MVME162FX Embedded Controller Programmer's Reference Guide

V162FXA/PG1

Notice

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

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

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

Restricted Rights Legend

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

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

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

Preface

This manual provides board level information and detailed ASIC chip information including register bit descriptions for the MVME162FX Embedded Controller. The information contained in this manual applies to the following MVME162FX models:

MVME162-410	MVME162-420	MVME162-430
MVME162-411	MVME162-421	MVME162-431
MVME162-412	MVME162-422	MVME162-432
MVME162-413	MVME162-423	MVME162-433
MVME162-440	MVME162-450	MVME162-460
MVME162-441	MVME162-451	MVME162-461
MVME162-442	MVME162-452	MVME162-462
MVME162-443	MVME162-453	MVME162-463
MVME162-510A	MVME162-520A	MVME162-530A
MVME162-511A	MVME162-521A	MVME162-531A
MVME162-512A	MVME162-522A	MVME162-532A
MVME162-513A	MVME162-523A	MVME162-533A

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

A basic knowledge of computers and digital logic is assumed.

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

Manual Terminology

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

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

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

Unless otherwise specified, all address references are in hexadecimal.

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

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.

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

WARNING

This equipment generates, uses, and can radiate radio frequency energy and if not installed and used in accordance with the instructions manual, may cause interference to radio communications. It has been tested and found to comply with the limits for a Class A Computing Device pursuant to Subpart J of part 15 of FCC rules, which are designed to provide reasonable protection against such interference when operated in a commercial environment. Operation of this equipment in a residential area is likely to cause interference in which case the user, at the user's own expense, will be required to take whatever measures necessary to correct the interference.

Related Documentation

The publications in the table below are applicable to the MVME162FX and may provide additional helpful information. If not shipped with this product, they may be purchased by contacting your local Motorola sales office.

Document Title	Motorola Publication Number
MVME162FX Embedded Controller Installation and Use	V162FXA/IH
MVME162Bug Diagnostics Manual	V162DIAA/UM
Debugging Package for Motorola 68K CISC CPUs User's Manual	68KBUG1/D and 68KBUG2/D
Single Board Computers SCSI Software User's Manual	SBCSCSI/D
MVME712M Transition Module and P2 Adapter Board User's Manual	MVME712M/D
MVME712-12, MVME712-13, MVME712A, MVME712AM, and MVME712B Transition Modules and LCP2 Adapter Board User's Manual	MVME712A/D
SIMM09 Serial Interface Module Installation Guide	SIMM09A/IH
M68040 Microprocessors User's Manual	M68040UM
M68000 Family Reference Manual	M68000FR

Notes Although not shown in the above list, each Motorola Computer Group manual publication number is suffixed with characters which represent the revision level of the document, such as "/*xx*2" (the second revision of a manual); a supplement bears the same number as a manual but has a suffix such as "/*xx*2A1" (the first supplement to the second edition of the manual).

You can obtain connector interconnect signal information, parts lists, and schematics for the MVME162FX free of charge by contacting your local Motorola sales office. They will relay the information.

The following publications are also available from the sources indicated.

Versatile Backplane Bus: VMEbus, ANSI/IEEE Std 1014-1987, The Institute of Electrical and Electronics Engineers, Inc., 345 East 47th Street, New York, NY 10017 (VMEbus Specification). (This is also *Microprocessor System Bus for 1 to 4 Byte Data*, IEC 821 BUS, Bureau Central de la Commission Electrotechnique Internationale; 3, rue de Varembé, Geneva, Switzerland.)

ANSI Small Computer System Interface-2 (SCSI-2), Draft Document X3.131-198X, Revision 10c; Global Engineering Documents, P.O. Box 19539, Irvine, CA 92714.

IndustryPack Logic Interface Specification, Revision 1.0; GreenSpring Computers, Inc., 1204 O'Brien Drive, Menlo Park, CA 94025.

Z85230 Serial Communications Controller data sheet; Zilog, Inc., 210 Hacienda Ave., Campbell, California 95008-6609.

82596CA Local Area Network Coprocessor data sheet, Order Number 290218; and 82596 User's Manual, Order Number 296443-001; Intel Corporation, Literature Sales, P.O. Box 58130, Santa Clara, CA 95052-8130.

NCR 53C710 SCSI I/O Processor, data manual document #SCSIP-53C710; NCR Corporation, Microelectronics Products Division, Colorado Springs, CO.

MK48T08(B) Timekeeper TM and 8Kx8 Zeropower TM RAM data sheet in Static RAMs Databook, Order Code DBSRAM71; SGS-THOMPSON Microelectronics Group; North & South American Marketing Headquarters, 1000 East Bell Road, Phoenix, AZ 85022-2699.

28F008SA Flash Memory Data Sheet, Order Number 290435-001; Intel Literature Sales, P.O. Box 7641, Mt. Prospect, IL 60056-7641.

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

Delta Series™, MC68040, VMEexec™,VMEmodule™, and VMEsystem™ are trademarks of Motorola, Inc.

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

PowerPC™ is a trademark of IBM Corp, and is used by Motorola, Inc. under license from IBM Corp.

Timekeeper™ and Zeropower™ are trademarks of Thompson Components.

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

©Copyright Motorola, Inc. 1996
All Rights Reserved
Printed in the United States of America
August 1996

Contents

Chapter 1 Board Description and Memory Maps

Introduction 1-1
Overview 1-1
Requirements 1-3
Features 1-3
Block Diagram 1-5
Functional Description 1-5
No-VMEbus-Interface Option 1-6
VMEbus Interface and VMEchip2 1-7
Memory Maps 1-8
Local Bus Memory Map 1-8
Normal Address Range 1-8
Detailed I/O Memory Maps 1-13
BBRAM/TOD Clock Memory Map 1-31
Interrupt Acknowledge Map 1-37
VMEbus Memory Map 1-37
VMEbus Accesses to the Local Bus 1-37
VMEbus Short I/O Memory Map 1-37
Software Support Considerations 1-38
Interrupts 1-38
Cache Coherency 1-38
Sources of Local BERR* 1-39
Local Bus Time-out 1-39
VMEbus Access Time-out 1-39
VMEbus BERR* 1-39
Local DRAM Parity Error 1-40
VMEchip2 1-40
Bus Error Processing 1-40
Description of Error Conditions on the MVME162FX 1-41
MPU Parity Error 1-41
MPU Off-board Error 1-41
MPU TEA - Cause Unidentified 1-42
MPU Local Bus Time-out 1-42
DMAC VMEbus Error 1-43

DMAC Parity Error 1-43
DMAC Off-board Error 1-44
DMAC LTO Error 1-44
DMAC TEA - Cause Unidentified 1-45
LAN Parity Error 1-45
LAN Off-board Error 1-46
LAN LTO Error 1-46
SCSI Parity Error 1-46
SCSI Off-board Error 1-47
SCSI LTO Error 1-47
Example of the Proper Use of Bus Timers 1-48
MVME162FX MC68040 Indivisible Cycles 1-49
Illegal Access to IP Modules from External VMEbus Masters 1-50

