

**MVME162LX  
200/300 Series  
Embedded Controller  
Programmer's  
Reference Guide**

**V162LX2-3A/PG2**

## **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.

This material may be read online, downloaded for personal use, or included in another document as a URL (from this server). The text itself may not be published commercially in print or electronic form, edited or otherwise altered without the permission of Motorola, Inc.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 announced in your country. Such references or information must not be construed to mean that Motorola intends to announce such Motorola products, programming, or services in your country.

## **Restricted Rights Legend**

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

Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013.

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

## Preface

This Programmer's Reference Guide provides board level information and detailed ASIC chip information, including register bit descriptions for the MVME162LX 200/300 Series VME Embedded Controller board. The information in this manual applies to the VME162LX models that are currently available in the 200/300 Series.

For information on the 700/800 Series boards, refer to the *MVME162LX 700/800 Series Programmer's Guide*. Both series are designated "LX" to distinguish them from the embedded controller boards in the MVME 162's FX family: the -LX boards route their Input/Output signals through the front panel of the board; the -FX send theirs through the P2 backplane connector.



**CAUTION**

In the event you upgrade from a 200/300 Series controller board to a 700/800 Series controller board, you must reference the *MVME162LX 700/800 Series Programmer's Guide*, Chapter 1, about several board-level changes affecting software compatibility.

The models in the following table comprise the 200/300 Series of the MVME162LX family of controllers at the time of this Guide's publication:

Series 162-2xx	Series 162-2xx (Cont'd)	Series 162-3xx
MVME162-200	MVME162-222	MVME162-322
MVME162-201	MVME162-223	MVME162-323
MVME162-202	MVME162-233	MVME162-333
MVME162-210	MVME162-243	
MVME162-211	MVME162-253	MVME162-353
MVME162-212	MVME162-262	
MVME162-213	MVME162-263	
MVME162-216		
MVME162-220		

## **Safety Summary**

### **Safety Depends On You**

The following general safety precautions must be observed during all phases of operation, service, and repair of this equipment. Failure to comply with these precautions or with specific warnings elsewhere in this manual violates safety standards of design, manufacture, and intended use of the equipment. Motorola, Inc. assumes no liability for the customer's failure to comply with these requirements.

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

#### **Ground the Instrument.**

To minimize shock hazard, the equipment chassis and enclosure must be connected to an electrical ground. The equipment is supplied with a three-conductor ac power cable. The power cable must be plugged into an approved three-contact electrical outlet. The power jack and mating plug of the power cable meet International Electrotechnical Commission (IEC) safety standards.

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

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

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

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

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

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

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

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

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

Because of the danger of introducing additional hazards, do not install substitute parts or perform any unauthorized modification of the equipment. Contact your local Motorola representative for service and repair to ensure that safety features are maintained.

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

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



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

This manual is intended for anyone who wants to program MVME162LX 200/300 Series boards in order to design OEM systems, supply additional capability to an existing compatible system, or work in a lab environment for experimental purposes. A basic knowledge of computers and digital logic is assumed.

For a listing of MCG documents referenced along with other applicable publications, refer to the Appendix C.

## Manual Terminology

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

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

For example, “12” is the decimal number twelve, and “\$12” is the decimal number eighteen. Unless otherwise specified, all address references are in hexadecimal.

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

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

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.



European Notice: Board 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 (CISPR 22) Radio Frequency Interference

EN50082-1 (IEC801-2, IEC801-3, IEEC801-4) Electromagnetic Immunity

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

This board product was 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.

The computer programs stored in the Read Only Memory of this device contain material copyrighted by Motorola Inc. and may be used only under a license such as those contained in Motorola's software licenses.

Motorola<sup>®</sup> and the Motorola symbol are registered trademarks of Motorola, Inc.

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

© Copyright Motorola, Inc. 1999  
All Rights Reserved

Printed in the United States of America

# Contents

---

## CHAPTER 1 Board Description and Memory Maps

Introduction.....	1-1
What this Guide Provides .....	1-1
Contents of the Chapters.....	1-2
Overview and Features of the MVME162LX 200/300 Series.....	1-3
Block Diagram .....	1-7
Functional Description.....	1-7
Real-time Clock and its operation .....	1-7
No VMEbus Interface Option.....	1-8
VMEbus Interface and VMEchip2 .....	1-11
Configuration Jumpers.....	1-11
Memory Maps.....	1-14
Local Bus Memory Map.....	1-14
Normal Address Range.....	1-14
Detailed I/O Memory Maps.....	1-20
BBRAM/TOD Clock Memory Map.....	1-36
BBRAM/TOD Clock Configuration Bytes .....	1-38
Interrupt Acknowledge Map.....	1-42
VMEbus Memory Map.....	1-42
VMEbus Accesses to the Local Bus.....	1-42
VMEbus Short I/O Memory Map.....	1-42
Software Support Considerations .....	1-43
Interrupts.....	1-43
Cache Coherency .....	1-43
Sources of Local BERR* .....	1-44
Local Bus Timeout .....	1-44
VMEbus Access Timeout.....	1-44
VMEbus BERR* .....	1-44
Local DRAM Parity Error .....	1-45
VMEchip2 .....	1-45
Bus Error Processing .....	1-45

---

Description of Error Conditions on the MVME162LX 200/300 Series .....	1-46
MPU Parity Error .....	1-46
MPU Offboard Error .....	1-46
MPU TEA - Cause Unidentified .....	1-47
MPU Local Bus Timeout .....	1-47
DMAC VMEbus Error .....	1-48
DMAC Parity Error .....	1-48
DMAC Offboard Error .....	1-49
DMAC LTO Error .....	1-49
DMAC TEA - Cause Unidentified .....	1-50
LAN Parity Error .....	1-50
LAN Offboard Error .....	1-51
LAN LTO Error .....	1-51
SCSI Parity Error .....	1-52
SCSI Offboard Error .....	1-52
SCSI LTO Error .....	1-53
Example of the Proper Use of Bus Timers .....	1-53
MVME162LX 200/300 Series MC68040 Indivisible Cycles .....	1-55
Illegal Access to IP Modules from External VMEbus Masters .....	1-55

## **CHAPTER 2   VMEchip2**

Introduction .....	2-1
Summary of Major Features .....	2-1
Functional Blocks .....	2-4
Local Bus to VMEbus Interface .....	2-6
Interface Components .....	2-6
Local Bus to VMEbus Requester .....	2-8
VMEbus to Local Bus Interface .....	2-10
Local Bus to VMEbus DMA Controller .....	2-12
No Address Increment DMA Transfers .....	2-14
DMAC VMEbus Requester .....	2-15
Tick and Watchdog Timers .....	2-16
Prescaler .....	2-16
Tick Timer .....	2-16
Watchdog Timer .....	2-17
VMEbus Interrupter .....	2-17



---

VMEbus System Controller.....	2-18
Arbiter.....	2-18
IACK Daisy-Chain Driver.....	2-18
Bus Timer .....	2-19
Reset Driver.....	2-19
Local Bus Interrupter and Interrupt Handler .....	2-19
Global Control and Status Registers.....	2-21
LCSR Programming Model .....	2-22
Programming the VMEbus Slave Map Decoders.....	2-28
VMEbus Slave Ending Address Register 1 .....	2-31
VMEbus Slave Starting Address Register 1.....	2-31
VMEbus Slave Ending Address Register 2.....	2-31
VMEbus Slave Starting Address Register 2.....	2-31
VMEbus Slave Address Translation Address Offset Register 1 .....	2-32
VMEbus Slave Address Translation Select Register 1 .....	2-33
VMEbus Slave Address Translation Address Offset Register 2 .....	2-34
VMEbus Slave Address Translation Select Register 2 .....	2-34
VMEbus Slave Write Post and Snoop Control Register 2 .....	2-35
VMEbus Slave Address Modifier Select Register 2 .....	2-36
VMEbus Slave Write Post and Snoop Control Register 1 .....	2-37
VMEbus Slave Address Modifier Select Register 1 .....	2-38
Programming the Local Bus to VMEbus Map Decoders .....	2-39
Local Bus Slave (VMEbus Master) Ending Address Register 1.....	2-41
Local Bus Slave (VMEbus Master) Starting Address Register 1.....	2-41
Local Bus Slave (VMEbus Master) Ending Address Register 2.....	2-42
Local Bus Slave (VMEbus Master) Starting Address Register 2.....	2-42
Local Bus Slave (VMEbus Master) Ending Address Register 3.....	2-42
Local Bus Slave (VMEbus Master) Starting Address Register 3.....	2-42
Local Bus Slave (VMEbus Master) Ending Address Register 4.....	2-43
Local Bus Slave (VMEbus Master) Starting Address Register 4.....	2-43
Local Bus Slave (VMEbus Master) Address Translation Address Reg. 4 .....	2-44
Local Bus Slave (VMEbus Master) Address Translation Select Reg. 4 .....	2-44
Local Bus Slave (VMEbus Master) Attribute Register 4.....	2-44
Local Bus Slave (VMEbus Master) Attribute Register 3.....	2-45
Local Bus Slave (VMEbus Master) Attribute Register 2.....	2-46
Local Bus Slave (VMEbus Master) Attribute Register 1 .....	2-47
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.....	2-52
DMAC Registers .....	2-53
PROM Decoder, SRAM and DMA Control Register .....	2-54
Local Bus To VMEbus Requester Control Register .....	2-55
DMAC Control Register 1 (bits 0-7) .....	2-56
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 .....	2-59
DMAC Byte Counter .....	2-59
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 Timeout Control Register.....	2-64
DMAC Ton/Toff Timers and VMEbus Global Timeout Control Register.....	2-65
VME Access, Local Bus and Watchdog Timeout 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) .....	2-76
Local Bus Interrupter Status Register (bits 16-23) .....	2-77
Local Bus Interrupter Status Register (bits 8-15) .....	2-78
Local Bus Interrupter Status Register (bits 0-7) .....	2-79
Local Bus Interrupter Enable Register (bits 24-31) .....	2-80
Local Bus Interrupter Enable Register (bits 16-23) .....	2-81
Local Bus Interrupter Enable Register (bits 8-15) .....	2-82
Local Bus Interrupter Enable Register (bits 0-7) .....	2-83
Software Interrupt Set Register (bits 8-15) .....	2-84

---

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) .....	2-88
Interrupt Level Register 1 (bits 16-23) .....	2-88
Interrupt Level Register 1 (bits 8-15) .....	2-89
Interrupt Level Register 1 (bits 0-7) .....	2-89
Interrupt Level Register 2 (bits 24-31) .....	2-90
Interrupt Level Register 2 (bits 16-23) .....	2-90
Interrupt Level Register 2 (bits 8-15) .....	2-91
Interrupt Level Register 2 (bits 0-7) .....	2-91
Interrupt Level Register 3 (bits 24-31) .....	2-92
Interrupt Level Register 3 (bits 16-23) .....	2-92
Interrupt Level Register 3 (bits 8-15) .....	2-93
Interrupt Level Register 3 (bits 0-7) .....	2-93
Interrupt Level Register 4 (bits 24-31) .....	2-94
Interrupt Level Register 4 (bits 16-23) .....	2-94
Interrupt Level Register 4 (bits 8-15) .....	2-95
Interrupt Level Register 4 (bits 0-7) .....	2-95
Vector Base Register .....	2-96
I/O Control Register 1 .....	2-97
I/O Control Register 2 .....	2-98
I/O Control Register 3 .....	2-99
Miscellaneous Control Register.....	2-100
GCSR Programming Model.....	2-102
Programming the GCSR .....	2-104
VMEchip2 Revision Register .....	2-106
VMEchip2 ID Register .....	2-106
VMEchip2 LM/SIG Register.....	2-106
VMEchip2 Board Status/Control Register .....	2-108
General Purpose Register 0 .....	2-108
General Purpose Register 1 .....	2-109
General Purpose Register 2 .....	2-109
General Purpose Register 3 .....	2-109
General Purpose Register 4 .....	2-110
General Purpose Register 5 .....	2-110

---

## CHAPTER 3 MCchip

Introduction .....	3-1
Summary of Major Features .....	3-1
Functional Description .....	3-2
MCchip 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
MC68040-Bus Master Support for 82596CA .....	3-4
LANC Bus Error .....	3-5
LANC Interrupt .....	3-5
53C710 SCSI Controller Interface .....	3-5
SRAM Memory Controller .....	3-5
Parity DRAM Memory Controller .....	3-6
Z85230 SCC Interface.....	3-7
Tick Timers.....	3-8
Watchdog Timer .....	3-8
Local Bus Timer .....	3-9
Memory Map of the MCchip Registers .....	3-10
Programming Model.....	3-12
MCchip ID Register .....	3-12
MCchip Revision Register .....	3-13
General Control Register.....	3-13
Interrupt Vector Base Register .....	3-14
Programming the Tick Timers.....	3-16
Tick Timer 1 and 2 Compare and Counter Registers.....	3-16
LSB Prescaler Count Register.....	3-17
Prescaler Clock Adjust Register.....	3-18
Tick Timer 1 and 2 Control Registers.....	3-19
Tick Timer Interrupt Control Registers.....	3-20
DRAM Parity Error Interrupt Control Register .....	3-22
SCC Interrupt Control Register.....	3-23
Tick Timer 3 and 4 Control Registers .....	3-24

---

Parity DRAM and SRAM Memory Controller Registers.....	3-25
Parity DRAM Space Base Address Register .....	3-25
SRAM Space Base Address Register .....	3-26
Parity DRAM Space Size Register .....	3-27
Parity DRAM/SRAM Options Register .....	3-28
SRAM Space Size Register .....	3-29
LANC Error Status Register .....	3-31
82596CA LANC Interrupt Control Register .....	3-32
LANC Bus Error Interrupt Control Register .....	3-33
SCSI Error Status Register .....	3-34
General Purpose Inputs Register.....	3-35
MVME162LX 200/300 Series Version Register .....	3-36
SCSI Interrupt Control Register .....	3-37
Tick Timer 3 and 4 Compare and Counter Registers.....	3-38
Bus Clock Register .....	3-39
EPROM Access Time Control Register .....	3-40
Flash Access Time Control Register.....	3-41
ABORT Switch Interrupt Control Register .....	3-42
RESET Switch Control Register.....	3-43
Watchdog Timer Control Register .....	3-44
Access and Watchdog Time Base Select Register .....	3-45
Parity DRAM Control Register .....	3-46
MPU Status Register.....	3-48
32-bit Prescaler Count Register .....	3-49

## CHAPTER 4 IPIC

Introduction.....	4-1
Features .....	4-1
Functional Description.....	4-2
General Description .....	4-2
Performance .....	4-2
Cache Coherency .....	4-3
Error Reporting .....	4-3
Interrupts .....	4-3
Overall Memory Map .....	4-5
Programming Model .....	4-6
Chip ID Register .....	4-9
Chip Revision Register .....	4-9

---

IP_a, IP_b, IP_c, IP_d Memory Base Address Registers.....	4-10
IP_a or Double Size IP_ab Memory Base Address Registers.....	4-11
IP_b Memory Base Address Registers.....	4-11
IP_c or Double Size IP_cd Memory Base Address Registers.....	4-11
IP_d Memory Base Address Registers.....	4-11
IP_a, IP_b, IP_c, IP_d Memory Size Registers.....	4-12
IP_a, IP_b, IP_c, IP_d: IRQ0/IRQ1 Interrupt Control Registers.....	4-13
IP_a, IP_b, IP_c, and IP_d: General Control Registers.....	4-14
IP RESET Register.....	4-16
Local Bus to IndustryPack Addressing.....	4-17
8-Bit Memory Space.....	4-17
16-Bit Memory Space.....	4-18
32-Bit Memory Space.....	4-19
IP_a I/O Space.....	4-20
IP_ab I/O Space.....	4-20
IP_a ID Space.....	4-21
IP to Local Bus Data Routing.....	4-22
Memory Space Accesses.....	4-22
I/O and ID Space Accesses.....	4-24

## CHAPTER 5 MCECC

Introduction.....	5-1
Features.....	5-1
Functional Description.....	5-2
General Description.....	5-2
Performance.....	5-2
Cache Coherency.....	5-4
ECC.....	5-5
Cycle Types.....	5-5
Error Reporting.....	5-5
Error Logging.....	5-7
Scrub.....	5-8
Refresh.....	5-8
Arbitration.....	5-8
Chip Defaults.....	5-9

---

Programming Model .....	5-10
Key for bit definitions in CSR .....	5-11
Memory Map of the MCECC Internal Register .....	5-12
Chip ID Register .....	5-15
Chip Revision Register .....	5-15
Memory Configuration Register .....	5-16
Dummy Register 0 .....	5-17
Dummy Register 1 .....	5-18
Base Address Register .....	5-18
DRAM Control Register .....	5-19
BCLK Frequency Register .....	5-21
Data Control Register .....	5-22
Scrub Control Register .....	5-23
Scrub Period Register Bits 15-8 .....	5-25
Scrub Period Register Bits 7-0 .....	5-25
Chip Prescaler Counter .....	5-25
Scrub Time On/Time Off Register .....	5-26
Scrub Prescaler Counter (Bits 21-16) .....	5-27
Scrub Prescaler Counter (Bits 15-8) .....	5-28
Scrub Prescaler Counter (Bits 7-0) .....	5-28
Scrub Timer Counter (Bits 15-8) .....	5-28
Scrub Timer Counter (Bits 7-0) .....	5-29
Scrub Address Counter (Bits 26-24) .....	5-29
Scrub Address Counter (Bits 23-16) .....	5-30
Scrub Address Counter (Bits 15-8) .....	5-30
Scrub Address Counter (Bits 7-4) .....	5-30
Error Logger Register .....	5-31
Error Address (Bits 31-24) .....	5-32
Error Address (Bits 23-16) .....	5-32
Error Address Bits (15-8) .....	5-32
Error Address Bits (7-4) .....	5-33
Error Syndrome Register .....	5-33
Defaults Register 1 .....	5-34
Defaults Register 2 .....	5-36
Initialization .....	5-37
Syndrome Decode .....	5-38

---

## APPENDIX A Serial Interface Connections

Introduction .....	A-1
Connection Diagrams .....	A-2

## APPENDIX B Interrupt Usage

Introduction .....	B-1
Interrupt Handler Routine Setup.....	B-1
Periodic Interrupt Example.....	B-2
Set Up Tick Timer 1 .....	B-2
Setup Local Bus Interrupter .....	B-2

## APPENDIX C Related Documentation

Motorola Computer Group Documents.....	C-1
Additional Motorola Publications .....	C-2
Manufacturers' Publications.....	C-3
Related Specifications .....	C-4

## FIGURES

Figure 1-1. MVME162LX 200/300 Series Block Diagram .....	1-9
Figure 1-2. MVME162LX 200/300 Series Board Layout .....	1-13
Figure 2-1. VMEchip2 Block Diagram .....	2-5
Figure A-1. Serial Interface Connections: Z85230-to-RJ45 Adapter .....	A-2
Figure A-2. DB25-DTE-to-RJ45 Adapter .....	A-3
Figure A-3. DB25-DCE-to-RJ45 Adapter .....	A-4
Figure A-4. Typical RJ45 Serial Cable .....	A-5

## TABLES

Table 1-1. MVME162LX 200/300 Series Features Summary .....	1-5
Table 1-2. Redundant Functions in the VMEchip2 and MCchip .....	1-10
Table 1-3. Jumper Settings for the 200/300 Series .....	1-12
Table 1-4. Local Bus Memory Map .....	1-15
Table 1-5. Local I/O Devices Memory Map .....	1-17
Table 1-6. VMEchip2 Memory Map (Sheet 1 of 3) .....	1-22
Table 1-6. VMEchip2 Memory Map (Sheet 2 of 3) .....	1-24
Table 1-6. VMEchip2 Memory Map (Sheet 3 of 3) .....	1-26
Table 1-7. MCchip Register Map .....	1-27
Table 1-8. IPIC Overall Memory Map .....	1-29



---

Table 1-9. IPIC Memory Map - Control and Status Registers .....	1-30
Table 1-10. MCECC Internal Register Memory Map .....	1-32
Table 1-11. Z85230 SCC Register Addresses .....	1-33
Table 1-12. 82596CA Ethernet LAN Memory Map .....	1-34
Table 1-13. 53C710 SCSI Memory Map .....	1-34
Table 1-14. MK48T08 BBRAM/TOD Clock Memory Map .....	1-36
Table 1-15. BBRAM Configuration Area Memory Map .....	1-36
Table 1-16. TOD Clock Memory Map .....	1-38
Table 2-1. VMEchip2 Memory Map - LCSR Summary (Sheet 1 of 2) .....	2-24
Table 2-1. VMEchip2 Memory Map - LCSR Summary (Sheet 2 of 2) .....	2-26
Table 2-2. DMAC Command Table Format .....	2-53
Table 2-3. Local Bus Interrupter Summary .....	2-75
Table 2-4. VMEchip2 Memory Map - GCSR Summary .....	2-105
Table 3-1. Parity DRAM Performance .....	3-6
Table 3-2. MCchip Register Map .....	3-10
Table 3-3. Interrupt Vector Base Register Encoding and Priority .....	3-15
Table 3-4. Parity DRAM Size Control Bit Encoding .....	3-27
Table 3-5. SRAM Size Control Bit Encoding .....	3-29
Table 3-6. SRAM Size Control Bit Encoding .....	3-29
Table 4-1. IPIC Overall Memory Map .....	4-5
Table 4-2. IPIC Memory Map - Control and Status Registers .....	4-7
Table 5-1. MCECC Specifications .....	5-3
Table 5-2. MCECC Internal Register Memory Map, Part 1 .....	5-12
Table 5-3. MCECC Internal Register Memory Map, Part 2 .....	5-13
Table C-1. Motorola Computer Group Documents .....	C-2
Table C-2. Additional Motorola Documents .....	C-2
Table C-3. Manufacturers' Documents .....	C-3

---

# Board Description and Memory Maps

# 1

## Introduction

### What this Guide Provides

This guide presents programming information for the MVME162LX 200/300 Series Embedded Controller. Extensive programming information is provided for the Application-Specific Integrated Circuit (ASIC) devices used on these models of the board. Reference information is included for the Large Scale Integration (LSI) devices that are used and sources for additional information are also provided.

For the corresponding information on the boards in the 700/800 Series of the MVME162, see the *MVME162LX 700/800 Series Programmer's Guide*. Both sets of series are designated LX to distinguish them from the embedded controller boards in the MVME 162's FX series: the -LX boards have their Input/Output connectors at the front of the board; the -FX route their I/O to the P2 adapter.



#### Caution

If upgrading from a 200/300 Series 162LX to a 700/800 Series board, you must consult the *MVME162LX 700/800 Series Programmer's Guide*, Chapter 1, about several board-level changes affecting software compatibility.

Be sure that your board's real-time clock has been started and set. See below, under [Functional Description on page 1-7](#).

## Contents of the Chapters

This first chapter briefly describes the board level hardware features of the MVME162LX 200/300 Series Embedded Controller. The chapter begins with a board level overview and features summary. A summary of Memory maps is next, and the chapter closes with some general software considerations such as cache coherency, interrupts, and bus errors. For a more detailed hardware functional description, including front panel switches and indicators, see the *MVME162LX Embedded Controller Installation and Use* manual, the *200/300 Series* version.

All programmable registers in the MVME162LX 200/300 Series that reside in ASICs are covered in the chapters on those ASICs. Coverage is as follows:

- ❑ Chapter 2: VMEchip2 (for Versa Module Europa bus)
- ❑ Chapter 3: MCchip (Memory Controller)
- ❑ Chapter 4: IPIC chip (IndustryPack Interface Controller)
- ❑ Chapter 5: MCECC chip (Memory Controller for ECC-protected DRAM)
- ❑ Appendix A: Serial interface connections
- ❑ Appendix B: Interrupt usage
- ❑ Appendix C: Related documentation

For those interested in programmable register bit definitions and less interested in hardware functionality, focus on Chapters 2, 3, 4, and 5. In some cases, however, Chapter 1 gives related background information.

# Overview and Features of the MVME162LX 200/300 Series

The MVME162LX 200/300 Series is based on the MC68040 or MC68LC040 microprocessor. The -LC040, a subset of the -040 line for low cost embedded controller applications, is the -040 without a built-in math coprocessor (FPU). The MVME162LX 200/300 Series is available in various versions with the features listed in [Table 1-1 on page 1-5](#).

The I/O on the MVME162LX 200/300 Series is routed through front panel connectors, and the board uses no transition boards. These characteristics also apply to the other models of the 162LX board, those in the 700/800 Series.

The I/O connection for the serial ports on the MVME162LX is provided by four RJ45 front panel I/O connectors. (For more information on the Serial Communications Interface see the Hardware Description chapter of the *MVME162LX Embedded Controller Installation and Use* manual, the *200/300 Series* version.)

Various versions of the MVME162LX 200/300 Series have either 1 or 4MB of parity-protected DRAM, or 4, 8, 16, or 32MB of ECC-protected DRAM on a mezzanine board. (Note the CAUTION on a Parity DRAM-related issue in the Features Summary; see [Table 1-1 on page 1-5](#); note also that, in the 700/800 Series, parity protection for DRAM is unsupported). The boards also come with either 128 KB of SRAM (with battery backup) or 2 MB SRAM on a mezzanine board (with battery backup). The MVME162LX 200/300 Series also comes with a time of day clock (with battery backup), an optional LAN Ethernet transceiver interface, four serial connections with EIA-232-D interface, six tick timers with watchdog timer(s), four EPROM sockets, 1 MB Flash memory (one Flash device), two IndustryPack (IP) interfaces, optional SCSI bus interface with DMA, and optional VMEbus interface (local bus to VMEbus/VMEbus to local bus, with A16/A24/A32, D8/D16/D32 bus widths and a VMEbus system controller). Note that currently one model comes with no VMEbus to provide an additional option.

The following ASICs are used on the MVME162LX 200/300 Series:


- ❑ **Versa Module Europa-bus (VMEchip2).** Provides the VMEbus interface. The VMEchip2 includes two tick timers, a watchdog timer, programmable map decoders for the master and slave interfaces, and a VMEbus to/from local bus DMA controller. The VMEchip2 ASIC also includes 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.

- ❑ **Memory Controller (MCchip).** Provides four tick timers, the interface to the LAN chip, SCSI chip, serial port chip, BBRAM, EPROM/Flash, Parity DRAM and SRAM.
- ❑ **Memory Controller for ECC-protected DRAM (MCECC chip).** Provides the programmable interface for the 4, 8, 16, or 32MB ECC-protected DRAM mezzanine board.
- ❑ **IndustryPack Interface Controller (IPIC chip).** Provides control and status information for up to two single size IndustryPacks (IPs) or one double size IP that can be plugged into the MVME162LX 200/300 Series main module.

**Table 1-1. MVME162LX 200/300 Series Features Summary**

Feature	Description
Processor	25 MHz 32-bit MC68040 or MC68LC040 microprocessor
DRAM	<p>1 or 4MB of shared DRAM with parity protection on a mezzanine module; or 4, 8, 16, or 32MB of shared DRAM with ECC protection on a mezzanine board</p> <p><b>Note</b> the following known problem and its work-around: When the RESET button on an MVME162LX 200/300 Series board is depressed for longer than a minimum length of time, Parity DRAM refresh may be inhibited and memory contents may be lost. This timelength may differ from one board to another and may vary with environmental conditionals.</p> <p> <b>Caution</b> If you intend to examine memory subsequent to a reset, release the RESET button as quickly as possible after depressing it to ensure that Parity DRAM contents will not be altered.</p> <p><b>Notes</b> On 200/300 Series boards, DRAM not specified as ECC-protected is parity-protected. The Parity DRAM's information is given in the <i>MCchip</i> chapter of this Guide, the ECC DRAM's is in the <i>MCECC</i> chapter.</p> <p>(In the later, 700/800 Series parity is unsupported on non-ECC DRAM, which currently comes on only one board in that Series. Parity must remain <i>disabled</i> on all 700/800 Series boards.)</p>
SRAM	128 KB of SRAM with battery backup; also available: 2 MB SRAM on a mezzanine board with battery backup
PROM/ EPROM Sockets	Four JEDEC standard 32-pin DIP PROM sockets for EPROM and Flash
Flash	One Intel 28F008SA 1M x 8 Flash memory device with write protection and MVME162BUG.
NVRAM and TOD clock	8K by 8 Non-Volatile RAM (NVRAM) and Time-of-Day (TOD) clock with battery backup: MK48T08

**Table 1-1. MVME162LX 200/300 Series Features Summary (Continued)**

<b>Feature</b>	<b>Description</b>
Timers	In the MCchip ASIC: Four 32-bit Tick Timers and Watchdog Timer for periodic interrupts
	In the VMEchip2 ASIC: Two 32-bit Tick Timers and Watchdog Timer for periodic interrupts
Software Interrupts	For MVME162LX models with the VMEchip2 (which means most models): eight software interrupts
I/O	Four serial ports with EIA-232-D interface, connection via four, front-panel RJ-45 connectors
	Serial port controllers: Zilog Z85230
	Optional: Small Computer Systems Interface (SCSI) bus interface with 32-bit local bus burst Direct Memory Access (DMA); controller is an NCR 53C710
	Optional: Ethernet transceiver interface with 32-bit local bus DMA; controller is an Intel 82596CA
	Two IndustryPack interfaces
VMEbus interface (boards may be special ordered without the VMEbus interface)	VMEbus system controller functions
	VMEbus interface to local bus (A24/A32, D8/D16/D32 [D8/D16/D32/D64 BLT]) (BLT = Block Transfer)
	Local bus to VMEbus interface (A16/A24/A32, D8/D16/D32)
	VMEbus interrupter
	VMEbus interrupt handler
	Global CSR for interprocessor communications
	DMA for fast local memory - VMEbus transfers (A16/A24/A32, D16/D32 [D16/D32/D64 BLT])
Switches	Two pushbutton switches: ABORT and RESET
Light-Emitting Diodes (LEDs)	Four status LEDs: FAIL, RUN, SCON, FUSES



## Block Diagram

Figure 1-1 on page 1-9 is a general block diagram of the MVME162LX 200/300 Series.

