA refers to the IndustryPack address. IPA<22-7> is the value on signal pins IPAD<15-0>/IPBD<15-0> during the select state (these only apply to memory accesses); IPA<6-1> is the value on signal pins IPA<6-1>; and IPA<0> is the value inferred by IPBS1\*, where IPA<0> is 0 if IPBS1\* is asserted and 1 if IPBS1\* is negated.

## Eight-Bit Memory Space

This example is for IP\_a, where the IP\_a memory space is programmed with a base address of \$00000000, a size of 4MB, and a port width of 8 bits. The relationship of the IndustryPack address to the local bus address is:  $IPA = (LBA \times 2) + 1$ .

4

LBA	IPA	Comments
\$00000000	\$000001	
\$00000001	\$000003	
\$00000002	\$000005	
\$00000003	\$000007	
\$003FFFFC	\$7FFFF9	
\$003FFFFD	\$7FFFFB	
\$003FFFFE	\$7FFFFD	
\$003FFFFF	\$7FFFFF	

## 16-Bit Memory Space

This example is for IP\_a, where the IP\_a memory space is programmed with a base address of \$00000000, a size of 8MB, and a port width of 16 bits. The relationship of the IndustryPack address to the local bus address is: IPA = LBA.

LBA	IPA	Comments
\$00000000	\$000000	
\$00000001	\$000001	
\$00000002	\$000002	
\$00000003	\$000003	
\$007FFFFC	\$7FFFFC	
\$007FFFFD	\$7FFFFD	
\$007FFFFE	\$7FFFFE	
\$007FFFFF	\$7FFFFF	

## 32-Bit Memory Space

This example is for IP\_ab, where the IP\_ab memory space is programmed with a base address of \$00000000, a size of 16MB, and a port width of 32 bits. The relationship of the IndustryPack address to the local bus address is:  $IPA\langle 22-1 \rangle = LBA\langle 23-2 \rangle$ , and  $IPA\langle 0 \rangle = LBA\langle 0 \rangle$ .

4

LBA	IPA	Comments
\$00000000	\$000000	IP_b or ab
\$00000001	\$000001	IP_b
\$00000002	\$000000	IP_a
\$00000003	\$000001	IP_a
\$00000004	\$000002	IP_b or ab
\$00000005	\$000003	IP_b
\$00000006	\$000002	IP_a
\$00000007	\$000003	IP_a
\$00000008	\$000004	IP_b or ab
\$00FFFFFFB	\$7FFFFFFD	IP_a
\$00FFFFFFC	\$7FFFFFFE	IP_b or ab
\$00FFFFFFD	\$7FFFFFFF	IP_b
\$00FFFFFFE	\$7FFFFFFE	IP_a
\$00FFFFFFF	\$7FFFFFFF	IP_a

## IP\_a I/O Space

This example is for IP\_a I/O space. The relationship of the IndustryPack address to the local bus address is:  $IPA\langle 6-0 \rangle = LBA\langle 6-0 \rangle$ . Note that  $IPA\langle 22-7 \rangle$  do not pertain to I/O space.

LBA	IPA<6-0>	Comments
\$FFF58000	%0000000	
\$FFF58001	%0000001	
\$FFF58002	%0000010	
\$FFF58003	%0000011	
\$FFF5807C	%1111100	
\$FFF5807D	%1111101	
\$FFF5807E	%1111110	
\$FFF5807F	%1111111	

## IP\_ab I/O Space

This example is for 32-bit, IP\_ab I/O space. The relationship of the IndustryPack address to the local bus address is:  $IPA\langle 6-1 \rangle = LBA\langle 7-2 \rangle$  and  $IPA\langle 0 \rangle = LBA\langle 0 \rangle$ . Note that  $IPA\langle 22-7 \rangle$  do not pertain to I/O space.

4

LBA	IPA<6-0>	Comments
\$FFF58400	%000000	IP_b or ab
\$FFF58401	%000001	IP_b
\$FFF58402	%000000	IP_a
\$FFF58403	%000001	IP_a
\$FFF58404	%000010	IP_b or ab
\$FFF58405	%000011	IP_b
\$FFF584FC	%111110	IP_b or ab
\$FFF584FD	%111111	IP_b
\$FFF584FE	%111110	IP_a
\$FFF584FF	%111111	IP_a



## IP\_a ID Space

This example is for IP\_a ID space. The relationship of the IndustryPack address to the local bus address is:  $IPA\langle 5-0 \rangle = LBA\langle 5-0 \rangle$ . Note that  $IPA\langle 22-6 \rangle$  do not pertain to ID space.

LBA	IPA<5-0>	Comments
\$FFF58080	%000000	
\$FFF58081	%000001	
\$FFF58082	%000010	
\$FFF58083	%000011	
\$FFF580BC	%111100	
\$FFF580BD	%111101	
\$FFF580BE	%111110	
\$FFF580BF	%111111	

## IP to Local Bus Data Routing

This section shows data routing from an IP to the local bus.

### Memory Space Accesses

The following table shows the data routing when accessing IP memory space.

IPWIDTH	LBSIZE	LBA	IPA	LD<31-24>	LD<23-16>	LD<15-8>	LD<7-0>
8 Bits	BYTE	0	1	IPXD<7-0>			
		1	3		IPXD<7-0>		
		2	5			IPXD<7-0>	
		3	7				IPXD<7-0>
	WORD	0	1,3	IPXD<7-0>	IPXD<7-0>		
		2	5,7			IPXD<7-0>	IPXD<7-0>
LWORD	0	1,3,5,7	IPXD<7-0>	IPXD<7-0>	IPXD<7-0>	IPXD<7-0>	
16 Bits	BYTE	0	0	IPXD<15-8>			
		1	1		IPXD<7-0>		
		2	2			IPXD<15-8>	
		3	3				IPXD<7-0>
	WORD	0	0	IPXD<15-8>	IPXD<7-0>		
		2	2			IPXD<15-8>	IPXD<7-0>
LWORD	0	0,2	IPXD<15-8>	IPXD<7-0>	IPXD<15-8>	IPXD<7-0>	
32 Bits	BYTE	0	0	IPBD<15-8>			
		1	1		IPBD<7-0>		
		2	0			IPAD<15-8>	
		3	1				IPAD<7-0>
	WORD	0	0	IPBD<15-8>	IPBD<7-0>		
		2	0			IPAD<15-8>	IPAD<7-0>
LWORD	0	0	IPBD<15-8>	IPBD<7-0>	IPAD<15-8>	IPAD<7-0>	

IPWIDTH refers to the memory space width that has been programmed into the general control register for the IndustryPack being accessed.

LBSIZE refers to local bus transfer size.

LBA refers to local bus address signals 1 and 0.

LD refers to the local data bus.

IPA refers to IndustryPack address signals 2,1,0. The IP2 sector implements dynamic bus sizing for memory space accesses whose local bus size is greater than the port width of the IndustryPack that is being accessed. Because of this, the IP2 sector performs 1, 2, or 4 IP memory space cycles for each local bus cycle. The IPA column in the table lists 1, 2, or 4 addresses to indicate the address for each IP cycle that is performed.

IPXD refers to the IP\_a data bus (IPAD) when accessing IP\_a or IP\_c. It refers to the IP\_b data bus (IPBD) when accessing IP\_b or IP\_d.

## I/O and ID Space Accesses

The following table shows the data routing when accessing IP I/O or ID space.

SPACE	LBSIZE	LBA	IPA	LD<31-24>	LD<23-16>	LD<15-8>	LD<7-0>	
IP_a,b,c or _d (I/O or ID)	BYTE	0	0	IPXD<15-8>				
		1	1		IPXD<7-0>			
		2	2	IPXD<15-8>				
		3	3				IPXD<7-0>	
	WORD	0	0	IPXD<15-8>	IPXD<7-0>			
		2	2			IPXD<15-8>	IPXD<7-0>	
	LWORD	0	0	IPXD<15-8>	IPXD<7-0>			
	IP_ab or _cd (I/O Only)	BYTE	0	0	IPBD<15-8>			
			1	1		IPBD<7-0>		
2			0			IPAD<15-8>		
3			1				IPAD<7-0>	
WORD		0	0	IPBD<15-8>	IPBD<7-0>			
		2	0			IPAD<15-8>	IPAD<7-0>	
LWORD		0	0	IPBD<15-8>	IPBD<7-0>	IPAD<15-8>	IPAD<7-0>	

SPACE refers to the IndustryPack space being accessed.

LBSIZE refers to local bus transfer size.

LBA refers to local bus address signals 1,0.

IPA refers to IndustryPack address signals 2,1,0.

LD refers to the local data bus.

IPXD refers to the IP\_a data bus (IPAD) when accessing IP\_a or IP\_c. It refers to the IP\_b data bus (IPBD) when accessing IP\_b or IP\_d.

---

## Introduction

The ECC DRAM Controller ASIC (MCECC) is one of three ASICs used on earlier MVME162/172 models (the MC2 chip, the IP2 chip, and the MCECC chip) whose functions are now incorporated into a single ASIC, the Petra chip, on the MVME1X2P4. The two memory controllers modeled in Petra duplicate the functionality of the “parity memory controller” found in the MC2 ASICs as well as that of the “single-bit error correcting/double-bit error detecting” memory controller found in the MCECC ASICs.

For ease of use in conjunction with programming models and documentation developed for earlier boards, however, the structure of this manual preserves the functional distinctions that formerly characterized the MC2, IP2, and MCECC ASICs. This chapter describes the MCECC sector of the Petra chip.

*The MCECC ASICs, used in a set of two, provided the interface to a 144-bit wide DRAM memory array. The Petra implementation provides an interface to a 40-bit SDRAM memory array. There are 32 bits for data, 7 for check bits, and one bit that is not used. SDRAM configurations that allow array sizes of 16MB to 128MB are supported. For a complete description of the memory configurations that are supported, refer to the definition of the SDCFG2-SDCFG0 bits in the SDRAM Configuration register.*

## Features

MCECC functions now implemented on the Petra chip include:

**Table 5-1. MCECC Functions on the Petra ASIC**

Function	Features
Memory Control	2-1-1-1 memory accesses (sustained) for burst writes
	4-1-1-1 memory accesses (sustained) for burst reads (5-1-1-1 with BERR on or when FSTRD is cleared)
	Support for byte, two-byte, four-byte, and cache line read or write transfers
	Programmable base address for DRAM
	Built-in refresh timer and refresh controller
	Programmable-period automatic scrub operation
Error Handling	ECC Single-Bit Error Detect and Correct
	Software-enabled Interrupt on Single-Bit Error
	Double-Bit Error Detect
	Software-programmable Bus Error and/or Interrupt on double-bit error

---

# Functional Description

The following sections provide an overview of the functions provided by the Petra MCECC sector. For a detailed programming model for the Petra/MCECC control and status registers, refer to the *Programming Model* section.

## General Description

The Petra MCECC sector is a single-chip solution for memory control functions. The memory architecture is a single bank of SDRAM, 32 bits wide plus seven check bits. The MCECC sector provides all the functions required to implement a memory system. These include programmable map decoding, memory control, refresh, and a memory scrubber. The scrubber, when enabled, periodically scans memory for errors. If the scrubber finds a single-bit error in the memory array, it corrects the error. This prevents soft single-bit errors from becoming double-bit errors.

5

## Performance

The Petra MCECC sector maintains tags for each internal bank of SDRAM. Each bank may be in an active or idle state. SDRAM access time is a function of the state of the bank of memory being addressed. If the bank addressed is active, performance is additionally a function of the page of memory being referenced. If the page referenced is open, access time is the shortest possible. Maximum access time will occur when the page referenced is not open, since the open page must be first closed and the desired page then opened.

Page sizes are determined by the configuration of the SDRAM device. Sizes range from 256 to 1024 memory locations per page.

The Petra MCECC sector design is targeted for SDRAM devices of the PC100 type. Memory access time are not influenced by the settings of mode bits or SDRAM speed selections.

The basic performance specifications for the MCECC sector are listed in [Table 5-2](#).