Chapter 2 VMEchip2

Introduction 2-1
Summary of Major Features 2-1
Functional Blocks 2-4
Local Bus to VMEbus Interface 2-4
Local Bus to VMEbus Requester 2-8
VMEbus to Local Bus Interface 2-9
Local Bus to VMEbus DMA Controller 2-11
No Address Increment DMA Transfers 2-13
DMAC VMEbus Requester 2-14
Tick and Watchdog Timers 2-15
Prescaler 2-15
Tick Timers 2-16
Watchdog Timer 2-17
VMEbus Interrupter 2-17
VMEbus System Controller 2-18
Arbiter 2-18
IACK Daisy-Chain Driver 2-19
Bus Timer 2-19
Reset Driver 2-19
Local Bus Interrupter and Interrupt Handler 2-20
Global Control and Status Registers 2-22
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 2-32
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 2-42
Local Bus Slave (VMEbus Master) Starting Address Register 1 ... 2-42
Local Bus Slave (VMEbus Master) Ending Address Register 2 2-42
Local Bus Slave (VMEbus Master) Starting Address Register 2 ... 2-43
Local Bus Slave (VMEbus Master) Ending Address Register 3 2-43
Local Bus Slave (VMEbus Master) Starting Address Register 3 ... 2-43
Local Bus Slave (VMEbus Master) Ending Address Register 4 2-44
Local Bus Slave (VMEbus Master) Starting Address Register 4 ... 2-44
Local Bus Slave (VMEbus Master) Address Translation Address Register 4 2-44
Local Bus Slave (VMEbus Master) Address Translation Select Register 4 2-45
Local Bus Slave (VMEbus Master) Attribute Register 4..... 2-45
Local Bus Slave (VMEbus Master) Attribute Register 3..... 2-46
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 2-48
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) 2-59
DMAC Local Bus Address Counter 2-60
DMAC VMEbus Address Counter 2-61
DMAC Byte Counter 2-61
Table Address Counter 2-62
VMEbus Interrupter Control Register 2-62
VMEbus Interrupter Vector Register 2-63
MPU Status and DMA Interrupt Count Register 2-64
DMAC Status Register 2-65
Programming the Tick and Watchdog Timers 2-66
VMEbus Arbiter Time-out Control Register 2-66
DMAC Ton/Toff Timers and VMEbus Global Time-out Control Register 2-67
VME Access, Local Bus, and Watchdog Time-out Control Register 2-68
Prescaler Control Register 2-69
Tick Timer 1 Compare Register 2-70
Tick Timer 1 Counter 2-70
Tick Timer 2 Compare Register 2-71
Tick Timer 2 Counter 2-71
Tick Timer 2 Compare Register 2-72
Tick Timer 2 Counter 2-72
Board Control Register 2-73
Watchdog Timer Control Register 2-74
Tick Timer 2 Control Register 2-75
Tick Timer 1 Control Register 2-76
Prescaler Counter 2-76
Programming the Local Bus Interrupter 2-77
Local Bus Interrupter Status Register (bits 24-31) 2-80
Local Bus Interrupter Status Register (bits 16-23) 2-81
Local Bus Interrupter Status Register (bits 8-15) 2-82
Local Bus Interrupter Status Register (bits 0-7) 2-83
Local Bus Interrupter Enable Register (bits 24-31) 2-84
Local Bus Interrupter Enable Register (bits 16-23) 2-85
Local Bus Interrupter Enable Register (bits 8-15) 2-86
Local Bus Interrupter Enable Register (bits 0-7) 2-87

Software Interrupt Set Register (bits 8-15) 2-88
Interrupt Clear Register (bits 24-31) 2-88
Interrupt Clear Register (bits 16-23) 2-89
Interrupt Clear Register (bits 8-15) 2-90
Interrupt Level Register 1 (bits 24-31) 2-90
Interrupt Level Register 1 (bits 16-23) 2-91
Interrupt Level Register 1 (bits 8-15) 2-91
Interrupt Level Register 1 (bits 0-7) 2-92
Interrupt Level Register 2 (bits 24-31) 2-92
Interrupt Level Register 2 (bits 16-23) 2-93
Interrupt Level Register 2 (bits 8-15) 2-93
Interrupt Level Register 2 (bits 0-7) 2-94
Interrupt Level Register 3 (bits 24-31) 2-94
Interrupt Level Register 3 (bits 16-23) 2-95
Interrupt Level Register 3 (bits 8-15) 2-95
Interrupt Level Register 3 (bits 0-7) 2-96
Interrupt Level Register 4 (bits 24-31) 2-96
Interrupt Level Register 4 (bits 16-23) 2-97
Interrupt Level Register 4 (bits 8-15) 2-97
Interrupt Level Register 4 (bits 0-7) 2-98
Vector Base Register 2-98
I/O Control Register 1 2-99
I/O Control Register 2 2-100
I/O Control Register 3 2-100
Miscellaneous Control Register 2-100
GCSR Programming Model 2-103
Programming the GCSR 2-105
VMEchip2 Revision Register 2-107
VMEchip2 ID Register 2-107
VMEchip2 LM/SIG Register 2-107
VMEchip2 Board Status/Control Register 2-109
General Purpose Register 0 2-110
General Purpose Register 1 2-110
General Purpose Register 2 2-111
General Purpose Register 3 2-111
General Purpose Register 4 2-112
General Purpose Register 5 2-112

Chapter 3 MC2 Chip

Introduction 3-1
Summary of Major Features 3-1
Functional Description 3-2
MC2 chip Initialization 3-2
Flash and EPROM Interface 3-2
BBRAM Interface 3-3
82596CA LAN Interface 3-3
MPU Port and MPU Channel Attention 3-3
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 3-6
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 MC2 chip Registers 3-9
Programming Model 3-11
MC2 chip ID Register 3-11
MC2 chip Revision Register 3-12
General Control Register 3-12
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 3-18
Prescaler Clock Adjust Register 3-18
Tick Timer 1 and 2 Control Registers 3-19
Tick Timer Interrupt Control Registers 3-21
DRAM Parity Error Interrupt Control Register 3-23
SCC Interrupt Control Register 3-24
Tick Timer 3 and 4 Control Registers 3-25
Tick Timer 4 Control Register 3-25
Tick Timer 3 Control Register 3-25
DRAM and SRAM Memory Controller Registers 3-26
DRAM Space Base Address Register 3-26

SRAM Space Base Address Register 3-27
DRAM Space Size Register 3-28
DRAM/SRAM Options Register 3-29
SRAM Space Size Register 3-31
LANC Error Status Register 3-32
82596CA LANC Interrupt Control Register 3-33
LANC Bus Error Interrupt Control Register 3-34
SCSI Error Status Register 3-35
General Purpose Inputs Register 3-36
MVME162FX Version Register 3-37
SCSI Interrupt Control Register 3-38
Tick Timer 3 and 4 Compare and Counter Registers 3-39
Bus Clock Register 3-40
EPROM Access Time Control Register 3-41
Flash Parameter Register 3-42
ABORT Switch Interrupt Control Register 3-43
RESET Switch Control Register 3-44
Watchdog Timer Control Register 3-45
Access and Watchdog Time Base Select Register 3-46
DRAM Control Register 3-47
MPU Status Register 3-49
32-bit Prescaler Count Register 3-50

Chapter 4 IP2 Chip

Introduction 4-1
Summary of Major Features 4-1
Functional Description 4-2
General Description 4-2
Cache Coherency 4-2
Local Bus to IndustryPack DMA Controllers 4-3
Clocking Environments and Performance 4-5
Pacer Clock 4-6
Error Reporting 4-7
Error Reporting as a Local Bus Slave 4-7
Error Reporting as a Local Bus Master 4-7
IndustryPack Error Reporting 4-7
Interrupts 4-8
Overall Memory Map 4-9

Programming Model 4-10
Chip ID Register 4-17
Chip Revision Register 4-17
Vector Base Register 4-18
IP_a, IP_b, IP_c, IP_d Memory Base Address Registers 4-19
IP_a or Double Size IP_ab Memory Base Address Registers 4-20
IP_b Memory Base Address Registers 4-20
IP_c or Double Size IP_cd Memory Base Address Registers..... 4-21
IP_d Memory Base Address Registers..... 4-21
IP_a, IP_b, IP_c, IP_d Memory Size Registers..... 4-22
IP_a, IP_b, IP_c, and IP_d; IRQ0 and IRQ1 Interrupt Control Registers 4-23
IP_a, IP_b, IP_c, and IP_d; General Control Registers 4-25
IP Clock Register..... 4-29
DMA Arbitration Control Register 4-30
IP RESET Register..... 4-31
Programming the DMA Controllers 4-31
DMA Enable Function..... 4-33
DMA Control and Status Register Set Definition 4-34
Programming the Pacer Clock 4-44
Local Bus to IndustryPack Addressing..... 4-47
8-Bit Memory Space 4-47
16-Bit Memory Space 4-48
32-Bit Memory Space 4-49
IP_a I/O Space 4-50
IP_ab I/O Space..... 4-51
IP_a ID Space 4-52
IP to Local Bus Data Routing 4-53
Memory Space Accesses 4-53
I/O and ID Space Accesses 4-55

Chapter 5 Serial Port Connections

Introduction 5-1

Appendix A Using Interrupts on the MVME162FX

Introduction A-1
VMEchip2 Tick Timer 1 Periodic Interrupt Example..... A-1

Figures

Figure 1-1. MVME162FX Block Diagram 1-5
Figure 2-1. VMEchip2 Block Diagram 2-5
Figure 5-1. MVME162FX Port 1 EIA-232 DCE, MVME712M Port 2 DTE 5-3
Figure 5-2. MVME162FX Port 1 EIA-232 DCE, MVME712M Port 2 DCE 5-4
Figure 5-3. MVME162FX Port 2 EIA-232 DTE, MVME712M Port 4 DTE 5-5
Figure 5-4. MVME162FX Port 2 EIA-232 DCE, MVME712M Port 4 DCE 5-6
Figure 5-5. MVME162FX Port 2 EIA-232 DTE, MVME712M Port 4 DCE 5-7
Figure 5-6. MVME162FX Port 2 EIA-232 DCE, MVME712M Port 4 DTE 5-8
Figure 5-7. MVME162FX Port 2 EIA-530 DTE 5-9
Figure 5-8. MVME162FX Port 2 EIA-530 DCE 5-10
Figure 5-9. MVME162FX Port 1 EIA-232 DCE, MVME712A/AM/-12/-13 Port 2 DTE 5-11
Figure 5-10. MVME162FX Port 1 EIA-232 DCE, MVME712AM/-13 Port 2 as Modem Port 5-12
Figure 5-11. MVME162FX Port 2 EIA-232 DTE, MVME712A/AM/-12/-13 Port 4DTE 5-13
Figure 5-12. MVME162FX Port 2 EIA-232 DCE, MVME712A/AM/-12/-13 Port 4DTE 5-14
Figure 5-13. MVME162FX EIA-485/EIA-422 Connections 5-15