## Functional Description

This section covers only a few specific features of the MVME162LX 200/300 Series.

A complete functional description of the major blocks of the MVME162LX 200/300 Series Embedded Controller is provided in your *MVME162LX Embedded Controller Installation and Use* manual, the *200/300 Series* version. The Installation and Use manual also covers startup procedures and use of the board's debugger firmware.

## Real-time Clock and its operation

Each MVME162LX 200/300 Series board has a real-time clock (RTC) with self-contained battery. Before shipment of the board this clock is stopped to preserve battery life.

The board's self-tests and operating systems require that its real-time clock be running, so after the initial hardware installation the clock must be started before the board is used. If necessary, start the real-time clock, and set the date and time, using the debugger firmware with the following command line structure:

```
162-Bug> SET [mddyhhmm][<+/-CAL>;C]
```

For example, the following command line starts the real-time clock and sets the date and time to 10:37 a.m., November 7, 1999:

```
162-Bug> SET 1107991037
```

Note that the **SET** command alone will *not* start the clock (gives error message); the **C** option allows you to calibrate the real-time clock. Refer to the *MVME162Bug Debugging Package User's Manual* for details.

If an MVME162LX 200/300 Series board is put into storage, its real-time clock must be put into power save mode to extend the life of the battery. To do this, use the **PS** command of the debugger firmware (no arguments required):

```
162-Bug> ps
```

## No VMEbus Interface Option

An MVME162LX board may be set up to operate as an embedded controller without the VMEbus interface. For this model option, the VMEchip2 ASIC and the VMEbus buffers are not populated. Also, the bus grant daisy chain and the interrupt acknowledge daisy chain have zero-ohm bypass resistors installed.

**Note** Currently only *one* MVME162LX 200/300 Series model comes with the no VMEbus option.

To support this board model, certain logic in the VMEchip2 has been duplicated in the MCchip. [Table 1-2 on page 1-10](#) defines the location of the redundant logic. This logic is inhibited in the MCchip if the VMEchip2 is present. The enables for these functions are controlled by software and MCchip hardware initialization.

(Note that the MVME162LX 200/300 Series model without the VMEbus interface is shipped with Flash memory blank; the factory uses the VMEbus to program the Flash memory with debugger code. To use MVME162Bug, the 162Bug package, in such a model be sure that the General Purpose Readable Jumpers Header is configured for the EPROM memory map. Refer to Chapters 1 and 3 of your *MVME162LX Embedded Controller Installation and Use* manual, the *200/300 Series* version, for further details. But note, again, that currently only one MVME162LX 200/300 Series model comes with the no VMEbus interface option.)

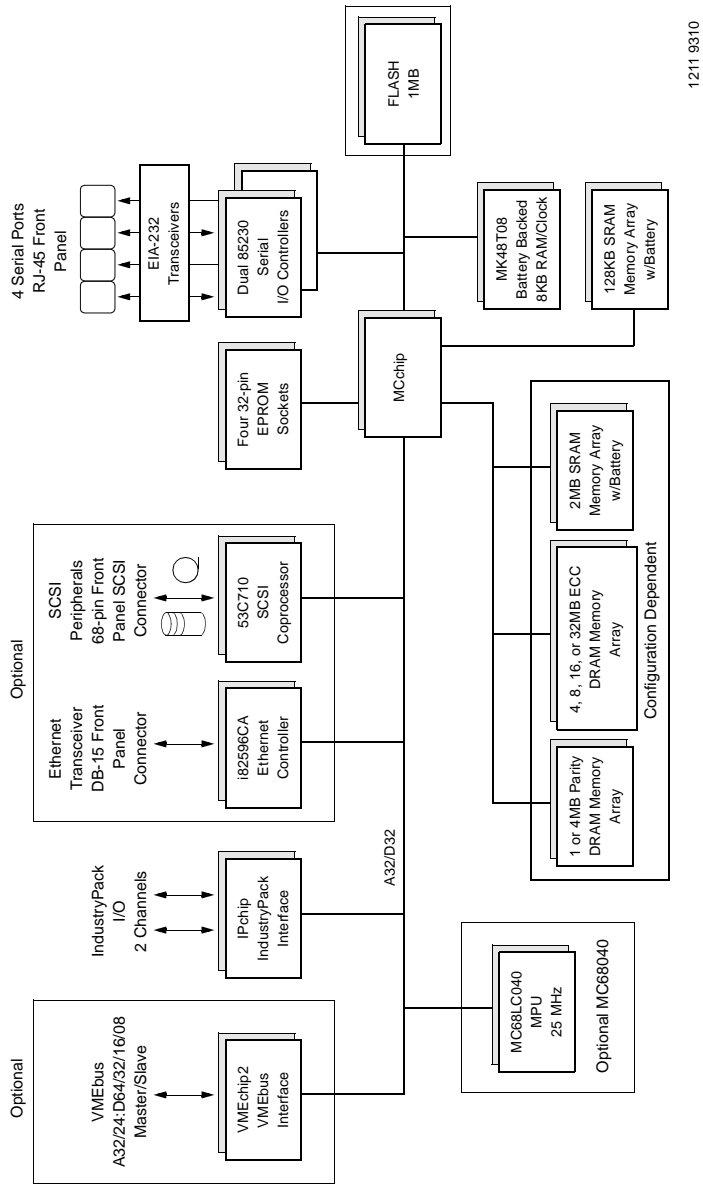


Figure 1-1. MVME162LX 200/300 Series Block Diagram

**Table 1-2. Redundant Functions in the VMEchip2 and MCchip**

VMEchip2		MCchip		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	31 - 0	8
		\$FFF42040	6 - 0	6
\$FF800000-\$FFBFFFFFFF	31 - 0	\$FF800000-\$FFBFFFFFFF	31 - 0	7
\$FFE00000-\$FFEFFFFFFF	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 VMEchip2 and MCchip have a one-to-one correspondence.
6. The ABORT switch interrupt control is implemented in the VMEchip2 but with a different bit organization. Refer to the VMEchip2 description in Chapter 2. The ABORT switch is wired to the MCchip and not the VMEchip2 in the MVME162LX 200/300 Series implementation.
7. The SRAM and PROM 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.

## VMEbus Interface and VMEchip2

The local bus to VMEbus interface and the VMEbus to local bus interface are provided by the VMEchip2. The VMEchip2 can also provide the VMEbus system controller functions. Refer to the chapter on the VMEchip2 for detailed programming information.

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

## Configuration Jumpers

The MVME162LX 200/300 Series provides software control over most options: by setting bits in control registers after installing the module in a system, you can modify its configuration. These registers and their programming is covered throughout the rest of this Guide in the chapters for each of the four ASICs in the MVME162LX 200/300 Series.

Some options, however, are not software-programmable. Such options are controlled manually, by changing the default, factory jumper settings with which the MVME162LX 200/300 Series has been factory tested and shipped. (Note that the board's required and factory-installed Debug Monitor, MVME162Bug (162Bug), operates with the default, factory jumper settings.)

[Table 1-3 on page 1-12](#) lists the manually configurable items on the 200/300 Series board, with the default settings enclosed in brackets. This table also gives the settings for the corresponding jumpers on the board. [Figure 1-2 on page 1-13](#) follows the table and shows the jumper/connectors in the board diagram. For more detailed information, see the section on Configuration Jumpers in the *MVME162LX Embedded Controller Installation and Use* manual, the 200/300 Series version.

**Table 1-3. Jumper Settings for the 200/300 Series**

<b>Jumper</b>	<b>Function</b>	<b>Settings (defaults bracketed)</b>	
J1	System controller selection	[1-2] No jumper	System controller. Not system controller.
J11	General-purpose readable jumper configuration	[No jumper] 7-8	EPROM selected. Flash selected. Other headers are user-definable (see <i>Installation and Use</i> manual).
J12	EPROM/Flash configuration	[3-4, 5-6, 9-11, 10-12] 3-4, 9-11, 10-12 5-6, 8-10, 11-13 7-9, 8-10 1-2, 7-9, 8-10	128K x 8 EPROMs 256K x 8 EPROMs. 512K x 8 EPROMs. 1M x 8 EPROMs. 1M x 8 EPROMs (onboard Flash disabled).
J13	SRAM backup power source selection (onboard SRAM)	No jumper [1-3, 2-4] 3-5, 4-6 1-3, 4-6 2-4, 3-5	Backup power disabled (storage only) Primary : VMEbus +5V STBY — secondary : VMEbus +5V STBY. Primary : onboard battery — secondary : onboard battery. Primary : VMEbus +5V STBY — secondary : onboard battery. Primary : onboard battery — secondary : VMEbus +5V STBY. (For the 2MB SRAM on mezzanine board option: Header J1 on the mezzanine board--see <i>Installation and Use</i> manual)
J14	SCSI termination	No jumper [1-2]	Onboard terminators disabled. Onboard terminators enabled.

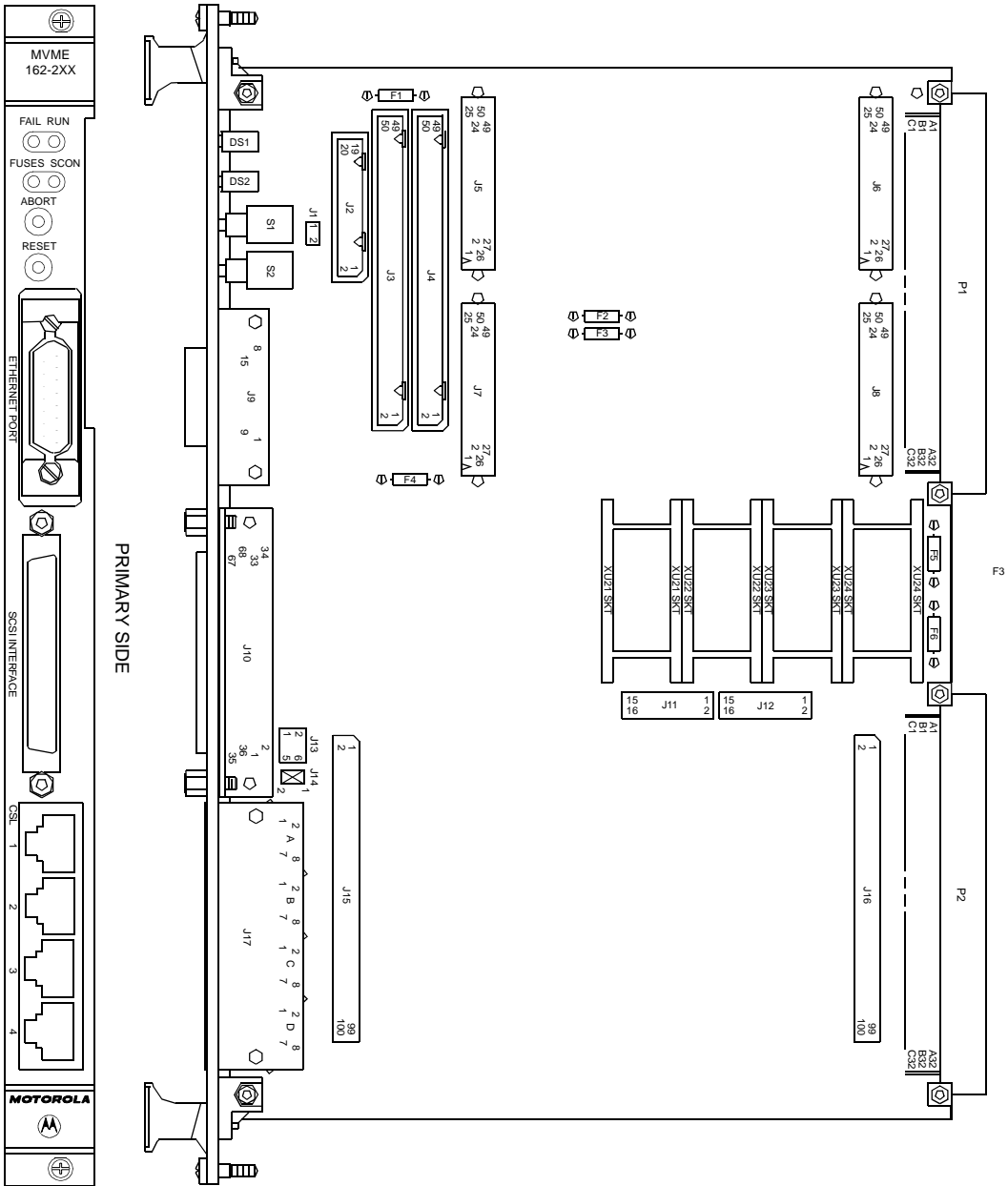


Figure 1-2. MVME162LX 200/300 Series Board Layout

## 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), and 2) the mapping of onboard resources as viewed by external masters (VMEbus memory map).

The memory and I/O maps which are described in the next three tables are correct for all local bus masters. There is some address translation capability in the VMEchip2. This allows multiple MVME162LXs 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 memory map of devices that respond to the normal address range is shown in the following tables. The normal address range is defined by the Transfer Type (TT) signals on the local bus. On the MVME162LX 200/300 Series, Transfer Types 0, 1, and 2 define the normal address range. [Table 1-4 on page 1-15](#) 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 MC68xx040 MMU. The onboard I/O space must be marked cache inhibit and serialized in its page table. [Table 1-5 on page 1-17](#) further defines the map for the local I/O devices.



**Table 1-4. Local Bus Memory Map**

Address Range	Devices Accessed	Port Width	Size	Software Cache Inhibit	Notes
Programmable	DRAM on Parity Mezzanine	D32	1MB-4MB	N	2
Programmable	DRAM on ECC Mezzanine	D32	4-32MB	N	2
Programmable	On-Board SRAM	D32	128 KB	N	2, 7
Programmable	SRAM on Mezzanine	D32	2 MB	N	2, 7
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
\$FF800000-\$FF9FFFFFFF	Flash/EPROM	D32	2 MB	N	1, 5, 8
\$FFA00000-\$FFBFFFFFFF	EPROM/Flash	D32	2 MB	N	5, 8
\$FFC00000-\$FFDFFFFFFF	Not Decoded	D32	2 MB	N	7
\$FFE00000-\$FFE1FFFF	On-Board SRAM Default	D32	128 KB	N	7
\$FFE80000-\$FFEFFFFFFF	Not Decoded	--	512 KB	N	6

**Table 1-4. Local Bus Memory Map (Continued)**

Address Range	Devices Accessed	Port Width	Size	Software Cache Inhibit	Notes
\$FFF00000-\$FFFEFFFF	Local I/O Devices (Refer to next table)	D32-D8	878 KB	Y	3
\$FFFF0000-\$FFFFFFF	VMEbus A16	D32/D16	64 KB	?	2, 4

## NOTES:

1. Devices mapped at \$FFF80000-\$FFF9FFFF also appear at \$00000000-\$001FFFFFF when the ROM0 bit in the MCchip EPROM control register is high (ROM0=1). ROM0 is set to 1 after each reset. The ROM0 bit must be cleared before other resources (Parity DRAM or SRAM) can be mapped in this range (\$00000000 - \$001FFFFFF).

The EPROM/Flash memory map is also controlled by the EPROM size and by control bit V11 in the MCchip ASIC. Refer to the EPROM/Flash configuration tables in your board's *Installation and Use Guide* for further details.

2. This area is user-programmable. The Parity DRAM and SRAM decoder is programmed in the MCchip, the local-to-VMEbus decoders are programmed in the VMEchip2, and the IP memory space is programmed in the IPIC.
3. Size is approximate.
4. Cache inhibit depends on the devices in the area mapped.
5. The EPROM and Flash are dynamically sized by the MCchip ASIC from an 8-bit private bus to the 32-bit MPU local bus.
6. These areas are not decoded unless one of the programmable decoders is initialized to decode this space. If they are not decoded and the local timer is enabled, an access to this address range will generate a local bus timeout.
7. SRAM is 128 KB when no SRAM mezzanine is present. With an SRAM mezzanine board, the SRAM size is 2 MB and the on-board 128 KB SRAM is disabled.
8. The 2MB "Size" given for the Flash/EPROM and the EPROM/Flash rows indicates the size of their address ranges. 1MB Flash is actually available as a physically present chip for each MVME162LX 200/300 Series board, and each board has sockets to accommodate 1MB, while its PROM is shipped as a .5MB part.

Table 1-5 on page 1-17 focuses on the “Local I/O Devices” portion of the local bus main memory map.

**Note** The IPIC chip on the MVME162LX 200/300 Series supports up to four IndustryPack (IP) interfaces, designated IP\_a through IP\_d. The 200/300 Series board itself accommodates two IPs: IP\_a and IP\_b. In the following map, the segments applicable to IP\_c and IP\_d are not used in the MVME162LX 200/300 Series.

**Table 1-5. Local I/O Devices Memory Map**

Address Range	Devices Accessed	Port Width	Size	Notes
\$FFF00000 - \$FFF3FFFF	Reserved	--	256 KB	4
\$FFF40000 - \$FFF400FF	VMEchip2 (LCSR)	D32	256 B	1, 3
\$FFF40100 - \$FFF401FF	VMEchip2 (GCSR)	D32-D8	256 B	1, 3
\$FFF40200 - \$FFF40FFF	Reserved	--	3.5 KB	4, 5
\$FFF41000 - \$FFF41FFF	Reserved	--	4 KB	4
\$FFF42000 - \$FFF42FFF	MCchip	D32-D8	4 KB	1
\$FFF43000 - \$FFF430FF	MCECC #1	D8	256 B	1, 9
\$FFF43100 - \$FFF431FF	MCECC #2	D8	256 B	1, 9
\$FFF43200 - \$FFF43FFF	MCECCs (repeated)	--	3.5 KB	1, 5, 9
\$FFF44000 - \$FFF44FFF	Reserved	--	8 KB	4
\$FFF45000 - \$FFF45800	SCC #1 (Z85230)	D8	2 KB	1, 2
\$FFF45801 - \$FFF45FFF	SCC #2 (Z85230)	D8	2 KB	1, 2
\$FFF46000 - \$FFF46FFF	LAN (82596CA)	D32	4 KB	1, 6
\$FFF47000 - \$FFF47FFF	SCSI (53C710)	D32-D8	4 KB	1
\$FFF48000 - \$FFF57FFF	Reserved	--	64 KB	4
\$FFF58000 - \$FFF5807F	IPIC IP_a I/O	D16	128 B	1
\$FFF58080 - \$FFF580FF	IPIC IP_a ID	D16	128 B	1

**Table 1-5. Local I/O Devices Memory Map (Continued)**

Address Range	Devices Accessed	Port Width	Size	Notes
\$FFF58100 - \$FFF5817F	IPIC IP_b I/O	D16	128 B	1
\$FFF58180 - \$FFF581FF	IPIC IP_b ID Read	D16	128 B	1
\$FFF58200 - \$FFF5827F	IPIC IP_c I/O	D16	128 B	8
\$FFF58280 - \$FFF582FF	IPIC IP_c ID	D16	128 B	8
\$FFF58300 - \$FFF5837F	IPIC IP_d I/O	D16	128 B	8
\$FFF58380 - \$FFF583FF	IPIC IP_d ID Read	D16	128 B	8
\$FFF58400 - \$FFF584FF	IPIC IP_ab I/O	D32-D16	256 B	1
\$FFF58500 - \$FFF585FF	IPIC IP_cd I/O	D32-D16	256 B	8
\$FFF58600 - \$FFF586FF	IPIC IP_ab I/O Repeated	D32-D16	256 B	1
\$FFF58700 - \$FFF587FF	IPIC IP_cd I/O Repeated	D32-D16	256 B	8
\$FFF58800 - \$FFF5887F	Reserved	--	128 B	1
\$FFF58880 - \$FFF588FF	Reserved	--	128 B	1
\$FFF58900 - \$FFF5897F	Reserved	--	128 B	1
\$FFF58980 - \$FFF589FF	Reserved	--	128 B	1
\$FFF58A00 - \$FFF58A7F	Reserved	--	128 B	1
\$FFF58A80 - \$FFF58AFF	Reserved	--	128 B	1
\$FFF58B00 - \$FFF58B7F	Reserved	--	128 B	1
\$FFF58B80 - \$FFF58BFF	Reserved	--	128 B	1
\$FFF58C00 - \$FFF58CFF	Reserved	--	256 B	1
\$FFF58D00 - \$FFF58DFE	Reserved	--	256 B	1
\$FFF58E00 - \$FFF58EFF	Reserved	--	256 B	1
\$FFF58F00 - \$FFF58FFF	Reserved	--	256 B	1
\$FFFBC000 - \$FFFBC01F	IPIC Registers	D32-D8	2 KB	1
\$FFFBC800 - \$FFFBC81F	Reserved	--	2 KB	1
\$FFFBD000 - \$FFFBDFFF	Reserved	--	12 KB	4

**Table 1-5. Local I/O Devices Memory Map (Continued)**

Address Range	Devices Accessed	Port Width	Size	Notes
\$FFFC0000 - \$FFFC7FFF	MK48T08 (BBRAM, TOD Clock)	D32-D8	32 KB	1
\$FFFC8000 - \$FFFCBFFF	MK48T08 & Disable Flash writes	D32-D8	16 KB	1, 7
\$FFFC0000 - \$FFFCFFFF	MK48T08 & Enable Flash writes	D32-D8	16 KB	1, 7
\$FFFD0000 - \$FFFEFFFF	Reserved	- -	128 KB	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 an MCchip private data bus. Byte access is required.
3. Writes to the LCSR in the VMEchip2 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 does not return an 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. Refer to the Flash and EPROM Interface section in the MCchip description in Chapter 3.
8. Not used.
9. To use this area, the ECC mezzanine board must be installed. If it is not installed, no acknowledge signal is returned; if the local bus timer is enabled, the access times out and is terminated by a TEA signal.

## Detailed I/O Memory Maps

Tables 1-7 through 1-16 give the detailed memory maps for the:

VMEchip2	<a href="#">Table 1-6 on page 1-22</a>
MCchip	<a href="#">Table 1-7 on page 1-27</a>
IPIC interface chip	<a href="#">Table 1-8 on page 1-29</a>
IPIC chip -- Control & Status Registers	<a href="#">Table 1-9 on page 1-30</a>
MCECC chip	<a href="#">Table 1-10 on page 1-32</a>
Z85230 Serial Communications Controller (SCC) chip Register addresses	<a href="#">Table 1-11 on page 1-33</a>
82596CA Ethernet LAN chip	<a href="#">Table 1-12 on page 1-34</a>
53C710 SCSI chip	<a href="#">Table 1-13 on page 1-34</a>
MK48T08 BBRAM/TOD clock	<a href="#">Table 1-14 on page 1-36</a>
BBRAM configuration area	<a href="#">Table 1-15 on page 1-36</a>
TOD clock	<a href="#">Table 1-16 on page 1-38</a>

Unlike the MVME162LX 200/300 Series models described in this Guide, boards in the 700/800 Series have MC2chip and IP2 chip ASICs in place of the MC and IPIC chips. As a result of this upgrade, I/O Memory Maps for these two ASICs differ somewhat from their predecessors' in this Guide (to compare details see the *MVME162LX 700/800 Series Programmer's Guide*).



### Caution

The upgrade also results in some incompatibilities between the two series in board-level software; for details see the *MVME162LX 700/800 Series Programmer's Guide*, Chapter 1.

**Note** For further information on the vendor-supplied chips and other third-party board components, see the section entitled "Applicable Non-Motorola Documentation" in the *Related Documentation* Appendix.

[This page left intentionally blank.]

**Table 1-6. VMEchip2 Memory Map (Sheet 1 of 3)**

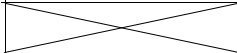
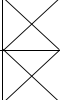

VMEchip2 LCSR Base Address = \$FFF40000

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	X				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	X										WAIT RMW	ROM ZERO	DMA TB SNP MODE	SRAM SPEED				
34	X																	
38	DMA CONTROLLER																	
3C	DMA CONTROLLER																	
40	DMA CONTROLLER																	
44	DMA CONTROLLER																	
48	X	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 1-6. 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				✗		CLR OVF 2	COC EN 2	TIC EN 2	OVERFLOW COUNTER 1				✗		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	✗									
CLR IRQ 15	CLR IRQ 14	CLR IRQ 13	CLR IRQ 12	CLR IRQ 11	CLR IRQ 10	CLR IRQ 9	CLR IRQ 8										
✗	P ERROR IRQ LEVEL			✗	IRQ1E IRQ LEVEL			✗	TIC TIMER 2 IRQ LEVEL			✗	TIC TIMER 1 IRQ LEVEL				
✗	SIG 1 IRQ LEVEL			✗	SIG 0 IRQ LEVEL			✗	LM 1 IRQ LEVEL			✗	LM 0 IRQ LEVEL				
✗	SW3 IRQ LEVEL			✗	SW2 IRQ LEVEL			✗	SW1 IRQ LEVEL			✗	SW0 IRQ LEVEL				
✗	VME IRQ 4 IRQ LEVEL			✗	VMEB IRQ 3 IRQ LEVEL			✗	VME IRQ 2 IRQ LEVEL			✗	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 1-6. 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	LM3	LM2	LM1	LM0	SIG 3	SIG 2	SIG 1	SIG 0	RST	ISF	BF	SC ON	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															

**Note** The DRAM referred to in the following map (Table 1-7) is all parity-protected, the type of DRAM supplied with many of the boards in the MVME162LX 200/300 Series. The chip registers outlined by this map are covered in the MCchip chapter of this Guide. For the boards with ECC-protected DRAM see below, Table 1-10, and the MCECC chip chapter of this Guide.

**Table 1-7. MCchip Register Map**

MCchip Base Address = \$FFF42000

Offset	D31-D24	D23-D16	D15-D8	D7-D0
\$00	MCchip ID	MCchip 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	Parity DRAM Space Base Address Register		SRAM Space Base Address Register	
\$24	Parity DRAM Space Size	Parity 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	MVME162 Version	SCSI Interrupt Control

**Table 1-7. MCchip Register Map (Continued)**

MCchip Base Address = \$FFF42000

\$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	PROM Access Time Control	Flash Access Time Control	ABORT Switch Interrupt Control
\$44	RESET Switch Control	Watchdog Timer Control	Access & Watchdog Time Base Select	(reserved)
\$48	Parity DRAM Control	(reserved)	MPU Status	(reserved)
\$4C	32-bit Prescaler Count Register			

The memory map [Table 1-8 on page 1-29](#) includes all devices selected by the IPIC map decoder.

**Note** The IPIC chip on the MVME162LX 200/300 Series supports up to four IndustryPack (IP) interfaces, designated IP\_a through IP\_d. The 200/300 Series itself accommodates two IPs: IP\_a and IP\_b. In the maps that follow, the segments applicable to IP\_c and IP\_d are not used in the MVME162LX 200/300 Series.

**Table 1-8. IPIC 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

**Table 1-8. IPIC Overall Memory Map (Continued)**

Address Range	Selected Device	Port Width	Size
\$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-\$FFFBC01F	Control/Status Registers	D32-D8	32B

A summary of the IPIC Control and Status Registers (CSRs) is shown in [Table 1-9](#). The CSRs can be accessed as bytes, words, or longwords. They should not be accessed as lines. They are shown in the table as bytes.

**Table 1-9. IPIC Memory Map - Control and Status Registers**

IPIC Chip 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	0	0
\$02	RESERVED	0	0	0	0	0	0	0	0
\$03	RESERVED	0	0	0	0	0	0	0	0
\$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



**Table 1-9. IPIC Memory Map - Control and Status Registers**IPIC Chip Base Address = \$FFFBC000 **(Continued)**

\$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	0	a_MEN
\$19	IP_b GENERAL CONTROL	b_ERR	0	b_RT1	b_RT0	b_WIDTH1	b_WIDTH0	0	b_MEN
\$1A	IP_c GENERAL CONTROL	c_ERR	0	c_RT1	c_RT0	c_WIDTH1	c_WIDTH0	0	c_MEN
\$1B	IP_d GENERAL CONTROL	d_ERR	0	d_RT1	d_RT0	d_WIDTH1	d_WIDTH0	0	d_MEN
\$1C	RESERVED	0	0	0	0	0	0	0	0
\$1D	RESERVED	0	0	0	0	0	0	0	0
\$1E	RESERVED	0	0	0	0	0	0	0	0
\$1F	IP RESET	0	0	0	0	0	0	0	RES

**Note** The DRAM referred to in the following map (Table 1-10) is all ECC-protected, the type of DRAM supplied with most of the boards in the MVME162LX 200/300 Series. The chip registers outlined by this map are covered in the MCECC chapter of this Guide. For the boards with parity-protected DRAM see above, Table 1-7, and the MCchip chapter of this Guide.