**Note** The table is not complete, because it cannot account for the effects of a write posting operation. If the Petra MCECC sector is idle and a write cycle is initiated on the local bus, the cycle is write-posted and the local bus is acknowledged in two clock ticks. If another bus cycle is initiated while the write post operation is in progress, the new cycle is stalled until the write posting is complete. *The Read cycles are extended by one clock cycle if the NCEBEN bit is set in the SDRAM Control register.*

*Since the bandwidth between the SDRAM and the processor local bus is generally higher than that of the logic it replaces (the MCECC pair and EDO DRAMs), software will take less time to execute. This could change the behavior of certain applications.*

**Table 5-2. Memory System Cycle Timing**

Access Description	Memory States		
	Idle	Active Hit	Active Miss
Read Single	4 clock cycles	3 clock cycles	5 clock cycles
Read Burst	4-1-1-1 clock cycles	3-1-1-1 clock cycles	5-1-1-1 clock cycles
Write Burst	2-1-1-1 clock cycles	2-1-1-1 clock cycles	2-1-1-1 clock cycles
Write Longword	2 clock cycles	2 clock cycles	2 clock cycles
Write 1 or 2 Bytes	9 clock cycles	8 clock cycles	10 clock cycles

## Cache Coherency

*The MCECC sector supports the MC680x0 caching scheme on the local bus by always providing 32 bits of valid data during DRAM read cycles regardless of the number of bytes requested by the local bus master for the cycle. It also supports cache coherency by monitoring the memory inhibit (MI\*) signal.*

*For a write or read cycle, the MCECC sector always waits for MI\* to be negated before it begins a read or write cycle to the DRAM. If another local bus slave asserts TA\* or TEA\* before MI\* is negated, then the MCECC sector never begins the DRAM write cycle.*



## ECC

*The Petra MCECC sector pair performs single-bit error correction and double-bit error detection (SECDED). Since the SDRAM device can deliver data from incremental addresses with each clock tick (subject to boundary limitations), the Petra MCECC sector does not implement an interleaved memory architecture. The SDRAM array is 32 data bits plus seven checkbits wide. The depth is dependent on the number and type of SDRAM devices.*

### Cycle Types

The Petra MCECC sector always initiates burst read/write accesses to the SDRAM device. If the *bus* access is not a burst, the cycle is terminated early. Single- and double-byte write cycles are read-modify-write accesses, but longword write accesses require no read cycle.

### Error Reporting

The Petra MCECC sector generates ECC check bits for write cycles. It also checks read data from the DRAM and corrects the data if it contains a single-bit error. If a non-correctable error occurs within the read data, the Petra MCECC sector so indicates by asserting its non-correctable error (NCE\*) pin.

The following paragraphs describe the actions that the MCECC sector will take in different error situations.

#### **single-Bit Error (Cycle Type = Burst Read or Non-Burst Read)**

1. Correct the data that is driven to the local MC680x0 bus.
2. Do not correct the data in DRAM. The DRAM is not corrected until the next scrub of that address, which happens only if scrubbing is enabled.
3. Terminate the cycle normally. (Assert TA\* to the local bus.)
4. Log the error if not already logged.
5. Notify the local MPU via interrupt, if so enabled.

**Double-Bit Error (Cycle Type = Burst Read or Non-Burst Read)**

You cannot correct the data that is driven to the local MC680x0 bus.

1. Leave the error in DRAM. (Note that the error is not corrected in DRAM during the next scrub of that address.)
2. Terminate the cycle with Bus Error (assert TEA\* to the local bus) if so enabled.
3. Log the error if not already logged.
4. Notify the local MPU via interrupt, if so enabled.

**Triple (or Greater) Bit Error (Cycle Type = Burst Read or Non-Burst Read)**

Some of these errors are detected correctly and are treated the same as a double-bit error. The rest could show up as "no error" or "single-bit error", both of which are incorrect.

**Cycle Type = Burst Write**

Because all of the bits are written during a burst write, no checking is done.

**Single-Bit Error (Cycle Type = Non-Burst Write)**

1. Correct the data read from the DRAM, merge with the write data, and write the correct, merged data to the DRAM.
2. Terminate the cycle normally. (Assert TA\* to the local bus.)
3. Log the error if not already logged.
4. Notify the local MPU via interrupt if so enabled.

**Double-Bit Error (Cycle Type = Non-Burst Write)**

1. Do not perform the write portion of the cycle. (This causes the location to continue to indicate a non-correctable error when accessed.)
2. Terminate the cycle normally. (Assert TA\* to the local bus.)
3. Log the error if not already logged.

4. Notify the local MPU via interrupt if so enabled.

**Triple (or Greater) Bit Error (Cycle Type = Non-Burst Write)**

Some of these errors are detected correctly and are handled the same as a double-bit error. The rest may show up as "no error" or "single-bit error", both of which are incorrect.

**Single-Bit Error (Cycle Type = Scrub)**

1. Write corrected data to the DRAM.
2. Log the error if not already logged.
3. Notify the local MPU via interrupt if so enabled.

**Double-Bit Error (Cycle Type = Scrub)**

1. Do not perform the write portion of the cycle. (This causes the location to continue to indicate a non-correctable error when accessed.)
2. Log the error if not already logged.
3. Notify the local MPU via interrupt if so enabled.

**Triple (or Greater) Bit Error (Cycle Type = Scrub)**

Some of these errors are detected correctly and are treated the same as a double-bit error. The rest may show up as "no error" or "single-bit error", both of which are incorrect.

## Error Logging

ECC error logging is facilitated by the Petra MCECC sector because of its internal latches. When an error (single- or double-bit) occurs in the DRAMs to which the MCECC sector is connected, it freezes the address of the error and the syndrome bits associated with the data that is in error. Each MCECC sector segment performs this logging function independently of the other. Once the MCECC sector has logged an error, it does not log any new errors that occur until the ERRLOG control/status bit has been cleared by software.

## Scrub

The MCECC sector contains programmable registers and circuitry to implement the memory scrubbing function. Programmable registers determine how often the entire DRAM is scrubbed. During a scrub, the scrubber holds the memory for a programmable amount of time and then releases it for the local bus, or for a refresher if one of them is requesting local bus mastership. The scrubber then refrains from using the DRAM again for a programmable amount of time. Each scrub cycle is made up of a full 39-bit read of DRAM, a correction of any single-bit errors, and a write of the full 39 corrected bits back to the same location. If a single- or double-bit error occurs and if such interrupts are enabled in the control register, the local bus master is notified. A software bit is available to disable the read portion of the scrub cycle.

## Refresh

The MCECC sector provides refresh control for the DRAM. It performs a single CAS-before-RAS refresh cycle to the two DRAM blocks approximately once every 15.6  $\mu\text{s}$ .

## Arbitration

The MCECC sector has three different entities that can request use of the DRAM cycle controller: (1) the local bus master, (2) the refresher, and (3) the scrubber.

The MCECC pair arbiter accepts requests and provides grants to the requesting entities as follows:

- ❑ Priority is (highest to lowest) refresher, local bus, and scrubber.
- ❑ When no requests are pending, the arbiter defaults to providing a local bus grant for fast response to local 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.

5

## Chip Defaults

*Certain parameters in the Petra MCECC sector have to be configured. These include DRAM size, DRAM speed, Control and Status register selection, etc. The configuration parameters are loaded into the Defaults 1, Defaults 2, and SDRAM Configuration registers on the first clock edge after reset negation. Software can override this initial setting by writing to the Defaults registers. It is not recommended that non-test software alter the contents of the Defaults registers, however. The actual values loaded into the Defaults registers are determined by board-level jumpers and configuration resistors.*

## Programming Model

This section defines the programming model for the control and status registers (CSRs) in the MCECC sector. The base address of the CSRs is hard-coded to the address \$FFF43000 for the MCECC sector on the first mezzanine board and to \$FFF43100 for the MCECC sector on the second mezzanine board.

*Note that several bits in the register map have changed in functionality from the MCECC ASIC pair. In most cases these bits were defined to be nonoperational in the MCECC model, but were also defined as to their original intent. This specification entirely omits those bit definitions.*

The possible operations for each bit in the CSR are as follows:

- R**            The bit is a read only status bit.
- R/W**        The bit is readable and writable.
- R/C**        The status bit is cleared by writing a 1 to it.
- C**            Writing a 0 to the bit clears it or another bit.  
This bit reads 0.
- S**            Writing a 1 to the bit sets it or another bit.  
This bit reads 0.

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

- P**            The bit is affected by power-up reset.
- L**            The bit is affected by local reset.
- S**            The bit is affected by software reset (writing \$0F to the Chip ID register).
- X**            The bit is not affected by reset.
- V**            The effect of reset on this bit is variable.

**Table 5-3. MCECC Sector Internal Register Memory Map**MCECC Sector Base Address = \$FFF43000 (first *board*); \$FFF43100 (second *board*)

Register		Register Bit Names							
Offset	Name	D31	D30	D29	D28	D27	D26	D25	D24
\$00	CHIP ID	CID7	CID6	CID5	CID4	CID3	CID2	CID1	CID0
\$04	CHIP REVISION	REV7	REV6	REV5	REV47	REV3	REV2	REV1	REV0
\$08	MEMORY CONFIG	0	0	FSTRD	1	0	MSIZ2	MSIZ1	MSIZ0
\$0C		0	0	0	0	0	0	0	0
\$10		0	0	0	0	0	0	0	0
\$14	BASE ADDRESS	BAD31	BAD30	BAD29	BAD28	BAD27	BAD26	BAD25	BAD24
\$18	DRAM CONTROL	BAD23	BAD22	RWB5	<b>RWB4</b>	RWB3	NCEIEN	NCEBEN	RAMEN
\$1C	BCLK FREQUENCY	BCK7	BCK6	BCK5	BCK47	BCK3	BCK2	BCK1	BCK0
\$20	DATA CONTROL	0	0	DERC	ZFILL	RWCKB	0	0	0
\$24	SCRUB CONTROL	<b>0</b>	<b>0</b>	<b>0</b>	SCRB	SCRBEN	0	SBEIEN	<b>RWB0</b>
\$28	SCRUB PERIOD	SBPD15	SBPD14	SBPD13	SBPD12	SBPD11	SBPD10	SBPD9	SBPD8
\$2C	SCRUB PERIOD	SBPD7	SBPD6	SBPD5	SBPD4	SBPD3	SBPD2	SBPD1	SBPD07
\$30	CHIP PRESCALE	CPS7	CPS6	CPS5	CPS4	CPS3	CPS2	CPS1	CPS0
\$34	SCRUB TIME ON/OFF	SRDIS	0	STON2	STON1	STON0	STOFF2	STOFF1	SRDIS
\$38	SCRUB PRESCALE	0	0	SPS21	SPS20	SPS19	SPS18	SPS17	SPS16
\$3C	SCRUB PRESCALE	SPS15	SPS14	SPS13	SPS12	SPS11	SPS10	SPS9	SPS85
\$40	SCRUB PRESCALE	SPS7	SPS6	SPS5	SPS4	SPS3	SPS2	SPS1	SPS0
\$44	SCRUB TIMER	ST15	ST14	ST13	ST12	ST11	ST10	ST9	ST8

**Table 5-3. MCECC Sector Internal Register Memory Map (Continued)**MCECC Sector Base Address = \$FFF43000 (first *board*); \$FFF43100 (second *board*)

Register		Register Bit Names							
Offset	Name	D31	D30	D29	D28	D27	D26	D25	D24
\$48	SCRUB TIMER	ST7	ST6	ST5	ST4	ST3	ST2	ST1	ST0
\$4C	SCRUB ADDR CNTR	0	0	0	0	0	SAC26	SAC25	SAC24
\$50	SCRUB ADDR CNTR	SAC23	SAC22	SAC21	SAC20	SAC19	SAC18	SAC17	SAC16
\$54	SCRUB ADDR CNTR	SAC15	SAC14	SAC13	SAC12	SAC11	SAC10	SAC9	SAC8
\$58	SCRUB ADDR CNTR	SAC7	SAC6	SAC5	SAC4	07	0	0	0
\$5C	ERROR LOGGER	ERRLO G	ERD	ESCRB	ERA	EALT	0	MBE	SBE
\$60	ERROR ADDRESS	EA31	EA30	EA29	EA28	EA27	EA26	EA25	EA24
\$64	ERROR ADDRESS	EA23	EA22	EA21	EA20	EA19	EA18	EA17	EA16
\$68	ERROR ADDRESS	EA15	EA14	EA13	EA12	EA11	EA10	EA9	EA8
\$6C	ERROR ADDRESS	EA7	EA6	EA5	EA4	07	0	0	0
\$70	ERROR SYNDROME	S7	S6	S5	S4	S3	S2	S1	S0
\$74	DEFAULTS1	<b>RWB7</b>	<b>RWB6</b>	FSTRD	SELI1	SELI0	RSIZ2	RSIZ1	RSIZ0
\$78	DEFAULTS2	<b>RWB7</b>	<b>RWB6</b>	REFDIS	TVECT	NOCACHE	RESST2	RESST1	RESST0
<b>\$7C</b>	<b>SDRAM CONFIG</b>						<b>SDCFG2</b>	<b>SDCFG1</b>	<b>SDCFG0</b>



## Chip ID Register

The Chip ID register is hard-wired to a hexadecimal value of \$81. The Petra MCECC sector can be given a software reset by writing a value of \$0F to this register. This write is terminated properly with TA\* and sets most internal registers to their default (power-up) state. Although writes of any value other than \$0F to this register are ignored, the MCECC sector always terminates the cycles properly with TA\*.

ADR/SIZ	1st \$FFF43000/2nd \$FFF43100 (8-bits)							
BIT	31	30	29	28	27	26	25	24
NAME	CID7	CID6	CID5	CID4	CID3	CID2	CID1	CID0
OPER	R	R	R	R	R	R	R	R
RESET	X	X	X	X	X	X	X	X

5

## Chip Revision Register

The Chip Revision register is hard-wired to reflect the revision level of the Petra/MCECC ASIC. The current value of the register is \$01. Although writes to this register are ignored, the MCECC sector pair always terminates the cycles properly with TA\*.