Tables

Table 1-1. Redundant Functions in the VMEchip2 and MC2 chip 1-6
Table 1-2. Local Bus Memory Map 1-9
Table 1-3. Local Bus I/O Devices Memory Map 1-11
Table 1-4. VMEchip2 Memory Map (Sheet 1 of 3) 1-15
Table 1-4. VMEchip2 Memory Map (Sheet 2 of 3) 1-17
Table 1-4. VMEchip2 Memory Map (Sheet 3 of 3) 1-20
Table 1-5. MC2 chip Register Map 1-21
Table 1-6. IP2 chip Overall Memory Map 1-22
Table 1-7. IP2 chip Memory Map - Control and Status Registers 1-23
Table 1-8. Z85230 SCC Register Addresses 1-30
Table 1-9. 82596CA Ethernet LAN Memory Map 1-30
Table 1-10. 53C710 SCSI Memory Map 1-31
Table 1-11. MK48T08 BBRAM/TOD Clock Memory Map 1-32
Table 1-12. BBRAM Configuration Area Memory Map 1-32
Table 1-13. TOD Clock Memory Map 1-33
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 2-78
Table 2-4. VMEchip2 Memory Map (GCSR Summary) 2-106
Table 3-1. DRAM Performance 3-6
Table 3-2. MC2 chip Register Map 3-10
Table 3-3. Interrupt Vector Base Register Encoding and Priority 3-15
Table 3-4. DRAM Size Control Bit Encoding 3-28
Table 3-5. DRAM Size Control Bit Encoding 3-29
Table 3-6. SRAM Size Control Bit Encoding 3-30
Table 3-7. SRAM Size Control Bit Encoding 3-31
Table 4-1. IP2 chip Clock Cycles 4-5
Table 4-2. IP2 chip Overall Memory Map 4-9
Table 4-3. IP2 chip Memory Map - Control and Status Registers 4-11

Board Description and Memory Maps

1

Introduction

This manual provides programming information for the MVME162FX Embedded Controller. Extensive programming information is provided for the Application-Specific Integrated Circuit (ASIC) devices used on the board. Reference information is included for the Large Scale Integration (LSI) devices used on the board and sources for additional information are provided.

This chapter briefly describes the board level hardware features of the MVME162FX Embedded Controller. The chapter begins with a board level overview and features list. Memory maps are next, and the chapter closes with some general software considerations such as cache coherency, interrupts, and bus errors.

All programmable registers in the MVME162FX that reside in ASICs are covered in the chapters on those ASICs. Chapter 2 covers the VMEchip2, Chapter 3 covers the MC2 chip, and Chapter 4 covers the IP2 chip. Chapter 5 covers certain serial port connections. Appendix A details using interrupts. For those interested in programmable register bit definitions and less interested in hardware functionality, focus on Chapters 2, 3, and 4. In some cases, however, Chapter 1 gives related background information.

Overview

The MVME162FX is based on the MC68040 or MC68LC040 microprocessor. Various versions of the MVME162FX have 4MB, 8MB, or 16MB of unprotected DRAM, 8KB of SRAM (with battery backup), time of day clock (with battery backup), Ethernet transceiver interface, two serial ports with EIA-232-D or EIA-530 or EIA-485/422 interface, six tick timers, watchdog timer, a PROM socket, 1MB Flash memory (one Flash device), four IndustryPack

(IP) interfaces with DMA, SCSI bus interface with DMA, VMEbus controller, and 512KB of SRAM with battery backup. A “No VMEbus” option is also available.

The I/O on the MVME162FX is connected to the VMEbus P2 connector. The main board is connected through a P2 transition board and cables to the transition boards. The MVME162FX supports the transition boards MVME712-12, MVME712-13, MVME712M, MVME712A, MVME712AM, and MVME712B (referred to in this manual as MVME712*x*, unless separately specified). The MVME712x transition boards provide configuration headers and industry-standard connectors for I/O devices.

The I/O connection for the serial ports on the MVME162FX is also implemented with two DB-25 front panel I/O connectors. The MVME712 series transition boards were designed to support the MVME167 boards, but can be used on the MVME162FX if you take some special precautions. (For more information, refer to the section on the Serial Communications Interface in the *MVME162FX Embedded Controller Installation and Use* manual.) These transition boards provide configuration headers, serial port drivers and industry standard connectors for the I/O devices.

The VMEbus interface is provided by an ASIC called the VMEchip2. 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, 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.

The MC2 chip ASIC provides four tick timers, the interface to the LAN chip, SCSI chip, serial port chip, BBRAM, the programmable interface for the DRAM and/or SRAM mezzanine board, and Flash write enable signal.

The IndustryPack Interface Controller (IP2 chip) ASIC provides control and status information, including DMA control, for up to four single-size IndustryPacks (IPs) or up to two double-size IPs that can be plugged into the MVME162FX main module.

Requirements

These boards are designed to conform to the requirements of the following documents:

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

Features

- 32MHz 32-bit MC68040 Microprocessor, or 25MHz 32-bit MC68040 or MC68LC040 Microprocessor
- 4MB, 8MB, or 16MB of shared DRAM with no parity protection
- 512KB of SRAM with battery backup
- One JEDEC standard 32-pin PLCC EPROM socket (EPROMs may be shipped separately from the MVME162FX)
- One Intel 28F008SA 1M x 8 Flash memory device (1MB Flash memory total)
- 8K by 8 Non-Volatile RAM and time-of-day clock with battery backup
- Four 32-bit Tick Timers (in the MC2 chip ASIC) for periodic interrupts
- Two 32-bit Tick Timers (in the VMEchip2 ASIC) for periodic interrupts

- Watchdog timer
- Eight software interrupts (for MVME162FX versions that have the VMEchip2)
- I/O
  - Two serial ports (one EIA-232-D DCE; one EIA-232-D DCE/DTE or EIA-530 DCE/DTE or EIA-42 DCE/DTE or EIA-485)
  - Serial port controller (Zilog Z85230)
  - Optional Small Computer Systems Interface (SCSI) bus interface with 32-bit local bus burst Direct Memory Access (DMA) (NCR 53C710 controller)
  - Optional LAN Ethernet transceiver interface with 32-bit local bus DMA (Inter 82596CA controller)
  - Four MVIP IndustryPack interfaces with DMA
- 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 and Light-Emitting Diodes
  - Two pushbutton switches (ABORT and RESET)
  - Eight LEDs (FAIL, STAT, RUN, SCON, LAN, FUSE (LAN power), SCSI, and VME)

Block Diagram

Figure 1-1 is a general block diagram of the MVME162FX.

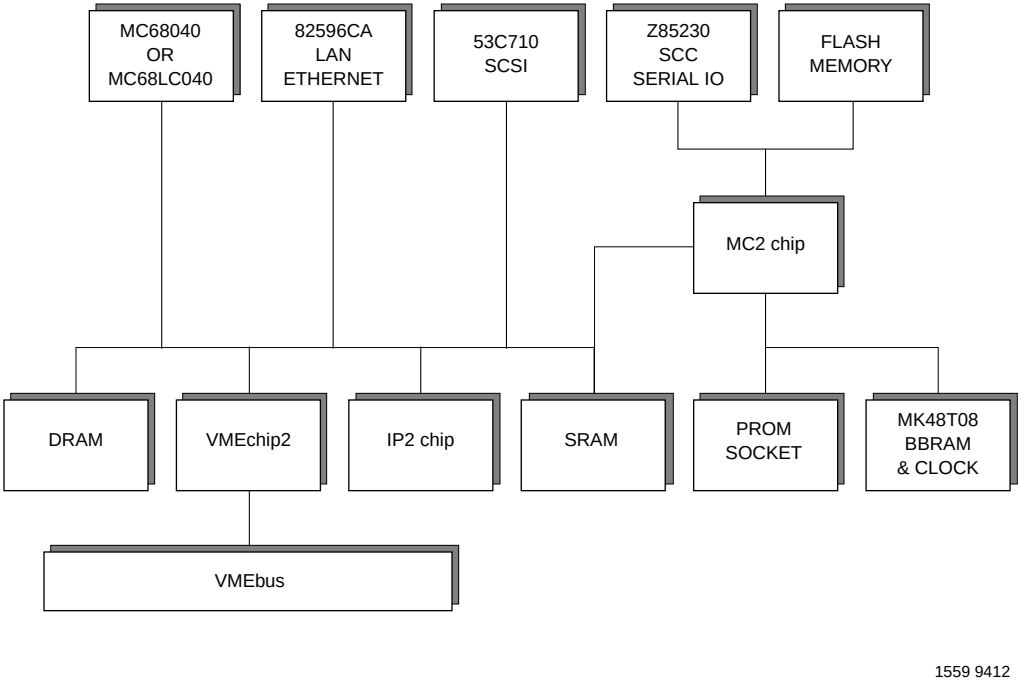

Figure 1-1. MVME162FX Block Diagram

Functional Description

This section covers only a few specific features of the MVME162FX.