**Table 1-10. MCECC Internal Register Memory Map**

MCECC Base Address = \$FFF43000 (1st); \$FFF43100 (2nd)

Register Offset	Register Name	Register Bit Names							
		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	REV4	REV3	REV2	REV1	REV0
\$08	MEM CONFIG			FSTRD	1	0	MSIZ2	MSIZ1	MSIZ0
\$0C	DUMMY 0	0	0	0	0	0	0	0	0
\$10	DUMMY 1	0	0	0	0	0	0	0	0
\$14	BASE ADDRESS	BAD31	BAD30	BAD29	BAD28	BAD27	BAD26	BAD25	BAD24
\$18	DRAM CNTRL	BAD23	BAD22	RWB5	SWAIT	RWB3	NCEIEN	NCEBEN	RAMEN
\$1C	BCLK FREQ	BCK7	BCK6	BCK5	BCK4	BCK3	BCK2	BCK1	BCK0
\$20	DATA CNTRL	0	0	DERC	ZFILL	RWCKB	0	0	0
\$24	SCRUB CNTRL	RACODE	RADATA	HITDIS	SCRB	SCRBEN	0	SBEIEN	IDIS
\$28	SCRUB PERIOD	SBPD15	SBPD14	SBPD13	SBPD12	SBPD11	SBPD10	SBPD9	SBPD8
\$2C	SCRUB PERIOD	SBPD7	SBPD6	SBPD5	SBPD4	SBPD3	SBPD2	SBPD1	SBPD0
\$30	CHIP PRESCALE	CPS7	CPS6	CPS5	CPS4	CPS3	CPS2	CPS1	CPS0
\$34	SCRUB TIME ON/OFF	SRDIS	0	STON2	STON1	STON0	STOFF2	STOFF1	STOFF0
\$38	SCRUB PRESCALE	0	0	SPS21	SPS20	SPS19	SPS18	SPS17	SPS16
\$3C	SCRUB PRESCALE	SPS15	SPS14	SPS13	SPS12	SPS11	SPS10	SPS9	SPS8
\$40	SCRUB PRESCALE	SPS7	SPS6	SPS5	SPS4	SPS3	SPS2	SPS1	SPS0
\$44	SCRUB TIMER	ST15	ST14	ST13	ST12	ST11	ST10	ST9	ST8
\$48	SCRUB TIMER	ST7	ST6	ST5	ST4	ST3	ST2	ST1	ST0
\$4C	SCRUB ADDR CNTRL	0	0	0	0	0	SAC26	SAC25	SAC24
\$50	SCRUB ADDR CNTRL	SAC23	SAC22	SAC21	SAC20	SAC19	SAC18	SAC17	SAC16

**Table 1-10. MCECC Internal Register Memory Map (Continued)**

MCECC Base Address = \$FFF43000 (1st); \$FFF43100 (2nd)

Register Offset	Register Name	Register Bit Names							
		D31	D30	D29	D28	D27	D26	D25	D24
\$54	SCRUB ADDR CNTRL	SAC15	SAC14	SAC13	SAC12	SAC11	SAC10	SAC9	SAC8
\$58	SCRUB ADDR CNTRL	SAC7	SAC6	SAC5	SAC4	0	0	0	0
\$5C	ERROR LOGGER	ERRLOG	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	0	0	0	0
\$70	ERROR SYNDROME	S7	S6	S5	S4	S3	S2	S1	S0
\$74	DEFAULTS1	WRHDIS	STATCOL	FSTRD	SELI1	SELI0	RSIZ2	RSIZ1	RSIZ0
\$78	DEFAULTS2	FRC_OPN	XY_FLIP	REFDIS	TVECT	NOCACHE	RESST2	RESST1	RESST0

**Table 1-11. 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 1-12. 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)					
<p><b>Notes</b></p> <ol style="list-style-type: none"> <li>1. Refer to the MPU Port and MPU Channel Attention registers in Chapter 3.</li> <li>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.</li> </ol>						

**Table 1-13. 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

**Table 1-13. 53C710 SCSI Memory Map (Continued)****Base Address is \$FFF47000**

<b>Big Endian Mode</b>	<b>53C710 Register Address Map</b>				<b>SCRIPTs Mode and Little Endian Mode</b>
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
<b>Note</b> Accesses may be 8-bit or 32-bit, but not 16-bit.					

## BBRAM/TOD Clock Memory Map

The MK48T08 BBRAM (also called Non-Volatile RAM or NVRAM) is divided into six areas as shown in [Table 1-14 on page 1-36](#). 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 MVME162LX board debugger (MVME162Bug). The fifth area, detailed in [Table 1-15 on page 1-36](#), is the configuration area. The sixth area, the TOD clock, detailed in [Table 1-16 on page 1-38](#), is defined by the chip hardware.

**Table 1-14. MK48T08 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 1-15. 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

**Table 1-15. BBRAM Configuration Area Memory Map (Continued)**

Address Range	Description	Size (Bytes)
\$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
\$FFFC1FD6 - \$FFFC1FDD	IP_d Board Serial Number	8
4FFFC1FDE - \$FFFC1FE5	IP_d Board PWB	8
\$FFFC1FE6 - \$FFFC1FF6	Reserved	65
\$FFFC1FF7	Checksum	1
<b>Note</b> IP_c and IP_d are not used on MVME162LX 200/300 Series modules.		

**Table 1-16. TOD Clock Memory Map**

Address	Data Bits								Function		
	D7	D6	D5	D4	D3	D2	D1	D0			
\$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	
<b>Notes</b>											
W = Write Bit R = Read Bit S = Signbit					ST = Stop Bit FT = Frequency Test x = Unused						

## BBRAM/TOD Clock Configuration Bytes

The data structure of the configuration bytes starts at \$FFFC1EF8 and is as follows.

```
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 mem_pwb[8];
    char mem_serial[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[65];
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 MVME162LX 200/300 Series board with MC68040, SCSI, Ethernet, 4 MB Parity DRAM, and 128 KB SRAM, this field contains:

```
MVME162-223
```

(The 11 characters are followed by five 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 MVME162LX 200/300 Series board with MC68040, SCSI, Ethernet, 4 MB Parity DRAM, and 128 KB SRAM, at revision A, the PWA field contains:

```
01-W3866B01A
```

(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 25.00 MHz board, this field contains:

```
2500
```

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. This does *not* include the 01-w prefix. For example, for a 4 MB Parity DRAM mezzanine at revision A, the PWB field contains:

```
3913B01A
```

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.
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 a.

14. Eight bytes are reserved for the serial number, in ASCII, assigned to the optional first IndustryPack a.
15. Eight bytes are reserved for the printed wiring board (PWB) number assigned to the optional first IndustryPack a.
16. Eight bytes are reserved for the board identifier, in ASCII, assigned to the optional second IndustryPack b.
17. Eight bytes are reserved for the serial number, in ASCII, assigned to the optional second IndustryPack b.
18. Eight bytes are reserved for the printed wiring board (PWB) number assigned to the optional second IndustryPack b.
19. Eight bytes are reserved for the board identifier, in ASCII, assigned to the optional third IndustryPack c.
20. Eight bytes are reserved for the serial number, in ASCII, assigned to the optional third IndustryPack c.
21. Eight bytes are reserved for the printed wiring board (PWB) number assigned to the optional third IndustryPack c.
22. Eight bytes are reserved for the board identifier, in ASCII, assigned to the optional fourth IndustryPack d.
23. Eight bytes are reserved for the serial number, in ASCII, assigned to the optional fourth IndustryPack d.
24. Eight bytes are reserved for the printed wiring board (PWB) number assigned to the optional fourth IndustryPack d.
25. Growth space (65 bytes) is reserved. This pads the structure to an even 256 bytes.
26. The final one byte of the area is reserved for a checksum (as defined in the *MVME162Bug Debugging Package 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.

### **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 the MVME162LX 200/300 Series responds to.

### **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 MVME162LX 200/300 Series is a complex board that interfaces to the VMEbus and 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.



## Caution

If upgrading from a 200/300 Series 162LX to a 700/800 Series board, you must consult the *MVME162LX 700/800 Series Programmer's Guide*, Chapter 1, about several board-level changes affecting software compatibility.

## Interrupts

The MC68040 uses hardware-vectored interrupts.

Most interrupt sources are level and base vector programmable. Interrupt vectors from the MCchip 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 MCchip having the highest priority, those from the IPIC having the next highest priority, and interrupt sources from the VMEchip2 having the lowest priority. Refer to Appendix A for an example of interrupt usage.

The MCchip, IPIC, and VMEchip2 ASICs are used to implement the multi-level MC680x0 interrupt architecture. A PLD is used to combine the individual IPLx signals from each ASIC.

## Cache Coherency

The MC68040 has the ability to watch 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. When snooping is enabled, the MPU can source data and invalidate cache entries as required by the current cycle. The MPU can not watch VMEbus cycles which do not access the local bus on the MVME162LX 200/300 Series.

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 offboard 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 timeout occurs, a DRAM parity error occurs and parity checking is enabled (boards with Parity DRAM only), or a VME bus error occurs during a VMEbus access.

The devices on the MVME162LX 200/300 Series that are able to assert a local bus error are described below.

### Local Bus Timeout

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

### VMEbus Access Timeout

A VMEbus Access Timeout 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: an initialization routine samples to see if a device is present on the VMEbus and it is not, software accesses a non-existent 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, or a VMEbus slave reports an access error (such as parity error on boards with Parity DRAM).

### Local DRAM Parity Error

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

**Note** This only applies to 200/300 Series boards with Parity 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 error 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. The software must be written to deal with this.

## Description of Error Conditions on the MVME162LX 200/300 Series

This section lists the various error conditions that are reported by the MVME162LX 200/300 Series 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

**Description:**

A Parity DRAM parity error.

**MPU Notification:**

TEA is asserted during an MPU Parity DRAM access.

**Status:**

Bit 9 of the MPU Status and DMA Interrupt Count Register in the VMEchip2 at 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 MVME162LX 200/300 Series 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 at 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) at 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**

**Description:**

Parity error while the DMAC was reading Parity 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

**Description:**

Parity error while the LANCE was reading Parity DRAM

**MPU Notification:**

MCchip Interrupt (LAN ERROR IRQ)

**Status:**

MCchip LAN Error Status Register (\$FFF42028)

**Comments:**

The LANCE has no ability to respond to TEA so the error interrupt and status are provided in the MCchip. Control for the interrupt is in the MCchip LAN Error Interrupt Control Register (\$FFF4202B).

## LAN Offboard Error

**Description:**

Error encountered while the LANCE was attempting to go to the VMEbus.

**MPU Notification:**

MCchip Interrupt (LAN ERROR IRQ)

**Status:**

MCchip LAN Error Status Register (\$FFF42028)

**Comments:**

The LANCE has no ability to respond to TEA so the error interrupt and status are provided in the MCchip. Control for the interrupt is in the MCchip LAN Error Interrupt Control Register (\$FFF4202B).

## LAN LTO Error

**Description:**

Local Bus Timeout occurred while the LANCE was local bus master.

**MPU Notification:**

MCchip Interrupt (LAN ERROR IRQ)

**Status:**

MCchip LAN Error Status Register (\$FFF42028)

**Comments:**

The LANCE has no ability to respond to TEA so the error interrupt and status are provided in the MCchip. Control for the interrupt is in the MCchip LAN Error Interrupt Control Register (\$FFF4202B).

## SCSI Parity Error

Description:

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

MPU Notification:

53C710 Interrupt

Status:

53C710 DMA Status Register

53C710 DMA Interrupt Status Register

MCchip SCSI Error Status Register (\$FFF4202C)

Comments:

53C710 interrupt enables are controlled in the 53C710 and in the MCchip 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

MCchip SCSI Error Status Register (\$FFF4202C)

Comments:

53C710 interrupt enables are controlled in the 53C710 and in the MCchip 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

MCchip SCSI Error Status Register (\$FFF4202C)

Comments:

53C710 interrupt enables are controlled in the 53C710 and in the MCchip SCSI Interrupt Control Register (\$FFF4202F).

## Example of the Proper Use of Bus Timers

In this example, the use of the bus timers is illustrated by describing the sequence of events when the MPU on one MVME162LX 200/300 Series accesses the local bus memory on another MVME162LX 200/300 Series using the VMEbus. In this scenario there are three bus timers involved. These are the local bus timer, the VMEbus access timer, and 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. Normally these timers should be set to quite different values.

An example of one MVME162LX 200/300 Series board accessing another 200/300 Series board illustrates the use of these timers.

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 local bus. Then it 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 are normally very short, such as the time to access onboard memory. Therefore, it is recommended this timer be set to a small value, such as 256  $\mu$ sec.

The next timer to take over when one MVME162LX 200/300 Series board 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 when VMEbus mastership has been granted. Because we have found in the past that some VME systems can become very busy, we recommend this timeout be set at 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 MVME162LX 200/300 Series access to another 200/300 Series can complete, however, the VMEchip2 on the accessed MVME162LX must decode a slave access and request the local bus of the second MVME162LX. When the local bus is granted (any in-process onboard transfers have completed) then the local bus timer of the accessed MVME162LX 200/300 Series 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.



## MVME162LX 200/300 Series MC68040 Indivisible Cycles

The MC68040 performs 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 offboard resources. The MVME162LX 200/300 Series does not fully support all RMW operations in all possible cases.

The MVME162LX 200/300 Series makes the following assumptions and supports a limited subset of RMW instructions. The 200/300 Series supports single-address RMW cycles caused by TAS and CAS instructions. Because it is not possible to tell if the MC68040 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 MVME162LX 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. As a result of 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, although only Industry Packs a and b are used by MVME162LX 200/300 Series boards.



---

## Introduction

This chapter defines the VMEchip2, local bus to VMEbus interface chip.

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

## Summary of Major 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:
    - Software-enabled FAIR request mode.
    - Software-configured release modes:
      - Release-When-Done (RWD).
      - Release-On-Request (ROR).
    - Software-configured BR0\*-BR3\* request levels.

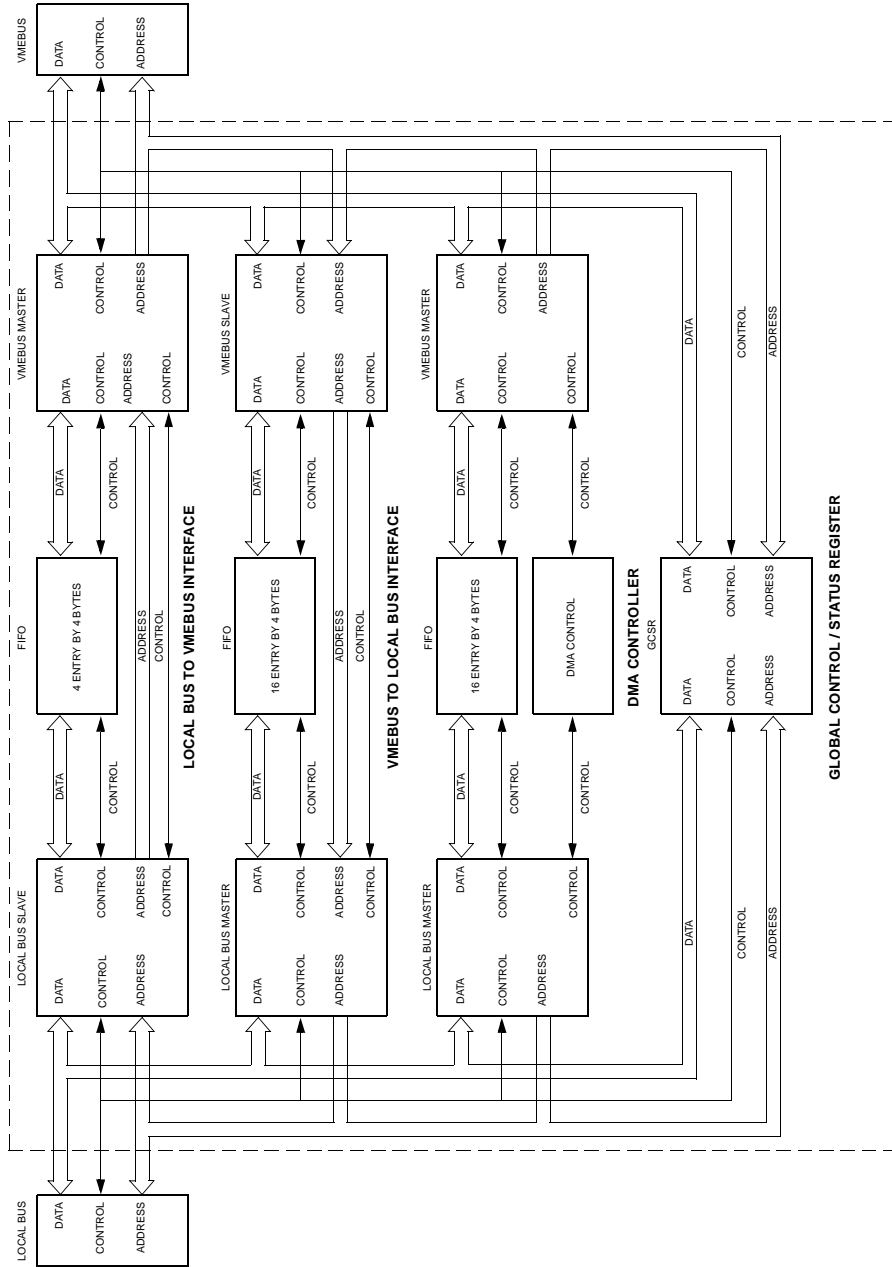
- VMEbus Bus to Local Bus Interface:
  - Programmable VMEbus map decoder.
  - Programmable AM decoder.
  - Programmable local bus snoop enable.
  - Simple VMEbus to local bus address translation.
  - 8-bit, 16-bit and 32-bit VMEbus data width.
  - 8-bit, 16-bit and 32-bit block transfer.
  - Standard and extended VMEbus addressing.
  - Software-enabled write posting mode.
  - Write post buffer (17 four-bytes in BLT mode, 2 four-bytes in non-BLT mode).
  - An eight four-byte read ahead buffer (BLT mode only).
- 32-Bit Local - 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.
  - A 16 four-byte FIFO data buffer.
  - Supports up to 4 GB of data per DMA request.
  - Automatically adjusts transfer size to optimize bus utilization.
  - DMA complete interrupt.
  - DMAC command chaining is supported by a singly-linked list of DMA commands.
  - VMEbus DMA controller requester:
    - Software-enabled FAIR request modes.
    - Software-configured release modes:
      - Release-On-Request (ROR).
      - Release-On-End-Of-Data (ROEOD).
    - Software-configured BR0-BR3 request levels.
    - Software enabled bus-tenure timer.

- VMEbus Interrupter:
  - Software-configured IRQ1-IRQ7 interrupt request level.
  - 8-bit software-programmed status/ID register.
- 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.
  - Four 16-bit dual-ported general purpose registers.
- Interrupt Handler:
  - All interrupts are level-programmable.
  - All interrupts are maskable.
  - All interrupts provide a unique vector.
  - Software and external interrupts.
- Watchdog timer.
- Two 32-bit tick timers.

## Functional Blocks

This section provides an overview of the functions provided by the VMEchip2. See [Figure 2-1 on page 2-5](#) for a block diagram of the VMEchip2.

A detailed programming model for the local control and status registers (LCSR) is provided in the next main section, *LCSR Programming Model*, beginning on page [2-22](#). A detailed programming model for the global control and status registers (GCSR) is provided in the *GCSR Programming Model* section of this chapter.



1344 9403

Figure 2-1. VMEchip2 Block Diagram

## Local Bus to VMEbus Interface

### Interface Components

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

### Local Bus Slave

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 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.

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, and the other provides an A24/D16 space at \$F0000000 to \$F0FFFFFF and an A32/D16 space at \$F1000000 to \$FF7FFFFFFF.

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.

### **Write Post Buffer**

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 one byte, two-byte, four-byte, or one cache line (four four-bytes). Write posting should only be enabled when bus errors are not expected. If a bus error is returned on a write posted cycle, the local processor is interrupted, if the interrupt is enabled. 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 MC68040 block transfer capability is not compatible with the VMEbus.

### **VMEbus Master**

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, and 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, and 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 send 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 the release-when-done (RWD) mode, the requester releases the bus when the above conditions are satisfied.
2. When the chip is configured in the 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 offboard 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 the non-block transfer mode and 16 deep in the 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 onchip buffer, significantly improving the response time of the slave in the block transfer mode.

The VMEchip2 includes an onchip 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 set each bank's address modifier codes and 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 be different 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 MC68040.

When snooping is enabled, the local bus master requests the cache controller in the MC68040 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 one 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 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 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-byte, double-byte, 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-byte, double-byte, or a mixture of both to end the transfer.

To optimize local bus use when the VMEbus is operating in the 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.

To allow 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, however the MVME162LX does not have any 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 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 MC68040 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 1 MHz 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 Timer

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 and 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 and when enabled in the free run mode, it increments every 1  $\mu$ s. 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 one to the overflow clear bit.

Tick timer one or two 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 the 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 one 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 (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 lock-up 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 timeout period. The timeout period can be set to 8, 64, or 256  $\mu$ 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 timeout period. This timer can be enabled or disabled under software control. The timeout period can be programmed for 8, 64, or 256  $\mu$ secs.

## 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 timeout, a VMEbus SYSRESET, or a control bit in the GCSR.

## Local Bus Interrupter and Interrupt Handler

There are 31 interrupt sources in the VMEchip2: VMEbus ACFAIL, ABORT switch, VMEbus SYSFAIL, write post bus error, external input, VMEbus IRQ1 edge-sensitive, VMEchip2 VMEbus interrupter acknowledge, tick timer 2-1, DMAC done, GCSR SIG3-0, GCSR location monitor 1-0, software interrupts 7-0, and VMEbus IRQ7-1. 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 IRQ7-1 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 MC68040 microprocessor.

## Global Control and Status Registers

The VMEchip2 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. 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. Byte, two-byte and four-byte read operations are permitted: however, byte and two-byte write operations are not permitted. Byte and two-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 5 lines.

- ❑ Line 1 is the base address of the register and the number of bits defined in the table.
- ❑ Line 2 shows the bits defined by this table.
- ❑ Line 3 defines the name of the register or the name of the bits in the register.
- ❑ Line 4 defines the operations possible on the register bits as follows:

**R** This bit is a read-only status bit.

**R/W** This bit is readable and writable.

**W/AC** This bit can be set and it is automatically cleared. This bit can also be read.

**C** Writing a one to this bit clears this bit or another bit. This bit reads zero.

**S** Writing a one to this bit sets this bit or another bit. This bit reads zero.



□ Line 5 defines the state of the bit following a reset as follows:

- P**      The bit is affected by powerup reset.
- S**      The bit is affected by SYSRESET.
- L**      The bit is affected by local reset.
- X**      The bit is not affected by reset.

A summary of the LCSR is given as Table 2-1.

**Table 2-1. VMEchip2 Memory Map - LCSR Summary (Sheet 1 of 2)**

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	<del>X</del>		<del>X</del>		ADDER	SNP	WP	SUP	USR	A32	A24	BLK D64	BLK	PRGM	DATA		
					2	2	2	2	2	2	2	2	2	2	2	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	<del>X</del>										WAIT RMW	ROM ZERO	DMA TB SNP MODE	SRAM SPEED			
34											<del>X</del>						
38	DMA CONTROLLER																
3C	DMA CONTROLLER																
40	DMA CONTROLLER																
44	DMA CONTROLLER																
48	<del>X</del>		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																
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 2-1. 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 slave map decoders described in this section are disabled by local reset, SYSRESET, or power-up reset. Caution must be used 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. The following procedure should be used to modify the map decoder registers. 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, starting and ending address registers should be programmed first, and then the map decoders should be enabled by programming the address modifier select registers.

The VMEbus to local bus interface allows offboard VMEbus masters access to local onboard 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 which allow the upper 16 bits of the local bus address to be provided by the address translation address register rather than the upper 16 bits of the VMEbus.

Each VMEbus slave map decoder has an address translation address register, an address translation select register, a start address register, an end address register, an address modifier select register, and an attribute register. The addresses and bit definitions of these registers are shown in the following tables.

A VMEbus 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. If the VMEbus address modifier codes indicate an A24 VMEbus address cycle, then the upper eight bits of the VMEbus address are forced to zero 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 zero.

The address translation address register and the address translation select register operate in the following way. If a bit in the address translation select register is set, then the corresponding local bus address line is driven from the corresponding bit in the address translation address register. If the bit is cleared in the address translation select register, then the corresponding local bus address line is driven from the corresponding VMEbus 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 Revision 1 of the VMEchip2 used on the MVME162LX 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 add. The adders are enable 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	<b>\$FFF40000</b> (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	<b>\$FFF40000</b> (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**

ADR/SIZ	<b>\$FFF40000</b> (16 bits of 32)		
BIT	31	. . .	16
NAME	Ending Address Register 2		
OPER	R/W		
RESET	0 PS		

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

**VMEbus Slave Starting Address Register 2**

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

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

### VMEbus Slave Address Translation Address Offset Register 1

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

**VMEbus Slave Address Translation Select Register 1**

ADR/SIZ	<b>\$FFF40008</b> (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.

<b>Segment Size</b>	<b>Address Translation Select Value</b>
64KB	FFFF
128KB	FFFE
256KB	FFFC
512KB	FFF8
1MB	FFF0
2MB	FFE0
4MB	FFC0
8MB	FF80
16MB	FF00
32MB	FE00
64MB	FC00
128MB	F800
256MB	F000
512MB	E000
1GB	C000
2GB	8000
4GB	0000

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

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

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

### VMEbus Slave Write Post and 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 are:

- 0 Snoop inhibited
- 1 Write - Sink data  
Read - Supply dirty data and leave dirty
- 2 Write - Invalidate  
Read - Supply dirty data and mark invalid
- 3 Snoop inhibited

**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)							
BIT	23	22	21	20	19	18	17	16
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 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 respond 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					ADDER1	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 are:

- 0 Snoop inhibited
- 1 Write - Sink data  
Read - Supply dirty data and leave dirty
- 2 Write - Invalidate  
Read - Supply dirty data and mark invalid
- 3 Snoop inhibited

**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	<b>\$FFF40010</b> (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 offboard 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 \$FFFFFFFF 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 \$FF7FFFFFFF.

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 in to 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 zero.

The address translation address register and the address translation select register operate in the following way. If a bit in the address translation select register is set, then the corresponding VMEbus address line is driven from the corresponding bit in the address translation address register. If the bit is cleared in the address translation select register, then the corresponding VMEbus address line is driven from the corresponding local bus address line. The most significant bit of the address translation select register corresponds to the most significant bit of 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 \$FFFFFFF 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 \$FOFFFFFF to the A24 map of the VMEbus and the address range \$F1000000 to \$FF7FFFFF 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. However, the map decoders may be programmed 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</b> (16 bits of 32)		
<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</b> (16 bits of 32)		
<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</b> (16 bits of 32)		
<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</b> (16 bits of 32)		
<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</b> (16 bits of 32)		
<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**

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

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

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

<b>ADR/SIZ</b>	<b>\$FFF40020</b> (16 bits of 32)		
<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</b> (16 bits of 32)		
<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</b> (16 bits of 32)		
<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**

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

<b>ADR/SIZ</b>	<b>\$FFF40028</b> (8 bits of 32)							
<b>BIT</b>	31	30	29	28	27	26	25	24
<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 fourth local bus to VMEbus bus map decoder.

**AM**

These bits define the VMEbus address modifier codes 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

ADR/SIZ	\$FFF40028 (8 bits of 32)							
BIT	23	22	21	20	19	18	17	16
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 third local bus to VMEbus bus map decoder.

- AM** These bits define the VMEbus address modifier codes 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**

<b>ADR/SIZ</b>	<b>\$FFF40028</b> (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 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 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

ADR/SIZ	\$FFF4002C (8 bits of 32)		
BIT	31	. . .	24
NAME	GCSR Group Address Register		
OPER	R/W		
RESET	\$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 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 MVME162LX 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: 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)							
					19	18	17	16
<b>NAME</b>					EN4	EN3	EN2	EN1
<b>OPER</b>					R/W	R/W	R/W	R/W
<b>RESET</b>					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 PSL	0 PS	0 PS	0 PS

This register controls the VMEbus short I/O map and the F page (\$F0000000 through \$FF7FFFFF) 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 thru \$FF7FFFFF) 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**

<b>ADR/SIZ</b>	<b>\$FFF4002C</b> (8 bits of 32)							
<b>BIT</b>	7	6	5	4	3	2	1	0
<b>NAME</b>	SIZE		BSPD			ASPD		
<b>OPER</b>	R/W		R/W			R/W		
<b>RESET</b>	0 PS		0 PS			0 PS		

This function is not used on the MVME162LX.

## 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 - VMEbus DMA controller (DMAC). The DMAC has two modes of operation: command chaining, and direct. In the 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 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. 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 the direct mode and by the DMAC in the command chaining mode.

Once the DMAC is enabled, the counter and control registers should not be modified by software. When the command chaining mode is used, the list of commands must be in local 32-bit memory and the entries must be four-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 next command address. The command packet format is shown in Table 2-2.

**Table 2-2. DMAC Command Table 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.

## PROM 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 MVME162LX.
- TBLSC**        These bits control the snoop signal lines on the local bus when the DMAC is table walking.
- 0        Snoop inhibited
  - 1        Write - Sink data  
          Read - Supply dirty data and leave dirty
  - 2        Write - Invalidate  
          Read - Supply dirty data and mark invalid
  - 3        Snoop inhibited
- ROM0**        This VMEchip2 bit is not used on the MVME162LX. Its function is performed by the ROM0 bit in the PROM Access Time Control Register in the MCchip. Refer to Chapter 3.
- WAIT RMW**    This function is not used on the MVME162LX.



## 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	LVRWD	LVREQL	
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.

**LVREQL** These bits define the VMEbus request level. The request is only changed when 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 rerequested 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 the release-when-done mode. When this bit is low, the requester operates in the release-on-request mode.

**LVFAIR** When this bit is high, the requester operates in the fair mode. When this bit is low, the requester does not operate in the fair mode. In the 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 the round robin mode. When this bit is low, the arbiter operates in the priority mode.

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

ADR/SIZ	<b>\$FFF40030</b> (8 bits of 32)							
BIT	7	6	5	4	3	2	1	0
NAME	DHALT	DEN	DTBL	DFAIR	DRELM		DRELQ	
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.

**DRELQ** These bits define the VMEbus request level for the DMAC requester. The request is only changed when 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 rerequested 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 the 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 the fair mode.

**DTBL** The DMAC operates in the direct mode when this bit is low, and it operates in the 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 the 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 the 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.
- 0 Snoop inhibited

- 1 Write - Sink data  
Read - Supply dirty data and leave dirty
- 2 Write - Invalidate  
Read - Supply dirty data and mark invalid
- 3 Snoop inhibited

**INTE**

This bit is used only in the command chaining mode and 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 the 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 the 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 the 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

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

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

### DMAC VMEbus Address Counter

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

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

### DMAC Byte Counter

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

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

### Table Address Counter

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

In the 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 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 and writing a 0 to this bit 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 IRQ1 signal line on the VMEbus.
  - 1 The output from tick timer 1 is connected to the IRQ1 signal line on the VMEbus.
  - 2 The IRQ1 signal from the interrupter is connected to the IRQ1 signal line on the VMEbus.
  - 3 The output from tick timer 2 is connected to the IRQ1 signal line on the VMEbus.

## VMEbus Interrupter Vector Register

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

This register controls the VMEbus interrupter vector.

## MPU Status and DMA Interrupt Count Register

ADR/SIZ	<b>\$FFF40048</b> (8 bits of 32)							
BIT	15	14	13	12	11	10	9	8
NAME	DMAIC				MCLR	MLBE	MLPE	MLOB
OPER	R				C	R	R	R
RESET	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 offboard. This bit is cleared by writing a one 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 one to the MCLR bit in this register.
- MLBE** When this bit is set, the MPU received a TEA and additional status was not provided. This bit is cleared by writing a one to the MCLR bit in this register.
- MCLR** Writing a one 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 the command chaining mode. If 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	<b>\$FFF40048</b> (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 and there were no errors or the DMAC has finished executing command because the halt bit was set. This bit is cleared when the DMAC is enabled.
- VME** When this bit is set, the DMAC received a VMEbus BERR during a data transfer. This bit is cleared when the DMAC is enabled.
- TBL** When this bit is set, the DMAC 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** When this bit is set, the DMAC received a TEA and the status indicated a local bus timeout. This bit is cleared when the DMAC is enabled.
- DLOB** When this bit is set, the DMAC received a TEA and the status indicated offboard. This bit is cleared when the DMAC is enabled.
- DLPE** When this bit is set, the DMAC received a TEA and the status indicated a parity error during a DRAM data transfer. This bit is cleared when the DMAC is enabled.
- DLBE** When this bit is set, the DMAC received a TEA and additional status was not provided. This bit is cleared when the DMAC is enabled.
- MLTO** When this bit is set, the MPU received a TEA and the status indicated a local bus timeout. This bit is cleared by a writing a one 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 Timeout 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 timeout timer.

**ARBTO** When this bit is high, the VMEbus grant timeout 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 re-arbitrates any pending requests.

## DMAC Ton/Toff Timers and VMEbus Global Timeout 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 timeout timer.

**VGTO** These bits define VMEbus global timeout 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 timeout 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 Timeout 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 timeout period:

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