ADR/SIZ	1st \$FFF43004/2nd \$FFF43104 (8-bits)							
BIT	31	30	29	28	27	26	25	24
NAME	REV7	REV6	REV5	REV4	REV3	REV2	REV1	REV0
OPER	R	R	R	R	R	R	R	R
RESET	X	X	X	X	X	X	X	X

## Memory Configuration Register

ADR/SIZ	1st \$FFF43008/2nd \$FFF43108 (8-bits)							
BIT	31	30	29	28	27	26	25	24
NAME	0	0	FSTRD	0	0	MSIZ2	MSIZ1	MSIZ0
OPER	R	R	R	R	R	R	R	R
RESET	X	X	X	X	X	X	X	X

5

### MSIZ2-MSIZ0

MSIZ2-MSIZ0 together define the size of the total memory to be controlled by the MCECC sector. These bits reflect the RSIZ2-RSIZ0 bits in Defaults Register 1.

MSIZ2	MSIZ1	MSIZ0	Memory Size	MSIZ2	MSIZ1	MSIZ0	Memory Size
0	0	0	4MB	1	0	0	64MB
0	0	1	8MB	1	0	1	128MB
0	1	0	16MB	1	1	0	Reserved
0	1	1	32MB	1	1	1	Reserved

**Note** Remember that the DRAM organization presented in the table above is relevant to the extent that it aids in emulating DRAM configurations from earlier programming models. For the *actual* SDRAM device and size options now applicable to the MVME1X2P4 boards, refer to [Table 1-1](#).

**FSTRD** FSTRD reflects the state of the FSTRD bit in Defaults Register 1. When 1, this bit indicates that DRAM reads are operating at full speed. When 0, it indicates that DRAM read accesses are slowed by one clock cycle.

## Base Address Register

These eight bits are combined with the two most significant bits in Register 7 (the next register) to form BAD31-BAD22, which defines the base address of the memory. For larger memory sizes, the lower significant bits are ignored.

The bit assignments for the Base Address register are:

ADR/SIZ	1st \$FFF43014/2nd \$FFF43114 (8-bits)							
<b>BIT</b>	31	30	29	28	27	26	25	24
<b>NAME</b>	BAD31	BAD30	BAD29	BAD28	BAD27	BAD26	BAD25	BAD24
<b>OPER</b>	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
<b>RESET</b>	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS

5

## DRAM Control Register

The bit assignments for the DRAM Control register are:

ADR/SIZ	1st \$FFF43018/2nd \$FFF43118 (8-bits)							
<b>BIT</b>	31	30	29	28	27	26	25	24
<b>NAME</b>	BAD23	BAD22	RWB5	RWB4	RWB3	NCEIEN	NCEBEN	RAMEN
<b>OPER</b>	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
<b>RESET</b>	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS

**RAMEN** RAM Enable. This control bit is used to enable the local bus to perform read/write accesses to the memory. Accesses are enabled when this bit is set and are disabled when this bit is cleared. *This bit should only be set after BAD31-BAD22 have been initialized.*

**NCEBEN** Setting the NCEBEN control bit enables the MCECC pair to assert TEA\* when a non-correctable error occurs during a local bus access to memory. In some cases setting NCEBEN causes DRAM accesses to be delayed by one clock. This delay is incurred when the access is a local bus (or scrub) read and the FSTRD bit is set.

**NCEIEN** When NCEIEN is set, the logging of a non-correctable error causes the INT signal pin to pulse true. Note that NCEIEN has no effect on DRAM access time.

**RWB3** Read/Write Bit 3 is a general purpose read/write bit.

**RWB4** Read/Write Bit 4 is a general purpose read/write bit.

**RWB5** Read/Write Bit 5 is a general purpose read/write bit.

**BAD22, BAD23**

These are the lower two bits of the DRAM base address described in the previous register.

## BCLK Frequency Register

The Bus Clock (BCLK) Frequency register should be programmed with the hexadecimal value of the operating clock frequency in MHz (i.e., \$19 for 25MHz and \$21 for 33MHz). The MCECC sector pair uses the value programmed in this register to control the Prescaler counter. The Prescaler counter increments to \$FF and then it is loaded with the two's complement of the value in the BCLK Frequency register. This produces a 1MHz clock that is used by the refresh timer and the scrubber. When the BCLK Frequency register is correctly programmed with the BCLK frequency, the DRAMs are refreshed approximately once every 15.6 microseconds. After power-up, this register is initialized to \$19 (for 25MHz).

**Note** This register is configured only during power-up-reset and is unchanged by software or local reset.

ADR/SIZ	1st \$FFF4301C/2nd \$FFF4311C (8-bits)							
BIT	31	30	29	28	27	26	25	24
NAME	BCK7	BCK6	BCK5	BCK4	BCK3	BCK2	BCK1	BCK0
OPER	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RESET	0 P	0 P	0 P	1 P	1 P	0 P	0 P	1P

## Data Control Register

ADR/SIZ	1st \$FFF43020/2nd \$FFF43120 (16-bits)							
BIT	31	30	29	28	27	26	25	24
NAME	0	0	DERC	ZFILL	RWCKB	0	0	0
OPER	R	R	R/W	R/W	R/W	R	R	R
RESET	X	X	1 PLS	0 PLS	0 PLS	X	X	X

**RWCKB** *READ/WRITE CHECKBITS*, when set, enables the data from the seven checkbits in the Petra MCECC sector (bits 30-24) to be written and read on the local MC680x0 data bus. This bit should be cleared for normal system operation.

Note that if test software forces a single-bit error to a location (line) 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 scrubbing and making sure that all previous scrubs have completed before performing the test. Also note that writing bad checkbits can set the ERRLOG bit in the Error Logger register.

The writing of checkbits causes the MCECC sector 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

checkbits operation. To avoid this, it is recommended that the DERC bit also be set while the RWCKB bit is set. A suggested sequence for performing read-write checkbits is as follows:

1. Stop all scrub operations by clearing all of the STON bits and setting all of the STOFF bits in the Scrub Time On/Time Off register.
2. Set the DERC and RWCKB bits in the Data Control register.
3. Perform the desired read and/or write checkbit operations.
4. Clear the DERC and RWCKB bits in the Data Control register.
5. Perform the desired testing related to the location/locations that have had their checkbits altered.
6. Allow the scrubber to proceed by restoring the STON and STOFF bits to their original state.

**ZFILL** ZERO FILL memory, when set, forces all zeros to be written to the DRAM during any kind of write cycle or scrub cycle. It is intended for use with the zero-fill function (refer to the section on *Initialization* at the end of this chapter). This bit should be cleared for normal system operation.

*When ZFILL is set, the read portion of a scrub cycle is suppressed and writes of all zeros are executed.*

**DERC** DISABLE ERROR CORRECTION, when set to 1, disables single-bit error correction by the Petra MCECC sector. Specifically, data read from the DRAM array is presented to the local MC680x0 data bus unaltered. Less-than-line write data performs a read-modify-write without correcting single-bit errors that may occur on the read portion of the read-modify-write. Note that DERC does not affect the generation of check bits. DERC should be cleared during normal system operation. DERC also allows the write portion of a read-modify-write to

complete regardless of whether or not there was a multiple bit error during the read portion of the read-modify-write. DERC also affects scrub cycles.

## Scrub Control Register

ADR/SIZ	1st \$FFF43024/2nd \$FFF43124 (8-bits)							
BIT	31	30	29	28	27	26	25	24
NAME	0	0	0	SCRB	SCRBEN	0	SBEIEN	RWB0
OPER	R	R	R	R	R/W	R	R/W	R/W
RESET	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	X	0 PLS	0 PLS

**RWB0** *RWB0 is a general-purpose read/write bit.*

**SBEIEN** Setting SBEIEN causes the logging of a single-bit error to create a true pulse on the INT signal pin.

**SCRBEN** This control bit enables the scrubber to operate. When SCRBEN is set, the MCECC sector immediately performs a scrub of the entire DRAM array. When the scrub is complete, if software has cleared SCRBEN, then scrubbing is not done again, until software sets the SCRBEN bit. If software has not cleared the SCRBEN bit, then when the amount of time indicated in the Scrub Period (SBPD) register expires, the MCECC sector scrubs the DRAM array again. It continues to perform scrubs of the entire DRAM array at the frequency indicated in the SBPD register. The scrubber does not start a new scrub once the SCRBEN bit is cleared. The time between scrubs is approximately two seconds times the value stored in the SBPD register. Note that a power-up, local, or software reset stops the scrubber.

**SCRB** This status bit reflects the state of the scrubber. When the scrubber is in the process of doing a scrub, this bit is set. When the scrubber is between scrubs, this bit is cleared.

## Scrub Period Register Bits 15-8

The Scrub Period Control register controls how often a scrub of the entire memory is performed if the SCR BEN bit is set in the Scrub Control register. The time between scrubs is approximately two seconds times the value programmed into the Scrub Period register. The scrub period can be programmed from once every four seconds to once every 36 hours. This register contains bits 15-8 of the Scrub Period register.

ADR/SIZ	1st \$FFF43028/2nd \$FFF43128 (8-bits)							
<b>BIT</b>	31	30	29	28	27	26	25	24
<b>NAME</b>	SBPD15	SBPD14	SBPD13	SBPD12	SBPD11	SBPD10	SBPD9	SBPD8
<b>OPER</b>	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
<b>RESET</b>	1 PLS	1 PLS	1 PLS	1 PLS	1 PLS	1 PLS	1 PLS	1 PLS

## Scrub Period Register Bits 7-0

This register contains bits 7-0 of the Scrub Period register.

ADR/SIZ	1st \$FFF4302C/2nd \$FFF4312C (8-bits)							
<b>BIT</b>	31	30	29	28	27	26	25	24
<b>NAME</b>	SBPD7	SBPD67	SBPD5	SBPD4	SBPD3	SBPD2	SBPD1	SBPD0
<b>OPER</b>	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
<b>RESET</b>	1 PLS	1 PLS	1 PLS	1 PLS	1 PLS	1 PLS	1 PLS	1 PLS



## Chip Prescaler Counter

This register reflects the current value in the prescaler counter. The Prescaler counter is used with the BCLK Frequency register to produce a 1MHz clock signal for use by the refresher and by the scrubber. The register is readable and writable for test purposes. Programming of this register is not recommended.

ADR/SIZ	1st \$FFF43030/2nd \$FFF43130 (8-bits)							
<b>BIT</b>	31	30	29	28	27	26	25	24
<b>NAME</b>	CPS7	CPS6	CPS57	CPS4	CPS3	CPS2	CPS1	CPS0
<b>OPER</b>	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
<b>RESET</b>	0 P	0 P	0 P	0 P	0 P	0 P	0 P	0 P

5

## Scrub Time On/Time Off Register

ADR/SIZ	1st \$FFF43034/2nd \$FFF43134 (8-bits)							
<b>BIT</b>	31	30	29	28	27	26	25	24
<b>NAME</b>	RWB7	0	STON2	STON1	STON0	STOFF2	STOFF1	STOFF0
<b>OPER</b>	R/W	R	R/W	R/W	R/W	R/W	R/W	R/W
<b>RESET</b>	0 PLS	0	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS

### STOFF2-STOFF0

STOFF2-STOFF0 control the amount of time that the scrubber refrains from requesting use of the DRAM each time it gives it up during a scrub. They control the off time as follows:

<b>STOFF2</b>	<b>STOFF1</b>	<b>STOFF0</b>	<b>Scrubber Time Off</b>
0	0	0	Request DRAM immediately
0	0	1	Request DRAM after 16 BCLK cycles
0	1	0	Request DRAM after 32 BCLK cycles
0	1	1	Request DRAM after 64 BCLK cycles
1	0	0	Request DRAM after 128 BCLK cycles
1	0	1	Request DRAM after 256 BCLK cycles
1	1	0	Request DRAM after 512 BCLK cycles
1	1	1	Request DRAM never

**STON2-STON0**

STON2-STON0 control the amount of time that the scrubber occupies the DRAM before providing a window during which the local bus and refresher might use it. They control the on time as follows:

<b>STON2</b>	<b>STON1</b>	<b>STON0</b>	<b>Scrubber Time On</b>
0	0	0	Keep DRAM for 1 memory cycle
0	0	1	Keep DRAM for 16 BCLK cycles
0	1	0	Keep DRAM for 32 BCLK cycles
0	1	1	Keep DRAM for 64 BCLK cycles
1	0	0	Keep DRAM for 128 BCLK cycles
1	0	1	Keep DRAM for 256 BCLK cycles
1	1	0	Keep DRAM for 512 BCLK cycles
1	1	1	Keep DRAM for TOTAL SCRUB TIME

Note that if STON2-0 is zero, the scrubber always releases the DRAM after one memory cycle, even if neither the local bus nor refresher need it.