A complete functional description of the major blocks on the MVME162FX Embedded Controller is provided in the *MVME162FX Embedded Controller Installation and Use* manual.

No-VMEbus-Interface Option

The MVME162FX can be operated as an embedded controller without the VMEbus interface. For this option, the VMEchip2 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.

To support this feature, certain logic in the VMEchip2 has been duplicated in the MC2 chip. The following table defines the location of the redundant logic. This logic is inhibited in the MC2 chip if the VMEchip2 is present. The enables for these functions are controlled by software and MC2 chip hardware initialization.

Table 1-1. Redundant Functions in the VMEchip2 and MC2 chip

VMEchip2		MC2 chip		Notes
Address	Bit #	Address	Bit #	
$FFF40060	28 - 24	$FFF42044	28 - 24	1,5
$FFF40060	22 - 19,17,16	$FFF42044	22 - 19,17,16	2,5
$FFF4004C	13 - 8	$FFF42044	13 - 8	3,5
$FFF40048	7	$FFF42048	8	4
$FFF40048	9	$FFF42048	9	4,5
$FFF40048	10	$FFF42048	10	4,5
$FFF40048	11	$FFF42048	11	4,5
$FFF40064	31 - 0	$FFF4204C	31 - 0	8
		$FFF42040	6 - 0	6
$FF800000-$FFBFFFFF	31 - 0	$FF800000- $FFBFFFFF	31 - 0	7
$FFE00000-$FFEFFFFF	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 MC2 chip have a one-to-one correspondence.

6. ABORT switch interrupt control. Implemented in the VMEchip2 also, but with a different bit organization (refer to the VMEchip2 description in Chapter 2). The ABORT switch is wired to the MC2 chip but not the VMEchip2 in the MVME162FX 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.

Note that MVME162FX 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 J22 is configured for the EPROM memory map. Refer to Chapters 3 and 4 of the *MVME162FX Installation and Use* documentation, V162FXA, for further details.

VMEbus Interface and VMEchip2

The local-bus-to-VMEbus interface and the VMEbus-to-local-bus interface are provided by the optional VMEchip2. The VMEchip2 can also provide the VMEbus system controller functions. Refer to the VMEchip2 description in Chapter 2 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 MC2 chip ASIC at location $FFF42043. The GPI inputs are integrated into the MC2 chip ASIC at location $FFF4202C bits 23-16.

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 VMEbus masters (VMEbus memory map).

The memory and I/O maps which are described in the following tables are correct for all local bus masters. There is some address translation capability in the VMEchip2. This allows multiple MVME162FXs 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 MVME162FX, Transfer Types 0, 1, and 2 define the normal address range. Table 1-2 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 MC68*xx*040 MMU. The onboard I/O space must be marked cache inhibit and serialized in its page table. Table 1-3 further defines the map for the local I/O devices.

Table 1-2. Local Bus Memory Map

Address Range	Devices Accessed	Port Width	Size	Software Cache Inhibit	Note(s)
Programmable	DRAM on board	D32	4MB-16MB	N	2
Programmable	SRAM	D32	128KB-2MB	N	2
Programmable	VMEbus A32/A24	D32/D16	--	?	4
Programmable	IP a Memory	D32-D8	64KB-8MB	?	2, 4
Programmable	IP b Memory	D32-D8	64KB-8MB	?	2, 4
Programmable	IP c Memory	D32-D8	64KB-8MB	?	2, 4
Programmable	IP d Memory	D32-D8	64KB-8MB	?	2, 4
$FF800000 - $FF9FFFFF	Flash/PROM	D32	2MB	N	1, 5
$FFA00000 - $FFBFFFFF	PROM/Flash	D32	2MB	N	6
$FFC00000 - $FFCFFFFF	Not decoded	--	1MB	N	7
$FFD00000 - $FFDFFFFF	Not decoded	--	1MB	N	7
$FFE00000 - $FFE7FFFF	SRAM default	D32	512KB	N	--
$FFE80000 - $FFEFFFFF	Not decoded	--	512KB	N	7
$FFF00000 - $FFFEFFFF	Local I/O	D32-D8	878KB	Y	3
$FFFF0000 - $FFFFFFFF	VMEbus A16	D32/D16	64KB	?	2, 4

Notes

1. Reset enables the decoder for this space of the memory map so that it will decode address spaces $FF800000-$FF9FFFFF and $00000000-$003FFFFF. The decode at 0 must be disabled in the MC2 chip before DRAM is enabled. DRAM is enabled with the DRAM Control Register at address $FFF42048, bit 24. PROM/Flash is disabled at the low address space with PROM Control Register at address $FFF42040, bit 20.

2. This area is user-programmable. The DRAM and SRAM decoder is programmed in the MC2 chip, the local-to-VMEbus decoders are programmed in the VMEchip2, and the IP memory space is programmed in the IP2 chip.

3. Size is approximate.

4. Cache inhibit depends on devices in area mapped.

5. The PROM and Flash are sized by the MC2 chip ASIC from an 8-bit private bus to the 32-bit MPU local bus. Because the device size is less than the allocated memory map size for some entries, the device contents repeat for those entries.

   If jumper GPI3 is installed, the Flash device is accessed. If GPI3 is not installed, the PROM is accessed.

6. The PROM and Flash are sized by the MC2 chip ASIC from an 8-bit private bus to the 32-bit MPU local bus. Because the device size is less than the allocated memory map size for some entries, the device contents repeat for those entries.

   If jumper GPI3 is installed, the PROM is accessed. If GPI3 is not installed, the Flash device is accessed.

7. These areas are not decoded unless one of the programmable decoders are initialized to decode this space. If they are not decoded, an access to this address range will generate a local bus time-out. The local bus timer must be enabled.

The following table focuses on the Local I/O Devices portion of the local bus Main Memory Map.

Table 1-3. Local Bus I/O Devices Memory Map

Address Range	Device	Port Width	Size	Note(s)
$FFF00000 - $FFF3FFFF	Reserved	--	256KB	4
$FFF40000 - $FFF400FF	VMEchip2 (LCSR)	D32	256B	1, 3
$FFF40100 - $FFF401FF	VMEchip2 (GCSR) registers	D32-D8	256B	1, 3
$FFF40200 - $FFF40FFF	Reserved	--	3.5KB	4, 5
$FFF41000 - $FFF41FFF	Reserved	--	4KB	4
$FFF42000 - $FFF42FFF	MC2 chip	D32-D8	4KB	1
$FFF43000 - $FFF44FFF	Reserved	--	8KB	4
$FFF45000 - $FFF45FFF	SCC (Z85230)	D8	4KB	1, 2
$FFF46000 - $FFF46FFF	LAN (82596CA)	D32	4KB	1, 6
$FFF47000 - $FFF47FFF	SCSI (53C710)	D32-D8	4KB	1
$FFF48000 - $FFF57FFF	Reserved	--	64KB	4
$FFF58000 - $FFF5807F	IP2 chip IP a I/O	D16	128B	1
$FFF58080 - $FFF580FF	IP2 chip IP a ID	D16	128B	1
$FFF58100 - $FFF5817F	IP2 chip IP b I/O	D16	128B	1
$FFF58180 - $FFF581FF	IP2 chip IP b ID Read	D16	128B	1
$FFF58200 - $FFF5827F	IP2 chip IP c I/O	D16	128B	1
$FFF58280 - $FFF582FF	IP2 chip IP c ID	D16	128B	1
$FFF58300 - $FFF5837F	IP2 chip IP d I/O	D16	128B	1
$FFF58380 - $FFF583FF	IP2 chip IP d ID Read	D16	128B	1
$FFF58400 - $FFF584FF	IP2 chip IP ab I/O	D32-D16	256B	1
$FFF58500 - $FFF585FF	IP2 chip IP cd I/O	D32-D16	256B	1
$FFF58600 - $FFF586FF	IP2 chip IP ab I/O repeated	D32-D16	256B	1
$FFF58700 - $FFF587FF	IP2 chip IP cd I/O repeated	D32-D16	256B	1
$FFF58800 - $FFF5887F	Reserved	--	128B	1
$FFF58880 - $FFF588FF	Reserved	--	128B	1
$FFF58900 - $FFF5897F	Reserved	--	128B	1
$FFF58980 - $FFF589FF	Reserved	--	128B	1