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 timeout 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.

08 $\mu$ s
164 $\mu$ s
2256 $\mu$ s
3The timer is disabled.

<b>VATO</b>	These bits define the VMEbus access timeout 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
1	1 ms
2	32 ms
3	The timer is disabled.

### Prescaler Control Register

ADR/SIZ	<b>\$FFF4004C</b> (8 bits of 32)		
BIT	7	. . .	0
NAME	Prescaler Adjust		
OPER	R/W		
RESET	SDF 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 one MHz clock to the Tick timers.

$$\text{Prescaler register} = 256 - \text{B clock (MHz)}$$

For example, for operation at 20 MHz the prescaler value is \$EC, at 25 MHz it is \$E7, and at 33 MHz it is \$DF.

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.

$$\text{Tick timer clock} = \text{B clock} / (256 - \text{Prescaler Value})$$

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

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

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

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

ADR/SIZ	<b>\$FFF4005C</b> (32 bits)		
BIT	31	. . .	0
NAME	Tick timer 2 Counter		
OPER	R/W		
RESET	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	<b>\$FFF40060</b> (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 duplicated at the same bit location in the MCchip at location \$FFF42044. When this bit is high, or the duplicate bit in the MCchip 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 power-up reset status bit is cleared. This bit is always read zero.
- 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 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 timeout.
- 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 PS	0	0	0 P	0 PSL	0 PSL	0 PSL	0 PSL

- 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 timeout occurs, a SYSRESET or LRESET is generated. The WDS/L bit in this register selects the reset. When this bit is low, a watchdog timeout 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 timeout has occurred. When this status bit is low, a watchdog timeout has not occurred. This bit is cleared by writing a one to the WDCC bit in this register.
- WDCC** When this bit is set high, the watchdog counter is reset. The counter must be reset within the timeout period or a watchdog timeout occurs.
- WDCS** When this bit is set high, the watchdog timeout 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	<b>\$FFF40060</b> (8 bits [7 used] of 32)							
BIT	15	14	13	12	11	10	9	8
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 zero 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 one 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 one to the COVF bit.

## Tick Timer 1 Control Register

ADR/SIZ	\$FFF40060 (8 bits [7 used] of 32)							
BIT	7	6	5	4	31	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 zero 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 one 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 one 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 wish 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 for the software interrupts, a set bit. 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 is shown in [Table 2-3 on page 2-75](#).

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 zero. 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 enabled. The master interrupt enable (MIEN) bit must be set before the VMEchip2 can generate any interrupts. The MIEN bit is in the I/O Control Register 1.

**Table 2-3. Local Bus Interrupter Summary**

Interrupt	Vector	Priority for Simultaneous Interrupts
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	
DMAC	\$X6	
VMEbus Interrupter Acknowledge	\$X7	
Tick Timer 1	\$X8	
Tick Timer 2	\$X9	
VMEbus IRQ1 Edge-Sensitive	\$XA	
ECC memory double-bit error	\$XB	
VMEbus Master Write Post Error	\$XC	
VMEbus SYSFAIL	\$XD	
(Not used on MVME162LX)	\$XE	∅
VMEbus ACFAIL	\$XF	Highest

**NOTES:** X = The contents of vector base register 0.

Y = The contents of vector base register 1.

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)

ADR/SIZ	\$FFF40068 (8 bits of 32)							
BIT	31	30	29	28	27	26	25	24
NAME	ACF	AB	SYSF	MWP	PE	VIIE	TIC2	TIC1
OPER	R	R	R	R	R	R	R	R
RESET	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>	ECC memory double-bit error
<b>MWP</b>	VMEbus master write post error interrupt
<b>SYSF</b>	VMEbus SYSFAIL interrupt
<b>AB</b>	Not used on MVME162LX.
<b>ACF</b>	VMEbus ACFAIL interrupt

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

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

ADR/SIZ	<b>\$FFF40068</b> (8 bits of 32)							
BIT	15	14	13	12	11	10	9	8
NAME	SW7	SW6	SW5	SW4	SW3	SW2	SW1	SW0
OPER	R	R	R	R	R	R	R	R
RESET	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>SW0</b>	Software 0 interrupt
<b>SW1</b>	Software 1 interrupt
<b>SW2</b>	Software 2 interrupt
<b>SW3</b>	Software 3 interrupt
<b>SW4</b>	Software 4 interrupt
<b>SW5</b>	Software 5 interrupt
<b>SW6</b>	Software 6 interrupt
<b>SW7</b>	Software 7 interrupt



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

ADR/SIZ	<b>\$FFF40068</b> (8 bits of 32)							
BIT	7	6	5	4	3	2	1	0
NAME	SPARE	VME7	VME6	VME5	VME4	VME3	VME2	VME1
OPER	R	R	R	R	R	R	R	R
RESET	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>VME1</b>	VMEbus IRQ1 Interrupt
<b>VME2</b>	VMEbus IRQ2 Interrupt
<b>VME3</b>	VMEbus IRQ3 Interrupt
<b>VME4</b>	VMEbus IRQ4 Interrupt
<b>VME5</b>	VMEbus IRQ5 Interrupt
<b>VME6</b>	VMEbus IRQ6 Interrupt
<b>VME7</b>	VMEbus IRQ7 Interrupt
<b>SPARE</b>	This bit is not used.

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

ADR/SIZ	<b>\$FFF4006C</b> (8 bits of 32)							
BIT	31	30	29	28	27	26	25	24
NAME	EACF	EAB	ESYSF	EMWP	EPE	EVI1E	ETIC2	ETIC1
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 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 enabled.

<b>ETIC1</b>	Enable tick timer 1 interrupt
<b>ETIC2</b>	Enable tick timer 2 interrupt
<b>EVI1E</b>	Enable VMEbus IRQ1 edge-sensitive interrupt
<b>EPE</b>	Not used on MVME162LX.
<b>EMWP</b>	Enable VMEbus master write post error interrupt
<b>ESYSF</b>	Enable VMEbus SYSFAIL interrupt
<b>EAB</b>	Not used on MVME162LX.
<b>EACF</b>	Enable VMEbus ACFAIL interrupt

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

ADR/SIZ	\$FFF4006C (8-bits)							
BIT	23	22	21	20	19	18	17	16
NAME	EVIA	EDMA	ESIG3	ESIG2	ESIG1	ESIG0	ELM1	ELM0
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 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 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)**

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

<b>ESW0</b>	Enable software 0 interrupt
<b>ESW1</b>	Enable software 1 interrupt
<b>ESW2</b>	Enable software 2 interrupt
<b>ESW3</b>	Enable software 3 interrupt
<b>ESW4</b>	Enable software 4 interrupt
<b>ESW5</b>	Enable software 5 interrupt
<b>ESW6</b>	Enable software 6 interrupt
<b>ESW7</b>	Enable software 7 interrupt

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

ADR/SIZ	<b>\$FFF4006C</b> (8 bits of 32)							
BIT	7	6	5	4	3	2	1	0
NAME	SPARE	EIRQ7	EIRQ6	EIRQ5	EIRQ4	EIRQ3	EIRQ2	EIRQ1
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 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 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	<b>\$FFF40070</b> (8 bits of 32)							
BIT	15	14	13	12	11	10	9	8
NAME	SSW7	SSW6	SSW5	SSW4	SSW3	SSW2	SSW17	SSW07
OPER	S	S	S	S	S	S	S	S
RESET	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 one to it. The software interrupt set bits are:

<b>SSW0</b>	Set software 0 interrupt
<b>SSW1</b>	Set software 1 interrupt
<b>SSW2</b>	Set software 2 interrupt
<b>SSW3</b>	Set software 3 interrupt
<b>SSW4</b>	Set software 4 interrupt
<b>SSW5</b>	Set software 5 interrupt
<b>SSW6</b>	Set software 6 interrupt
<b>SSW7</b>	Set software 7 interrupt

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

ADR/SIZ	<b>\$FFF40074</b> (8 bits of 32)							
25	24							
NAME	CACF	CAB	CSYSF	CMWP	CPE	CVIIE	CTIC2	CTIC1
OPER	C	C	C	C	C	C	C	C
RESET	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 one to its clear bit. The clear bits are defined below.

<b>CTIC1</b>	Clear tick timer 1 interrupt
<b>CTIC2</b>	Clear tick timer 2 interrupt
<b>CVIIE</b>	Clear VMEbus IRQ1 edge-sensitive interrupt
<b>CPE</b>	Not used on MVME162LX.
<b>CMWP</b>	Clear VMEbus master write post error interrupt
<b>CSYSF</b>	Clear VMEbus SYSFAIL interrupt
<b>CAB</b>	Not used on MVME162LX.
<b>CACF</b>	Clear VMEbus ACFAIL interrupt

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

ADR/SIZ	<b>\$FFF40074</b> (8 bits of 32)							
BIT	23	22	21	20	19	18	17	16
NAME	CVIA	CDMA	CSIG3	CSIG2	CSIG1	CSIG0	CLM1	CLM0
OPER	C	C	C	C	C	C	C	C
RESET	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 one 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	<b>\$FFF40074</b> (8 bits of 32)							
BIT	15	14	13	12	11	10	9	8
NAME	CSW7	CSW6	CSW5	CSW4	CSW3	CSW2	CSW1	CSW0
OPER	C	C	C	C	C	C	C	C
RESET	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 one to its clear bit. The clear bits are:

<b>CSW0</b>	Clear software 0 interrupt
<b>CSW1</b>	Clear software 1 interrupt
<b>CSW2</b>	Clear software 2 interrupt
<b>CSW3</b>	Clear software 3 interrupt
<b>CSW4</b>	Clear software 4 interrupt
<b>CSW5</b>	Clear software 5 interrupt
<b>CSW6</b>	Clear software 6 interrupt
<b>CSW7</b>	Clear software 7 interrupt

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

ADR/SIZ	<b>\$FFF40078</b> (8 bits [6 used] of 32)							
BIT	31	30	29	28	27	26	25	24
NAME	ACF LEVEL				AB LEVEL			
OPER	R/W				R/W			
RESET	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 MVME162LX.

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

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

ADR/SIZ	<b>\$FFF40078</b> (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** These bits define the level of the ECC memory double-bit error interrupt.

**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	<b>\$FFF4007C</b> (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	<b>\$FFF4007C</b> (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	<b>\$FFF4007C</b> (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	<b>\$FFF4007C</b> (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	<b>\$FFF40080</b> (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	<b>\$FFF40080</b> (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	<b>\$FFF40084</b> (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 MVME162LX.

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

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

This register is used to define the level of the VMEbus IRQ5 interrupt and the VMEbus IRQ6 interrupt. The VMEbus level 5 (IRQ5) interrupt and the VMEbus level 6 (IRQ6) interrupt 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 LEVEL				VIRQ3 LEVEL			
OPER	R/W				R/W			
RESET	0 PSL				0 PSL			

This register is used to define the level of the VMEbus IRQ3 interrupt and the VMEbus IRQ4 interrupt. The VMEbus level 3 (IRQ3) interrupt and the VMEbus level 4 (IRQ4) interrupt 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 LEVEL				VIRQ1 LEVEL			
OPER	R/W				R/W			
RESET	0 PSL				0 PSL			

This register is used to define the level of the VMEbus IRQ1 interrupt and the VMEbus IRQ2 interrupt. The VMEbus level 1 (IRQ1) interrupt and the VMEbus level 2 (IRQ2) interrupt 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	<b>\$FFF40088</b> (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 2-3, Local Bus Interrupter Summary, earlier in this chapter, for further information.

A suggested setting for the Vector Base Register for the VMEchip2 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 2-3, and a Vector Base1 of \$70 corresponding to the "Y" in Table 2-3.

## 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	GPOEN 3	GPOEN 2	GPOEN 1	GPOEN 0
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) on the VMEchip2 ASIC.

- ❑ GPIO0 is logically connected to the fuse LED. For details refer to the Note under I/O Control Register 2's description.
- ❑ GPIO1-3 are each directly connected to the board's J2 pins 16-18, respectively. These signals are available for customer use.



GPIO0 should not be programmed as an output. Refer to the Note under I/O Control Register 2's description for details.

- 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 function is not used on the MVME162LX.
- 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

This register is a general purpose I/O control register.

Bits 8-11 reflect the status of the four General Purpose I/O pins (GPIO0-3).

- GPIOI0** When this bit is low, the GPIO0 pin is low. When this bit is high, the GPIO0 pin is high.
- GPIOI1** When this bit is low, the GPIO1 pin is low. When this bit is high, the GPIO1 pin is high.
- GPIOI2** When this bit is low, the GPIO2 pin is low. When this bit is high, the GPIO2 pin is high.
- GPIOI3** When this bit is low, the GPIO3 pin is low. When this bit is high, the GPIO3 pin is high.

Bits 12-15 determine the driven level of the four General Purpose I/O pins (GPIO0-3) when they are defined as outputs.

- GPIOO0** When this bit is low, the GPIO0 pin is driven low if it is defined as an output. When this bit is high, the GPIO0 pin is driven high if it is defined as an output.
- GPIOO1** When this bit is low, the GPIO1 pin is driven low if it is defined as an output. When this bit is high, the GPIO1 pin is driven high if it is defined as an output.

**GPIO2** When this bit is low, the GPIO2 pin is driven low if it is defined as an output. When this bit is high, the GPIO2 pin is driven high if it is defined as an output.

**GPIO3** When this bit is low, the GPIO3 pin is driven low if it is defined as an output. When this bit is high, the GPIO3 pin is driven high if it is defined as an output.

**Notes** The GPIO0 pin is logically connected to the fuse LED, a composite indicator of the condition of all fuses on the board. When the GPIO0 pin is high, all fuses are functional. When it is low, one or more fuses are inoperable.

GPIO1-3 are each directly connected to the board's J2 pins 16-18, respectively. These signals are available for customer use.



### Caution

GPIO0 should not be programmed as an output because it receives the composite fuse signal.

### 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 MVME162LX.

## 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.

---

<b>DISMST</b>	When this bit is high, the VME LED on the MVME162LX is lit when local bus reset is asserted or the VMEchip2 is driving local bus busy. When this bit is low, the VME LED on the MVME162LX is lit when local bus reset is asserted, the VMEchip2 is driving local bus busy, or the VMEchip2 is driving the VMEbus address strobe.
<b>DISSRAM</b>	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 MVME162LX, this bit must be set.
<b>REVEROM</b>	This function is not used on the MVME162LX. This bit must not be set.
<b>MPIRQEN</b>	This function is not used on the MVME162LX. 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 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 zero and one are used by the old VMEchip. The initial revision of the VMEchip2 is zero. 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 one to the

corresponding bit in the location monitor register. LM0 and LM1 can also be set by writing a one 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 one 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.

**Note** 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 power-up 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 is provided in the following tables. Each register definition includes a table with 5 lines. Line 1 is the base address of the register as viewed from the local bus and as viewed from the VMEbus, and the number of bits defined in the table. Line 2 shows the bits defined by this table. Line 3 defines the name of the register or the name of the bits in the register.

Line 4 defines the operations possible on the register bits as follows:

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

Line 5 defines the state of the bit following a reset as defined below.

- P** The 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 any reset.

A summary of the GCSR is shown in [Table 2-4 on page 2-105](#).

**Table 2-4. VMEchip2 Memory Map - GCSR Summary**

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	LM3	LM2	LM1	LM0	SIG 3	SIG 2	SIG 1	SIG 0	RST	ISF	BF	SC ON	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															

**NOTES:**

VMEchip2 GCSR Base Address = \$FFF40100

L = Local bus offset.

V = VMEbus offset.

## VMEchip2 Revision Register

ADR/SIZ	Local bus: <b>\$FFF40100</b> /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 zero and is incremented if mask changes are required. The VMEchip2 used on the MVME162LX is revision \$01 or greater.

## VMEchip2 ID Register

ADR/SIZ	Local bus: <b>\$FFF40100</b> /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: <b>\$FFF40104</b> /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 one 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 one 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 one 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 one 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 one 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 one 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 one 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 one 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 one 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 one 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 one 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 one to the LM3 bit in this register.

## VMEchip2 Board Status/Control Register

ADR/SIZ	Local bus: <b>\$FFF40104</b> /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: <b>\$FFF40108</b> /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: <b>\$FFF4010C</b> /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: <b>\$FFF40110</b> /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: <b>\$FFF40114</b> /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: <b>\$FFF40118</b> /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: <b>\$FFF4011C</b> /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.



---

## Introduction

The Memory Controller ASIC (MCchip) is one of four ASICs that are part of the MVME162LX 200/300 Series hardware set.

**Notes** The “DRAM” in this chapter refers only to the parity protected (instead of ECC) DRAM that comes on many models of the 200/300 Series. For boards with ECC protected DRAM, see the *MCECC* chapter of this Guide.

(In the later, 700/800 Series, parity is unsupported on non-ECC DRAM, and parity must remain *disabled* on all 700/800 Series boards.)

## Summary of Major Features

- ❑ BBRAM and time-of-day clock (MK48T08) interface with bus sizing.
- ❑ EPROM interface with bus sizing.
- ❑ Flash interface with bus sizing.
- ❑ SRAM controller supporting several configurations.
- ❑ Parity DRAM controller supporting several configurations; see Notes, above.
- ❑ Four serial interfaces implemented with two Zilog Z85230 SCC devices.
- ❑ NCR 53C710 SCSI Coprocessor interface.
- ❑ Intel 82596CA LAN Coprocessor interface.
- ❑ Four 32-bit tick timers.
- ❑ Interrupt support for ABORT switch, LAN, SCSI, SCC, Parity DRAM, and Timers.
- ❑ Local bus access timer.
- ❑ Watchdog timer.

## Functional Description

This section provides an overview of the functions provided by the MCchip. After the memory map, a detailed programming model for the MCchip control and status registers is then provided in the section entitled *Programming Model* on page 3-12.

### MCchip Initialization

The MCchip ASIC is designed to accommodate several memory configurations and MVME162LX 200/300 Series population versions. A factory-programmed resident device is used to initialize the board's Version Register, General Purpose Inputs Register, and Parity DRAM/SRAM Options Register (read only). A different initialization device is used for each version of the board in the 200/300 Series. Refer to the section *Programming Model* on page 3-12 for more information.

### Flash and EPROM Interface

The MCchip interfaces the MC68040 local bus to one 1M X 8 Intel 28F008SA Flash device and to four 32-pin DIP JEDEC standard EPROM sockets. The Flash and EPROM memory map locations can be swapped based upon a jumper input (J11, pins 7 and 8, GPI3) to the initialization PAL. (The initialization device was discussed in a previous section.) This enables the MVME162LX 200/300 Series to execute reset code from either the EPROM or Flash.

Note that MVME162LX 200/300 Series models ordered without the VMEbus interface are shipped with Flash memory blank (the factory uses the VMEbus to program the Flash memory with debugger code). To use the 162Bug package, MVME162Bug, in such models, be sure that jumper header J11 is configured for the EPROM memory map. Refer to the *Jumper Settings* summary table in the first chapter of this Guide (*Board Description and Memory Maps*), or to the *MVME162LX Embedded Controller Installation and Use* manual, the 200/300 Series version.

The MCchip 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 Flash memory has a write-protect feature. An access to any location between \$FFFC8000 through \$FFFCBFFF disables Flash write cycles. An access to any location between \$FFFC0000 through \$FFFCFFFF enables Flash write cycles. If the Flash memory is in the write-protect mode and a write is attempted, the cycle completes normally but the Flash memory contents are not changed.

## BBRAM Interface

The MCchip provides a read/write interface to the BBRAM by any bus master on the MC68040 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 MCchip.

## 82596CA LAN Interface

The LAN controller interface is described in the following sections.

### MPU Port and MPU Channel Attention

The MCchip allows the MC68040 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 selftest.

Each 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 MCchip 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 MC68040-bus master performs a simple read or write to address \$FFF46004.

### **MC68040-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 MC68040-bus cycles. When the 82596CA has local bus mastership, the MCchip drives the following MC68040 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 error are processed in the following way. 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 MCchip is updated and a LANC bus error interrupt is generated if it is enabled in the MCchip. 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 MCchip provides an interrupt control register for normal LANC termination and another register for bus error termination of LANC operation. The MCchip 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 MCchip 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 MCchip 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 Parity DRAM/SRAM Options Register when the MVME162LX 200/300 Series board is reset. SRAM performance at 25 MHz is 5,3,3,3 for read and write cycle. SRAM performance at 32 MHz is 6,4,4,4 for read cycles and 6,3,3,3 for write cycles (currently, however, all 200/300 Series boards are set to 25 MHz).

## Parity DRAM Memory Controller

When a 200/300 Series board's DRAM is non-ECC (that is, when it is Parity-protected), the MCchip ASIC determines the DRAM performance. This subsection describes the DRAM options for that case. (For boards with ECC DRAM, see the *Programming Model* section of the *MCECC* chapter in this Guide.)

The Parity DRAM base address, DRAM array size, and DRAM device size are programmable. The Parity DRAM controller assumes an interleaved architecture if the DRAM size requires eight physical devices. That is, there are two memory configurations which allow interleaved Parity DRAM architecture: when memory array size is four MB and DRAM technology is four Mbits per device; and when memory array size is 16 MB and DRAM technology is 16 Mbits per device. (*Note*: currently, however, boards with Parity-protected DRAM in the MVME162LX 200/300 Series only come with 1 or 4 MB arrays.)

Parity checking and parity exception action is also programmable. The DRAM array size and DRAM device size is initialized in the Parity DRAM/SRAM Options Register.

**Table 3-1. Parity DRAM Performance**

Clock Budget	Operating Conditions
4,2,2,2	non-interleaved, read, 25 MHz, without TEA on parity error
4,1,1,1	interleaved, read, 25 MHz, without TEA on parity error
5,3,3,3	non-interleaved, read, 25 MHz, with TEA on parity error
5,2,2,2	interleaved, read, 25 MHz, with TEA on parity error
3,2,2,2	write, 25 MHz
<p><b>NOTE:</b> TEA (Transfer Error Acknowledgement) is the MC68040 bus error transaction signal. "With TEA" indicates that a bus error cycle occurs if a DRAM parity error was detected.</p> <p>Also see <i>note</i>, above Table, about the sizes of Parity DRAM currently available in the 200/300 Series.</p>	

Be aware of the following, known problem and its work-around that are Parity DRAM-related:

When the RESET button on an MVME162LX 200/300 Series board is depressed for longer than a minimum length of time, Parity DRAM refresh may be inhibited and memory contents may be lost. This timelength may differ from one board to another and may vary with environmental conditions.



**Caution**

If you intend to examine memory subsequent to a reset, release the RESET button as quickly as possible after depressing it to ensure that Parity DRAM contents will not be altered.

## Z85230 SCC Interface

The MCchip provides a map decoder and an interrupt handler for the two Zilog Z85230s. The base addresses are \$FFF45000 and \$FFF45800. The MCchip 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. During the interrupt acknowledge cycle, interrupts from the first Z85230 have priority over those from the second Z85230.

For the MVME162LX 200/300 Series, the MCchip supports two Z85230 SCC devices (refer to the *Board Level Hardware Description* information in the *MVME162LX Embedded Controller Installation and Use* manual, *200/300 Series* version). The addresses for the two devices are defined as follows. Note that CSR bits were added to the General Control Register to control the delay time for the Z85230 IACK cycle.

Address Range	SCC Device Number
\$FFF45000 - \$FFF453FF	0
\$FFF45800 - \$FFF45BFF	1

## Tick Timers

The MCchip implements four 32-bit tick timers. These timers are identical to the timers in the VMEchip2. The timers run on a 1 MHz 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. These 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 the 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 one to the overflow clear bit.

## Watchdog Timer

A watchdog timer function is provided in the VMEchip2 and the MCchip. The watchdog timer implemented in the MCchip is used when the "No VMEbus Interface" option is enabled. 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 MCchip timer cannot generate a system reset because it is not connected to the VMEbus.



## Local Bus Timer

The MVME162LX 200/300 Series provides a timeout 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 timeout value is selectable by software for 8  $\mu$ sec, 64  $\mu$ sec, 256  $\mu$ sec, or infinite. 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 MCchip ASIC. Because the local bus timer in the VMEchip2 can detect an offboard access and the MCchip local bus timer cannot, the timer in the VMEchip2 is used in all cases except when the "No VMEbus Interface" option is enabled.

## Memory Map of the MCchip Registers

The register map and address of the memory controller ASIC (MCchip) 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 a 8- or 16-bit entity, it can be accessed as a byte, word, or longword.

The base address of the register map is \$FFF42000.

**Table 3-2. MCchip Register Map**  
MCchip Base Address = \$FFF42000

Offset	D31-D24	D23-D16	D15-D8	D7-D0
\$00	MCchip ID	MCchip 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	Parity DRAM Space Base Address Register		SRAM Space Base Address Register	
\$24	Parity DRAM Space Size	Parity DRAM/SRAM Options	SRAM Space Size	(reserved)

**Table 3-2. MCchip Register Map (Continued)**

MCchip Base Address = \$FFF42000

\$28	LANC Error Status	(reserved)	LANC Interrupt Control	LANC Bus Error Interrupt Control
\$2C	SCSI Error Status	General Purpose Inputs	MVME162LX 200/300 Series 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 Access Time Control	ABORT Switch Interrupt Control
\$44	RESET Switch Control	Watchdog Timer Control	Access & Watchdog Time Base Select	(reserved)
\$48	Parity DRAM Control	(reserved)	MPU Status	(reserved)
\$4C	32-bit Prescaler Count Register			

## Programming Model

This section defines the programming model for the control and status registers (CSR) in the MCchip. The base address of the CSR is \$FFF42000. 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.
- C** Writing a one to this bit clears this bit or another bit. This bit reads zero.

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

- P** The bit is affected by power-up reset.
- 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.

Note also the Caution about RESET that is DRAM-related, see page [3-7](#).

## MCchip 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.

## MCchip 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	0 PL	0 PL

**RV7-RV0** The current value of the chip revision is \$00. 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							MIEN	FAST
OPER	R	R	R	R	R	R	R/W	R/W
RESET	0 PL	0 PL	0 PL	0 PL	0 PL	0 PL	0 PL	0 P

**FAST** This control bit tailors the control circuit for BBRAM to the speed of BBRAM. When operating at 25 MHz, the FAST bit should be cleared for devices with access times longer than 200 ns (5 CLK cycles). The bit can be set for devices that have access times of 200 ns or faster. It is not allowed to use devices slower than 360 ns (9 CLK cycles), at 25 MHz. When operating at 32 MHz, the FAST bit should be cleared for devices with access times longer than 150 ns (5 CLK cycles). The bit can be set for devices that have access times of 150 ns or faster. It is not allowed to use devices slower than 270 ns (9 CLK cycles), at 32 MHz. (Note that currently no MVME162LX 200/300 Series boards operate at 32 MHz.)

**MIEN** Master Interrupt Enable. When this bit is high, interrupts from and via the MCchip are allowed to reach the MPU. When it is low, all interrupts from the MCchip are disabled. Also, when the bit is low, all interrupt acknowledge cycles to the MCchip are passed on, via the IACKOUT\* pin. This bit is cleared by a reset.

## 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 MC68xx040 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 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	<b>\$FFF42000</b> (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 MCchip 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 MCchip is wired to have the highest priority followed by the IndustryPack interface ASIC (IPIC chip) and then the VMEchip2 ASIC.

**Table 3-3. 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 (Z85230s)	Note 1	Next Highest
ABORT Switch	\$E	Highest
unused	\$F	..

**Note:** The Z85230 controllers have an integrated interrupt vector register which is separate from the vector generation found on the MCchip. The Z85230 also supports a scheme where the base register value is changed based upon the interrupt requested. During the interrupt acknowledge cycle, interrupts from the first Z85230 have priority over those from the second Z85230.

## Programming the Tick Timers

There are four programmable tick timers in the MCchip. 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 zero, 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

ADR/SIZ	<b>\$FFF42004</b> (32 bits)		
BIT	31	. . .	0
NAME	Tick Timer 1 Compare Register		
OPER	R/W		
RESET	0 P		



**Tick Timer 1 Counter**

ADR/SIZ	<b>\$FFF42008</b> (32 bits)		
BIT	31	...	0
NAME	Tick Timer 1 Counter		
OPER	R/W		
RESET	X		

**Tick Timer 2 Compare Register**

ADR/SIZ	<b>\$FFF4200C</b> (32 bits)		
BIT	31	...	0
NAME	Tick Timer 2 Compare Register		
OPER	R/W		
RESET	0 P		

**Tick Timer 2 Counter**

ADR/SIZ	<b>\$FFF42010</b> (32 bits)		
BIT	31	...	0
NAME	Tick Timer 2 Counter		
OPER	R/W		
RESET	X		

**LSB Prescaler Count Register**

This register is used to generate the 1 MHz 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.

ADR/SIZ	<b>\$FFF42014</b> (8 bits)		
BIT	31	...	24
NAME	LSB Prescaler Count		
OPER	R		
RESET	X		

## Prescaler Clock Adjust Register

This register adjusts the prescaler so that it maintains a 1 MHz clock source for the tick timers. To provide a 1 MHz clock, the prescaler adjust register should be programmed based on the following equation:

$$\text{Prescaler Clock Adjust Register} = 256 - \text{processor clock (MHz)}$$

For example, for operation at 20 MHz the prescaler value is \$EC, at 25 MHz it is \$E7, and at 32 MHz 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.

ADR/SIZ	<b>\$FFF42014</b> (8 bits)		
BIT	23	...	16
NAME	Prescaler Clock Adjust		
OPER	R/W		
RESET	\$DF P		

## Tick Timer 1 and 2 Control Registers

Each tick timer has a control register. The control registers for one and two are defined in this section. Control registers for three and four 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 zero 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 one 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 one 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