**RWB7** *Read/Write Bit 7 is a general-purpose read/write bit.*

## Scrub Prescaler Counter (Bits 21-16)

The Scrub Prescaler counter uses the 1MHz clock as an input to create the 0.5 Hz clock that is used for the scrub period. Writes to this address update the scrub prescaler. Reads to this address yield the value in the scrub prescaler. The ability to read and write to the scrub prescaler is provided for test purposes. Programming this counter is not recommended. This register reflects the current value in the scrub prescaler bits 21-16.

ADR/SIZ	1st \$FFF43038/2nd \$FFF43138 (8-bits)							
<b>BIT</b>	31	30	29	28	27	26	25	24
<b>NAME</b>	0	0	SPS21	SPS20	SPS19	SPS18	SPS17	SPS16
<b>OPER</b>	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
<b>RESET</b>	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS

## Scrub Prescaler Counter (Bits 15-8)

This register reflects the current value in the scrub prescaler bits 15-8.

ADR/SIZ	1st \$FFF4303C/2nd \$FFF4313C (8-bits)							
<b>BIT</b>	31	30	29	28	27	26	25	24
<b>NAME</b>	SPS15	SPS14	SPS13	SPS12	SPS11	SPS10	SPS9	SPS8
<b>OPER</b>	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
<b>RESET</b>	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS

## Scrub Prescaler Counter (Bits 7-0)

This register reflects the current value in the scrub prescaler bits 7-0.

ADR/SIZ	1st \$FFF43040/2nd \$FFF43140 (8-bits)							
<b>BIT</b>	31	30	29	28	27	26	25	24
<b>NAME</b>	SPS7	SPS6	SPS5	SPS4	SPS3	SPS2	SPS1	SPS0
<b>OPER</b>	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
<b>RESET</b>	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS

5

## Scrub Timer Counter (Bits 15-8)

This read/write register is the Scrub Timer counter. If scrubbing is enabled and the Scrub Period register is non-zero, the Scrub Timer counter increments approximately once every two seconds until it matches the value programmed into the Scrub Period register, at which time it clears and resumes incrementing. Writes to this address update the Scrub Timer counter and reads to this address yield the counter's value. The ability to read and write this register is provided for test purposes. Programming this counter is not recommended. This register reflects the current value in the Scrub Timer counter bits 15-8.

ADR/SIZ	1st \$FFF43044/2nd \$FFF43144 (8-bits)							
<b>BIT</b>	31	30	29	28	27	26	25	24
<b>NAME</b>	ST15	ST14	ST13	ST12	ST11	ST10	ST9	ST8
<b>OPER</b>	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
<b>RESET</b>	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS

## Scrub Timer Counter (Bits 7-0)

This register reflects the current value in the Scrub Timer counter bits 7-0.

ADR/SIZ	1st \$FFF43048/2nd \$FFF43148 (8-bits)							
<b>BIT</b>	31	30	29	28	27	26	25	24
<b>NAME</b>	ST7	ST6	ST5	ST4	ST3	ST2	ST1	ST0
<b>OPER</b>	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
<b>RESET</b>	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS

5

## Scrub Address Counter (Bits 26-24)

This read/write register is the Scrub Address counter. Each time the scrubber performs a scrub memory cycle, the Scrub Address counter increments. For an entire scrub, the Scrub Address counter starts at 0 and increments until it reaches the DRAM size that is indicated by the MEMSIZ pins. Writes to this address update the Scrub Address counter; reads to this address yield the value in the counter. The ability to read and write this counter is provided for test purposes. Note that if scrubbing is in process, the Scrub Time On/Time Off register should be set for the minimum time on and the maximum time off during any writes to this register. This register reflects the current value in the Scrub Address counter bits 26-24.

ADR/SIZ	1st \$FFF4304C/2nd \$FFF4314C (8-bits)							
<b>BIT</b>	31	30	29	28	27	26	25	24
<b>NAME</b>	0	0	0	0	0	SAC26	SAC25	SAC24
<b>OPER</b>	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
<b>RESET</b>	X	X	X	X	X	0 PLS	0 PLS	0 PLS

## Scrub Address Counter (Bits 23-16)

This register reflects the current value in the Scrub Address counter bits 23-16.

ADR/SIZ	1st \$FFF43050/2nd \$FFF43150 (8-bits)							
<b>BIT</b>	31	30	29	28	27	26	25	24
<b>NAME</b>	SAC23	SAC22	SAC21	SAC20	SAC19	SAC18	SAC17	SAC16
<b>OPER</b>	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
<b>RESET</b>	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS

## Scrub Address Counter (Bits 15-8)

This register reflects the current value in the Scrub Address counter bits 15-8.

ADR/SIZ	1st \$FFF43054/2nd \$FFF43154 (8-bits)							
<b>BIT</b>	31	30	29	28	27	26	25	24
<b>NAME</b>	SAC15	SAC14	SAC13	SAC12	SAC11	SAC10	SAC9	SAC8
<b>OPER</b>	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
<b>RESET</b>	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS

## Scrub Address Counter (Bits 7-4)

This register reflects the current value in the Scrub Address counter bits 7-4.

ADR/SIZ	1st \$FFF43058/2nd \$FFF43158 (8-bits)							
<b>BIT</b>	31	30	29	28	27	26	25	24
<b>NAME</b>	SAC7	SAC6	SAC5	SAC4	0	0	0	0
<b>OPER</b>	R/W	R/W	R/W	R/W	R	R	R	R
<b>RESET</b>	0 PLS	0 PLS	0 PLS	0 PLS	X	X	X	X

## Error Logger Register

ADR/SIZ	1st \$FFF4305C/2nd \$FFF4315C (8-bits)							
<b>BIT</b>	31	30	29	28	27	26	25	24
<b>NAME</b>	ERRLOG	ERD	ESCRB	ERA	EALT	0	MBE	SBE
<b>OPER</b>	R/C	R	R	R	R	R	R	R
<b>RESET</b>	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	X	0 PLS	0 PLS

- SBE** SINGLE BIT ERROR is set when the last error logged was due to a single-bit error. It is cleared when a 1 is written to the ERRLOG bit. The syndrome code reflects the bit in error. (Refer to the *Syndrome Decoding* section.)
- MBE** MULTIPLE BIT ERROR is set when the last error logged was due to a multiple bit error. It is cleared when a 1 is written to the ERRLOG bit. The syndrome code is meaningless if MBE is set.
- ERA** This bit provides status for a function that is not currently used in the MCECC sector.
- EALT** EALT indicates that the last logging of an error occurred on a DRAM access by an alternate (MI\* not asserted) local bus master.
- ESCRB** ESCRB indicates the entity that was accessing DRAM at the last logging of a single- or double-bit error. If ESCRB is 1, it indicates that the scrubber was accessing DRAM. If ESCRB is 0, it indicates that the local MC680x0 bus master was accessing DRAM.
- ERD** ERD reflects the state of the local bus READ signal pin at the last logging of a single- or double-bit error. ERD = 1 corresponds to READ = high and ERD = 0 to READ = low. ERD is meaningless if ESCRB is set.
- ERRLOG** When set, ERRLOG indicates that a single- or a double-bit error has been logged by this MCECC, and that no more will be logged until it is cleared. The bit can only be set by logging an error and cleared by writing a 1 to it.

When ERRLOG is cleared, the MCECC is ready to log a new error. Note that because hardware duplicates control register writes to both MCECCs, clearing ERRLOG in one MCECC sector clears it in the other. Any available error information in either MCECC should be recovered before clearing ERRLOG.

## Error Address (Bits 31-24)

5

This register reflects the value that was on bits 31-24 of the local MC680x0 address bus at the last logging of an error.

ADR/SIZ	1st \$FFF43060/2nd \$FFF43160 (8-bits)							
<b>BIT</b>	31	30	29	28	27	26	25	24
<b>NAME</b>	EA31	EA30	EA29	EA28	EA27	EA26	EA25	EA24
<b>OPER</b>	R	R	R	R	R	R	R	R
<b>RESET</b>	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS

## Error Address (Bits 23-16)

This register reflects the value that was on bits 23-16 of the local MC680x0 address bus at the last logging of an error.

ADR/SIZ	1st \$FFF43064/2nd \$FFF43164 (8-bits)							
<b>BIT</b>	31	30	29	28	27	26	25	24
<b>NAME</b>	EA23	EA22	EA21	EA20	EA19	EA18	EA17	EA16
<b>OPER</b>	R	R	R	R	R	R	R	R
<b>RESET</b>	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS



## Error Address Bits (15-8)

This register reflects the value that was on bits 15-8 of the local MC680x0 address bus at the last logging of an error.

ADR/SIZ	1st \$FFF43068/2nd \$FFF43168 (8-bits)							
<b>BIT</b>	31	30	29	28	27	26	25	24
<b>NAME</b>	EA15	EA14	EA13	EA12	EA11	EA10	EA9	EA8
<b>OPER</b>	R	R	R	R	R	R	R	R
<b>RESET</b>	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS

5

## Error Address Bits (7-4)

This register reflects the value that was on bits 7-4 of the local MC680x0 bus at the last logging of an error.

ADR/SIZ	1st \$FFF4306C/2nd \$FFF4316C (8-bits)							
<b>BIT</b>	31	30	29	28	27	26	25	24
<b>NAME</b>	EA7	EA6	EA5	EA4	0	0	0	0
<b>OPER</b>	R	R	R	R	R	R	R	R
<b>RESET</b>	0 PLS	0 PLS	0 PLS	0 PLS	X	X	X	X

## Error Syndrome Register

ADR/SIZ	1st \$FFF43070/2nd \$FFF43170 (16-bits)							
<b>BIT</b>	31	30	29	28	27	26	25	24
<b>NAME</b>	0	S6	S5	S4	S3	S2	S1	S0
<b>OPER</b>	R	R	R	R	R	R	R	R
<b>RESET</b>	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS

5

### S6-S0

*Bits SYNDROME6-0 reflect the syndrome value at the last logging of an error. The seven-bit code indicates the position of the data error. When all the bits are 0, there is no error. Note that if the logged error was non-correctable, then these bits are meaningless (refer to the [Syndrome Decoding](#) section).*

## Defaults Register 1

ADR/SIZ	1st \$FFF43074/2nd \$FFF43174 (8-bits)							
<b>BIT</b>	31	30	29	28	27	26	25	24
<b>NAME</b>	RWB7	RWB6	FSTRD	SELI1	SELI0	RSIZ2	RSIZ1	RSIZ0
<b>OPER</b>	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
<b>RESET</b>	0 PL	V PLS	V PLS	V PLS	V PLS	V PLS	V PLS	V PLS

It is not recommended that non-test software write to this register.

**RSIZ2-RSIZ0**

Bits RSIZ2-RSIZ0 determine the size of the DRAM array that is assumed by the MCECC. They control the size as follows:

RSIZ2	RSIZ1	RSIZ0	DRAM Array Size
0	0	0	4MB
0	0	1	8MB
0	1	0	16MB
0	1	1	32MB
1	0	0	64MB
1	0	1	128MB
1	1	0	Reserved
1	1	1	Reserved

The states of RSIZ2-0 after reset (power-up, soft, or local) match those of the RSIZ2-0 bits from the reset serial bit stream.

**SELI1, SELI0**

The SELI1, SELI0 control bits determine the base address at which the control and status registers respond, as shown below:

SELI1	SELI0	Register Base Address
0	1	\$FFF43000
1	0	\$FFF43100

*SELI1 and SELI0 are initialized by hardware after a power-up, soft, or local reset. Their initialized state is determined by board-level configuration resistors.*

**FSTRD**

The FSTRD control bit determines the speed at which SDRAM reads occur. When it is 1, SDRAM reads happen at full speed. When it is 0, SDRAM reads are slowed by one clock, unless they are already slowed by NCEBEN

being set. The state of FSTRD after a reset (power-up, soft, or local) is determined by board-level configuration resistors.

**RWB6** *Read/Write Bit 6 is a general-purpose read/write bit.*

**RWB7** *Read/Write Bit 7 is a general-purpose read/write bit.*

## Defaults Register 2

5

ADR/SIZ	1st \$FFF43078/2nd \$FFF43178 (8-bits)							
BIT	31	30	29	28	27	26	25	24
NAME	0	0	0	0	0	RESST2	RESST1	RESST0
OPER	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RESET	0 PLS	0 PLS	0 PLS	V PLS	V PLS	V PLS	V PLS	V PLS

It is not recommended that non-test software write to this register.

### **RESST2-RESST0**

These general-purpose read/write bits are initialized by a power-up, soft, or local reset to match the RESST2-RESST0 bits from the reset serial bit stream.