Table 1-3. Local Bus I/O Devices Memory Map (Continued)

Address Range	Device	Port Width	Size	Note(s)
$FFF58A00 - $FFF58A7F	Reserved	--	128B	1
$FFF58A80 - $FFF58AFF	Reserved	--	128B	1
$FFF58B00 - $FFF58B7F	Reserved	--	128B	1
$FFF58B80 - $FFF58BFF	Reserved	--	128B	1
$FFF58C00 - $FFF58CFF	Reserved	--	256B	1
$FFF58D00 - $FFF58DFF	Reserved	--	256B	1
$FFF58E00 - $FFF58EFF	Reserved	--	256B	1
$FFF58F00 - $FFF58FFF	Reserved	--	256B	1
$FFFBC000 - $FFFBC01F	IP2 chip registers	D32-D8	2KB	1
$FFFBC800 - $FFFBC81F	Reserved	--	2KB	1
$FFFBD000 - $FFFBFFFF	Reserved	--	12KB	4
$FFFC0000 - $FFFC7FFF	MK48T08 (BBRAM, TOD clock)	D32-D8	32KB	1
$FFFC8000 - $FFFCBFFF	MK48T08	D32-D8	16KB	1, 7
$FFFCC000 - $FFFCFFFF	MK48T08	D32-D8	16KB	1, 7
$FFFD0000 - $FFFEFFFF	Reserved	--	128KB	4

Notes 1. For a complete description of the register bits, refer to the data sheet for the specific chip. For a more detailed memory map, refer to the following detailed peripheral device memory maps.

2. The SCC is an 8-bit device located on an MC2 chip private data bus. Byte access is required.

The data register of the Zilog Z85230 device which is interfaced by the MC2 chip ASIC cannot be accessed. The Zilog Z85230 has an indirect access mode to the data registers which is functional and must be used.

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 PROM Interface section in the MC2 chip description in Chapter 3.

Detailed I/O Memory Maps

Tables 1-4 through 1-13 give the detailed memory maps for:

1-4	VMEchip2
1-5	MC2 chip
1-6	IP2 chip
1-7	IP2 chip Control and Status Registers

1-8 Z85230 SCC Register addresses
1-9 82596CA Ethernet LAN chip
1-10 53C710 SCSI chip
1-11 MK48T08 BBRAM/TOD clock
1-12 BBRAM configuration area
1-13 TOD clock

Note Manufacturers' errata sheets for the various chips are available by contacting your local Motorola sales representative. A non-disclosure agreement may be required.

Table 1-4. VMEchip2 Memory Map (Sheet 1 of 3)

VMEchip2 LCSR Base Address = $FFF40000

OFFSET:

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	SLAVE ENDING ADDRESS 1															
4	SLAVE ENDING ADDRESS 2															
8	SLAVE ADDRESS TRANSLATION ADDRESS 1															
C	SLAVE ADDRESS TRANSLATION ADDRESS 2															
10					ADDER 2	SNP 2			WP 2	SUP 2	USR 2	A32 2	A24 2	BLK D64 2	BLK 2	PRGM 2	DATA2
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
14	MASTER ENDING ADDRESS 1															
18	MASTER ENDING ADDRESS 2															
1C	MASTER ENDING ADDRESS 3															
20	MASTER ENDING ADDRESS 4															
24	MASTER ADDRESS TRANSLATION ADDRESS 4															
28	MAST D16 EN	MAST WP EN	MASTER AM 4						MAST D16 EN	MAST WP EN	MASTER AM 3					
2C	GCSR GROUP SELECT								GCSR BOARD SELECT				MAST 4 EN	MAST 3 EN	MAST 2 EN	MAST 1 EN
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
30											WAIT RMW	ROM ZERO	DMA TB SNP MODE		SRAM SPEED	
34																
38	DMA CONTROLLER															
3C	DMA CONTROLLER															
40	DMA CONTROLLER															
44	DMA CONTROLLER															
48		TICK 2/1	TICK IRQ 1 EN	CLR IRQ	IRQ STAT	VMEBUS INTERRUPT LEVEL			VMEBUS INTERRUPT VECTOR							

This sheet continues on facing page. ⟶

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SLAVE STARTING ADDRESS 1															
SLAVE STARTING ADDRESS 2															
SLAVE ADDRESS TRANSLATION SELECT 1															
SLAVE ADDRESS TRANSLATION SELECT 2															
				ADDER 1	SNP 1		WP 1	SUP 1	USR 1	A32 1	A24 1	BLK D64 1	BLK 1	PRGM 1	DATA1

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 1-4. 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								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		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																
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		AC FAIL IRQ LEVEL				ABORT IRQ LEVEL				SYS FAIL IRQ LEVEL				MST WP ERROR IRQ LEVEL		
7C		VME IACK IRQ LEVEL				DMA IRQ LEVEL				SIG 3 IRQ LEVEL				SIG 2 IRQ LEVEL		
80		SW7 IRQ LEVEL				SW6 IRQ LEVEL				SW5 IRQ LEVEL				SW4 IRQ LEVEL		
84		SPARE IRQ LEVEL				VME IRQ 7 IRQ LEVEL				VME IRQ 6 IRQ LEVEL				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																

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 1-4. 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	SIG3	SIG2	SIG1	SIG0	RST	ISF	BF	SCON	SYSFL	X	X	X
4	8	GENERAL PURPOSE CONTROL AND STATUS REGISTER 0															
6	C	GENERAL PURPOSE CONTROL AND STATUS REGISTER 1															
8	10	GENERAL PURPOSE CONTROL AND STATUS REGISTER 2															
A	14	GENERAL PURPOSE CONTROL AND STATUS REGISTER 3															
C	18	GENERAL PURPOSE CONTROL AND STATUS REGISTER 4															
E	1C	GENERAL PURPOSE CONTROL AND STATUS REGISTER 5															

Table 1-5. MC2 chip Register Map

MC2 chip Base Address = $FFF42000

Offset	D31-D24	D23-D16	D15-D8	D7-D0
$00	MC2 chip ID	MC2 chip Revision	General Control	Interrupt Vector Base Register
$04	Tick Timer 1 Compare Register			
$08	Tick Timer 1 Counter Register			
$0C	Tick Timer 2 Compare Register			
$10	Tick Timer 2 Counter Register			
$14	LSB Prescaler Count Register	Prescaler Clock Adjust	Tick Timer 2 Control	Tick Timer 1 Control
$18	Tick Timer 4 Interrupt Control	Tick Timer 3 Interrupt Control	Tick Timer 2 Interrupt Control	Tick Timer 1 Interrupt Control
$1C	DRAM Parity Error Interrupt Control	SCC Interrupt Control	Tick Timer 4 Control	Tick Timer 3 Control
$20	DRAM Space Base Address Register		SRAM Space Base Address Register	
$24	DRAM Space Size	DRAM/SRAM Options	SRAM Space Size	(reserved)
$28	LANC Error Status	(reserved)	LANC Interrupt Control	LANC Bus Error Interrupt Control
$2C	SCSI Error Status	General Purpose Inputs	MVME162FX Version	SCSI Interrupt Control
$30	Tick Timer 3 Compare Register			
$34	Tick Timer 3 Counter Register			
$38	Tick Timer 4 Compare Register			
$3C	Tick Timer 4 Counter Register			
$40	Bus Clock	EPROM Access Time Control	Flash Parameter Control	ABORT Switch Interrupt Control
$44	RESET Switch Control	Watchdog Timer Control	Access & Watchdog Time Base Select	(reserved)
$48	DRAM Control	(reserved)	MPU Status	(reserved)
$4C	32-bit Prescaler Count Register			

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

Table 1-6. IP2 chip Overall Memory Map

Address Range	Selected Device	Port Width	Size
Programmable	IP_a/IP_ab Memory Space	D32-D8	64KB-16MB
Programmable	IP_b Memory Space	D16-D8	64KB-8MB
Programmable	IP_c/IP_cd Memory Space	D32-D8	64KB-16MB
Programmable	IP_d Memory Space	D16-D8	64KB-8MB
$FFF58000-$FFF5807F	IP_a I/O Space	D16	128B
$FFF58080-$FFF580BF	IP_a ID Space	D16	64B
$FFF580C0-$FFF580FF	IP_a ID Space Repeated	D16	64B
$FFF58100-$FFF5817F	IP_b I/O Space	D16	128B
$FFF58180-$FFF581BF	IP_b ID Space	D16	64B
$FFF581C0-$FFF581FF	IP_b ID Space Repeated	D16	64B
$FFF58200-$FFF5827F	IP_c I/O Space	D16	128B
$FFF58280-$FFF582BF	IP_c ID Space	D16	64B
$FFF582C0-$FFF582FF	IP_c ID Space Repeated	D16	64B
$FFF58300-$FFF5837F	IP_d I/O Space	D16	128B
$FFF58380-$FFF583BF	IP_d ID Space	D16	64B
$FFF583C0-$FFF583FF	IP_d ID Space Repeated	D16	64B
$FFF58400-$FFF584FF	IP_ab I/O Space	D32-D16	256B
$FFF58500-$FFF585FF	IP_cd I/O Space	D32-D16	256B
$FFF58600-$FFF586FF	IP_ab I/O Space Repeated	D32-D16	256B
$FFF58700-$FFF587FF	IP_cd I/O Space Repeated	D32-D16	256B
$FFFBC000-$FFFBC083	Control/Status Registers	D32-D8	32B

A summary of the IP2 chip CSR registers is shown in Table 1-7. 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.