<b>IL2-IL0</b>	These three bits select the interrupt level for the tick timers. Level 0 does not generate an interrupt.
<b>ICLR</b>	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 zero.
<b>IEN</b>	When this bit is set high, the interrupt is enabled. The interrupt is disabled when this bit is low.
<b>INT</b>	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 *only* on the MVME162LX 200/300 Series boards with Parity- (instead of ECC-) protected DRAM. The parity control and interrupt logic is contained in the DRAM Parity Error Interrupt Control Register and in the Parity DRAM Control Register located at \$FFF4201C and \$FFF42048 respectively.

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 Parity DRAM parity error detection. Level 0 does not generate an interrupt.
- ICLR** Writing a logic 1 to this bit clears the Parity DRAM parity error detection interrupt. This clears the INT bit in this register. This bit is always read as zero.
- IEN** This bit set to a one enables the parity error interrupt. If this bit is set to a one, 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.



### Caution

Do not set the IEN bit unless parity is supported by the DRAM mezzanine. Most current MVME162LX 200/300 Series models come with ECC and not Parity DRAM. For boards with ECC DRAM, see the *MCECC* chapter of this Guide.

- INT** When this bit is high, an interrupt is being generated due to a Parity 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 either 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 zero. During the interrupt acknowledge cycle, interrupts from the first Z85230 have priority over those from the second Z85230.

## 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 zero 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 one 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 one to COVF.



## Parity DRAM and SRAM Memory Controller Registers

The Parity 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.

**Note** For boards with ECC DRAM, see the *MCECC* chapter of this Guide.

The reset logic initializes the Parity DRAM and SRAM Base registers so that DRAM space starts at address 0 and SRAM space starts at \$FFE00000. Parity DRAM and SRAM are inhibited by reset. Software can examine the MVME162LX 200/300 Series Parity DRAM/SRAM Options Register at address \$FFF42024 bits 20-16 to determine the size of the SRAM and DRAM.

Note also the Caution about the RESET button that is Parity DRAM-related; see page 3-7.

### Parity 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 Parity 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 4096 KB, the Base Register must be set to 0, or 4096 KB, or 8192 KB, or 12288 KB, etc.

**SRAM Space Base Address Register**

ADR/SIZ	<b>\$FFF42020</b> (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 Parity DRAM Base and Size Registers also exists for the SRAM decode logic. Refer to the Parity DRAM Space Base Register description.

## Parity DRAM Space Size Register

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** These size bits apply only to boards with *Parity-* (i.e., instead of ECC-) protected DRAM. The bits configure the Parity DRAM decoder for a particular memory size. The [Table 3-4](#), below, defines their encoding. Note that the table specifies the allowed bit combinations for DZ2 - DZ0: any other combinations generate unpredictable results.

**Note** For the size bits of boards with ECC DRAM, see the *Programming Model* section in the *MCECC* chapter of this Guide, the *Defaults Register 1* subsection.

DZ2 - DZ0 are set equal to the DZ2 - DZ0 bits of the Parity DRAM/SRAM Options Register. Note that changing DZ2 - DZ0 so that the Parity DRAM architecture changes between interleaved and non-interleaved relocates the data (but also see NOTE at bottom of [Table 3-4](#), below). DZ2 - DZ0 are programmable to facilitate diagnostic software.

**Table 3-4. Parity DRAM Size Control Bit Encoding**

DZ2 - DZ0	Memory Size (see Note at bottom for 200/300 Series boards)
\$0	1 MB with 4 Mbit Parity DRAMs
\$1	2 MB with 4 Mbit Parity DRAMs
\$3	4 MB with 4 Mbit Parity DRAMs (interleaved)
\$4	4 MB with 16 Mbit Parity DRAMs

**Table 3-4. Parity DRAM Size Control Bit Encoding (Continued)**

\$5	8 MB with 16 Mbit Parity DRAMs
\$6	Parity DRAM Mezzanine is not present.
\$7	16 MB with 16 Mbit Parity DRAMs (interleaved)
<b>NOTE:</b> 1 and 4 MB are the only sizes currently available for 200/300 Series boards with Parity DRAM.	

**Parity DRAM/SRAM Options Register**

Note that this register is read only and 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**

DZx bits indicate the size and architecture of the Parity (*non-ECC*) DRAM array (see explanation of preceding register for details). Software must initialize the Parity 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.

See [Table 3-4 on page 3-27](#), above, for Parity DRAM Size Control Bit Encoding (and see NOTE at bottom of Table for 200/300 Series boards).

**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 (see [Table 3-5 on page 3-29](#)). SZ1 - SZ0 are initialized at reset to a value which is determined by the contents of a factory-programmed resident device.

**Table 3-5. SRAM Size Control Bit Encoding**

<b>SZ1 - SZ0</b>	<b>SRAM Configuration</b>
\$0	128 KB
\$1	512 KB
\$2	1 MB
\$3	2 MB

**F0** F0 set to a 0 indicates that one 28F008SA 1M 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

<b>ADR/SIZ</b>	<b>\$FFF42024 (8 bits)</b>							
<b>BIT</b>	15	14	13	12	11	10	9	8
<b>NAME</b>						SEN	SZ1	SZ0
<b>OPER</b>	R							
<b>RESET</b>	0 PL							

**SEN** SRAM ENABLE must be set to a one 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 Parity DRAM/SRAM Options Register. SZ1 - SZ0 are programmable to facilitate diagnostic software.

**Table 3-6. SRAM Size Control Bit Encoding**

<b>SZ1 - SZ0</b>	<b>Memory Size</b>
\$0	Reserved (Don't use)
\$1	512 KB (or 128 KB)

**Table 3-6. SRAM Size Control Bit Encoding (Continued)**

\$2	1 MB
\$3	2 MB

**Note** For an MVME162LX 200/300 Series board with 128 KB of SRAM, the software must program SZ1-SZ0 = \$1 (512 KB). 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 timeout, 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 parity check error on a board with Parity DRAM (only), 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 zero.
- 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 zero.
- 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 MCchip drives onto the local MC68xx040 bus SC1 and SC0 pins, when the 82596CA (LANC) performs DMA accesses. During LANC DMA, if bit SC0 is 0, local bus pin SC0 is low, and when bit SC0 is 1, pin SC0 is high. The same relationship holds true for bit and pin SC1. Refer to the M68040 user's manual for details on how it uses the snoop control signals.

## 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 timeout, 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 parity check error on a board with Parity DRAM (only), 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 and the state of an 8-position jumper block are translated to bit settings of the General Purpose Inputs Register, Version Register and Parity DRAM/SRAM Options Register when the MCchip is reset. These registers are read only. 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 which are connected to three jumpers on the MVME162LX 200/300 Series board. Refer to Chapter 1 for a jumper pin summary. If the bit is set to a one, the jumper is absent; if it is a zero, the jumper is present. The jumpers for V10 - V8 are located at J11 pins 6 and 5, 4 and 3, 2 and 1 (for GPI2, GPI1, and GPIO), respectively.

**V11** Refer to the *Local Bus Memory Map* table, Notes 5 and 6, in Chapter 1 of this Guide (*Board Description and Memory Maps*). The jumper for V11 is located at J11 pins 8 and 7 (for GPI3). Refer to Chapter 1 for a jumper pin summary. If jumper is installed, Flash is accessed at \$FF800000-\$FF9FFFFFFF, and EPROM is accessed at \$FFA00000-\$FFBFFFFFFF. If jumper is not installed, the EPROM and Flash locations are reversed. Refer to the *MVME162LX Embedded Controller Installation and Use* manual, the *200/300 Series* version, for full jumper pin definitions.



### Caution

If you remove the jumper from J11, pins 8 and 7 (selects the EPROM memory map), make sure that the EPROM socket is populated with code that will be executed when the system is reset.

**V15-V12** V15 - V12 are general purpose inputs. Refer to the description for V10 - V8. The jumpers for V15 - V12 are located at J11 pins 16 and 15, 14 and 13, 12 and 11, 10 and 9 (for GPI7, GPI6, GPI5, and GPI4), respectively.

## MVME162LX 200/300 Series Version Register

The contents of a PAL and the state of an 8-position jumper block are translated to bit settings of the General Purpose Inputs Register, Version Register and Parity DRAM/SRAM Options Register when the MCchip is reset. These registers are read only. 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 set to a one indicates that the processor and associated logic is operating at 32 MHz. V0 set to a zero indicates 25 MHz. Note that currently all 200/300 Series boards operate at 25 MHz.
- V1** V1 set to a one indicates that the VMEchip2 ASIC is not present. V1 set to a zero indicates that a VMEbus interface is present. If V1 = 0, the MCchip local bus access timer is inhibited.
- V2** V2 set to a one indicates that the SCSI interface is not present. V2 set to a zero indicates that a SCSI interface is present.
- V3** V3 set to a one indicates that the Ethernet interface is not present. V3 set to a zero indicates that a Ethernet interface is present.
- V4** V4 set to a one indicates that the MC68040 processor is present. V4 set to a zero indicates that a MC68LC040 is present.
- V5** Reserved for internal use only.
- V6** V6 = 0 indicates the board is an MVME162. V6 = 1 indicates the board is an MVME162LX 200/300 Series.
- 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	ILO
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-ILO** 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-ILO. 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 three and four are defined here because they maintain this relative position in the memory map. Refer to *Tick Timer 1 and 2 Compare and Counter Registers* on page 3-16 for a description of tick timers one and two.

### Tick Timer 3 Compare Register

ADR/SIZ	<b>\$FFF42030</b> (32 bits)		
BIT	31	...	0
NAME	Tick Timer 3 Compare Register		
OPER	R/W		
RESET	0 P		

### Tick Timer 3 Counter

ADR/SIZ	<b>\$FFF42034</b> (32 bits)		
BIT	31	...	0
NAME	Tick Timer 3 Counter		
OPER	R/W		
RESET	X		

### Tick Timer 4 Compare Register

ADR/SIZ	<b>\$FFF42038</b> (32 bits)		
BIT	31	...	0
NAME	Tick Timer 4 Compare Register		
OPER	R/W		
RESET	0 P		

### Tick Timer 4 Counter

ADR/SIZ	<b>\$FFF4203C</b> (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., \$20 for 32 MHz). The MCchip uses the value programmed in this register to control the refresh timer so that the Parity DRAMs are refreshed every 15.6 microseconds. After power-up, this register is initialized to \$10 (for 16 MHz).

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:

$$\text{Refresh Rate} = \text{BCK}/\text{BUS CLOCK} * 16$$

where BCK is the value programmed in the Bus Clock Register, and BUS CLOCK is the MC68xx040 bus clock frequency.

## EPROM Access Time Control Register

The MVME162LX 200/300 Series 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 encodings.

ET2-ET0	EPROM ACCESS $\leq$ N at 25 MHz where N =	EPROM ACCESS $\leq$ N at 32 MHz 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

**Note** (Currently no boards in the MVME162LX 200/300 Series operate at 32 MHz.)

**ROM0** Refer to the *Local Bus Memory Map* table, Note 1, in the first chapter of this Guide (*Board Description and Memory Maps*).



## Flash Access Time Control Register

The MVME162LX 200/300 Series 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						FT2	FT1	FT0
OPER	R	R	R	R	R	R/W	R/W	R/W
RESET	0	0	0	0	0	1 PL	1 PL	1 PL

**FT2-FT0** Flash memory access time is controlled by the state of FT2-FT0. The following table defines the FT2-FT0 encodings.

FT2-FT0	Flash Access $\leq N$ at 25 MHz where $N =$	Flash Access $\leq N$ at 32 MHz 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

**Note** (Currently no boards in the MVME162LX 200/300 Series operate at 32 MHz.)

## ABORT Switch Interrupt Control Register

The following table describes the ABORT switch interrupt logic in the MCchip.

ADR/SIZ	<b>\$FFF42040</b> (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 zero.
- IEN** When 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 set to a one indicates that the ABORT switch is pressed. When it is a zero, the switch is inactive.

## RESET Switch Control Register

The RESET switch on the MVME162LX 200/300 Series front panel and several status and 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 with the “No VMEbus Interface” option. This bit is duplicated at the same bit location 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 MCchip asserts the BRDFAIL signal pin. This signal is wired-or to the VMEchip2 board fail pin. It controls the board fail (FAIL) LED on the MVME162LX 200/300 Series.
- CPURS** When this bit is set high, the power-up reset status bit is cleared. This bit is always read zero.
- 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 MCchip is asserted. When this status bit is low, the BRDFAIL signal pin on the MCchip is not asserted. The BRDFAIL pin may be asserted by an external device, the BDFLO bit in this register, or a watchdog timeout.

## Watchdog Timer Control Register

The watchdog timer control logic in the MCchip is used with the "No VMEbus Interface" option. 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 and the MCchip.

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 timeout occurs, a LRESET is generated. When this bit is low, a watchdog timeout does not cause a reset.
- WDBFE** When this bit is high and the watchdog timer has timed out, the MCchip asserts the BRDFAIL signal pin. When this bit is low, the watchdog timer does not contribute to the BRDFAIL signal on the MCchip.
- WDTO** When this status bit is high, a watchdog timeout has occurred. When this status bit is low, a watchdog timeout has not occurred. This bit is cleared by writing a one 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 timeout period or a watchdog timeout occurs.
- WDCS** When this bit is set high, the watchdog timeout status bit (WDTO bit in this register) is cleared.

## Access and Watchdog Time Base Select Register

The watchdog timer control logic in the MCchip is used with the "No VMEbus Interface" option. 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 MCchip.

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 timeout period:

**Bit Encoding    Timeout**

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
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 timeout 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. Note that the Version Register bit V1 must be set to a 1 to enable the MCchip access timer (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.

## Parity DRAM Control Register

This register controls the parity checking mode and Parity DRAM enable for Parity DRAM applications only, not for 200/300 Series boards with ECC DRAM.



**Caution**

Do *not* enable parity checking unless it is supported by the appropriate type of DRAM mezzanine. Currently more than half of the models in the MVME162LX 200/300 Series come with ECC-protected DRAM on mezzanines, not Parity DRAM.

**Note** For corresponding information on the 200/300 Series boards with ECC DRAM, see the *Programming Model* section in the *MCECC* chapter of this Guide.

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 *only* for boards with Parity DRAM. The DRAM should be enabled after the Parity DRAM Space Base Address Register is enabled and the ROM0 bit has been cleared. The Parity DRAM Space Base Address Register is located at \$FFF42020 bits 31 - 16 and the ROM0 bit is located at \$FFF42040 bit 20.

**PAREN-PARINT**

These bits are to be enabled *only* on boards with Parity DRAM:

PAREN	PARINT	MPU	Alternate
0	0	NONE	NONE
0	1	INTERRUPT	NONE
1	0	CHECKED	CHECKED
1	1	INTERRUPT	CHECKED

(On boards with Parity DRAM *only*.)

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 TAE\* if a parity error occurs. Note that CHECKED cycles lengthen the Parity DRAM accesses by one clock tick.

**WWP**

*Only* on boards with Parity DRAM: setting WWP to a one causes inverted parity to be written to the DRAM . This is used for diagnostic software.

## MPU Status Register

This logic is duplicated in the VMEchip2 at location \$FFF40048, bits 11, 10, 9, and 7. The duplication is to enable "No VMEbus Interface" operation.

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 timeout. This bit is cleared by a writing a one to the MCLR bit in this register. This bit is used with the "No VMEbus Interface" option and is duplicated in the VMEchip2 at address \$FFF40048 bit 7.
- MLPE** *Only on boards with Parity DRAM:* when this bit is set, the MPU received a TEA and the status indicated a parity error during a Parity DRAM data transfer. This bit is cleared by writing a one to the MCLR bit in this register. This bit is used with the "No VMEbus Interface" option 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 one to the MCLR bit in this register. This bit is used with the "No VMEbus Interface" option and is duplicated in the VMEchip2 at address \$FFF40048 bit 10.
- MCLR** Writing a one 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 MCchip. The lower 8 bits of the prescaler is programmed to generate an output with a one microsecond period. Refer to *LSB Prescaler Count Register on page 3-17*. 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 to. Writes to this register must be 32 bits.

ADR/SIZ	<b>\$FFF4204C</b> (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 MCchip at \$FFF42014.

**MSB31-8** The most significant bits of the prescaler.  
 Note that for the "No VMEbus Interface" option, 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 MVME162LX 200/300 Series board with the VMEchip2 as a 200/300 Series board without the VMEchip2. This feature is provided for those applications which require a Prescaler Count Register to run on all MVME162LX 200/300 Series versions.



---

## Introduction

This chapter describes the IndustryPack Interface Controller (IPIC) ASIC for the MC68040 bus. The IPIC chip is designed for the MVME162LX board and supports up to four IndustryPack (IP) interfaces, designated IP\_a through IP\_d.

**Note** MVME162LX 200/300 Series models implement interfaces to only two IPs: IP\_a and IP\_b. While this chapter will document all four interfaces (and their physical control registers are present in all IPIC chips), the segments applicable to IP\_c and IP\_d are *not* used with 200/300 Series models.

## Features

- ❑ Provides all logic required to interface the MC68040 bus to four IndustryPacks.
- ❑ Supports IndustryPack I/O, Memory, Interrupt Acknowledge, and ID cycles.
- ❑ Supports 8-bit, 16-bit, and 32-bit (double size) IndustryPack cycles.
- ❑ Provides dynamic bus sizing for accesses to IndustryPack Memory Space.
- ❑ Fixed base address for IndustryPack I/O, ID, spaces.
- ❑ Programmable base address/size for IndustryPack Memory Space.
- ❑ Eight Interrupt Handler Control Registers, two for each IndustryPack.
- ❑ Recovery timer for each IndustryPack to provide dead time between back to back accesses.

## Functional Description

The following sections provide an overview of the functions provided by the IPIC chip. A detailed programming model for the IPIC control and status registers is provided in a later section of this chapter.

### 4

## General Description

The IPIC chip converts IP-bound MC68040 read/write/interrupt acknowledge cycles to IndustryPack cycles. The IPIC 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.)

**Note** MVME162LX 200/300 Series models implement interfaces to only two IPs: IP\_a and IP\_b. While this chapter will document all four interfaces (and their physical control registers are present in all IPIC chips), the segments applicable to IP\_c and IP\_d are *not* used with 200/300 Series models.

## Performance

The clock speed for the IndustryPack logic interface on MVME162LX 200/300 Series boards is 8MHz.

The BCLK (local bus clock) and IPCLK frequencies are not assumed by the IPIC logic to be synchronized to each other. Because of this, the IPIC provides internal synchronization from the MC68040 bus to the IndustryPacks bus, and vice versa. Between this synchronization time, and the fact that the IndustryPack clock cycle is 8 MHz, accesses by the MC68040 to the IndustryPacks are relatively slow compared to accesses such as the MC68040 to DRAM.

## Cache Coherency

The IPIC chip observes the snoop control (SC1, SC0) and memory inhibit (MI\*) signals to maintain cache coherency. When SC1, SC0 indicate that snooping is inhibited, the IPIC pair ignores the memory inhibit (MI\*) signal line. When SC1, SC0 do not indicate that snooping is inhibited, the IPIC chip 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 IPIC assumes that the cycle is over and that it is not to participate.

## Error Reporting

The IPIC chip does not have the ability to drive the TEA\* signal. Consequently, the only error reporting that is available from the IPIC is in the ERR status bits in the General Control Registers.

## Interrupts

The IPIC chip 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 IPIC 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 there are multiple IndustryPack interrupts pending at the level being acknowledged, the IPIC performs the interrupt acknowledge for the one with the highest priority. The priority is as follows:

Interrupt Source	Priority
IP_a0	Highest
IP_a1	Next Highest
IP_b0	
IP_b1	
IP_c0 (Note)	
IP_c1 (Note)	∅
IP_d0(Note)	Next Lowest
IP_d1(Note)	Lowest

**Note** The \_c and \_d interfaces are not used on MVME162LX 200/300 Series models.

When the local bus master acknowledges an interrupt, if the IPIC 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\*.

# Overall Memory Map

The following memory map table includes all devices selected by the IPIC map decoder.

**Note** The IP\_c, IP\_d, and IP\_cd interfaces are not used on MVME162LX 200/300 Series models.

4

**Table 4-1. IPIC 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 (Note)	D16	128B
\$FFF58280-\$FFF582BF	IP_c ID Space (Note)	D16	64B
\$FFF582C0-\$FFF582FF	IP_c ID Space Repeated (Note)	D16	64B
\$FFF58300-\$FFF5837F	IP_d I/O Space (Note)	D16	128B
\$FFF58380-\$FFF583BF	IP_d ID Space (Note)	D16	64B
\$FFF583C0-\$FFF583FF	IP_d ID Space Repeated (Note)	D16	64B
\$FFF58400-\$FFF584FF	IP_ab I/O Space	D32-D16	256B
\$FFF58500-\$FFF585FF	IP_cd I/O Space	D32-D16	256B

**Table 4-1. IPIC Overall Memory Map (Continued)**

Address Range	Selected Device	Port Width	Size
\$FFF58600-\$FFF586FF	IP_ab I/O Space Repeated	D32-D16	256B
\$FFF58700-\$FFF587FF	IP_cd I/O Space Repeated (Note)	D32-D16	256B
\$FFFBC000-\$FFFBC01F	Control/Status Registers	D32-D8	32B
NOTE: These items are present but are not used on MVME162LX 200/300 Series models.			

## Programming Model

This section defines the programming model for the control and status registers (CSRs) in the IPIC chip. 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 one to it.
- C** Writing a zero to this bit clears this bit or another bit. This bit reads as zero.
- S** Writing a one to this bit sets this bit or another bit. This bit reads as zero.

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 IPIC CSR registers is shown in [Table 4-2 on page 4-7](#). 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 the following register descriptions are labeled as bits 7 through 0.



**Note** The IP\_c, IP\_d, and IP\_cd interfaces are not used on MVME162LX 200/300 Series models.

**Table 4-2. IPIC Memory Map - Control and Status Registers**

IPIC Chip Base Address = \$FFFBC000

4

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	0	0
\$02	RESERVED	0	0	0	0	0	0	0	0
\$03	RESERVED	0	0	0	0	0	0	0	0
\$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

**Table 4-2. IPIC Memory Map - Control and Status Registers (Continued)**

IPIC Chip Base Address = \$FFFBC000

\$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 INTO 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	0	a_MEN
\$19	IP_b GENERAL CONTROL	b_ERR	0	b_RT1	b_RT0	b_WIDTH1	b_WIDTH0	0	b_MEN
\$1A	IP_c GENERAL CONTROL	c_ERR	0	c_RT1	c_RT0	c_WIDTH1	c_WIDTH0	0	c_MEN
\$1B	IP_d GENERAL CONTROL	d_ERR	0	d_RT1	d_RT0	d_WIDTH1	d_WIDTH0	0	d_MEN
\$1C	RESERVED	0	0	0	0	0	0	0	0
\$1D	RESERVED	0	0	0	0	0	0	0	0
\$1E	RESERVED	0	0	0	0	0	0	0	0
\$1F	IP RESET	0	0	0	0	0	0	0	RES

## Chip ID Register

The read-only Chip ID Register is hard-wired to a hexadecimal value of \$23. Writes to this register do nothing, however the IPIC 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

4

## Chip Revision Register

The read-only Chip Revision Register is hard-wired to reflect the revision level of the IPIC ASIC. The current value of this register is \$00. Writes to this register do nothing, however the IPIC terminates them normally with TA\*.

ADR/SIZ	\$FFFBC001 (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	0	0	0	0	0	0

## IP\_a, IP\_b, IP\_c, IP\_d Memory Base Address Registers

The memory base address registers define the base address at which the IPIC 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.

**Note** The IP\_c, IP\_d, and IP\_cd interfaces are not used on MVME162LX 200/300 Series models.

For each of the four sets of registers, BASE31-BASE16 are compared to MC68040 address signals 31-16 respectively. The IPIC 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 MC68040 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 of IP\_a, IP\_b, IP\_c, IP\_d, that are enabled, should *not* be programmed to overlap each other. (The IP\_c, IP\_d, and IP\_cd interfaces are not used on MVME162LX 200/300 Series models.)

**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_BASE31	a_BASE30	a_BASE29	a_BASE28	a_BASE27	a_BASE26	a_BASE25	a_BASE24
NAME(\$05)	a_BASE23	a_BASE22	a_BASE21	a_BASE20	a_BASE19	a_BASE18	a_BASE17	a_BASE16
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	\$FFFBC006 and \$FFFBC007 (8 bits each)							
BIT	7	6	5	4	3	2	1	0
NAME(\$06)	b_BASE31	b_BASE30	b_BASE29	b_BASE28	b_BASE27	b_BASE26	b_BASE25	b_BASE24
NAME(\$07)	b_BASE23	b_BASE22	b_BASE21	b_BASE20	b_BASE19	b_BASE18	b_BASE17	b_BASE16
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\_c or Double Size IP\_cd Memory Base Address Registers**

(not used on MVME162LX)

ADR/SIZ	\$FFFBC008 and \$FFFBC009 (8 bits each)							
BIT	7	6	5	4	3	2	1	0
NAME(\$08)	c_BASE31	c_BASE30	c_BASE29	c_BASE28	c_BASE27	c_BASE26	c_BASE25	c_BASE24
NAME(\$09)	c_BASE23	c_BASE22	c_BASE21	c_BASE20	c_BASE19	c_BASE18	c_BASE17	c_BASE16
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\_d Memory Base Address Registers**

(not used on MVME162LX)

ADR/SIZ	\$FFFBC00A and \$FFFBC00B (8 bits each)							
BIT	7	6	5	4	3	2	1	0
NAME(\$0A)	d_BASE31	d_BASE30	d_BASE29	d_BASE28	d_BASE27	d_BASE26	d_BASE25	d_BASE24
NAME(\$0B)	d_BASE23	d_BASE22	d_BASE21	d_BASE20	d_BASE19	d_BASE18	d_BASE17	d_BASE16
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\_a, IP\_b, IP\_c, IP\_d Memory Size Registers

**Note** Registers for IP\_c, IP\_d, and IP\_cd are not used on the MVME162LX 200/300 Series models.

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 IPIC performs its IndustryPack 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 Being 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	16KB

Note that 16MB is only possible using a double size IP.

## IP\_a, IP\_b, IP\_c, IP\_d: IRQ0/IRQ1 Interrupt Control Registers

**Note** The segments applicable to IP\_c and IP\_d are not used in the MVME162LX 200/300 Series..

ADR/SIZ	\$FFFBC010 through \$FFFBC017 (8 bits each)							
BIT	7	6	5	4	3	2	1	0
NAME(\$10)	a0_PLTY	a0_E/L*	a0_INT	a0_IEN	a0_ICLR	a0_IL2	a0_IL1	a0_IL0
NAME(\$11)	a1_PLTY	a1_E/L*	a1_INT	a1_IEN	a1_ICLR	a1_IL2	a1_IL1	a1_IL0
NAME(\$12)	b0_PLTY	b0_E/L*	b0_INT	b0_IEN	b0_ICLR	b0_IL2	b0_IL1	b0_IL0
NAME(\$13)	b1_PLTY	b1_E/L*	b1_INT	b1_IEN	b1_ICLR	b1_IL2	b1_IL1	b1_IL0
NAME(\$14)	c0_PLTY	c0_E/L*	c0_INT	c0_IEN	c0_ICLR	c0_IL2	c0_IL1	c0_IL0
NAME(\$15)	c1_PLTY	c1_E/L*	c1_INT	c1_IEN	c1_ICLR	c1_IL2	c1_IL1	c1_IL0
NAME(\$16)	d0_PLTY	d0_E/L*	d0_INT	d0_IEN	d0_ICLR	d0_IL2	d0_IL1	d0_IL0
NAME(\$17)	d1_PLTY	d1_E/L*	d1_INT	d1_IEN	d1_ICLR	d1_IL2	d1_IL1	d1_IL0
OPER	R/W	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-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\*s are active low, PLTY would normally be cleared.

## IP\_a, IP\_b, IP\_c, and IP\_d: General Control Registers

**Note** The segments applicable to IP\_c and IP\_d are not used in the MVME162LX 200/300 Series..

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	0	a_MEN
NAME(\$19)	b_ERR	0	b_RT1	b_RT0	b_WIDTH1	b_WIDTH0	0	b_MEN
NAME(\$1A)	c_ERR	0	c_RT1	c_RT0	c_WIDTH1	c_WIDTH0	0	c_MEN
NAME(\$1B)	d_ERR	0	d_RT1	d_RT0	d_WIDTH1	d_WIDTH0	0	d_MEN6
OPER	R	R	R/W	R/W	R/W	R/W	R	R/W
RESET	? R	0 R	0 R	0 R	0 R	0 R	0 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.

**WIDTH1, WIDTH0** The IPIC 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. 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



**Notes:** The segments applicable to IP\_c and IP\_d are not used in the MVME162LX 200/300 Series.

When programming b\_WIDTH1-b\_WIDTH0 for either 8 bits or 16 bits, a\_WIDTH1-a\_WIDTH0 must be programmed for one 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 one 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 IPIC 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.

RT1	RT0	Recovery Time
0	0	0 microseconds
0	1	2 microseconds
1	0	4 microseconds
1	1	8 microseconds

There are some restrictions for 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.

**ERR** When one of these bits is set to a one, its corresponding IndustryPack Error\* signal is asserted. a\_ERR, b\_ERR, c\_ERR, and d\_ERR are from IndustryPack Errors a through d respectively. The state of these status bits at reset depends on the level driven onto their corresponding Error\* signals by the specific IndustryPacks that are installed in the system.

## IP RESET Register

ADR/SIZ	\$FFFBC01F (8 bits)							
BIT	7	6	5	4	3	2	1	0
NAME	0	0	0	0	0	0	0	RES
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

**RES** Setting RES to a one asserts the IPIC IPRESET\* signal. IPRESET\* is connected to the Reset\* signal on all four IndustryPacks. When software sets the RES bit, it remains set for at least 1 millisecond, then it clears. Consequently, the duration of the assertion of IPRESET\* is at least 1 millisecond. Note that the IPIC also asserts IPRESET\* whenever its RESET\* input signal is asserted (powerup reset or any local reset).

# 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.

**Note** Segments applicable to IP\_c and IP\_d are not used in the MVME162LX 200/300 Series.

## 8-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:  $IPA=(LBA*2)+1$ .

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.

4

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  $IPA\langle 0 \rangle = LBA\langle 0 \rangle$ .

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.

LBA	IPA<6-0>	Comments
\$FFF58400	%0000000	IP_b or ab
\$FFF58401	%0000001	IP_b
\$FFF58402	%0000000	IP_a
\$FFF58403	%0000001	IP_a
\$FFF58404	%0000010	IP_b or ab
\$FFF58405	%0000011	IP_b
\$FFF584FC	%1111110	IP_b or ab
\$FFF584FD	%1111111	IP_b
\$FFF584FE	%1111110	IP_a
\$FFF584FF	%1111111	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.

**Note** Segments applicable to IP\_c and IP\_d are not used in the MVME162LX 200/300 Series.

4

### Memory Space Accesses

The following table shows the data routing when accessing IP memory space.

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 IPIC 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 IPIC 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.



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>

## I/O and ID Space Accesses

**Note** Segments applicable to IP\_c and IP\_d are not used in the MVME162LX 200/300 Series.

The following table shows the data routing when accessing IP I/O or ID space.

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.

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>
		0	0	IPBD<15-8>	IPBD<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>

---

## Introduction

This chapter describes the Memory Controller chip for ECC DRAM (MCECC). The MCECC is an ASIC used on DRAM mezzanine boards with ECC protection, and thus it only appears on the MVME162LX 200/300 Series boards with ECC DRAM.

The MCECC is designed for the MVME162LX 200/300 Series of boards and is used in a set of two, to provide the interface to a 144-bit wide DRAM memory system.

**Note** In some MVME product families, the MCECC corresponds to a parity memory controller (MEMC040) with which it shares a group of similarly-defined registers. For reference purposes, differences between these registers are briefly documented in this chapter (*Programming Model* section), but the MEMC040 is *not* used on any MVME162LX boards.

## Features

- ❑ Allows 2-1-1-1 memory accesses (sustained) for burst writes