## SDRAM Configuration Register

ADR/SIZ	1st \$FFF4307c/2nd \$FFF4317c (8-bits)							
BIT	31	30	29	28	27	26	25	24
NAME	0	0	0	0	0	SDCFG2	SDCFG1	SDCFG0
OPER	R	R	R	R	R	R	R	R
RESET	0 PLS	0 PLS	0 PLS	V PLS	V PLS	V PLS	V PLS	V PLS

### SDCFG2-SDCFG0

Define the physical SDRAM memory population on the printed circuit board:

SDCFG2	SDCFG1	SDCFG0	DRAM Array Size
0	0	0	SDRAM device is 64MBit x 16 data with one bank composed of 3 devices
0	0	1	SDRAM device is 64MBit x 8 data with one bank composed of 5 devices
0	1	0	SDRAM device is 64MBit x 8 data with two banks composed of 5 devices each
0	1	1	SDRAM device is 64MBit x 8 data with four banks composed of 5 devices each
1	0	0	SDRAM device is 128MBit x 8 data with one bank composed of 5 devices
1	0	1	SDRAM device is 128MBit x 8 data with two banks composed of 5 devices each
1	1	0	reserved
1	1	1	reserved

## Initialization

Most DRAM vendors require that the DRAMs be subjected to some number of access cycles before the DRAMs are fully operational. The MCECC does not perform this initialization automatically, but depends on software to perform enough dummy accesses to DRAM to meet the requirement. The number of cycles required is fewer than 10. If there are multiple blocks of DRAM, software has to perform at least 10 accesses to each block.

The MCECC pair provides a fast zero-fill capability. The sequence shown below performs such a zero fill. It zeroes all of the DRAM controlled by this MCECC pair at the rate of 100MB/second when the BCLK pin is operating at 25MHz. This sequence may have to be altered to perform the scrub more slowly if the scrub causes the DRAM to consume too much power at full speed.

1. Make sure that the scrubber is disabled by clearing the SCRBNEN bit in the Scrub Control register. (Clear bit 27 of offset \$24.)
2. Make sure that the scrubber is done with any old scrub cycles by waiting for the SCRBN bit in the Scrub Control register to be cleared. (Wait for bit 28 of offset \$24 = 0.)
3. Discontinue all accesses from the MC680x0 bus to the DRAM.
4. Ensure that all accesses have stopped by clearing the RAMEN bit in the DRAM Control register. (Clear bit 0 of offset \$18)
5. Set the ZFILL bit in the MCECC pair. (Set Bit 28 of offset \$20)
6. Set the Scrub Time On/Time Off register for the maximum rate and to do write cycles, by setting the SRDIS bit, setting all of the STON bits, and clearing all of the STOFF bits. (Write \$B8 to offset \$34)
7. Enable scrubbing by setting the SCRBNEN bit in the Scrub Control register. (Set bit 27 of offset \$24.)
8. Ensure that the zero-fill has started by waiting for the SCRBN bit in the Scrub Control register to be set. (Wait for bit 28 of offset \$24 = 1.)

9. Ensure that the zero-fill stops after one pass by clearing the SCRBN bit in the Scrub Control register. (Clear bit 27 of offset \$24.)
10. Wait for the zero-fill to complete by waiting for the SCRBN bit in the Scrub Control register to be cleared. (Wait for bit 28 of offset \$24 = 0.)
11. Clear the ZFILL bit in the MCECC pair. (Clear Bit 28 of offset \$20.)
12. The entire DRAM that is controlled by this MCECC is now zero-filled. The software can now program the appropriate scrubbing mode and other desired initialization, and enable DRAM for operation.

## Syndrome Decoding

The following table defines the syndrome bit encoding for the Petra/MCECC ASIC. A syndrome code value of \$00 indicates no error found. All other syndrome code values denote an error. The bit in error is decoded as shown in the table.

**Table 5-4. Syndrome Bit Encoding**

Bit in Error	Syndrome Code	Bit in Error	Syndrome Code
Bit 0	\$4F	Bit 16	\$0E
Bit 1	\$4A	Bit 17	\$0B
Bit 2	\$52	Bit 18	\$13
Bit 3	\$54	Bit 19	\$15
Bit 4	\$57	Bit 20	\$16
Bit 5	\$58	Bit 21	\$19
Bit 6	\$5B	Bit 22	\$1A
Bit 7	\$5D	Bit 23	\$1C
Bit 8	\$23	Bit 24	\$62
Bit 9	\$25	Bit 25	\$64
Bit 10	\$26	Bit 26	\$67
Bit 11	\$29	Bit 27	\$68
Bit 12	\$2A	Bit 28	\$6B
Bit 13	\$2C	Bit 29	\$6D
Bit 14	\$31	Bit 30	\$70
Bit 15	\$34	Bit 31	\$77
Check Bit 0	\$01	Check Bit 4	\$10
Check Bit 1	\$02	Check Bit 5	\$20
Check Bit 2	\$04	Check Bit 6	\$40
Check Bit 3	\$08		



Since the memory architecture is 32 data bits plus seven syndrome bits with a non-interleaved architecture, there is no corresponding entry for Bank in Error. The selection of the physical SDRAM bank is decoded from the address bus. Consequently, the Error Address register must be examined to determine which bank contains the error. Given a specific SDRAM configuration, the following table relates bits in the Error Address register to the physical bank where the error originated.

**Table 5-5. Identifying SDRAM Bank in Error**

SDCFG2	SDCFG1	SDCFG0	DRAM Array Size and Bank with the Error
0	0	0	Device is 64Mbit x 16 data with one bank composed of 3 devices.
0	0	1	Device is 64Mbit x 8 data with one bank composed of 5 devices.
0	1	0	Device is 64Mbit x 8 data with two banks composed of 5 devices each. If EA24 = 0, Bank 0 If EA24 = 1, Bank 1
0	1	1	Device is 64Mbit x 8 data with four banks composed of 5 devices each. If EA[25:24] = 00, Bank 0 If EA[25:24] = 01, Bank 1 If EA[25:24] = 10, Bank 2 If EA[25:24] = 11, Bank 3
1	0	0	Device is 128Mbit x 8 data with one bank composed of 5 devices.
1	0	1	Device is 128Mbit x 8 data with two banks composed of 5 devices each. If EA25 = 0, Bank 0 If EA25 = 1, Bank 1



# Summary of Changes



## Introduction

This appendix summarizes the modifications that accompanied the introduction of the Petra ASIC on the MVME162P4 and MVME172P4 Embedded Controller boards. Differences in function and implementation between previous MVME162FX and MVME172FX models and the new MVME1X2P4 boards are listed in the following table.

**Table A-1. List of Changes**

Function	Previous Implementation	MVME1X2P4 Implementation
MC2 memory control	MC2 ASIC, revision 01.	Petra ASIC, revision 03 (page 3-12).
Serial I/O	MC2 ASIC did not support direct access to data port. Software used indirect access.	Both direct and indirect access modes supported (page 3-7).
Flash memory write enable	Flash writes enabled/disabled via control bit in register.	Flash writes enabled/disabled via board configuration jumper (page 3-39).
MC2 memory model	No parity protection for MC2 memory due to availability of DRAM devices.	MC2 memory has ECC capability if parity is enabled. Multibit errors are announced as parity errors. Scrubbing is not enabled. Debugger firmware should enable parity for optimum functionality (page 3-6).
IP interface control	IP2 ASIC, revision 00.	Petra ASIC, revision 03 (page 4-18).
IP reset	IP Reset Control bit in local bus memory map at location \$FFFBC01F. Local bus resets, power-up resets, and software writes to this location all influence IP reset status.	IP Reset bit is under control of software configuration switch (page 4-30).
IP DMA flush	DMA bug in IP2 ASIC: no flush mechanism.	New control register bit supplies flush mechanism. IP DMA FIFOs will flush if a DMA data stream ends before the byte count reaches zero (page 4-47).

**Table A-1. List of Changes (Continued)**

<b>Function</b>	<b>Previous Implementation</b>	<b>MVME1X2P4 Implementation</b>
IP DMA chaining	IP chained DMA functional at 8MHz, but not at 32MHz.	Chained DMA now functional with 32MHz IP bus frequency as well as 8MHz (page 4-2).
IP DMA snoop	DMA Snoop control bug in IP2 ASIC: occasional loss of snoop control signals SC[1:0].	Two reliable snoop control bits for each of the four DMA channels (page 4-31).
IP cache coherency	Cache coherency bug with IP memory space: IP2 interface did not ignore lower address bits (0,1) during burst cycles.	Cache coherency with IP memory space now functional (page 4-4).
MCECC memory control	MCECC ASIC, revision 00.	Petra ASIC, revision 01 (page 5-13).
Check bits	Two banks of eight bits each.	Single bank of seven bits (page 5-3).
Register shadowing	Registers shadowed by a second set of registers.	No register shadowing (page 5-5).
Performance	Standard MCECC performance.	Faster than previous MCECC (page 5-3).
Ethernet	N82C501AD device.	LXT901 device (software-transparent).
DRAM	No parity, fast page.	SDRAM with ECC protection (page 1-2).
User configuration	Jumper headers.	Jumper headers and hardware/software switches.
Flash memory device	Combination of 28F008SA and 28F016SA-100 devices. 12V programming voltage. Manufacturing ID: 89; device ID: 0A.	28F008SA in MVME162P4 versions; 28F160S5 in MVME172P4 versions. 5V programming voltage. Manufacturing ID: B0; device ID: D0 (page 1-6).
LEDs	Through-hole, wave soldered.	Surface mount, with light pipes.
SRAM battery	Through-hole, dual battery.	Surface mount holder.

# Using Interrupts on the MVME1X2P4

## B

### Example: Tick Timer 1 Periodic Interrupt

This appendix describes how to use interrupts on the MVME1X2P4. It gives an example of how to generate and handle a VMEchip2 Tick Timer 1 interrupt on an MVME1X2P4 that has a VMEbus connection. Specific values have been given for the register writes.

Read this entire appendix before performing any of these procedures.

#### 1. Set up Tick Timer:

Step	Register and Address	Action and Reference
1	Prescaler Control register \$FFF4004C	If not already initialized by the debugger, initialize as follows: Prescaler register = $256 - \mathbf{Bclock}$ (MHz). This gives a 1 MHz clock to the tick timers. <b>Bclock</b> is the bus clock rate, such as 25MHz. $256 - 25 = \$E7$ .
2	Tick Timer 1 Compare register \$FFF40050	For periodic interrupts, set the Compare Register value = <b>Period</b> (s). For example, if you want an interrupt every millisecond, set the register value to 1000 (\$3E8). Refer to the <i>Tick Timer 1 Compare Register</i> description in Chapter 2.
3	Tick Timer 1 Counter register \$FFF40054	Write a zero to clear the register.
4	Tick Timer 1 Control register \$FFF40060 (8 bits)	Write \$07 to this register (set bits 0, 1, and 2). This enables the Tick Timer 1 counter to increment, resets the count to zero on compare, and clears the overflow counter.

**2. Set up local bus interrupter:**

Step	Register and Address	Action and Reference
5	Vector Base register \$FFF40088 (8 of 32 bits)	If not already initialized by the debugger, set Interrupt Base register 0 by writing to bits 28-31. Refer to the <i>Vector Base Register</i> description and to Table 2-4. <i>Local Bus Interrupter Summary</i> , in Chapter 2.
6	Interrupt Level register 1 (bits 0-7) \$FFF40078 (8 of 32 bits)	Write desired level of Tick Timer 1 interrupt to bits 0-2.
7	Local Bus Interrupter Enable register \$FFF4006C (8 of 32 bits)	Set bit 24 (ETIC1) to 1 to enable Tick Timer 1 interrupts.
8	I/O Control Register 1 \$FFF40088 (8 of 32 bits)	Write a 1 to bit 23 to enable interrupts from the VMEchip2. A 0 masks <i>all</i> interrupts from the VMEchip2.

Periodic Tick Timer 1 interrupts now occur, so you need an interrupt handler. Section 3 gives the details, as follows.

**3. Set up an interrupt handler routine:**

Step	Action and Reference
Your interrupt handler should include the following features.	
1	Be sure the MC680x0 Vector Base register is set up. Set the proper MC680x0 exception vector location so the processor vectors to your interrupt handler location. You can determine the proper exception vector location to set from the MC680x0 Vector Base register, the VMEchip2 Base register, and Table 2-4, <i>Local Bus Interrupter Summary</i> in Chapter 2, from which you can determine the actual interrupt vector given on a Tick Timer 1 interrupt. Lower the MC680x0 mask so the interrupt level you programmed is accepted. The <i>interrupt handler itself</i> should include the following (steps 2 through 5).

Step	Action and Reference
2	Confirm that the Tick Timer 1 interrupt occurred, by reading the status of bit 24 in the Interrupter Status register at \$FFF40068. A high indicates an interrupt present.
3	Clear the Tick Timer 1 interrupt by writing a 1 to bit 24 of the Interrupt Clear register at \$FFF40074.
4	Increment a software counter to keep track of the number of interrupts, if desired. Output a character or some other action (such as toggling the <b>FAIL LED</b> ) on an appropriate count, such as 1000.
5	Return from exception.