Table 1-7. IP2 chip Memory Map - Control and Status Registers

IP2 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	1
$02	RESERVED	0	0	0	0	0	0	0	0
$03	VECTOR BASE	IV7	IV6	IV5	IV4	IV3	IV2	IV1	IV0
$04	IP_a MEM BASE UPPER	a_BASE31	a_BASE30	a_BASE29	a_BASE28	a_BASE27	a_BASE26	a_BASE25	a_BASE24
$05	IP_a MEM BASE LOWER	a_BASE23	a_BASE22	a_BASE21	a_BASE20	a_BASE19	a_BASE18	a_BASE17	a_BASE16
$06	IP_b MEM BASE UPPER	b_BASE31	b_BASE30	b_BASE29	b_BASE28	b_BASE27	b_BASE26	b_BASE25	b_BASE24
$07	IP_b MEM BASE LOWER	b_BASE23	b_BASE22	b_BASE21	b_BASE20	b_BASE19	b_BASE18	b_BASE17	b_BASE16
$08	IP_c MEM BASE UPPER	c_BASE31	c_BASE30	c_BASE29	c_BASE28	c_BASE27	c_BASE26	c_BASE25	c_BASE24
$09	IP_c MEM BASE LOWER	c_BASE23	c_BASE22	c_BASE21	c_BASE20	c_BASE19	c_BASE18	c_BASE17	c_BASE16
$0A	IP_d MEM BASE UPPER	d_BASE31	d_BASE30	d_BASE29	d_BASE28	d_BASE27	d_BASE26	d_BASE25	d_BASE24
$0B	IP_d MEM BASE LOWER	d_BASE23	d_BASE22	d_BASE21	d_BASE20	d_BASE19	d_BASE18	d_BASE17	d_BASE16
$0C	IP_a MEM SIZE	a_SIZE23	a_SIZE22	a_SIZE21	a_SIZE20	a_SIZE19	a_SIZE18	a_SIZE17	a_SIZE16
$0D	IP_b MEM SIZE	b_SIZE23	b_SIZE22	b_SIZE21	b_SIZE20	b_SIZE19	b_SIZE18	b_SIZE17	b_SIZE16
$0E	IP_c MEM SIZE	c_cSIZE23	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

Table 1-7. IP2 chip Memory Map - Control and Status Registers (Continued)
IP2 chip Base Address = $FFFBC000

Register Offset	Register Name	Register Bit Names							
		D7	D6	D5	D4	D3	D2	D1	D0
$18	IP_a GENERAL CONTROL	a_ERR	0	a_RT1	a_RT0	a_WIDTH1	a_WIDTH0	a_BTD	a_MEN
$19	IP_b GENERAL CONTROL	b_ERR	0	b_RT1	b_RT0	b_WIDTH1	b_WIDTH0	b_BTD	b_MEN
$1A	IP_c GENERAL CONTROL	c_ERR	0	c_RT1	c_RT0	c_WIDTH1	c_WIDTH0	c_BTD	c_MEN
$1B	IP_d GENERAL CONTROL	d_ERR	0	d_RT1	d_RT0	d_WIDTH1	d_WIDTH0	d_BTD	d_MEN
$1C	RESERVED	0	0	0	0	0	0	0	0
$1D	IP CLOCK	0	0	0	0	0	0	0	IP32
$1E	DMA ARBITRATION CONTROL	0	0	0	0	0	ROTAT	PRI1	PRI0
$1F	IP RESET	0	0	0	0	0	0	0	RES

Table 1-7. IP2 chip Memory Map - Control and Status Registers (Continued)

IP2 chip Base Address = $FFFBC000

Register Offset	Register Name	Register Bit Names							
		D7	D6	D5	D4	D3	D2	D1	D0
DMAC for IndustryPack a, request 0. This register set is referred to as DMACa in the text.									
$20	DMA_a STATUS	0	DLBE	0	IPEND	CHANI	TBL	IPTO	DONE
$21	DMA_a INT CTRL	0	0	DINT	DIEN	DICLR	DIL2	DIL1	DIL0
$22	DMA ENABLE	0	0	0	0	0	0	0	DEN
$23	RESERVED	0	0	0	0	0	0	0	0
$24	DMA_a CONTROL 1	DHALT	0	DTBL	ADMA	WIDTH1	WIDTH0	0	XXX
$25	DMA_a CONTROL 2	INTE	0	DMAEI	DMAEO	ENTO	TOIP	0	0
$26	RESERVED	0	0	0	0	0	0	0	0
$27	RESERVED	0	0	0	0	0	0	0	0
$28	DMA_a LB ADDR	LBA31	LBA30	LBA29	LBA28	LBA27	LBA26	LBA25	LBA24
$29	DMA_a LB ADDR	LBA23	LBA22	LBA21	LBA20	LBA19	LBA18	LBA17	LBA16
$2A	DMA_a LB ADDR	LBA15	LBA14	LBA13	LBA12	LBA11	LBA10	LBA9	LBA8
$2B	DMA_a LB ADDR	LBA7	LBA6	LBA5	LBA4	LBA3	LBA2	LBA1	LBA0
$2C	DMA_a IP ADDR	0	0	0	0	0	0	0	0
$2D	DMA_a IP ADDR	IPA23	IPA22	IPA21	IPA20	IPA19	IPA18	IPA17	IPA16
$2E	DMA_a IP ADDR	IPA15	IPA14	IPA13	IPA12	IPA11	IPA10	IPA9	IPA8
$2F	DMA_a IP ADDR	IPA7	IPA6	IPA5	IPA4	IPA3	IPA2	IPA1	IPA0
$30	DMA_a BYTE CNT	0	0	0	0	0	0	0	0
$31	DMA_a BYTE CNT	BCNT23	BCNT22	BCNT21	BCNT20	BCNT19	BCNT18	BCNT17	BCNT16
$32	DMA_a BYTE CNT	BCNT15	BCNT14	BCNT13	BCNT12	BCNT11	BCNT10	BCNT9	BCN8
$33	DMA_a BYTE CNT	BCNT7	BCNT6	BCNT5	BCNT4	BCNT3	BCNT2	BCNT1	BCNT0
$34	DMA_a TBL ADDR	TA31	TA30	TA29	TA28	TA27	TA26	TA25	TA24
$35	DMA_a TBL ADDR	TA23	TA22	TA21	TA20	TA19	TA18	TA17	TA16
$36	DMA_a TBL ADDR	TA15	TA14	TA13	TA12	TA11	TA10	TA9	TA8
$37	DMA_a TBL ADDR	TA7	TA6	TA5	TA4	TA3	TA2	TA1	TA0

Table 1-7. IP2 chip Memory Map - Control and Status Registers (Continued)