- ❑ Allows 4-1-1-1 memory accesses (sustained) for burst reads (5-1-1-1 with BERR on or when FSTRD is cleared)
- ❑ Supports byte, 2 byte, 4 byte, and cache line read or write transfers
- ❑ Programmable base address for DRAM
- ❑ Built-in refresh timer and refresh controller
- ❑ ECC
  - Single Bit Error Detect and Correct
  - Software enabled Interrupt on Single Bit Error
  - Address and Syndrome Register For Single Bit Error Logging Support
  - Double Bit Error Detect
  - Software programmable Bus Error and/or Interrupt on Double Bit Error
- ❑ Programmable period automatic scrub operation

## Functional Description

The following subsections provide an overview of the functions provided by the MCECC. A detailed programming model for the MCECC control and status registers is provided in the section on *Programming Model*.

Note also the Caution about DRAM that is RESET-related; see page [5-11](#).

### 5

## General Description

The MCECC is designed to be used as a set of two chips. A pair of MCECCs works with x4 DRAM memory chips to form a memory system for the MVME162LX boards. A pair of MCECCs that is connected to implement a memory control function is referred to as an "MCECC pair". The MCECC pair provides all the functions required to implement a memory system. These include programmable map decoding, memory control, refresh, and a scrubber. The scrubber, when it is enabled, periodically scans memory looking for errors. If the scrubber finds a single bit error in the memory array, it corrects it. This prevents soft single bit errors from becoming double bit errors.

## Performance

The MCECC pair is specifically designed to provide maximum performance for cache line (burst) cycles to and from the MC68040 bus. This is done by providing a four-way interleave between the 32-bit MC68040 data bus and the 128 bit (144 with check bits) DRAM. This permits burst accesses to be pipelined, giving high performance from standard speed, static column, DRAMs. For example, burst reads can be sustained at speeds of 7 clocks per line of four four-bytes (8 clocks per line with BERR enabled or FSTRD cleared). If the local MC68040 bus clock frequency is 25MHz, this gives an average access time of 70ns (80ns with BERR or no FSTRD) per four-byte. Burst writes can be sustained at 5 clocks per line, for an average of 50 ns at 32 MHz.

Random (non-burst) reads and writes are pipelined to the extent possible. Random reads take four clocks (five clocks with BERR on or FSTRD cleared).

Random, non-burst writes are the slowest kind of access because they require that the MCECC pair perform a read-modify-write cycle to the DRAM in order to complete. The MCECC pair responds to the local bus in two clocks during random writes, but then it takes another eight clocks for the DRAM read-modify-write cycle to complete, thereby making the effective cycle time 10 clocks if the following access by the local bus master is to DRAM. This boils down to two clocks for one random write, and 10 clocks for sustained random writes.

The performance specifications for the MCECC are shown in Table 4-1.

**Table 5-1. MCECC Specifications**

Descriptions	Specifications
Reads, BERR off, FSTRD = 1	4 clock cycles for random reads 4-1-1-1 clock cycles for burst reads (sustained)
Reads, FSTRD = 0	5 clock cycles for random reads 5-1-1-1 clock cycles for burst reads (sustained)
Reads, BERR on	5 clock cycles for random reads 5-1-1-1 clock cycles for burst reads (sustained)
Writes	2 to 10 clock cycles for random non-burst writes 2-1-1-1 clock cycles for burst writes (sustained)

## Cache Coherency

The MCECC pair supports the MC68040 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 snoop control signal lines on the local bus and behaving appropriately based on their value.

5

When the snoop control signal lines (SC1, SC0) indicate that snooping is inhibited, the MCECC pair ignores the memory inhibit (MI\*) signal line.

When (SC1, SC0) do not indicate that snooping is inhibited, the MCECC pair responds differently to DRAM accesses, based on whether the cycle is a read or a write, and on the snoop wait (SWAIT) control bit.

For a read with SWAIT = 0, the MCECC pair immediately starts a read cycle to the DRAM and latches the data from the DRAMs. It waits, however, for MI\* to be negated before it enables the data (that has been latched) onto the local bus and asserts TA\* or TEA\*. If TA\* or TEA\* is asserted by another local bus slave before MI\* is negated, then the MCECC pair assumes that the cycle is over and that the DRAM is not to participate in that cycle.

For a read with SWAIT = 1, the MCECC pair behaves the same as with SWAIT = 0 except that it does not start the DRAM read cycle until it sees the MI\* signal negated. Note that this means that if another local bus slave asserts TA\* or TEA\* before MI\* is negated, then the MCECC pair never starts the DRAM read cycle.

For a write cycle, the MCECC pair always waits for MI\* to be negated before it begins a write cycle to the DRAM. If another local bus slave asserts TA\* or TEA\* before MI\* is negated, then the MCECC pair never starts the DRAM write cycle.

## ECC

The MCECC pair performs single bit error correction and double bit error detection (SECCDED). The 32 bit wide local data bus is divided into lower (D00-D15) and upper (D16-D31) halves. Each half is routed through an MCECC, which multiplexes it with half of the 128 bit wide DRAM. This allows each MCECC to connect to 64 bits of the DRAM. Each MCECC additionally connects to 8 bits of check bit DRAM. This actually makes the DRAM array 144 bits wide (128 bits of normal data and 16 bits of check data).

### Cycle Types

To support ECC, the MCECC pair always deals with DRAM using full width (144 bits, 72 bits for each MCECC) accesses. When the local bus master requests any size read of DRAM, the MCECC pair reads 144 bits. When the local bus master requests a line write to DRAM, the MCECC pair writes all 144 bits. When the local bus master requests a byte, word (two-byte), or longword write to DRAM, the MCECC pair performs a 144-bit wide read cycle to DRAM, merges the appropriate local bus write data in, and writes 144 bits to DRAM.

### Error Reporting

The MCECCs generate the ECC check bits for write cycles. They also check read data from the DRAM and correct it if it contains a single bit error. If a non-correctable error occurs within either of the MCECC 72 bits of read data, the affected MCECC indicates it by asserting its non-correctable error (NCE\*) pin.

The following paragraphs indicate the actions taken by the MCECC pair for different error situations.

#### **Single Bit Error (Cycle Type = Burst Read or Non-Burst Read)**

Correct the Data that is driven to the local MC68040 bus.

Do not correct the Data in DRAM. Note that the DRAM is not corrected until the next scrub of that address, which happens only if scrubbing is enabled.

Terminate the cycle normally. (Assert TA to the local bus.)

Log the error if one has not already been logged.

Notify the local MPU via interrupt if so enabled.

**Double Bit Error (Cycle Type = Burst Read or Non-Burst Read)**

Cannot correct the data that is driven to the local MC68040 bus.

Leave the error in DRAM. (Note that it is not corrected in DRAM during the next scrub of that address.)

Terminate the cycle with Bus Error (assert TEA to the local bus) if so enabled.

Log the error if one has not already been logged.

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)**

Correct the data read from the DRAM, merge with the write data, and write the correct, merged data to the DRAM.

Terminate the cycle normally. (Assert TA to the local bus.)

Log the error if one has not already been logged.

Notify the local MPU via interrupt if so enabled.

**Double Bit Error (Cycle Type = Non-Burst Write)**

Do not perform the write portion of the cycle. This causes the location to continue to indicate non-correctable error when accessed.

Terminate the cycle normally. (Assert TA to the local bus.)

Log the error if one has not already been logged.

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 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.

**Single Bit Error (Cycle Type = Scrub)**

Write corrected data to the DRAM.

Log the error if one has not already been logged.

Notify the local MPU via interrupt if so enabled.

**Double Bit Error (Cycle Type = Scrub)**

Do not perform the write portion of the cycle. This causes the location to continue to indicate non-correctable error when accessed.

Log the error if one has not already been logged.

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 could show up as "no error" or "single bit error", both of which are incorrect.

## Error Logging

ECC error logging is facilitated by the MCECC because of its internal latches. When an error (single or double bit) occurs in the DRAMs to which an MCECC is connected, it freezes the address of the error and the syndrome bits associated with the data that is in error. Each MCECC performs this logging function independently of the other. Once an MCECC 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 pair contains programmable registers and circuitry that provide the 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, then releases it for the local bus, or 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 144-bit read of DRAM, a correction of any single bit errors, and a write of the full 144 corrected bits back to the same location. If a single or double bit error occurs, the local bus master is notified if such interrupts are enabled in the control register. A software bit is available to disable the read portion of the scrub cycle.

## Refresh

The MCECC pair 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  $\mu$ s. To prevent undue noise generation, the MCECC pair does not refresh both blocks at once, but staggers the refreshes by one clock cycle.

## Arbitration

The MCECC pair has 3 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.

## Chip Defaults

Some jumper option kinds of parameters need to be configured in the MCECC pair. These options include DRAM size, DRAM speed, Control and Status Register Selection, etc. Rather than use pins (which are extremely scarce) for each of the options, the MCECC pair is designed to have an external PAL or other equivalent logic provide this information at reset time, using one pin as a serial input. The information provided to this input pin at power-up-reset or local bus reset, is called the "reset serial bit stream". The reset serial bit stream initializes the MCECC pair by setting or resetting the bits that appear in the Defaults 1 and Defaults 2 Registers. Software can override this initial setting by writing to the Defaults Registers. It is not recommended that non-test software alter the bits in the Defaults Registers.

## Programming Model

This section defines the programming model for the control and status registers (CSRs) in the MCECC pair. The base address of the CSRs is hard coded to the address \$FFF43000 for the MCECC pair on the first mezzanine board and \$FFF43100 for the MCECC pair on the second mezzanine board. The CSRs for the two MCECCs appear at the same address, (one on D16-D31, the other on D00-D15).

5

Hardware automatically duplicates the values that are written to the CSRs in the upper MCECC (the one that connects to D16-D31) to the lower MCECC (the one that connects to D0-D15). Hence Software only needs to write to the control registers in the upper MCECC. This duplicating function can be disabled by software for test purposes.

### **A Note on the “Differences with MEMC040” information in this section**

In some other MVME product families, the MCECC corresponds to a parity memory controller, the MEMC040, with which it shares a group of similarly-defined registers. For reference purposes, differences between these registers are briefly documented in this section as described below, but the MEMC040 is *not* used on any MVME162LX boards.

The first eight registers of the MCECC pair are those whose register map was made to look as similar as possible to the map for the same eight registers contained in the MEMC040. Where there are differences, they are noted. The remaining 18 registers contain functions unique to the MCECC pair.

## Key for bit definitions in CSR

The possible operations for each bit in the CSR are as follows:

<b>R</b>	This bit is a read only status bit.
<b>R/W</b>	This bit is readable and writable.
<b>R/C</b>	This status bit is cleared by writing a one to it.
<b>C</b>	Writing a zero to this bit clears this bit or another bit. This bit reads zero.
<b>S</b>	Writing a one to this bit sets this bit or another bit. This bit reads zero.

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

<b>P</b>	The bit is affected by power-up reset.
<b>L</b>	The bit is affected by local reset.
<b>S</b>	The bit is affected by software reset. (Writing \$0F to the Chip ID Register)
<b>X</b>	The bit is not affected by reset.
<b>V</b>	The effect of reset on this bit is variable.

Note the following, known RESET problem and its work-around that are DRAM-related: when the RESET button on an MVME162LX 200/300 Series board is depressed for longer than a minimum length of time, DRAM refresh may be inhibited and memory contents may be lost. This timelength may differ from one board to another and may vary with environmental conditionals.



### Caution

If you intend to examine memory subsequent to a reset, release the RESET button as quickly as possible after depressing it to ensure that DRAM contents will not be altered.

## Memory Map of the MCECC Internal Register

A summary of the first eight CSR registers (the ones that correspond to those found in the MEMC040; see explanation at start of this section) is shown in Table 5-2, following. Note that even though there are two sets of these registers, one for the lower MCECC and one for the upper MCECC, software should only perform read and write cycles to the control and status registers in the upper MCECC.

Hardware takes care of duplicating the information to the lower MCECC. The following descriptions show the upper MCECC bit positions. Upper MCECC bit positions 31-24 correspond to lower MCECC bit positions 15-8. The base address of the CSRs is hard coded to the address \$FFF43000 for the MCECC pair on the first mezzanine board and \$FFF43100 for the MCECC pair on the second mezzanine board.

**Table 5-2. MCECC Internal Register Memory Map, Part 1**

MCECC Base Address = \$FFF43000 (1st); \$FFF43100 (2nd)

Register Offset	Register Name	Register Bit Names							
		D31	D30	D29	D28	D27	D26	D25	D24
\$00	<b>CHIP ID</b>	CID7	CID6	CID5	CID4	CID3	CID2	CID1	CID0
\$04	<b>CHIP REVISION</b>	REV7	REV6	REV5	REV4	REV3	REV2	REV1	REV0
\$08	<b>MEMORY CONFIG</b>	0	0	FSTRD	1	0	MSIZ2	MSIZ1	MSIZ0
\$0C	<b>DUMMY 0</b>	0	0	0	0	0	0	0	0
\$10	<b>DUMMY 1</b>	0	0	0	0	0	0	0	0
\$14	<b>BASE ADDRESS</b>	BAD31	BAD30	BAD29	BAD28	BAD27	BAD26	BAD25	BAD24
\$18	<b>DRAM CONTROL</b>	BAD23	BAD22	BAD21	BAD20	BAD19	BAD18	BAD17	BAD16
\$1C	<b>BCLK FREQUENCY</b>	BCK7	BCK6	BCK5	BCK4	BCK3	BCK2	BCK1	BCK0

A summary of the remaining CSR registers is shown in [Table 5-3](#), following. As with the first eight CSR registers, the summary shows the registers for the upper MCECC. The registers for the lower MCECC appear on D8-D15.

As with the first eight CSR registers, software should read and write to only the upper MCECC CSRs. The exception to this is the error logger, error address, and error syndrome registers. These registers contain information specific to each MCECC and the DRAMs which it controls, and as such should be treated separately. The base address of the CSRs is hard coded to the address \$FFF43000 for the MCECC pair on the first mezzanine board and \$FFF43100 for the MCECC pair on the second mezzanine board.

**Table 5-3. MCECC Internal Register Memory Map, Part 2**

MCECC Base Address = \$FFF43000 (1st); \$FFF43100 (2nd)

Register Offset	Register Name	Register Bit Names							
		D31	D30	D29	D28	D27	D26	D25	D24
\$20	<b>DATA CONTROL</b>	0	0	DERC	ZFILL	RWCKB	0	0	0
\$24	<b>SCRUB CONTROL</b>	RACODE	RADATA	HITDIS	SCRB	SCRBEN	0	SBEIEN	IDIS
\$28	<b>SCRUB PERIOD</b>	SBPD15	SBPD14	SBPD13	SBPD12	SBPD11	SBPD10	SBPD9	SBPD8
\$2C	<b>SCRUB PERIOD</b>	SBPD7	SBPD6	SBPD5	SBPD4	SBPD3	SBPD2	SBPD1	SBPD07
\$30	<b>CHIP PRESCALE</b>	CPS7	CPS6	CPS5	CPS4	CPS3	CPS2	CPS1	CPS0
\$34	<b>SCRUB TIME ON/OFF</b>	SRDIS	0	STON2	STON1	STON0	STOFF2	STOFF1	SRDIS
\$38	<b>SCRUB PRESCALE</b>	0	0	SPS21	SPS20	SPS19	SPS18	SPS17	SPS16
\$3C	<b>SCRUB PRESCALE</b>	SPS15	SPS14	SPS13	SPS12	SPS11	SPS10	SPS9	SPS85

**Table 5-3. MCECC Internal Register Memory Map, Part 2 (Continued)**

MCECC Base Address = \$FFF43000 (1st); \$FFF43100 (2nd)

\$40	<b>SCRUB PRESCALE</b>	SPS7	SPS6	SPS5	SPS4	SPS3	SPS2	SPS1	SPS0
\$44	<b>SCRUB TIMER</b>	ST15	ST14	ST13	ST12	ST11	ST10	ST9	ST8
\$48	<b>SCRUB TIMER</b>	ST7	ST6	ST5	ST4	ST3	ST2	ST1	ST0
\$4C	<b>SCRUB ADDR CNTR</b>	0	0	0	0	0	SAC26	SAC25	SAC24
\$50	<b>SCRUB ADDR CNTR</b>	SAC23	SAC22	SAC21	SAC20	SAC19	SAC18	SAC17	SAC16
\$54	<b>SCRUB ADDR CNTR</b>	SAC15	SAC14	SAC13	SAC12	SAC11	SAC10	SAC9	SAC8
\$58	<b>SCRUB ADDR CNTR</b>	SAC7	SAC6	SAC5	SAC4	07	0	0	0
\$5C	<b>ERROR LOGGER</b>	ERRLOG	ERD	ESCRB	ERA	EALT	0	MBE	SBE
\$60	<b>ERROR ADDRESS</b>	EA31	EA30	EA29	EA28	EA27	EA26	EA25	EA24
\$64	<b>ERROR ADDRESS</b>	EA23	EA22	EA21	EA20	EA19	EA18	EA17	EA16
\$68	<b>ERROR ADDRESS</b>	EA15	EA14	EA13	EA12	EA11	EA10	EA9	EA8
\$6C	<b>ERROR ADDRESS</b>	EA7	EA6	EA5	EA4	07	0	0	0
\$70	<b>ERROR SYNDROME</b>	S7	S6	S5	S4	S3	S2	S1	S0
\$74	<b>DEFAULTS1</b>	WRHDIS	STATCOL	FSTRD	SELI1	SELI0	RSIZ2	RSIZ1	RSIZ0
\$78	<b>DEFAULTS2</b>	FRC_OPN	XY_FLIP	REFDIS	TVECT	NOCACHE	RESST2	RESST1	RESST0



## Chip ID Register

The Chip ID Register is hard-wired to a hexadecimal value of \$81. The MCECC 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. Writes of any value other than \$0F to this register are ignored; however, the MCECC always terminates the cycles properly with TA\*.

(Difference from *MEMC040*: value = \$80 for MEMC040;  
value = \$81 for MCECC.)

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

## Chip Revision Register

The Chip Revision Register is hard-wired to reflect the revision level of the MCECC ASIC. The current value of this register is \$00. Writes to this register are ignored; however, the MCECC pair always terminates the cycles properly with TA\*.

(Difference from *MEMC040*: none between corresponding revisions of the two parts.)

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	RB4	RB3	MSIZ2	MSIZ1	MSIZ0
OPER	R	R	R	R	R	R	R	R
RESET	X	X	X	X	X	X	X	X

**MSIZ2-MSIZ0** MSIZ2-MSIZ0 together define the size of the total memory to be controlled by the MCECC pair. These bits reflect the RSIZ2-RSIZ0 bits in the Defaults Register 1.

MSIZ2	MSIZ1	MSIZ0	Memory Size
0	0	0	4MB using one 144-bit wide block of 256Kx4 DRAMs
0	0	1	8MB using two 144-bit wide block of 256Kx4 DRAMs
0	1	0	16MB using one 144-bit wide block of 1Mx4 DRAMs
0	1	1	32MB using two 144-bit wide blocks of 1Mx4 DRAMs
1	0	0	64MB using one 144-bit wide block of 4Mx4 DRAMs
1	0	1	128MB using two 144-bit wide blocks of 4Mx4 DRAMs
1	1	0	reserved
1	1	1	reserved

(Difference from *MEMC040*: NONE except that they reflect input pins on the MEMC040; while they reflect register bits that are initialized by the reset serial bit stream on the MCECC.)

**RB3** Read Bit 3 is a read only bit that is always 0.

(Difference from *MEMC040*:) bit = WPB (write-per-bit input strap status) for MEMC040; bit = 0 for MCECC (WPB = 0 on current versions of MVME162LX).

**RB4** Read Bit 4 is a read only bit that is always 1.

(Difference from *MEMC040*:) bit = EXTPEN (external parity enable input strap status) for MEMC040; bit = 1 for MCECC (EXTPEN = 1 on current versions of MVME162LX).

**FSTRD** FSTRD reflects the state of the FSTRD bit in the 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 to accommodate slower DRAM devices.

(Difference from *MEMC040*: NONE except that it is an input pin on the MEMC040; while it is a register bit that is initialized by the reset serial bit stream on the MCECC.)

5

## Dummy Register 0

Dummy Register 0 is hard-wired to all zeros. Writes to this register are ignored; however, the MCECC always terminates the cycles properly with TA\*.

(Difference from *MEMC040*: register = Alternate Status for MEMC040; register = \$00 for MCECC.)

ADR/SIZ	1st \$FFF4300C/2nd \$FFF4310C (8-bits)							
BIT	31	30	29	28	27	26	25	24
NAME	0	0	0	0	0	0	0	0
OPER	R	R	R	R	R	R	R	R
RESET	X	X	X	X	X	X	X	X

## Dummy Register 1

Dummy Register 1 is hard-wired to all zeros. Writes to this register are ignored; however, the MCECC always terminates the cycles properly with TA\*.

(Difference from *MEMC040*: register = Alternate Control for MEMC040; register = \$00 for MCECC.)

ADR/SIZ	1st <b>\$FFF43010</b> /2nd <b>\$FFF43110</b> (8-bits)							
BIT	31	30	29	28	27	26	25	24
NAME	0	0	0	0	0	0	0	0
OPER	R	R	R	R	R	R	R	R
RESET	X	X	X	X	X	X	X	X

5

## 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.

(Difference from *MEMC040*: none.)

The bit assignments for the Base Address Register are:

ADR/SIZ	1st <b>\$FFF43014</b> /2nd <b>\$FFF43114</b> (8-bits)							
BIT	31	30	29	28	27	26	25	24
NAME	BAD31	BAD30	BAD29	BAD28	BAD27	BAD26	BAD25	BAD24
OPER	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RESET	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS

## DRAM Control Register

The bit assignments for the DRAM Control Register are:

ADR/SIZ	1st \$FFF43018/2nd \$FFF43118 (8-bits)							
BIT	31	30	29	28	27	26	25	24
NAME	BAD23	BAD22	RWB5	SWAIT	RWB3	NCEIEN	NCEBEN	RAMEN
OPER	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
RESET	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.*

(Difference from *MEMC040*: none.)

**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.

(Difference from *MEMC040*:) bit = PAREN for *MEMC040*; bit = NCEBEN for MCECC (both accomplish basically the same thing, enabling TEA assertion for non-correctable errors).

**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.

(Difference from *MEMC040*: bit = PARINT for *MEMC040*; bit = NCEIEN for MCECC.)

**RWB3** Read/Write Bit 3 is a general purpose read/write bit

(Difference from *MEMC040*:) bit = WWP (write-wrong-parity) for *MEMC040*; bit = RWB (general purpose read write bit) for MCECC.

**SWAIT** Setting the SWAIT control bit causes the MCECC pair to wait for MI\* to be negated before starting a DRAM cycle in response to a local bus cycle to DRAM that does not have snooping inhibited. Clearing the SWAIT bit causes the MCECC pair to start a DRAM read cycle even before MI\* is negated during a snooped, local bus cycle. Note that the MCECC pair still waits for MI\* to be negated before enabling its data onto the local data bus and asserting TA\*/TEA\*. Additionally, setting the SWAIT bit causes the MCECC pair to wait for LOCKOK to be asserted before starting a DRAM cycle in response to a local bus cycle to DRAM that has LOCKL asserted. Clearing the SWAIT bit causes the MCECC pair to start a DRAM read even before LOCKOK is asserted during a local bus cycle that has LOCKL asserted. As with MI\*, the MCECC pair still waits for LOCKOK to be asserted before enabling its data onto the local data bus and asserting TA\*/TEA\*. SWAIT should normally be cleared, as it can provide a slight performance gain.

(Difference from *MEMC040*: when bit set - no difference for snooping, when bit cleared - MEMC040 REV. 1 no difference, MEMC040 REV. 0 - MCECC pair waits for MI\* negated in all cases of snooped writes whereas MEMC040 REV. 0 does not wait if snooped write is a line push. Additionally, for the MEMC040, SWAIT does not affect LOCKL, LOCKOK operation. For the MCECC, SWAIT affects LOCKL, LOCKOK operation as explained.)

**RWB5** Read/Write Bit 5 is a general purpose read/write bit.

(Difference from *MEMC040*:) bit = DMCTL (data-mux-control) for MEMC040; bit = RWB (general purpose read write bit) for MCECC (data-mux-control not required for MCECC pair).

**BAD22, BAD23** These are the lower two bits of the DRAM base address described in the previous register.

(Difference from *MEMC040*: none.)

## 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 25 MHz and \$21 for 32 MHz). The MCECC 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 1 MHz 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 25 MHz).

(Difference from *MEMC040*: none.)

**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	1 P

**Note** (None of the remaining registers have counterparts in the *MEMC040* because they are associated with functions contained only in the MCECC pair.)

## 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

5

### RWCKB

READ/WRITE CHECKBITS, when set, enables the data from the eight checkbits in this MCECC to be written and read on the local MC68040 data bus (bits 24-31 for upper MCECC, bits 8-15 for lower MCECC). 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 to perform a read-modify-write to DRAM. If the location to which check bits are being written, has a single or double bit err, 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 to be used 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.
- DERC** DISABLE ERROR CORRECTION, when set to one, disables the MCECC from correcting single bit errors. Specifically, read data is presented to the local MC68040 data bus unaltered from the DRAM array. 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 happen regardless of whether or not there is 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	RACODE	RADATA	HITDIS	SCRB	SCRBEN	0	SBEIEN	IDIS
OPER	R/W	R/W	R/W	R	R/W	R	R/W	R/W
RESET	V PLS	0 PLS	V PLS	0 PLS	0 PLS	X	0 PLS	0 PLS

- IDIS** When cleared, the Image DISable bit allows writes to the upper MCECC control registers to duplicate the data to the lower MCECC control registers. When IDIS is set, the lower MCECC control registers are written separately by the data on D00-D16. IDIS should only be set for test purposes.
- SBEIEN** Setting SBEIEN causes the logging of a single bit error to create a true pulse on the INT signal pin.

<b>SCRBEN</b>	This control bit enables the scrubber to operate. When SCRBEN is set, the MCECC 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 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 power-up, local, or software reset stops the scrubber.
<b>SCRB</b>	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.
<b>HITDIS</b>	This bit controls a function that is not currently used in the MCECC.
<b>RADATA</b>	This bit controls a function that is not currently used in the MCECC.
<b>RACODE</b>	This bit controls a function that is not currently used in the MCECC.

## 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 <b>\$FFF43028</b> /2nd <b>\$FFF43128</b> (8-bits)							
BIT	31	30	29	28	27	26	25	24
NAME	SBPD15	SBPD14	SBPD13	SBPD12	SBPD11	SBPD10	SBPD9	SBPD8
OPER	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RESET	1 PLS	1 PLS	1 PLS	1 PLS	1 PLS	1 PLS	1 PLS	1 PLS

5

## Scrub Period Register Bits 7-0

This register contains bits 7-0 of the Scrub Period Register.

ADR/SIZ	1st <b>\$FFF4302C</b> /2nd <b>\$FFF4312C</b> (8-bits)							
BIT	31	30	29	28	27	26	25	24
NAME	SBPD7	SBPD6	SBPD5	SBPD4	SBPD3	SBPD2	SBPD1	SBPD0
OPER	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RESET	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 1 MHz 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 <b>\$FFF43030</b> /2nd <b>\$FFF43130</b> (8-bits)							
BIT	31	30	29	28	27	26	25	24
NAME	CPS7	CPS6	CPS5	CPS4	CPS3	CPS2	CPS1	CPS0
OPER	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RESET	0 P	0 P	0 P	0 P	0 P	0 P	0 P	0 P

## Scrub Time On/Time Off Register

ADR/SIZ	1st \$FFF43034/2nd \$FFF43134 (8-bits)							
BIT	31	30	29	28	27	26	25	24
NAME	SRDIS	0	STON2	STON1	STON0	STOFF2	STOFF1	STOFF0
OPER	R/W	R	R/W	R/W	R/W	R/W	R/W	R/W
RESET	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:

STOFF2	STOFF1	STOFF0	Scrubber Time Off
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:

STON2	STON1	STON0	Scrubber Time On
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.

**SRDIS** SRDIS disables the scrubber from performing reads during scrub cycles. This mode should only be used when using the scrub function to perform zero fill of the DRAM. Setting this bit causes the zero fill to happen faster. This bit should not be changed while scrubbing is in process.

## Scrub Prescaler Counter (Bits 21-16)

The Scrub Prescaler Counter uses the 1MHz clock as an input to create the .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)							
BIT	31	30	29	28	27	26	25	24
NAME	0	0	SPS21	SPS20	SPS19	SPS18	SPS17	SPS16
OPER	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RESET	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 <b>\$FFF4303C</b> /2nd <b>\$FFF4313C</b> (8-bits)							
BIT	31	30	29	28	27	26	25	24
NAME	SPS15	SPS14	SPS13	SPS12	SPS11	SPS10	SPS9	SPS8
OPER	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RESET	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 <b>\$FFF43040</b> /2nd <b>\$FFF43140</b> (8-bits)							
BIT	31	30	29	28	27	26	25	24
NAME	SPS7	SPS6	SPS5	SPS4	SPS3	SPS2	SPS1	SPS0
OPER	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RESET	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS

## 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, reads to this address yield its 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 <b>\$FFF43044</b> /2nd <b>\$FFF43144</b> (8-bits)							
BIT	31	30	29	28	27	26	25	24
NAME	ST15	ST14	ST13	ST12	ST11	ST10	ST9	ST8
OPER	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RESET	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)							
BIT	31	30	29	28	27	26	25	24
NAME	ST7	ST6	ST5	ST4	ST3	ST2	ST1	ST0
OPER	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RESET	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 Scrub Address 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)							
BIT	31	30	29	28	27	26	25	24
NAME	0	0	0	0	0	SAC26	SAC25	SAC24
OPER	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RESET	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)							
BIT	31	30	29	28	27	26	25	24
NAME	SAC23	SAC22	SAC21	SAC20	SAC19	SAC18	SAC17	SAC16
OPER	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RESET	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)							
BIT	31	30	29	28	27	26	25	24
NAME	SAC15	SAC14	SAC13	SAC12	SAC11	SAC10	SAC9	SAC8
OPER	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RESET	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)							
BIT	31	30	29	28	27	26	25	24
NAME	SAC7	SAC6	SAC5	SAC4	0	0	0	0
OPER	R/W	R/W	R/W	R/W	R	R	R	R
RESET	0 PLS	0 PLS	0 PLS	0 PLS	X	X	X	X



## Error Logger Register

ADR/SIZ	1st \$FFF4305C/2nd \$FFF4315C (8-bits)							
BIT	31	30	29	28	27	26	25	24
NAME	ERRLOG	ERD	ESCRB	ERA	EALT	0	MBE	SBE
OPER	R/C	R	R	R	R	R	R	R
RESET	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 section on *Syndrome Decode*.)
- 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.
- 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 MC68040 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 is logged until it is cleared. The bit can only be set by logging an error and cleared by writing a one 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 clears it in the other. Any available error information in either MCECC should be recovered before clearing ERRLOG.

## Error Address (Bits 31-24)

This register reflects the value that was on bits 31-24 of the local MC68040 address bus at the last logging of an error.

ADR/SIZ	1st <b>\$FFF43060</b> /2nd <b>\$FFF43160</b> (8-bits)							
BIT	31	30	29	28	27	26	25	24
NAME	EA31	EA30	EA29	EA28	EA27	EA26	EA25	EA24
OPER	R	R	R	R	R	R	R	R
RESET	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS

5

## Error Address (Bits 23-16)

This register reflects the value that was on bits 23-16 of the local MC68040 address bus at the last logging of an error.

ADR/SIZ	1st <b>\$FFF43064</b> /2nd <b>\$FFF43164</b> (8-bits)							
BIT	31	30	29	28	27	26	25	24
NAME	EA23	EA22	EA21	EA20	EA19	EA18	EA17	EA16
OPER	R	R	R	R	R	R	R	R
RESET	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 MC68040 address bus at the last logging of an error.

ADR/SIZ	1st <b>\$FFF43068</b> /2nd <b>\$FFF43168</b> (8-bits)							
BIT	31	30	29	28	27	26	25	24
NAME	EA15	EA14	EA13	EA12	EA11	EA10	EA9	EA8
OPER	R	R	R	R	R	R	R	R
RESET	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS

## Error Address Bits (7-4)

This register reflects the value that was on bits 7-4 of the local MC68040 bus at the last logging of an error.

ADR/SIZ	1st \$FFF4306C/2nd \$FFF4316C (8-bits)							
BIT	31	30	29	28	27	26	25	24
NAME	EA7	EA6	EA5	EA4	0	0	0	0
OPER	R	R	R	R	R	R	R	R
RESET	0 PLS	0 PLS	0 PLS	0 PLS	X	X	X	X

5

## Error Syndrome Register

ADR/SIZ	1st \$FFF43070/2nd \$FFF43170 (16-bits)							
BIT	31	30	29	28	27	26	25	24
NAME	S7	S6	S5	S4	S3	S2	S1	S0
OPER	R	R	R	R	R	R	R	R
RESET	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS	0 PLS

### S7-S0

SYNDROME7-0 reflects the syndrome value at the last logging of an error. The eight bit code indicates the position of the data error. When all the bits are zero, there is no error. Note that if the logged error was non-correctable, then these bits are meaningless. Refer to the section on *Syndrome Decode*.