## Motorola Computer Group Documents

The Motorola publications listed below are applicable to the MVME1X2P4. You can obtain paper or electronic copies of Motorola Computer Group publications by:

- ❑ Contacting your local Motorola sales office
- ❑ Visiting MCG's World Wide Web literature site, <http://www.motorola.com/computer/literature>

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

Document Title	Publication Number
MVME162P4 VME Embedded Controller Installation and Use	V162PFXA/IH
MVME172P4 VME Embedded Controller Installation and Use	V172PFXA/IH
MVME162Bug Diagnostics User's Manual	V162DIAA/UM
MVME172Bug Diagnostics User's Manual	V172DIAA/UM
Debugging Package for Motorola 68K CISC CPUs User's Manual (Parts 1 and 2)	68KBUG1/D and 68KBUG2/D
Single Board Computers SCSI Software User's Manual	SBCSCSI/D
MVME712M Transition Module and P2 Adapter Board Installation and Use	VME712MA/IH
MVME712-12, MVME712-13, MVME712A, MVME712AM, and MVME712B Transition Modules and LCP2 Adapter Board User's Manual	MVME712A/D
SIMM09 Serial Interface Module Installation Guide	SIMM09A/IH

To locate and view the most up-to-date product information in PDF or HTML format, visit <http://www.motorola.com/computer/literature>.

## Manufacturers' Documents

For additional information, refer to the following table for manufacturers' data sheets or user's manuals and related specifications. As an additional help, a source for the listed document is also provided. Please note that in many cases, the information is preliminary and the revision levels of the documents are subject to change without notice.

**Table C-2. Manufacturers' Documents**

Document Title and Source	Publication Number
<p><i>M68040 Microprocessors User's Manual</i>  <i>M68060 Microprocessors User's Manual</i>  <i>M68000 Family Reference Manual</i></p> <p>Literature Distribution Center for Motorola            Telephone: 1-800- 441-2447            FAX: (602) 994-6430 or (303) 675-2150            E-mail: ldcformotorola@hibbertco.com            Web: <a href="http://www.mot.com/SPS">http://www.mot.com/SPS</a></p>	<p>M68040UM            M68060UM            M68000FR</p>
<p><i>82596CA Local Area Network Coprocessor Data Sheet</i>  <i>82596CA Local Area Network Coprocessor User's Manual</i>  <i>28F008SA Flash Memory Data Sheet</i>  <i>28F016SA Flash Memory Data Sheet</i></p> <p>Intel Corporation            Web: <a href="http://developer.intel.com/design">http://developer.intel.com/design</a></p>	<p>290218            296853            209435            209435</p>
<p><i>M48T58(B) Timekeeper™ and 8Kx8 Zeropower™ RAM data sheet</i></p> <p>SGS-Thomson Microelectronics Group            North &amp; South American Marketing Headquarters            1000 East Bell Road            Phoenix, Arizona 85022            Telephone: (602) 867-6100            Web: <a href="http://www.st.com/stonline/books">http://www.st.com/stonline/books</a></p>	<p>M48T58</p>
<p><i>SYM 53C710 (was NCR 53C710) SCSI I/O Processor Data Manual</i>  <i>SYM 53C710 (was NCR 53C710) SCSI I/O Processor Programmer's Guide</i></p> <p>Symbios Logic Inc.            1731 Technology Drive, Suite 600            San Jose, CA 95110            NCR Managed Services Center — Telephone: 1-800-262-7782            Web: <a href="http://www.lsilogic.com/products/symbios">http://www.lsilogic.com/products/symbios</a></p>	<p>NCR53C710DM            NCR53C710PG</p>

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

Document Title and Source	Publication Number
<i>Z85230 Serial Communications Controller Product Brief</i> Zilog Inc. 210 Hacienda Avenue Campbell, CA 95008-6609 Web: <a href="http://www.zilog.com/products">http://www.zilog.com/products</a>	Z85230pb.pdf

C

## Related Specifications

For additional information, refer to the following table for related specifications. As a further aid, sources for the listed documents are also provided. Please note that in many cases, the information is preliminary and the revision levels of the documents are subject to change without notice.

**Table C-3. Related Specifications**

Document Title and Source	Publication Number
<i>Interface Between Data Terminal Equipment and Data Circuit-Terminating Equipment Employing Serial Binary Data Interchange (EIA-232-D)</i> Global Engineering Documents Suite 400 1991 M Street, NW Washington, DC 20036 Telephone: 1-800-854-7179 Telephone: (303) 397-7956 Web: <a href="http://global.ihs.com">http://global.ihs.com</a>	ANSI/EIA-232-D Standard
<i>VME64 Specification</i> <i>IndustryPack Logic Interface Specification, Revision 1.0</i> VITA (VMEbus International Trade Association) 7825 E. Gelding Drive, Suite 104 Scottsdale, AZ 85260 Telephone: (602) 951-8866 Web: <a href="http://www.vita.com">http://www.vita.com</a>	ANSI/VITA 1-1994 ANSI/VITA 4-1995

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

Document Title and Source	Publication Number
<p><i>Versatile Backplane Bus: VMEbus</i></p> <p>The Institute of Electrical and Electronics Engineers, Inc., 345 East 47th Street, New York, NY 10017 (VMEbus Specification)</p> <p>(This is also <i>Microprocessor System Bus for 1 to 4 Byte Data</i>, IEC 821 BUS: Bureau Central de la Commission Electrotechnique Internationale; 3, rue de Varembe, Geneva, Switzerland)</p>	<p>ANSI/IEEE Std 1014-1987</p>
<p><i>ANSI Small Computer System Interface-2 (SCSI-2)</i></p> <p>Global Engineering Documents 15 Inverness Way East Englewood, CO 80112-5704</p>	<p>Draft Document X3.131-198X, Revision 10c</p>

C

## Numerics

53C710 SCSI controller interface [3-5](#)

82596CA LAN interface [3-3](#)

## A

A16/D16 space, VMEbus [2-37](#)

A16/D16 space, VMEchip2 ASIC [2-6](#)

A16/D32 space, VMEbus [2-37](#)

A16/D32 space, VMEchip2 ASIC [2-6](#)

A24/D16 space, VMEbus [2-37](#), [2-51](#)

A24/D16 space, VMEchip2 ASIC [2-6](#)

A32/D16 space [2-37](#)

A32/D16 space, VMEbus [2-51](#)

A32/D16 space, VMEchip2 ASIC [2-6](#)

ABORT switch interrupt, address [1-9](#)

AC fail interrupter, VMEbus [2-19](#)

access cycles, VMEbus [2-34](#)

access timer, VMEbus [2-7](#)

ACFAIL\* signal line [2-19](#), [2-96](#)

adder, in address translation [2-31](#), [2-32](#), [2-35](#)

adders, VMEchip2 [2-27](#)

address

    GCSR, VMEchip2 [1-40](#), [2-100](#)

    LCSR, VMEchip2 [2-20](#)

    VMEbus resources [2-37](#)

address counter, VMEbus [2-13](#)

address decoders [1-13](#)

address modifier codes [2-43](#), [2-44](#)

address modifier codes (DMAC) [2-58](#)

address modifier register (VMEbus slave)  
[2-38](#)

address modifier select bits [2-33](#), [2-36](#)

address range

    devices [1-11](#)

    local bus [2-39](#)

address space, decoding [1-13](#)

address translation address register [2-38](#)

address translation address register, slave  
    map decoder [2-27](#)

address translation registers [2-38](#)

address translation registers, slave map  
    decoder [2-27](#)

address translation select register [2-38](#)

address translation select register, slave map  
    decoder [2-27](#)

address, Ethernet [1-37](#)

addressed DMA (aDMA) [4-4](#)

addressing capabilities

    local bus to IndustryPack [4-49](#)

    local-bus-to-VMEbus interface [2-4](#)

    VMEbus-to-local-bus interface [2-9](#)

    VMEchip2 DMAC [2-11](#)

aDMA (addressed DMA) [4-4](#)

alternate address register, VMEchip2 ASIC  
[2-10](#)

arbiter

    time-out timer, VMEbus [2-64](#)

    VMEbus [2-17](#)

arbitration (MCECC sector) [5-9](#)

arbitration modes, VMEbus [2-17](#)

ASICS replaced by Petra [1-2](#)

attribute register, VMEchip2 [2-28](#)  
    snoop bits [2-28](#)

**B**

backward compatibility 1-2, 1-38, 3-6, 3-25  
 base address  
   VMEchip2 LCSR 2-20  
 BBRAM  
   configuration area memory map 1-34  
   Flash write enable 3-3  
   memory map 1-33  
 BBSY\* signal, VMEbus 2-98  
 BERR\* signal, VMEbus 2-17  
 BGIN filters, VMEbus 2-98  
 binary number, symbol for *xxiv*  
 block (D64) access cycles, VMEbus 2-33, 2-36  
 block access cycles, VMEbus 2-33, 2-36  
 block diagram, MVME1x2P4 1-7  
 block diagram, VMEchip2 ASIC 2-5  
 block transfer  
   cycles, VMEchip2 DMAC 2-11  
   mode 2-9  
   modes, DMAC 2-59  
 board  
   address, GCSR 2-48  
   documentation C-1  
   failure signal, VMEchip2 ASIC 2-70  
   ID field (configuration data structure) 1-37  
   overview 1-2  
   status/control register, VMEchip2 2-106  
 Board Control register, VMEchip2 2-101  
 BRDFAIL\* signal pin, VMEchip2 ASIC 2-70, 2-71  
 broadcast interrupt function (VMEchip2 timers) 2-15  
 broadcast mode, VMEbus 2-16  
 BSY signal and arbitration timer 2-17  
 bug fixes, IP2 sector 4-2  
 bus  
   error handler 1-43  
   error processing 1-43  
   error signals 1-41  
   map decoder, LCSR 2-20

bus *continued*  
   sizing 2-6  
   timer enable/disable, VMEbus 2-17  
   timers, use of 1-50  
 byte counter, DMAC 2-60

**C**

cache coherency 1-41  
   MCECC sector 5-4  
   Petra/IP2 chip 4-4  
 cache inhibit function 1-11  
 CAS instruction 1-51  
 cautions for use of reset (VMEchip2) 2-101  
 changes from previous boards A-1  
 checksum field, board configuration structure 1-39  
 chip arbiter, VMEbus 2-17  
 chip defaults (MCECC sector) 5-9  
 chip ID and revision registers (VMEchip2 ASIC) 2-100  
 clear-on-compare mode, VMEchip2 counters 2-15  
 clock domains (IP2 sector) 4-6  
 clock programming, IP2 sector 4-44  
 clocks for VMEchip2 counters and timers 2-67  
 command chaining mode, VMEchip2 DMAC 2-12, 2-52  
 command packets, DMAC 2-52  
 compatibility, backward 1-2, 1-38, 3-6, 3-25  
 configuration bytes, data structure of 1-36  
 CSR decoding in Petra chip 3-25  
 cycle types (memory access) 5-5

**D**

data access cycles, VMEbus 2-33, 2-36  
 data transfer capabilities  
   local-bus-to-VMEbus interface 2-4  
   VMEbus-to-local-bus interface 2-9  
   VMEchip2 DMAC 2-11  
 data transfer size, VMEchip2 DMAC 2-11  
 data transfers, DMA (VMEchip2 ASIC) 2-52

---

data transfers, VMEbus [2-43](#), [2-44](#)  
decimal number, symbol for [xxiv](#)  
decoders  
    programmable [2-4](#)  
    VMEchip2 [2-26](#)  
devices, normal address range [1-11](#)  
DFAIR bit [2-14](#)  
differences from previous boards [A-1](#)  
direct mode, DMAC [2-51](#)  
DMA  
    control and status registers [4-34](#)  
    enable function [4-34](#)  
    implementation (IP2 sector) [4-4](#)  
    transfers, no-address-increment [2-12](#)  
DMA Controller (DMAC), VMEchip2 ASIC  
    [2-10](#), [2-51](#)  
DMAC  
    command packets [2-52](#)  
    interrupter, VMEbus [2-19](#)  
    LTO error [1-46](#)  
    off-board error [1-46](#)  
    parity error [1-45](#)  
    TEA error, cause unidentified [1-47](#)  
    VMEbus error [1-45](#)  
    VMEbus requester [2-13](#)  
DMAC registers (VMEchip2 ASIC)  
    DMAC byte counter [2-60](#)  
    DMAC Control register 1 (bits 0-7) [2-55](#)  
    DMAC Control register 2 (bits 0-7) [2-58](#)  
    DMAC Control register 2 (bits 8-15)  
        [2-57](#)  
    DMAC local bus address counter [2-59](#)  
    DMAC Status register [2-63](#)  
    DMAC VMEbus address counter [2-60](#)  
    Local-Bus-to-VMEbus Requester  
        Control register [2-54](#)  
    MPU Status and DMA Interrupt Count  
        register [2-62](#)  
    PROM Decoder, SRAM and DMA  
        Control register [2-53](#)  
    table address counter [2-60](#)

DMAC registers (VMEchip2) *continued*  
    VMEbus Interrupter Control register  
        [2-61](#)  
    VMEbus Interrupter Vector register [2-62](#)  
documentation [C-1](#)  
DRAM  
    and SRAM Memory Controller registers  
        [3-24](#)  
    initialization [5-34](#)  
    memory controller  
        see SDRAM memory controller,  
            Petra/MC2 chip [3-6](#)  
    size control bit encoding (Petra/MC2  
        chip) [3-26](#), [3-27](#)  
    specifications [1-6](#)  
DTACK\* signal [2-9](#)  
DWB bit (VMEchip2 LCSR) [2-8](#)