IP2 chip Base Address = $FFFBC000

Register Offset	Register Name	Register Bit Names							
		D7	D6	D5	D4	D3	D2	D1	D0
DMAC for IndustryPack b, request 0 or for IndustryPack a, request 1. This register set is referred to as DMACb in the text.									
$38	DMA_b STATUS	0	DLBE	0	IPEND	CHANI	TBL	IPTO	DONE
$39	DMA_b INT CTRL	0	0	DINT	DIEN	DICLR	DIL2	DIL1	DIL0
$3a	DMA ENABLE	0	0	0	0	0	0	0	DEN
$3b	RESERVED	0	0	0	0	0	0	0	0
$3c	DMA_b CONTROL 1	DHALT	0	DTBL	ADMA	WIDTH1	WIDTH0	A_CH1	XXX
$3d	DMA_b CONTROL 2	INTE	0	DMAEI	DMAEO	ENTO	TOIP	0	0
$3e	RESERVED	0	0	0	0	0	0	0	0
$3f	RESERVED	0	0	0	0	0	0	0	0
$40	DMA_b LB ADDR	LBA31	LBA30	LBA29	LBA28	LBA27	LBA26	LBA25	LBA24
$41	DMA_b LB ADDR	LBA23	LBA22	LBA21	LBA20	LBA19	LBA18	LBA17	LBA16
$42	DMA_b LB ADDR	LBA15	LBA14	LBA13	LBA12	LBA11	LBA10	LBA9	LBA8
$43	DMA_b LB ADDR	LBA7	LBA6	LBA5	LBA4	LBA3	LBA2	LBA1	LBA0
$44	DMA_b IP ADDR	0	0	0	0	0	0	0	0
$45	DMA_b IP ADDR	IPA23	IPA22	IPA21	IPA20	IPA19	IPA18	IPA17	IPA16
$46	DMA_b IP ADDR	IPA15	IPA14	IPA13	IPA12	IPA11	IPA10	IPA9	IPA8
$47	DMA_b IP ADDR	IPA7	IPA6	IPA5	IPA4	IPA3	IPA2	IPA1	IPA0
$48	DMA_b BYTE CNT	0	0	0	0	0	0	0	0
$49	DMA_b BYTE CNT	BCNT23	BCNT22	BCNT21	BCNT20	BCNT19	BCNT18	BCNT17	BCNT16
$4a	DMA_b BYTE CNT	BCNT15	BCNT14	BCNT13	BCNT12	BCNT11	BCNT10	BCNT9	BCN8
$4b	DMA_b BYTE CNT	BCNT7	BCNT6	BCNT5	BCNT4	BCNT3	BCNT2	BCNT1	BCNT0
$4c	DMA_b TBL ADDR	TA31	TA30	TA29	TA28	TA27	TA26	TA25	TA24
$4d	DMA_b TBL ADDR	TA23	TA22	TA21	TA20	TA19	TA18	TA17	TA16
$4e	DMA_b TBL ADDR	TA15	TA14	TA13	TA12	TA11	TA10	TA9	TA8
$4f	DMA_b TBL ADDR	TA7	TA6	TA5	TA4	TA3	TA2	TA1	TA0

Table 1-7. IP2 chip Memory Map - Control and Status Registers (Continued)

IP2 chip Base Address = $FFFBC000

Register Offset	Register Name	Register Bit Names							
		D7	D6	D5	D4	D3	D2	D1	D0
DMAC for IndustryPack c, request 0. This register set is referred to as DMACc in the text.									
$50	DMA_c STATUS	0	DLBE	0	IPEND	CHANI	TBL	IPTO	DONE
$51	DMA_c INT CTRL	0	0	DINT	DIEN	DICLR	DIL2	DIL1	DIL0
$52	DMA ENABLE	0	0	0	0	0	0	0	DEN
$53	RESERVED	0	0	0	0	0	0	0	0
$54	DMA_c CONTROL 1	DHALT	0	DTBL	ADMA	WIDTH1	WIDTH0	0	XXX
$55	DMA_c CONTROL 2	INTE	0	DMAEI	DMAEO	ENTO	TOIP	0	0
$56	RESERVED	0	0	0	0	0	0	0	0
$57	RESERVED	0	0	0	0	0	0	0	0
$58	DMA_c LB ADDR	LBA31	LBA30	LBA29	LBA28	LBA27	LBA26	LBA25	LBA24
$59	DMA_c LB ADDR	LBA23	LBA22	LBA21	LBA20	LBA19	LBA18	LBA17	LBA16
$5A	DMA_c LB ADDR	LBA15	LBA14	LBA13	LBA12	LBA11	LBA10	LBA9	LBA8
$5B	DMA_c LB ADDR	LBA7	LBA6	LBA5	LBA4	LBA3	LBA2	LBA1	LBA0
$5C	DMA_c IP ADDR	0	0	0	0	0	0	0	0
$5D	DMA_c IP ADDR	IPA23	IPA22	IPA21	IPA20	IPA19	IPA18	IPA17	IPA16
$5E	DMA_c IP ADDR	IPA15	IPA14	IPA13	IPA12	IPA11	IPA10	IPA9	IPA8
$5F	DMA_c IP ADDR	IPA7	IPA6	IPA5	IPA4	IPA3	IPA2	IPA1	IPA0
60	DMA_c BYTE CNT	0	0	0	0	0	0	0	0
$61	DMA_c BYTE CNT	BCNT23	BCNT22	BCNT21	BCNT20	BCNT19	BCNT18	BCNT17	BCNT16
$62	DMA_c BYTE CNT	BCNT15	BCNT14	BCNT13	BCNT12	BCNT11	BCNT10	BCNT9	BCN8
$63	DMA_c BYTE CNT	BCNT7	BCNT6	BCNT5	BCNT4	BCNT3	BCNT2	BCNT1	BCNT0
$64	DMA_c TBL ADDR	TA31	TA30	TA29	TA28	TA27	TA26	TA25	TA24
$65	DMA_c TBL ADDR	TA23	TA22	TA21	TA20	TA19	TA18	TA17	TA16
$66	DMA_c TBL ADDR	TA15	TA14	TA13	TA12	TA11	TA10	TA9	TA8
$67	DMA_c TBL ADDR	TA7	TA6	TA5	TA4	TA3	TA2	TA1	TA0

Table 1-7. IP2 chip Memory Map - Control and Status Registers (Continued)

IP2 chip Base Address = $FFFBC000

Register Offset	Register Name	Register Bit Names							
		D7	D6	D5	D4	D3	D2	D1	D0
DMAC for IndustryPack d, request 0 or for IndustryPack c, request 1, and for PACER CLOCK. This register set, not including the Pacer Clock, is referred to as DMACd in the text.									
$68	DMA_d STATUS	0	DLBE	0	IPEND	CHANI	TBL	IPTO	DONE
$69	DMA_d INT CTRL	0	0	DINT	DIEN	DICLR	DIL2	DIL1	DIL0
$6a	DMA ENABLE	0	0	0	0	0	0	0	DEN
$6b	RESERVED	0	0	0	0	0	0	0	0
$6c	DMA_d CONTROL 1	DHALT	0	DTBL	ADMA	WIDTH1	WIDTH0	C_CH1	XXX
$6d	DMA_d CONTROL 2	INTE	0	DMAEI	DMAEO	ENTO	TOIP	0	0
$6e	RESERVED	0	0	0	0	0	0	0	0
$6f	RESERVED	0	0	0	0	0	0	0	0
$70	DMA_d LB ADDR	LBA31	LBA30	LBA29	LBA28	LBA27	LBA26	LBA25	LBA24
$71	DMA_d LB ADDR	LBA23	LBA22	LBA21	LBA20	LBA19	LBA18	LBA17	LBA16
$72	DMA_d LB ADDR	LBA15	LBA14	LBA13	LBA12	LBA11	LBA10	LBA9	LBA8
$73	DMA_d LB ADDR	LBA7	LBA6	LBA5	LBA4	LBA3	LBA2	LBA1	LBA0
$74	DMA_d IP ADDR	0	0	0	0	0	0	0	0
$75	DMA_d IP ADDR	IPA23	IPA22	IPA21	IPA20	IPA19	IPA18	IPA17	IPA16
$76	DMA_d IP ADDR	IPA15	IPA14	IPA13	IPA12	IPA11	IPA10	IPA9	IPA8
$77	DMA_d IP ADDR	IPA7	IPA6	IPA5	IPA4	IPA3	IPA2	IPA1	IPA0
$78	DMA_d BYTE CNT	0	0	0	0	0	0	0	0
$79	DMA_d BYTE CNT	BCNT23	BCNT22	BCNT21	BCNT20	BCNT19	BCNT18	BCNT17	BCNT16
$7a	DMA_d BYTE CNT	BCNT15	BCNT14	BCNT13	BCNT12	BCNT11	BCNT10	BCNT9	BCN8
$7b	DMA_d BYTE CNT	BCNT7	BCNT6	BCNT5	BCNT4	BCNT3	BCNT2	BCNT1	BCNT0
$7c	DMA_d TBL ADDR	TA31	TA30	TA29	TA28	TA27	TA26	TA25	TA24
$7d	DMA_d TBL ADDR	TA23	TA22	TA21	TA20	TA19	TA18	TA17	TA16
$7e	DMA_d TBL ADDR	TA15	TA14	TA13	TA12	TA11	TA10	TA9	TA8
$7f	DMA_d TBL ADDR	TA7	TA6	TA5	TA4	TA3	TA2	TA1	TA0

Table 1-7. IP2 chip Memory Map - Control and Status Registers (Continued)
IP2 chip Base Address = $FFFBC000