## Defaults Register 1

ADR/SIZ	1st \$FFF43074/2nd \$FFF43174 (8-bits)							
BIT	31	30	29	28	27	26	25	24
NAME	WRHDIS	STATCOL	FSTRD	SEL11	SEL10	RSIZ2	RSIZ1	RSIZ0
OPER	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RESET	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.

5

**RSIZ2-RSIZ0** 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 using one 144-bit wide block of 256Kx4 DRAMs
0	0	1	8MB using two 144-bit wide blocks of 256Kx4 DRAMs
0	1	0	16MB using one 144-bit wide block of 1Mx4 DRAMs
0	1	1	32MB using two 144-bit wide blocks of 1Mx4 DRAMs
1	0	0	64MB using one 144-bit wide block of 4Mx4 DRAMs
1	0	1	128MB using two 144-bit wide blocks of 4Mx4 DRAMs
1	1	0	reserved
1	1	1	reserved

The states of RSIZ2-0 after power-up, soft, or local reset, 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	0	\$FFF43000
0	1	\$FFF43100
1	0	\$FFF43200
1	1	\$FFF43300

The states of SELI1 and SELI0 after power-up, soft, or local reset, match those of the SELI1 and SELI0 bits from the reset serial bit stream.

**FSTRD** The FSTRD control bit determines the speed at which DRAM reads occur. When it is 1, DRAM reads happen at full speed. When it is 0, DRAM reads are slowed by one clock, unless they are already slowed by NCEBEN being set. FSTRD is cleared by Power-up or Local Reset if the FSTRD bit in the reset serial bit stream is 0. It is set by Power-up, soft, or Local Reset if the FSTRD bit in the reset serial bit stream is 1. Note that this bit can also be read in the Memory Configuration Register.

**STATCOL** When the STATCOL bit is set, the RACODE and/or RADATA bits in the Scrub Control Register can be set. When it is cleared, they cannot. STATCOL is initialized by Power-up, soft, or Local Reset to match the value of the STATCOL bit in the reset serial bit stream.

**WRHDIS** This bit controls a function that is not currently used in the MCECC.

## Defaults Register 2

ADR/SIZ	1st \$FFF43078/2nd \$FFF43178 (8-bits)							
BIT	31	30	29	28	27	26	25	24
NAME	FRC_OPEN	XY_FLIP	REFDIS	TVECT	NOCACHE	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 power-up, soft, or local reset, to match the RESST2-RESST0 bits from the reset serial bit stream.

**NOCACHE** When NOCACHE is cleared, the HITDIS bit in the Scrub Control Register can be cleared by software. When it is set, the HITDIS bit cannot be cleared. NOCACHE is initialized by power-up, soft, or local reset to match the NOCACHE bit in the reset serial bit stream. It should always be left at the default value of 1.

**TVECT** TVECT makes bidirectional signals work while running the vendors test vectors on this chip. It should be cleared for normal operation. It is initialized by power-up, soft, or local reset, to match the TVECT bit from the reset serial bit stream.

**REFDIS** When REFDIS is set, refreshing is disabled. This mode should only be used for testing, as DRAM must have refresh to operate correctly. REFDIS is initialized by power-up, soft, or local reset to match the REFDIS bit in the reset serial bit stream.

**XY\_FLIP** When XY\_FLIP is set, the opposite internal set of cache latches is selected. This bit should be used with caution and is for test vector coverage improvement.

**FRC\_OPN** When FRC\_OPN is set, the internal DRAM read latches are forced continuously open. This bit should be used with caution and is for test vector coverage improvement.

## 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 automatically but depends on software to perform enough dummy accesses to DRAM to meet the requirement. The number of required cycles is less 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 zeros all of the DRAM controlled by this MCECC pair at the rate of 100 MB/second when the BCLK pin is operating at 25 MHz. 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 SCRIBEN 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 SCRIB bit in the Scrub Control Register to be cleared. (Wait for bit 28 of offset \$24 = 0.)
3. Discontinue all accesses from the MC68040 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 SCRIBEN bit in the Scrub Control Register. (Set bit 27 of offset \$24.)
8. Ensure that the zero-fill has started by waiting for the SCRIB 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 time through, 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 Decode

A syndrome code value of \$00 indicates no error found. All other syndrome code values indicate an error with the bit in error decoded as shown in the following table. Note that BANK A corresponds to A3,A2 = 00, BANK B to A3,A2 = 01, BANK C to A3,A2 = 10, and BANK D to A3,A2 = 11.

Bank in Error	Bit in Error	Syndrome Code
BANK D	BIT 0/16	\$8C
BANK D	BIT 1/17	\$0D
BANK D	BIT 2/18	\$0E
BANK D	BIT 3/19	\$F4
BANK D	BIT 4/20	\$15
BANK D	BIT 5/21	\$16
BANK D	BIT 6/22	\$26
BANK D	BIT 7/23	\$25
BANK D	BIT 8/24	\$19
BANK D	BIT 9/25	\$1A
BANK D	BIT 10/26	\$1C
BANK D	BIT 11/27	\$E9
BANK D	BIT 12/28	\$2A
BANK D	BIT 13/29	\$2C
BANK D	BIT 14/30	\$4C
BANK D	BIT 15/31	\$4A



<b>Bank in Error</b>	<b>Bit in Error</b>	<b>Syndrome Code</b>
BANK C	BIT 0/16	\$23
BANK C	BIT 1/17	\$43
BANK C	BIT 2/18	\$83
BANK C	BIT 3/19	\$3D
BANK C	BIT 4/20	\$45
BANK C	BIT 5/21	\$85
BANK C	BIT 6/22	\$89
BANK C	BIT 7/23	\$49
BANK C	BIT 8/24	\$46
BANK C	BIT 9/25	\$86
BANK C	BIT 10/26	\$07
BANK C	BIT 11/27	\$7A
BANK C	BIT 12/28	\$8A
BANK C	BIT 13/29	\$0B
BANK C	BIT 14/30	\$13
BANK C	BIT 15/31	\$92

<b>Bank in Error</b>	<b>Bit in Error</b>	<b>Syndrome Code</b>
BANK B	BIT 0/16	\$C8
BANK B	BIT 1/17	\$D0
BANK B	BIT 2/18	\$E0
BANK B	BIT 3/19	\$4F
BANK B	BIT 4/20	\$51
BANK B	BIT 5/21	\$61
BANK B	BIT 6/22	\$62
BANK B	BIT 7/23	\$52
BANK B	BIT 8/24	\$91
BANK B	BIT 9/25	\$A1
BANK B	BIT 10/26	\$C1
BANK B	BIT 11/27	\$9E
BANK B	BIT 12/28	\$A2
BANK B	BIT 13/29	\$C2
BANK B	BIT 14/30	\$C4
BANK B	BIT 15/31	\$A4

<b>Bank in Error</b>	<b>Bit in Error</b>	<b>Syndrome Code</b>
BANK A	BIT 0/16	\$32
BANK A	BIT 1/17	\$34
BANK A	BIT 2/18	\$38
BANK A	BIT 3/19	\$D3
BANK A	BIT 4/20	\$54
BANK A	BIT 5/21	\$58
BANK A	BIT 6/22	\$98
BANK A	BIT 7/23	\$94
BANK A	BIT 8/24	\$64
BANK A	BIT 9/25	\$68
BANK A	BIT 10/26	\$70
BANK A	BIT 11/27	\$A7
BANK A	BIT 12/28	\$A8
BANK A	BIT 13/29	\$B0
BANK A	BIT 14/30	\$31
BANK A	BIT 15/31	\$29

<b>Bank in Error</b>	<b>Bit in Error</b>	<b>Syndrome Code</b>
UPPER/LOWER CHECKBITS	BIT 0	\$01
UPPER/LOWER CHECKBITS	BIT 1	\$02
UPPER/LOWER CHECKBITS	BIT 2	\$04
UPPER/LOWER CHECKBITS	BIT 3	\$08
UPPER/LOWER CHECKBITS	BIT 4	\$10
UPPER/LOWER CHECKBITS	BIT 5	\$20
UPPER/LOWER CHECKBITS	BIT 6	\$40
UPPER/LOWER CHECKBITS	BIT 7	\$80

# Serial Interface Connections



---

## Introduction

This appendix provides connection diagrams for the four serial ports on the MVME162LX 200/300 Series Embedded Controller board. These ports are typically connected to external serial devices through cables attached to the controller's front panel. The figures showing these diagrams are listed in the following table:

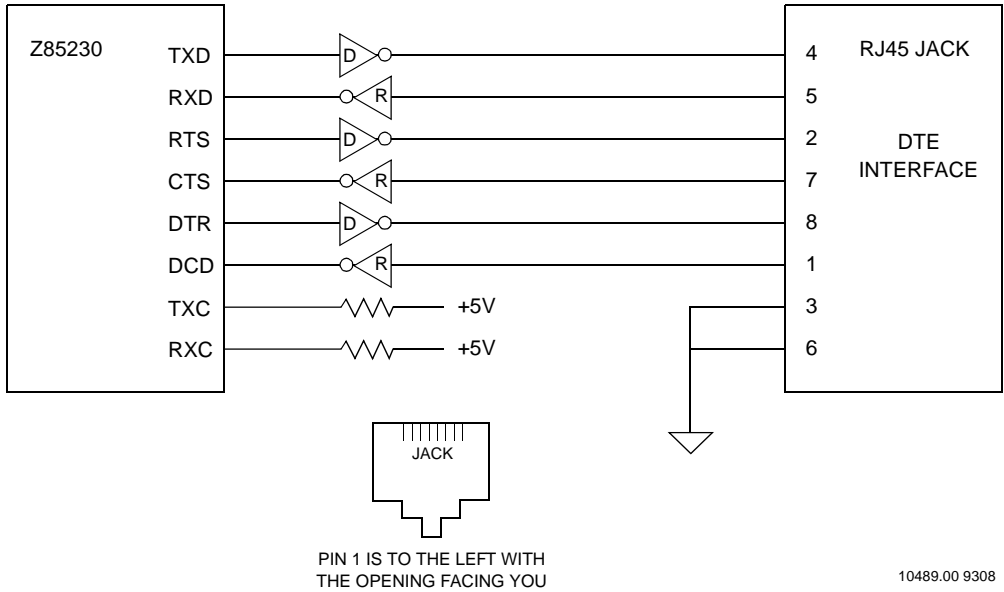
Figure Number	Name
A-1	Z85230 -to- RJ45 Adapter
A-2	DB25-DTE -to- RJ45 Adapter
A-3	DB25-DCE -to- RJ45 Adapter
A-4	RJ45 Adapter -to- RJ45 Adapter

The controller uses two Zilog Z85230 serial port controllers to implement the four serial communications interfaces. Each interface supports CTS, DCD, RTS, and DTR control signals as well as the TXD and RXD transmit/receive data signals. Because the serial clocks are omitted in the controller's design, serial communications are strictly asynchronous.

**Note** For additional information, refer to the *Serial Communications Interface* section in the *Operating Instructions and Functional Description* chapter of the *MVME162LX Embedded Controller Installation and Use* manual (200/300 Series version).

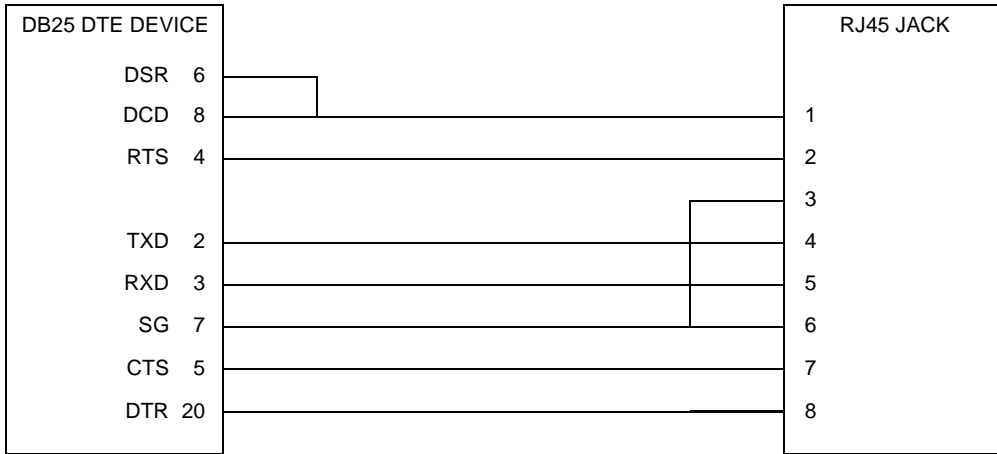
## Connection Diagrams

Figure A-1 shows the connections between the Z85230 and the RJ45 connectors.



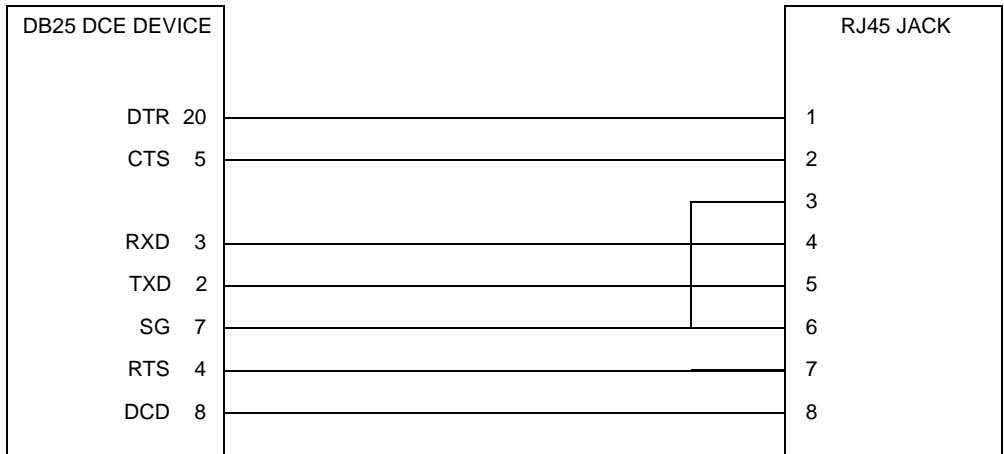
**Figure A-1. Serial Interface Connections: Z85230-to-RJ45 Adapter**

Figure A-2 shows the pin assignments required in a cable to adapt a DB25 DTE device to the RJ45 connectors.



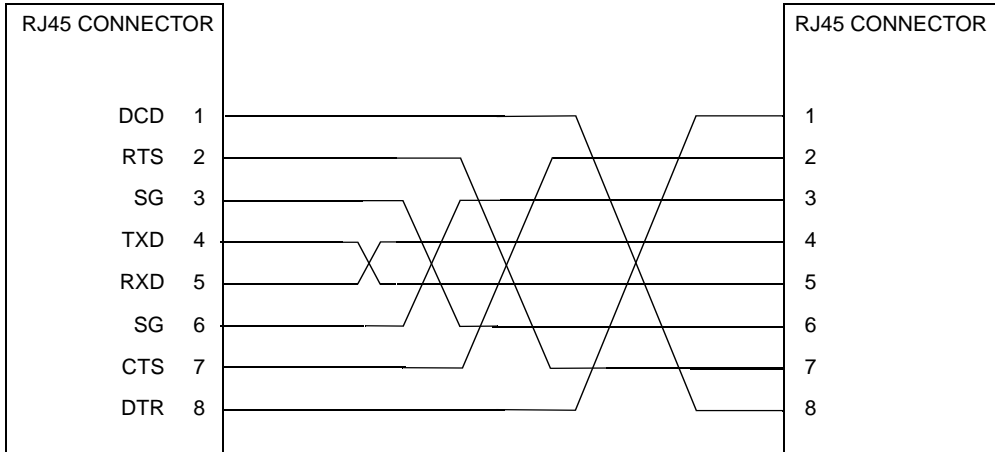
**Figure A-2. DB25-DTE-to-RJ45 Adapter**

Figure A-3 shows the pin assignments required in a cable to adapt a DB25 DCE device to the RJ45 connectors.



**Figure A-3. DB25-DCE-to-RJ45 Adapter**

Figure A-4 shows the pin assignments required in a typical eight-conductor serial cable having RJ45 connectors at both ends. Note that all wires are crossed.



**Figure A-4. Typical RJ45 Serial Cable**





---

## Introduction

This appendix provides information on how to use interrupts on the MVME162LX 200/300 Series Embedded Controller. It gives an example of how to generate and handle a VMEchip2 Tick Timer 1 interrupt on a controller board that has a VMEbus connection. Specific values have been given for the register writes.

**Note** Read this entire appendix before performing any of the procedures.

## Interrupt Handler Routine Setup

The following table explains how to set up a handler routine for the interrupt you will generate in the next section. Your interrupt handler should include the following features.

Step	Action and Reference
1.	Ensure the MC68040 vector base register is set up. Set the proper MC68040 exception vector location so the processor vectors to your interrupt handler location. You can determine proper exception vector location to set from the MC68040 vector base register, the VMEchip2 base register, and Table 2-3, in Chapter 2, from which you can determine the actual interrupt vector given on a Tick Timer 1 interrupt. Lower the MC68040 mask so the interrupt level you programmed is accepted. The <i>interrupt handler itself</i> should include the following (steps 2 through 5).
2.	Confirm the Tick Timer 1 interrupt occurred, by reading the status of bit 24 of the Interrupter Status Register at \$FFF40068. A high indicates an interrupt present.
3.	Clear Tick Timer 1 interrupt by writing a one 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 FAIL LED) on an appropriate count, such as 1000.
5.	Return from exception.

## Periodic Interrupt Example

This section explains how to generate an example of a VMEchip2 Tick Timer 1 periodic interrupt:

### Set Up Tick Timer 1

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 - <b>Bclock</b> (MHz). This gives a 1 MHz clock to the tick timers. <b>Bclock</b> is the bus clock rate, such as 25 MHz. $256 - 25 = \$E7$ .
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.
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.

### Setup 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 description of the <i>Vector Base Register</i> and to <a href="#">Table 2-3</a> , the Local Bus Interrupter Summary, both 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 one to enable Tick Timer 1 interrupts.
8.	I/O Control Register 1 \$FFF40088 (8 of 32 bits)	Write a one to bit 23 to enable interrupts from the VMEchip2. A zero masks <i>all</i> interrupts from the VMEchip2.



## Motorola Computer Group Documents

The Motorola publications listed below are referenced in this manual. You can obtain paper or electronic copies of Motorola Computer Group publications by:

- ❑ Contacting your local Motorola sales office
- ❑ Visiting MCG's World Wide Web literature site, <http://www.mcg.mot.com/literature>

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

<b>Document Title</b>	<b>Motorola Publication Number</b>
MVME162LX 200/300 Series Embedded Controller Installation and Use Guide	V162LX2-3A/IH2
MVME162Bug Diagnostics Manual	V162DIAA/UM
Debugging Package for Motorola 68K CISC CPUs User's Manual	68KBUG1/D and 2/D
MVME 162Bug Debugging Package User's Manual	MVME162BUG/D

To locate and view the most up-to-date product information in PDF or HTML format, visit <http://www.mcg.mot.com/literature>.

## Additional Motorola Publications

**C**

The following publications are also applicable to the MVME162LX 200/300 Series Embedded Controller and may provide you with additional information. They may be purchased through the sources listed in the table below.

**Table C-2. Additional Motorola Publications**

<b>Document Title and Source</b>	<b>Publication Number</b>
M68000 Family Reference Manual Literature Distribution Center for Motorola Telephone: (800) 441-2447 FAX: (602) 994-6430 or (303) 675-2150 E-mail: ldcformotorola@hibbertco.com	M68000FR
MC68040 Microprocessor User's Manual Literature Distribution Center for Motorola Telephone: (800) 441-2447 FAX: (602) 994-6430 or (303) 675-2150 E-mail: ldcformotorola@hibbertco.com	M68040UM

# Manufacturers' Publications

Refer to the table below for manufacturers' data sheets and/or user's manuals. For your convenience, 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

The following manufacturers' publications available:

**Table C-3. Manufacturers' Documents**

Document Title	Source
NCR 53C710 SCSI I/O Processor Data Manual, Order number: NCR53C710DM	NCR Corporation Microelectronics Products Div. 1635 Aero Plaza Drive Colorado Springs, CO 80916
NCR 53C710 SCSI I/O Processor Programmer's Guide, Order number: NCR53C710PG	
SGS-THOMSON 64K (8K x 8) Timekeeper® SRAM Data Sheet, Order number: M48T08/18	SGS-THOMSON Microelectronics Group Marketing Headquarters 1000 East Bell Road Phoenix, AZ 85022-2699
IndustryPack Logic Interface Specification, Revision 1.0, Order number: ANSI/VITA 4-1995	VITA (VMEbus International Trade Association) 7825 E. Gelding Dr., Ste. 104 Scottsdale, AZ 85260-3415
Z85230 Serial Communications Controller Data Sheet	Zilog Inc. 210 Hacienda Ave. Campbell, CA 95008-6609

## Related Specifications

Refer to the table below for related specifications. For your convenience, 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.

The following related specifications are available:

**Table C-4. Related Specifications**

Document Title and Source	Publication Number
Versatile Backplane Bus The Institute of Electrical and Electronics Engineers, Inc., 345 East 47th Street, New York, NY 10017 (VMEbus Specification). This is also available as Microprocessor system bus for 1 to 4 byte data, IEC 821 BUS, Bureau Central de la Commission Electrotechnique, Internationale; 3, rue de Varembe, Geneva, Switzerland.	ANSI/IEEE Std 1014-1987
ANSI Small Computer System Interface-2 (SCSI-2), Global Engineering Documents, P.O. Box 19539, Irvine, CA 92714.	3.131-198X, Revision 10c
IndustryPack Logic Interface Specification, GreenSpring Computers, Inc., 1204 O'Brien Drive, Menlo Park, CA 94025	Refer To Latest Revision
Z85230 Serial Communications Controller data sheet; Zilog, Inc., 210 Hacienda Ave., Campbell, California 95008-6609	Refer To Latest Revision
82596CA Local Area Network Coprocessor Data Sheet, 82596 User's Manual, Intel Corporation, Literature Sales, P.O. Box 58130, Santa Clara, CA 95052-8130.	290218 -and- 296853
NCR 53C710 SCSI I/O Processor, Data Manual Document NCR Corporation, Microelectronics Products Division, Colorado Springs, CO	#SCSIP-53C710

**Table C-4. Related Specifications (Continued)**

<b>Document Title and Source</b>	<b>Publication Number</b>
MK48T08(B) Timekeeper™ <i>-and-</i> 8Kx8 Zeropower™ RAM data sheet in Static RAMs Databook, SGS-THOMPSON Microelectronics Group; North & South American Marketing Headquarters, 1000 East Bell Road, Phoenix, AZ 85022-2699	DBSRAM71
8F008SA Flash Memory Data Sheet, Intel Literature Sales, P.O. Box 7641, Mt. Prospect, IL 60056-7641	2904351-001

**C**





## Numerics

- 16-bit memory space [4-18](#)
- 32-bit memory space [4-19](#)
- 32-Bit Memory Space, IPIC chip [4-19](#)
- 32-bit Prescaler Count Register,  
    MCchip [3-49](#)
- 53C710 SCSI controller interface,  
    MCchip [3-5](#)
- 82596CA LAN  
    Interface [3-3](#)  
    Interrupt Control Register,  
    MCchip [3-32](#)
- 8-bit memory space [4-17](#)

## A

- A16/D16 space [2-7](#)
- A16/D32 space [2-7](#)
- A24/D16 space [2-7](#)
- A32/D16 space [2-7](#)
- ABORT Switch  
    Interrupt Control Register,  
    MCchip [3-42](#)  
    Interrupt Address [1-11](#)
- Access and Watchdog Time Base Select  
    Register, MCchip [3-45](#)
- access timer, VMEbus [2-8](#)
- address modifier [2-36](#), [2-37](#)
- arbiter [2-18](#)
- arbitration, MCECC [5-8](#)

## B

- Base Address Register [5-18](#)
- BBRAM  
    configuration memory map [1-36](#)  
    interface [3-3](#)  
    TOD Clock Memory Map [1-36](#)
- BCLK Frequency Register [5-21](#)
- block diagram, full board [1-7](#)
- Board Control Register [2-70](#)
- board fail [2-70](#)
- board failure [2-70](#)
- Board Layout,  
    MVME162LX 200/300 Series [1-13](#)
- board level hardware description [1-1](#)
- board, Real-time Clock and operation [1-7](#)
- Bus Clock Register, MCchip [3-39](#)
- bus error processing [1-45](#)
- bus timers [1-53](#), [2-19](#)

## C

- cache coherency [1-43](#), [4-3](#)  
    MCECC [5-4](#)
- chip defaults, MCECC [5-9](#)
- Chip ID Register [4-9](#)  
    MCECC [5-15](#)
- Chip Prescaler Counter [5-25](#)
- Chip Revision Register [4-9](#)  
    MCECC [5-15](#)
- clock frequency [3-39](#)
- clocking timing functions, MCchip [3-49](#)
- configuration jumpers [1-11](#)
- cycle types, MCECC [5-5](#)

**D**

Data Control Register [5-22](#)  
date and time, starting/setting board  
real-time clock's [1-7](#)  
DB25-DCE-to-RJ45 Adapter, serial  
connection diagram [A-4](#)  
DB25-DTE-to-RJ45 Adapter, serial  
connection diagram [A-3](#)  
Defaults Register  
#1 [5-34](#)  
#2 [5-36](#)  
detailed I/O memory maps [1-20](#)  
DMAC  
Byte Counter [2-59](#)  
command table format [2-53](#)  
Control Register 1 (bits 0-7) [2-56](#)  
Control Register 2 (bits 0-7) [2-58](#)  
Control Register 2 (bits 8-15) [2-57](#)  
Local Bus Address Counter [2-59](#)  
LTO error [1-49](#)  
offboard error [1-49](#)  
parity error [1-48](#)  
Status Register [2-63](#)  
TEA - cause unidentified [1-50](#)  
Ton/Toff Timers and  
VMEbus Global Timeout Control  
Register [2-65](#)  
VMEbus Address Counter [2-59](#)  
VMEbus error [1-48](#)  
VMEbus requester [2-15](#)  
documentation  
non-Motorola [C-3](#)  
other applicable Motorola  
publications [C-2](#)  
double bit error  
(cycle type = burst read) [5-6](#)  
(cycle type = non-burst read) [5-6](#)  
(cycle type = non-burst write) [5-6](#)  
(cycle type = scrub) [5-7](#)  
DRAM  
Control Register [5-19](#)  
Parity Error Interrupt Control

Register, MCchip [3-22](#)  
types and amounts available in  
200/300 Series [1-5](#)

**Dummy Register**

#0 [5-17](#)

#1 [5-18](#)

dynamic bus sizing, supported by  
VMEbus master [2-7](#)

**E**

ECC DRAM protection, MCECC [5-5](#)

**EPROM**

Access time [3-40](#)

Access Time Control Register,  
MCchip [3-40](#)

EPROM/Flash interface [3-2](#)

**Error Address**

Bits 23-16 [5-32](#)

Bits 31-24 [5-32](#)

Bits 15-8 [5-32](#)

Bits 7-4 [5-33](#)

error conditions on the MVME162LX [1-46](#)

error indication, LANC [3-5](#)

Error Logger Register [5-31](#)

error logging,

MCECC [5-7](#)

reporting [4-3](#)

Error Syndrome Register [5-33](#)

**Ethernet**

LAN memory map [1-34](#)

Transceiver interface [1-6](#)

example of the proper use of bus timers [1-53](#)

**F****Features Summary**

MVME162LX 200/300 Series [1-5](#)

**features**

IPIC chip [4-1](#)

MCECC [5-1](#)

Flash Access Time Control Register,  
MCchip [3-41](#)

Flash memory device [1-5](#)

---

Flash/EPROM interface [3-2](#)  
functional blocks, VMEchip2 [2-4](#)  
functional description  
    board [1-7](#)  
    IPIC [4-2](#)  
    MC chip [3-2](#)

## G

GCSR [2-21](#)  
    board address [2-48](#)  
    group address [2-47](#)  
    programming model  
    VMEchip2 [2-102](#)  
General Control Register [3-13, 4-14](#)  
general description  
    IPIC [4-2](#)  
    MCECC [5-2](#)  
General Purpose  
    I/O (GPIO) pins, VMEchip2 [2-97](#)  
    MCchip [3-35](#)  
    Readable Jumpers Header [1-8](#)  
General Purpose  
    Register #0 [2-108](#)  
    Register #1 [2-109](#)  
    Register #2 [2-109](#)  
    Register #3 [2-109](#)  
    Register #4 [2-110](#)  
    Register #5 [2-110](#)  
Global Control and Status Registers [2-21](#)  
global reset [2-19](#)  
GPI inputs, addresses [1-11](#)

## I

I/O and ID space accesses [4-24](#)  
I/O Control  
    Register #1, VMEchip2 [2-97](#)  
    Register #2, VMEchip2 [2-98](#)  
    Register #3, VMEchip2 [2-99](#)  
I/O  
    interfaces [1-6](#)  
    map decoders [2-7](#)

IACK daisy-chain driver [2-18](#)  
ID Register  
    MCchip [3-12](#)  
indivisible cycles [1-54](#)  
IndustryPack  
    Interface Controller (IPIC) [4-1](#)  
    specification [C-3](#)  
initialization [5-37](#)  
interrupt acknowledge map [1-42](#)  
Interrupt Clear  
    Register (bits 16-23) [2-86](#)  
    Register (bits 24-31) [2-85](#)  
    Register (bits 8-15) [2-87](#)  
interrupt handler [2-19](#)  
interrupt level  
    LANC [3-32](#)  
Interrupt Level  
    Register 1 (bits 0-7) [2-89](#)  
    Register 1 (bits 16-23) [2-88](#)  
    Register 1 (bits 24-31) [2-88](#)  
    Register 1 (bits 8-15) [2-89](#)  
    Register 2 (bits 0-7) [2-91](#)  
    Register 2 (bits 16-23) [2-90](#)  
    Register 2 (bits 24-31) [2-90](#)  
    Register 2 (bits 8-15) [2-91](#)  
    Register 3 (bits 0-7) [2-93](#)  
    Level Register 3 (bits 16-23) [2-92](#)  
    Register 3 (bits 24-31) [2-92](#)  
    Register 3 (bits 8-15) [2-93](#)  
    Register 4 (bits 0-7) [2-95](#)  
    Register 4 (bits 16-23) [2-94](#)  
    Register 4 (bits 24-31) [2-94](#)  
    Register 4 (bits 8-15) [2-95](#)  
interrupt logic, ABORT switch [3-42](#)  
Interrupt Usage [B-1](#)  
Interrupt Vector  
    Base Register [3-14](#)  
    base register encoding  
    priority, MCchip [3-15](#)  
interrupt vectors [1-43](#)  
interrupter [2-19](#)

interrupts [4-3](#)  
 interrupts, used by MC68040 [1-43](#)  
 introduction  
     Guide [1-1](#)  
     IPIC chip [4-1](#)  
     MCchip [3-1](#)  
     MCECC [5-1](#)  
     VMEchip2 [2-1](#)  
 IP Modules  
     illegal accesses from external  
         VMEbus master [1-55](#)  
 IP RESET Register [4-16](#)  
 IP to local bus data routing [4-22](#)  
 IP\_a  
     I/O space [4-20](#)  
     ID space [4-21](#)  
 IP\_a or Double Size IP\_ab Memory  
     Base Address Registers [4-11](#)  
 IP\_a, IP\_b, IP\_c, and IP\_d [4-13](#), [4-14](#)  
     Memory Base Address  
     Registers [4-10](#)  
     Memory SizeRegisters [4-12](#)  
 IP\_ab I/O space [4-20](#)  
 IP\_b Memory Base Address Registers [4-11](#)  
 IP\_c or Double Size IP\_cd Memory Base Ad-  
     dress Registers [4-11](#)  
 IP\_d Memory Base Address Registers [4-11](#)  
 IPIC [4-1](#)  
     functional description [4-2](#)  
     general description [4-2](#)  
 IPIC chip  
     local bus to IndustryPack  
     addressing [4-17](#)  
     Overall Memory Map [4-5](#)  
 IPIC memory map [4-5](#)  
 IPIC overall memory map [4-5](#)  
 IRQ0 and IRQ1 Interrupt Control  
     Registers [4-13](#)

**J**

jumper header J11 [3-2](#)

**L****LAN**

interface, MC chip [3-3](#)  
 LTO error [1-51](#)  
 offboard error [1-51](#)  
 parity error [1-50](#)

**LANC**

interrupt [3-5](#)  
 Bus error [3-5](#)  
 Error Status Register [3-31](#)  
 Bus Error Interrupt Control  
 Register, MCchip [3-33](#)  
 LCSR memory map [2-24](#)

LCSR programming model [2-22](#)

light- emitting diodes (LEDs) [1-6](#)

**local bus**

access timer [3-1](#)  
 error condition [3-34](#)  
 slave [2-6](#)  
 timer, MC2 chip [3-9](#)

**local bus interrupter**

Interrupt handler [2-19](#)  
 Enable Register (bits 0-7) [2-83](#)  
 Enable Register (bits 16-23) [2-81](#)  
 Enable Register (bits 24-31) [2-80](#)  
 Enable Register (bits 8-15) [2-82](#)  
 Status Register (bits 0-7) [2-79](#)  
 Status Register (bits 16-23) [2-77](#)  
 Status Register (bits 24-31) [2-76](#)  
 Status Register (bits 8-15) [2-78](#)  
 Summary [2-75](#)

local bus memory map [1-14](#)

---

- Local Bus Slave (VMEbus Master)
  - Address Translation Address Register 4 [2-44](#)
  - Address Translation Select Register 4 [2-44](#)
  - Attribute Register 1 [2-47](#)
  - Attribute Register 2 [2-46](#)
  - Attribute Register 3 [2-45](#)
  - Attribute Register 4 [2-44](#)
  - Ending Address Register 1 [2-41](#)
  - Ending Address Register 2 [2-42](#)
  - Ending Address Register 3 [2-42](#)
  - Ending Address Register 4 [2-43](#)
  - Starting Address Register 1 [2-41](#)
  - Starting Address Register 2 [2-42](#)
  - Starting Address Register 3 [2-42](#)
  - Starting Address Register 4 [2-43](#)
- Local Bus To VMEbus
  - Enable Control Register [2-49](#)
  - I/O Control Register [2-50](#)
  - Interface [1-11](#)
  - VMEbus DMA controller [2-12](#)
  - VMEbus interface, VMEchip2 [2-6](#)
  - VMEbus Requester [2-8](#)
  - VMEbus Requester Control Register [2-55](#)
- Local Control and Status Registers (LCSR) [2-8, 2-21](#)
- local DRAM parity error [1-45](#)
- Local I/O Devices memory map [1-17](#)
- local reset [2-19, 2-70](#)
- LSB Prescaler Count Register [3-17](#)

## M

- map decoder
  - user-programmable [2-6](#)
- master interrupt enable (MIEN) [2-98](#)
- MC2 chip
  - local bus timer [3-9](#)
- MC68040-bus master support
  - for 82596CA [3-4](#)

- MCchip [3-1](#)
  - functional description [3-2](#)
  - General Purpose Inputs Register [3-35](#)
  - ID Register [3-12](#)
  - initialization [3-2](#)
  - Parity DRAM Control Register [3-46](#)
  - Parity DRAM memory controller [3-6](#)
  - programming model [3-12](#)
  - Revision Register [3-13](#)
  - summary of major features [3-1](#)
- MCchip registers, memory map [3-10](#)
- MCECC [5-1](#)
  - features [5-1](#)
  - internal register memory map [5-12](#)
  - specifications [5-3](#)
  - key for bit definitions in CSR [5-11](#)
  - programming model [5-10](#)
- Memory Configuration Register [5-16](#)
- memory map
  - BGRAM configuration area [1-36](#)
  - Ethernet LAN [1-34](#)
  - MCchip registers [3-10](#)
  - MCECC Internal Register [5-12](#)
  - SCSI [1-34](#)
  - time-of-day clock [1-38](#)
  - VMEchip2 GCSR [1-26, 2-105](#)
  - VMEchip2 LCSR [1-22, 2-24](#)
  - Z85230 SCC register [1-33](#)
- memory maps
  - detailed I/O [1-20](#)
  - overall board [1-14](#)
- memory space accesses, IPIC chip [4-22](#)
- microprocessor [1-5](#)
  - available in 200/300 Series [1-5](#)
- Miscellaneous Control Register, VMEchip2 [2-100](#)

- MPU
- local bus timeout 1-47
  - offboard error 1-46
  - parity error 1-46
  - Status and DMA Interrupt Count Register 2-62
  - Status Register, MCchip 3-48
  - TEA - cause unidentified 1-47
- MVIP IndustryPack interfaces 1-6
- MVME162Bug 3-2
- MVME162LX 200/300 Series
- Features Summary 1-5
  - Block Diagram 1-9
  - Version Register, MCchip 3-36
- N**
- no address increment DMA transfers 2-14
  - no VMEbus interface option 1-8
  - No VMEbus Interface Option (one model only) 1-8, 3-49
  - Non-Volatile RAM (NVRAM) 1-5
  - normal address range 1-14
- O**
- overall memory map 4-5
  - overview 1-3
    - MCchip 3-2
- P**
- PAL contents 3-35, 3-36
  - Parity DRAM and SRAM Memory Controller Registers, MCchip 3-25
  - Parity DRAM
    - Control Register, MCchip 3-46
    - memory controller, MCchip 3-6
    - performance, MCchip 3-6
    - Size Control Bit Encoding, MCchip 3-27
    - Space Base Address Register 3-25
    - Space Base Address Register, MCchip 3-25
    - Space Size Register, MCchip 3-27
  - Parity DRAM/SRAM Options Register, MCchip 3-28, 3-35, 3-36
  - performance
    - IPIC chip 4-2
    - MCECC 5-2
  - power save mode, board's real-time clock (storage) 1-8
  - Prescaler 2-16
    - Clock Adjust Register 3-18
    - Register 2-67
    - Counter 2-73
  - priority
    - interrupt 3-14
    - PRI mode 2-18
  - programming model 4-6
    - MCchip 3-12
    - MCECC 5-10
  - programming
    - GCSR 2-104
    - local bus interrupter 2-74
    - local bus to VMEbus map decoders 2-39
    - tick timers, MCchip 3-16
    - VMEbus slave map decoders 2-28
    - VMEchip2 DMA controller 2-52
  - PROM Decoder, SRAM and DMA Control Register 2-54
  - PROM/ EPROM sockets 1-5
  - publications
    - Non-Motorola C-3
- R**
- Real-time Clock and operation, board's 1-7
  - redundant functions in the VMEchip2 and MCchip 1-10
  - Refresh Rate 3-39
  - refresh, MCECC 5-8
  - register map, MCchip 3-10
  - Related Documentation C-1
  - reset driver 2-19

---

## Reset Status

MCchip 3-12

RESET Switch Control Register 3-43

ROM Control Register 2-51

ROM0 bit 2-54

Round Robin Select (RRS) Mode 2-18

## S

### SCC interface

MC chip 3-1

MC2 chip 3-1

MCchip 3-7

SCC Interrupt Control Register,

MCchip 3-23

SCC Register addresses 1-33

Scrub Address Counter

(Bits 15-8) 5-30

(Bits 23-16) 5-30

(Bits 26-24) 5-29

(Bits 7-4) 5-30

Scrub Control Register 5-23

Scrub Period Register

(Bits 15-8) 5-25

(Bits 7-0) 5-25

Scrub Prescaler Counter

(Bits 15-8) 5-28

(Bits 21-16) 5-27

(Bits 7-0) 5-28

Scrub Time On/Time Off Register 5-26

Scrub Timer Counter

(Bits 15-8) 5-28

(Bits 7-0) 5-29

scrub, MCECC 5-8

### SCSI

bus interface 1-6

controller interface, MCchip 3-5

coprocessor interface 3-1

Error Status Register 3-34

Error Status Register, MCchip 3-34

Interrupt Control Register,

MCchip 3-37

LTO error 1-53

### SCSI (Cont'd)

memory map 1-34

offboard error 1-52

parity error 1-52

specification C-3

serial cables A-3

Serial Interface Connections

Z85230-to-RJ45 Adapter

diagram A-2

serial port controllers 1-6

serial ports 1-6

short I/O area 2-7

single (SGL) mode 2-18

single bit error

(cycle type = burst read) 5-5

(cycle type = non-burst read) 5-5

(cycle type = non-burst write) 5-6

(cycle type = scrub) 5-7

snoop 2-34, 2-37

Snoop Control SC1-SC0 3-4

Software Interrupt Set Register

(bits 8-15) 2-84

software interrupts 1-6

software support considerations 1-43

sources of local BERR\* 1-44

specifications C-3

speeds

EPROM 3-40

Flash 3-41

SRAM 1-5, 2-54

memory controller, MCchip 3-5

size control bit encoding 3-29

Space Base Address Register 3-26

Space Size Register 3-29

with battery backup 1-5

summary of major features

MCchip 3-1

VMEchip2 2-1

switches 1-6

ABORT and RESET 1-6

syndrome decode 5-38

system reset 2-70

**T**

Table 2-60  
 Table Address Counter 2-60  
 tick and watchdog timers 2-16  
 Tick Timer 1 and 2  
   Compare and Counter  
   Registers 3-16  
   Control Registers, MCchip 3-19  
 Tick Timer 1  
   Compare Register 2-68, 3-16  
   Control Register 2-73, 3-19  
   Counter 2-68, 3-17  
   Interrupt Control Register 3-20  
 Tick Timer 2  
   Compare Register 2-69, 3-17  
   Control Register 2-72, 3-19  
   Counter 2-69, 3-17  
   Interrupt Control Register 3-20  
 Tick Timer 3 and 4  
   Compare and Counter Registers,  
   MCchip 3-38  
   Control Registers, MCchip 3-24  
 Tick Timer 3  
   Compare Register 3-38  
   Control Register 3-24  
   Counter 3-38  
   Interrupt Control Register 3-20  
 Tick Timer 4  
   Compare Register 3-38  
   Control Register 3-24  
   Counter 3-38  
   Interrupt Control Register 3-20  
 Tick Timer Interrupt Control Registers  
   MCchip 3-20  
 tick timers 1-6  
   MCchip 3-8  
 time-of-day clock 1-5  
   memory map 1-38  
 timers 1-6  
   VMEbus 2-8

**Transfer**

  Modifiers TM2-TM0 3-4  
   Types TT1-TT0 3-4  
 triple (or greater) bit error  
   (cycle type = burst read) 5-6  
   (cycle type = non-burst read) 5-6  
   (cycle type = non-burst write) 5-7  
   (cycle type = scrub) 5-7  
 Typical RJ45 Serial Cable, serial connection  
 diagram A-5

**V**

Vector Base Register 2-96  
 Version Register, MCchip 3-35  
 VME Access, Local Bus and Watchdog  
   Timeout Control Register 2-66  
 VMEbus  
   specification C-3  
   access timeout 1-44  
   accesses to the local bus 1-42  
   Arbiter Timeout Control  
   Register 2-64  
   BERR\* 1-44  
   interface 1-6  
   Interface, no VMEbus option  
   (one model only) 1-8  
   interrupter 2-17  
   Interrupter Control Register 2-61  
   Interrupter Vector Register 2-62  
   maps, creating 2-6  
   master 2-7  
   memory map 1-42  
   short I/O memory map 1-42  
   Slave Address Modifier Select  
   Register #1 2-38  
   Slave Address Modifier Select  
   Register #2 2-36  
   Slave Address Translation Address  
   Offset Register #1 2-32  
   Slave Address Translation Address  
   Offset Register #2 2-34



---

## VMEbus (Cont'd)

- Slave Address Translation Select Register #1 [2-33](#)
- Slave Address Translation Select Register #2 [2-34](#)
- Slave Ending Address Register #1 [2-31](#)
- Slave Ending Address Register#2 [2-31](#)
- Slave GCSR Group Address Register [2-47](#)
- Slave Starting Address Register #1 [2-31](#)
- Slave Starting Address Register #2 [2-31](#)
- Slave Write Post and Snoop Control Register #1 [2-37](#)
- Slave Write Post and Snoop Control Register #2 [2-35](#)
- system controller [2-18](#)
- to local bus interface [1-11](#), [2-10](#)

## VMEchip2 [1-45](#), [2-1](#)

- Block diagram [2-5](#)
- Board Status/Control, Memory map, LCSR Summary [2-24](#)
- ID Register [2-106](#)
- LM/SIG Register [2-106](#)
- Register [2-108](#)
- Revision Register [2-106](#)

## W

- watchdog timer [1-6](#), [2-17](#)
  - Control Register [2-71](#)
  - Control Register, MCchip [3-44](#)
  - MCchip [3-8](#)
- write post [2-34](#), [2-37](#)
  - buffer [2-7](#)
  - timer [2-8](#)
- write posting [2-40](#)
  - definition [2-7](#)
- write-protect feature, Flash memory [3-3](#)

## Z

- Z85230
  - SCC interface, MCchip [3-7](#)
  - SCC Register addresses [1-33](#)
- Z85230-to-RJ45 Adapter, serial connection diagram [A-2](#)



**MVME162LX  
200/300 Series  
Embedded Controller  
Programmer's  
Reference Guide**