## E

ECC (error-correcting code) [5-5](#)  
edge-sensitive interrupters, VMEbus [2-19](#)  
edge-sensitive interrupts, VMEchip2 ASIC  
    [2-74](#)  
ending address register  
    VMEbus slave [2-38](#)  
    VMEbus slave map decoder [2-27](#)  
EPROM socket [1-6](#)  
EPROM/Flash  
    interface [3-2](#)  
    selection [1-13](#)  
    sizing [1-13](#)  
errata sheets, chip [1-17](#)  
error conditions, descriptions of [1-43](#)  
error handler [1-43](#)  
error logging, ECC [5-8](#)  
error reporting, Petra/IP2 chip [4-9](#)  
    as a local bus master [4-8](#)  
    as a local bus slave [4-8](#)  
error reporting, Petra/MCECC chip [5-5](#)  
error sources, local [1-41](#)  
error types (MCECC sector) [5-5](#)

## errors

- DMAC local bus time-out [1-46](#)
- DMAC off-board access [1-46](#)
- DMAC TEA [1-47](#)
- LAN local bus time-out [1-48](#)
- LAN off-board access [1-48](#)
- MPU local bus time-out [1-44](#)
- MPU off-board access [1-44](#)
- MPU TEA [1-44](#)
- SCSI local bus time-out [1-50](#)
- SCSI off-board access [1-49](#)
- VMEbus DMAC [1-45](#)

Ethernet address [1-37](#)

Ethernet LAN memory map [1-31](#)

## examples

- generating tick timer periodic interrupt [B-1](#)
- setting up interrupt handler routine [B-2](#)
- setting up local bus interrupter [B-2](#)

extended access cycles, VMEbus [2-34](#), [2-37](#)

**F**

fair mode, VMEchip2 [2-8](#), [2-14](#)

## features

- IP2 function [4-1](#)
- MC2 function [3-1](#)
- MCECC sector [5-1](#)
- MVME1x2P4 [1-6](#)
- VMEchip2 ASIC [2-1](#)

Flash memory [1-6](#)

- write enable via BBRAM [3-3](#)
- write protection and MC1/MC2 mode [1-16](#), [3-39](#)

Flash/EPROM

- interface [3-2](#)
- selection [1-13](#)
- sizing [1-13](#)

functional description [1-9](#)

- MCECC sector [5-3](#)
- Petra MC2 sector [3-2](#)
- VMEchip2 ASIC [2-4](#)

**G**

GCSR (global control/status registers)

- programming [2-102](#)
- VMEchip2 [2-20](#), [2-100](#)
- base address registers, programming [2-37](#)
- board address [2-48](#)
- group address [2-47](#)
- map decoder [1-40](#)
- programming model [2-100](#)
- SIG3-0 interrupts, VMEbus [2-19](#)

General Purpose Register 1 [2-107](#)

general-purpose I/O pins, direction [2-96](#)

general-purpose registers, VMEchip2 [2-101](#)

global control/status registers

- General Purpose Register 0 [2-107](#)
- General Purpose Register 2 [2-107](#)
- General Purpose Register 3 [2-108](#)
- General Purpose Register 4 [2-108](#)
- General Purpose Register 5 [2-108](#)
- ID register, VMEchip2 [2-104](#)
- VMEchip2 Board Status/Control register [2-106](#)

VMEchip2 ID register [2-104](#)

VMEchip2 LM/SIG register [2-104](#)

VMEchip2 Revision register [2-103](#)

global control/status registers (GCSRs) [2-100](#)

global reset driver, VMEbus [2-18](#)

global reset, VMEbus [2-18](#)

GPI inputs, addresses [1-9](#)

GPI3 (general-purpose input 3) bit [1-13](#)

GPI3 bit (EPROM/Flash select) [3-33](#)

group address, GCSR [2-47](#)

**H**

hexadecimal number, symbol for [xxiv](#)

**I**

I/O and ID space accesses, IP [4-58](#)

I/O interfaces [1-6](#)

I/O map decoders [2-6](#), [2-37](#), [2-39](#)



---

I/O memory maps 1-17  
 I/O space  
     32-bit IP\_ab 4-54  
     IP\_a 4-53  
 IACK cycle, VMEbus 2-20  
 IACK daisy-chain driver, VMEbus and 2-17  
 IACK daisy-chain, VMEbus 2-16  
 ID space, IP 4-55  
 indivisible cycles, MC680x0 1-51  
 IndustryPack  
     addressing 4-49  
     error reporting 4-9  
     ID bytes 1-38  
     Interface Controller ASIC (Petra/IP2 chip) 4-1  
 initialization, DRAM 5-34  
 interrupt  
     base vectors, VMEbus 2-95  
     control (MCECC) 3-21  
     control register, VMEchip2 2-101  
     counter, DMAC 2-62  
     handler routine, how to set up B-2  
     handler, VMEbus 2-16  
     priority 1-40  
     sources, VMEchip2 2-18  
     status bit 2-77  
     Vector Base register encoding and priority 3-14  
     vectors 1-40  
 Interrupt Acknowledge map 1-39  
 interrupt handler, VMEchip2 ASIC 2-18  
 interrupter acknowledge interrupter,  
     VMEbus 2-19  
 interrupter control, VMEbus 2-61  
 interrupts  
     broadcast 2-15, 2-16  
     edge-sensitive (VMEchip2 ASIC) 2-74  
     hardware-vectored 1-40  
     how to use B-1  
     masked 2-96  
     Petra/IP2 chip 4-9  
     IP to local bus data routing 4-56  
     IP2 functions  
         overall memory map 1-24  
     IP2 sector  
         Alternate DMA Byte Count register 4-47  
         Chip ID register 4-18  
         Chip Revision register 4-18  
         DMA Arbitration Control register 4-29  
         DMA Byte counter 4-43  
         DMA Control Register 1 4-38  
         DMA Control Register 2 4-40  
         DMA Enable register 4-37  
         DMA IndustryPack Address counter 4-42  
         DMA Interrupt Control register 4-36  
         DMA Local Bus Address counter 4-42  
         DMA Status register 4-35  
         DMA Table Address counter 4-43  
         features 4-1  
         functional description 4-3  
         General Control registers 4-24  
         IP Clock register 4-28  
         IP RESET register 4-30  
         IP to local bus data routing 4-56  
         IRQ0, IRQ1 Interrupt Control registers 4-23  
         Memory Base Address registers 4-20  
         Memory Size registers 4-22  
         overall memory map 4-10  
         Programmable Clock General Control register 4-45  
         Programmable Clock Interrupt Control register 4-44  
         Programmable Clock Timer register 4-46  
         programming model 4-11  
         snoop control 4-4, 4-31  
         Vector Base register 4-19  
     IRQ1 interrupter, VMEbus 2-19  
     IRQ7-1 interrupters, VMEbus 2-19

**L**

## LAN

- interface [3-3](#)
- LTO error [1-48](#)
- off-board error [1-48](#)
- parity error [1-47](#)

LANC bus error processing [3-4](#)LANC interrupt processing [3-5](#)

## LCSR (Local Control/Status Registers),

VMEchip2 [2-20](#)

- base address [2-20](#)
- memory map [2-22](#)
- programming model [2-20](#)

LEDs [2-99](#)LM/SIG register, VMEchip2 [2-104](#)

## local

- control and status registers (LCSRs),  
VMEbus [2-7](#)
- DRAM parity error [1-42](#)
- I/O devices memory map [1-14](#)
- reset driver, VMEbus [2-18](#)
- reset, VMEbus [2-18](#)

## local bus

- accesses from VMEbus [1-40](#)
- address counter, DMAC [2-59](#)
- address range [2-39](#)
- base address, GCSR [2-100](#)
- error signals [1-41](#)
- interrupt filters, VMEchip2 ASIC [2-98](#)
- interrupter summary [2-75](#)
- interrupter, how to set up [B-2](#)
- map decoder registers [2-38](#)
- memory map [1-11](#), [1-12](#)
- reset [2-106](#)
- time-out [1-41](#)
- time-out value [2-66](#)

## local bus interrupter

- DMAC and [2-12](#)
- programming [2-74](#)
- VMEchip2 [2-18](#)

## local bus interrupter registers

- I/O Control register 1 [2-96](#)
- I/O Control register 2 [2-97](#)
- I/O Control register 3 [2-97](#)
- Interrupt Level register 4 (bits 0-7) [2-95](#)
- Miscellaneous Control register [2-98](#)
- Status register (bits 16-23) [2-78](#)
- Status register (bits 24-31) [2-77](#)
- Vector Base register [2-95](#)

local bus master [2-9](#)VMEbus and [2-10](#)

## local bus slave (VMEbus master) registers

- Address Translation Address Register 4  
[2-42](#)
- Address Translation Select Register 4  
[2-43](#)
- Attribute Register 1 [2-46](#)
- Attribute Register 2 [2-45](#)
- Attribute Register 3 [2-44](#)
- Attribute Register 4 [2-43](#)
- Ending Address Register 1 [2-40](#)
- Ending Address Register 2 [2-40](#)
- Ending Address Register 3 [2-41](#)
- Ending Address Register 4 [2-42](#)
- Starting Address Register 1 [2-40](#)
- Starting Address Register 2 [2-41](#)
- Starting Address Register 3 [2-41](#)
- Starting Address Register 4 [2-42](#)

local bus slave, composition of [2-4](#)

## local bus timer

- Petra/MC2 ASIC [3-9](#)
- VMEchip2 ASIC [2-18](#)

local-bus-to-IndustryPack addressing [4-49](#)

## local-bus-to-VMEbus

- Enable Control register [2-49](#)
- I/O Control register [2-50](#)
- interface [1-9](#)
- interface, VMEchip2 [2-4](#)
- map decoders, programming [2-37](#)
- requester [2-7](#)
- requester register, programming [2-51](#)

- 
- location monitors
    - interrupters, VMEbus [2-19](#)
    - LM0-LM3 (VMEchip2 ASIC) [2-100](#)
    - status register (VMEchip2 ASIC) [2-101](#)
  - LVFAIR bit (VMEchip2 ASIC) [2-8](#)
  - M**
  - M48T58 memory map [1-33](#)
  - manufacturers' documents [C-2](#)
  - map decoders [2-4](#), [2-37](#)
    - GCSR [1-40](#)
    - VMEbus interface [1-40](#)
    - VMEchip2 ASIC [2-6](#), [2-9](#)
  - master interrupt enable (MIEN) bit [2-74](#), [2-96](#), [3-12](#)
  - MC1/MC2 mode and Flash write protection [1-16](#), [3-39](#)
  - MC2 chip ASIC
    - see Petra ASIC (MC2 functions) [1-5](#)
  - MC2 functions [3-1](#)
    - features [3-1](#)
    - initialization [3-2](#)
    - memory map [1-23](#)
  - MC2 sector
    - ABORT Switch Interrupt Control register [3-40](#)
    - Access and Watchdog Time Base Select register [3-43](#)
    - Bus Clock register [3-37](#)
    - DRAM Control register [3-44](#)
    - DRAM Parity Error Interrupt Control register [3-21](#)
    - DRAM Space Base Address register [3-24](#)
    - DRAM Space Size register [3-25](#)
    - DRAM/SRAM Options register [3-27](#)
    - Flash Access Time Control register [3-39](#)
    - functional description [3-2](#)
    - General Control register [3-12](#)
    - General-Purpose Inputs register [3-33](#)
    - ID register [3-12](#)
    - Interrupt Vector Base register [3-13](#)
    - LANC Bus Error Interrupt Control register [3-31](#)
    - LANC Error Status register [3-29](#)
    - LANC Interrupt Control register [3-30](#)
    - LSB Prescaler Count register [3-17](#)
    - MPU Status register [3-46](#)
    - Prescaler Clock Adjust register [3-17](#)
    - Prescaler Count register [3-47](#)
    - registers [3-10](#)
    - RESET Switch Control register [3-41](#)
    - Revision register [3-12](#)
    - SCC Interrupt Control register [3-22](#)
    - SCSI Error Status register [3-32](#)
    - SDRAM Control register [3-44](#)
    - SRAM Space Base Address register [3-25](#)
    - SRAM Space Size register [3-28](#)
    - Tick Timer 1 and 2 Compare and Counter registers [3-15](#)
    - Tick Timer 1 and 2 Control registers [3-18](#)
    - Tick Timer 3 and 4 Compare and Counter registers [3-36](#)
    - Tick Timer Interrupt Control registers [3-19](#)
    - Watchdog Timer Control register [3-42](#)
  - MC680x0 processor
    - indivisible cycles [1-51](#)
    - indivisible RMW memory accesses [1-51](#)
  - MCECC chip Memory Controller ASIC
    - see Petra ASIC (MCECC functions) [1-5](#)
  - MCECC functions [5-1](#)
    - memory model (additional functionality) [3-25](#)
  - MCECC sector
    - arbitration [5-9](#)
    - Base Address register [5-15](#)
    - BCLK Frequency register [5-16](#)
    - Chip ID register [5-13](#)
    - Chip Prescaler counter [5-21](#)
    - Chip Revision register [5-13](#)
    - Data Control register [5-17](#)

MCECC sector *continued*

- Defaults register 1 [5-30](#)
  - Defaults register 2 [5-32](#)
  - DRAM Control register [5-15](#)
  - Error Address register (bits 15-8) [5-29](#)
  - Error Address register (bits 23-16) [5-28](#)
  - Error Address register (bits 31-24) [5-28](#)
  - Error Address register (bits 7-4) [5-29](#)
  - Error Logger register [5-27](#)
  - Error Syndrome register [5-30](#)
  - features [5-1](#)
  - functional description [5-3](#)
  - internal register memory map [5-11](#)
  - Memory Configuration register [5-14](#)
  - Scrub Address counter (bits 15-8) [5-26](#)
  - Scrub Address counter (bits 23-16) [5-26](#)
  - Scrub Address counter (bits 26-24) [5-25](#)
  - Scrub Address counter (bits 7-4) [5-26](#)
  - Scrub Control register [5-19](#)
  - Scrub Period register bits 15-8 [5-20](#)
  - Scrub Period register bits 7-0 [5-20](#)
  - Scrub Prescaler counter (bits 15-8) [5-23](#)
  - Scrub Prescaler counter (bits 21-16) [5-23](#)
  - Scrub Prescaler counter (bits 7-0) [5-24](#)
  - Scrub Time On/Time Off register [5-21](#)
  - Scrub Timer counter (bits 15-8) [5-24](#)
  - Scrub Timer counter (bits 7-0) [5-25](#)
  - SDRAM Configuration register [5-33](#)
  - SDRAM implementation [5-5](#)
  - specifications [5-4](#)
- memory controller models (parity vs. ECC) [1-3](#), [3-6](#)
- memory cycles [1-51](#)
- memory devices used on board [1-6](#)
- memory map
- BBRAM configuration area [1-34](#)
  - BBRAM, TOD clock [1-33](#)
  - Ethernet LAN [1-31](#)
  - interrupt acknowledge [1-39](#)
  - IP2 functions, all devices [1-24](#)
  - local bus [1-11](#), [1-12](#)

memory map *continued*

- local I/O devices [1-14](#)
  - MC2 functions [1-23](#)
  - MCECC sector internal registers [5-11](#)
  - Petra/ MC2 chip registers [3-10](#)
  - Petra/IP2 chip devices [4-10](#)
  - SCSI [1-32](#)
  - time-of-day clock [1-35](#)
  - VMEbus [1-39](#)
  - VMEchip2 GCSR [1-22](#)
  - VMEchip2 GCSR (global control/status registers) [2-103](#)
  - VMEchip2 LCSR [1-18](#)
  - VMEchip2 LCSR summary [2-22](#)
  - Z85230 SCC registers [1-31](#)
- memory mezzanine serial number [1-38](#)
- memory space
- 16-bit IP\_a [4-51](#)
  - 32-bit IP\_ab [4-52](#)
  - 8-bit IP\_a [4-50](#)
- memory space accesses, IP [4-56](#)
- microprocessors used on board [1-6](#)
- MIEN (master interrupt enable) bit [2-74](#), [2-96](#), [3-12](#)
- MPU
- local bus time-out [1-44](#)
  - off-board error [1-44](#)
  - parity error [1-43](#)
  - Status register, programming [2-51](#)
  - TEA error, cause unidentified [1-44](#)
- MVME1x2P4
- features [1-6](#)
  - functional description [1-9](#)
  - microprocessors [1-6](#)
  - Version register [3-34](#)
- MVME712x transition boards [1-3](#)

**N**

- no-address-increment DMA transfers [2-12](#)
- non-privileged access cycles, VMEbus [2-34](#), [2-37](#)

---

Non-Volatile RAM (NVRAM) 1-6

memory map 1-33

see BBRAM 1-33

## O

overflow counter, VMEchip2 ASIC 2-72,  
2-73

overview, MVME1x2P4 1-2

## P

parity errors 1-42, 1-43

performance, MCECC sector 5-3

Petra ASIC

functionality of 1-2

IP2 functions 4-1

MC2 functions 3-1

MCECC functions 5-1

periodic interrupt example B-1

Petra chip/VMEchip2 redundancies 1-9

power monitor function, VMEbus 2-17

powerup reset, VMEchip2 ASIC 2-70

prescaler, VMEchip2 2-14

Priority (PRI) arbitration mode, VMEbus  
2-17

priority of interrupts 1-40

processor bus master support for 82596CA  
3-4

processor-to-VMEbus transfers 1-5

program access cycles, VMEbus 2-33, 2-36

program address modifier code (VMEbus)  
2-50

programmable map decoders 2-37

programming

DMA controller, VMEchip2 ASIC 2-51

DMA controllers (IP2 sector) 4-31

GCSR (global control/status registers),  
VMEchip2 2-102

GCSR base address registers 2-37

IP2 sector programmable clock 4-44

LCSR, VMEchip2 2-20

local bus interrupter 2-74

programming *continued*

local-bus-to-VMEbus map decoders,  
VMEchip2 2-37

local-bus-to-VMEbus requester register  
2-51

MPU Status register 2-51

tick and watchdog timers, VMEchip2  
2-64

tick timers (Petra/MC2 chip) 3-15

VMEbus interrupter 2-51

VMEbus slave map decoders 2-26

programming model

MCECC control/status registers 5-10

Petra/IP2 chip 4-11

Petra/MC2 chip 3-11

VMEchip2 GCSR 2-100

VMEchip2 LCSR 2-20

PROM Access Time Control register 3-38

PROM/EPROM sockets 1-6

PROM/Flash interface 3-2

PWB number field 1-38

## R

refresh function (MCECC sector) 5-8

register definitions, LCSR 2-21

registers

local bus map decoders 2-38

VMEbus slave map decoder 2-26

related publications C-1

related specifications C-3

release-on-acknowledge (ROAK) mode,  
VMEbus 2-16

release-on-request (ROR) mode, VMEchip2  
ASIC 2-8

reset drivers, VMEbus 2-18

reset status, Petra/ IP2 chip 4-11

RESET switch, enabling/disabling 2-70

reset, Petra/MC2 chip 3-33

revision register

VMEchip2 2-103

ROM Control register, VMEchip2 ASIC  
2-51

Round Robin Select (RRS) arbitration mode,  
VMEbus 2-17

## S

SCC (serial communications controller)

interface 3-7

register addresses 1-31

scrubbing function (MCECC sector) 5-8

SCSI

controller interface 3-5

ID 1-38

Interrupt Control register 3-35

LTO error 1-50

memory map 1-32

off-board error 1-49

parity error 1-49

sDMA (standard DMA) 4-4

SDRAM implementation 1-3, 3-6, 5-1, 5-5

SDRAM memory controller, Petra/MC2 chip  
3-6

segment size (VMEbus addressing) 2-31

segment size, translating 2-30

SERCLK driver 2-17

serial communications interface 3-7

short I/O area, VMEchip2 ASIC 2-6

short I/O map decoder (VMEbus) 2-50

short I/O memory map 1-40

short I/O segment, VMEbus 2-50

short I/O space, VMEbus 2-37, 2-100

signal interrupts SIG0-SIG3 (VMEchip2  
ASIC) 2-100

Single (SGL) arbitration mode, VMEbus  
2-17

size, VMEbus segment 2-30, 2-31

slave map decoders, VMEbus 2-26

snoop bits, enabling 2-28

snoop control 4-4, 4-31

bits 2-53

register 2-32

snoop function enabling 2-32, 2-35

snoop signal lines (DMAC) 2-58

snooping, definition of 1-41, 2-10

software 7-0 interrupts, VMEbus 2-19

software interrupts 1-6

software support considerations 1-40

specification documents C-3

specifications

applicable 1-7

MCECC sector 5-4

speed, board 1-37

SRAM

memory controller 3-5

size control bit encoding (Petra/MC2  
chip) 3-28, 3-29

specifications 1-6

standard access cycles, VMEbus 2-34, 2-36

standard DMA (sDMA) 4-4

starting address register (VMEbus slave)  
2-38

starting address register, slave map decoder  
2-27

status LEDs 1-7

status register, DMAC 2-63

status register, MPU (DMAC) 2-62

supervisor address modifier code

VMEbus 2-50

VMEchip2 ASIC 2-50

supervisory access cycles, VMEbus 2-34,  
2-37

switches (ABORT and RESET) 1-7

syndrome decoding 5-36

SYSFAIL\* interrupter 2-19

SYSFAIL\* signal line 2-19, 2-70, 2-96

SYSRESET function, VMEchip2 ASIC 2-15

SYSRESET\* signal 2-18, 2-71

system controller function, VMEchip2 ASIC  
2-70

enable/disable 2-17

## T

TAS cycles 1-51

TEA\* signal (bus error) 1-41

Tick Timer 3 and 4 Control registers 3-23

- tick timers 1-6
  - interrupters 2-19
  - periodic interrupt example xxiii, B-1
  - Petra/MC2 ASIC 3-8
  - VMEchip2 2-14
- time-of-day clock 1-6
  - memory map 1-33, 1-35
- time-out
  - local bus 1-41
  - VMEbus access 1-42
- time-out period, VMEbus 2-17
- time-out period, watchdog 2-66
- timer registers (VMEchip2 ASIC)
  - Board Control register 2-70
  - DMAC time on/off timers 2-65
  - Prescaler Control register 2-67
  - Prescaler counter 2-73
  - Tick Timer 1 Compare register 2-68
  - Tick Timer 1 Control register 2-73
  - Tick Timer 1 counter 2-68
  - Tick Timer 2 Compare register 2-69
  - Tick Timer 2 Control register 2-72
  - Tick Timer 2 counter 2-69
  - VME Access, Local Bus, and Watchdog
    - Time-Out Control register 2-66
  - VMEbus Arbiter Time-Out Control
    - register 2-64
  - VMEbus global time-out timer 2-65
  - Watchdog Timer Control register 2-71
- timers 1-6
- timers, VMEbus 2-7
- transfer mode, VMEbus 2-12
- Transfer Type (TT) signals 1-11
- transition boards 1-3

## U

- updates from previous boards A-1
- user access cycles, VMEbus 2-34, 2-37
- using bus timers 1-50

## V

- V11 control bit, Petra/MC2 chip 3-33

- vector base registers, VMEchip2 ASIC 2-74
- VME LED 2-99
- VMEbus
  - access time-out 1-42
  - access time-out value (DMAC) 2-66
  - address counter, DMAC 2-60
  - BBSY\* signal 2-98
  - BERR\* signal 1-42
  - capabilities 2-4, 2-9, 2-11
  - interface 1-7
  - interrupter function, VMEchip2 ASIC
    - 2-16
  - interrupter, programming 2-51
  - IRQ1, IRQ2 interrupts 2-95
  - mapping 1-39
  - maps, creating 2-6
  - master, bus sizing and 2-6
  - request level (DMAC) 2-54, 2-55
  - requester, DMAC 2-13
  - segment size, translating 2-30
  - slave 2-9
  - slave map decoders 2-26
  - slave map decoders, programming 2-26
  - system controller function 2-17
  - timer 2-18
- VMEbus Slave registers
  - Address Modifier Select Register 1 2-36
  - Address Modifier Select Register 2 2-33
  - Address Translation Address Offset
    - Register 1 2-29
    - Register 2 2-31
  - Address Translation Select Register 1
    - 2-30
  - Address Translation Select Register 2
    - 2-31
  - Ending Address Register 1 2-28
  - Ending Address Register 2 2-29
  - GCSR Board Address Register 2-48
  - GCSR Group Address Register 2-47
  - Starting Address Register 1 2-28
  - Starting Address Register 2 2-29

- 
- VMEbus Slave registers *continued*
    - Write Post and Snoop Control Register 1  
[2-35](#)
    - Write Post and Snoop Control Register 2  
[2-32](#)
  - VMEbus-to-local-bus interface [1-9](#), [2-9](#)
  - VMEchip2 ASIC [1-5](#)
    - block diagram [2-5](#)
    - functional blocks [2-4](#)
    - GCSR programming model [2-100](#)
    - local BERR\* [1-42](#)
    - programming model [2-20](#)
  - VMEchip2/Petra chip redundancies [1-9](#)

## W

- watchdog timer
  - Petra/MC2 ASIC [3-9](#)
  - VMEchip2 [2-14](#), [2-15](#)
- write post
  - buffer [2-6](#), [2-9](#)
  - bus error interrupter, VMEbus [2-19](#)
  - enable bits [2-39](#)
  - enabling [2-32](#), [2-35](#), [2-43](#), [2-44](#), [2-50](#),  
[2-51](#)
  - register [2-32](#)
  - timer, VMEchip2 ASIC [2-7](#)
- write posting, VMEchip2 ASIC [2-6](#), [2-9](#)
- write-protect feature of Flash memory [3-2](#)

## Z

- Z85230 SCC interface [3-7](#)
  - register addresses [1-31](#)