Register Offset	Register Name	Register Bit Names							
		D7	D6	D5	D4	D3	D2	D1	D0
$80	PACER INT CONTROL	0	IRE	INT	IEN	ICLR	IL2	IL1	IL0
$81	PACER GEN CONTROL	PLTY	PLS	0	EN	CLR	PS2	PS1	PS0
$82	PACER TIMER	T15	T14	T13	T12	T11	T10	T9	T8
$83	PACER TIMER	T7	T6	T5	T4	T3	T2	T1	T0

Table 1-8. Z85230 SCC Register Addresses

Z85230 SCC Register	Address
Port B Control	$FFF45001
Port B Data	$FFF45003
Port A Control	$FFF45005
Port A Data	$FFF45007

Note A bug in MVME162FXs that have MC2 chip revision $01 does not allow the data registers to be accessed directly. You must access them indirectly via the SCC chip. The software must send a command to the control register that tells it that the next thing read or written to the control register will go to the data register. The following two macros are examples.

dev_addr is a pointer to the base address of the SCC. SCCR0 is the offset to the SCC control register #0.

```
#define READ_SCC(VAR_NAME)\
        dev_addr[SCCR0] = 0x08;\
        (VAR_NAME) = dev_addr[SCCR0]

#define WRITE_SCC(VAR_NAME)\
        dev_addr[SCCR0] = 0x08;\
        dev_addr[SCCR0] = (VAR_NAME)
```

Table 1-9. 82596CA Ethernet LAN Memory Map

82596CA Ethernet LAN
Directly Accessible Registers

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

Notes 1. Refer to the MPU Port and MPU Channel Attention registers in Chapter 3.

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

Table 1-10. 53C710 SCSI Memory Map

Base Address is $FFF47000

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

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

BBRAM/TOD Clock Memory Map

The MK48T08 BBRAM (also called Non-Volatile RAM or NVRAM) is divided into six areas as shown in Table 1-11. 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 MVME162FX board debugger (MVME162Bug). The fifth area, detailed in Table 1-12, is the configuration area. The sixth area, the TOD clock, detailed in Table 1-13, is defined by the chip hardware.

Table 1-11. 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 1-12. BBRAM Configuration Area Memory Map

Address Range	Description	Size (Bytes)
$FFFC1EF8 - $FFFC1EFB	Version	4
$FFFC1EFC - $FFFC1F07	Serial Number	12
$FFFC1F08 - $FFFC1F17	Board ID	16
$FFFC1F18 - $FFFC1F27	PWA	16
$FFFC1F28 - $FFFC1F2B	Speed	4
$FFFC1F2C - $FFFC1F31	Ethernet Address	6
$FFFC1F32 - $FFFC1F33	Reserved	2
$FFFC1F34 - $FFFC1F35	Local SCSI ID	2
$FFFC1F36 - $FFFC1F3D	Memory Mezz. PWB	8
$FFFC1F3E - $FFFC1F45	Memory Mezz. Serial Number	8
$FFFC1F46 - $FFFC1F4D	Ser. Port 2 Pers. PWB	8
$FFFC1F4E - $FFFC1F55	Ser. Port 2 Pers. Serial No.	8
$FFFC1F56 - $FFFC1F5D	IP a Board ID	8
$FFFC1F5E - $FFFC1F65	IP a Board Serial Number	8
$FFFC1F66 - $FFFC1F6D	IP a Board PWB	8
$FFFC1F6E - $FFFC1F75	IP b Board ID	8
$FFFC1F76 - $FFFC1F7D	IP b Board Serial Number	8
$FFFC1F7E - $FFFC1F85	IP b Board PWB	8

Table 1-12. BBRAM Configuration Area Memory Map (Continued)

Address Range	Description	Size (Bytes)
$FFFC1F86 - $FFFC1F8D	IP c Board ID	8
$FFFC1F8E - $FFFC1F95	IP c Board Serial Number	8
$FFFC1F96 - $FFFC1F9D	IP c Board PWB	8
$FFFC1F9E - $FFFC1FA5	IP d Board ID	8
$FFFC1FA6 - $FFFC1FAD	IP d Board Serial Number	8
$FFFC1FAE - $FFFC1FB5	IP d Board PWB	8
$FFFC1FB6 - $FFFC1FF6	Reserved	65
$FFFC1FF7	Checksum	1

Table 1-13. TOD Clock Memory Map

Address	Data Bits								Function	
	D7	D6	D5	D4	D3	D2	D1	D0		
$FFFC1FF8	W	R	S	--	--	--	--	--	CONTROL	
$FFFC1FF9	ST	--	--	--	--	--	--	--	SECONDS	00
$FFFC1FFA	x	--	--	--	--	--	--	--	MINUTES	00
$FFFC1FFB	x	x	--	--	--	--	--	--	HOUR	00
$FFFC1FFC	x	FT	x	x	x	--	--	--	DAY	01
$FFFC1FFD	x	x	--	--	--	--	--	--	DATE	01
$FFFC1FFE	x	x	x	--	--	--	--	--	MONTH	01
$FFFC1FFF	--	--	--	--	--	--	--	--	YEAR	00

Notes W = Write Bit R = Read Bit S = Signbit

ST = Stop Bit FT = Frequency Test x = Unused

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 MVME162FX board with MC68040, SCSI, Ethernet, 4MB DRAM, and 512KB SRAM, this field contains:

   ```
   MVME162-513A
   ```

   (The 12 characters are followed by four 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 MVME162FX board with MC68040, SCSI, Ethernet, 4MB DRAM, and 512KB SRAM, at revision A, the PWA field contains:

   ```
   01-W3960B01A
   ```

   (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 32.00 MHz board, this field contains:

   ```
   3200
   ```

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 4MB mezzanine at revision A, the PWB field contains:

   ```
   3992B03A
   ```

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. Default addresses for the slave, master, and GCSR address decoders are provided by the **ENV** command.

VMEbus Accesses to the Local Bus

The VMEchip2 includes a user-programmable map decoder for the VMEbus to local bus interface. The map decoder allows you to program the starting and ending address and the modifiers the MVME162FX 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 MVME162FX 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.

Interrupts

The MC68040 uses hardware-vectored interrupts.

Most interrupt sources are level and base vector programmable. Interrupt vectors from the MC2 chip 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 MC2 chip having the highest priority, those from the IP2 chip 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 MC2 chip, IP2 chip, and VMEchip2 ASICs are used to implement the multilevel MC680*x*0 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, and the IP DMA controller.

When snooping is enabled, the MPU can source data and invalidate cache entries as required by the current cycle. The MPU cannot watch VMEbus cycles which do not access the local bus on the MVME162FX. Software must ensure that data shared by multiple

processors is kept in memory that is not cached. The software must also mark all onboard and off-board I/O areas as cache inhibited and serialized.

Sources of Local BERR*

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

Note The present MVME162FX models do not contain parity DRAM.

The devices on the MVME162FX that are able to assert a local bus error are described below.

Local Bus Time-out

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

VMEbus Access Time-out

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

VMEbus BERR*

A VMEbus BERR* occurs when the BERR* signal line is asserted on the VMEbus while a local bus master is accessing the VMEbus. VMEbus BERR* should occur only if: an initialization routine samples to see if a device is present on the VMEbus and it is not,

software accesses a nonexistent device within the VMEbus range, incorrect configuration information causes the VMEchip2 to incorrectly access a device on the VMEbus (such as driving LWORD* low to a 16-bit board), a hardware error occurs on the VMEbus, or a VMEbus slave reports an access error (such as parity error).

Local DRAM Parity Error

Note The present MVME162FX models do not contain parity DRAM.

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.

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 MVME162FX

This section list the various error conditions that are reported by the MVME162FX hardware. A subsection heading identifies each type of error condition. A standard format gives a description of the error, indicates how notification of the error condition is made, indicates which status register(s) have information about the error, and concludes with some comments pertaining to each particular error.

MPU Parity Error

Note The present MVME162FX models do not contain parity DRAM.

Description:
A DRAM parity error.

MPU Notification:
TEA is asserted during an MPU DRAM access.

Status:
Bit 9 of the MPU Status and DMA Interrupt Count Register in the VMEchip2 at address $FFF40048.

Comments:
After memory has been initialized, this error normally indicates a hardware problem.

MPU Off-board Error

Description:
An error occurred while the MPU was attempting to access an off-board resource.

MPU Notification:
TEA is asserted during off-board access.

Status:
Bit 8 of the MPU Status and DMA Interrupt Count Register. Address $FFF40048.

Comments:
This can be caused by a VMEbus time-out, a VMEbus BERR, or an MVME162FX VMEbus access time-out. 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 Time-out

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

Note The present MVME162FX models do not contain parity DRAM.

Description:
Parity error while the DMAC was reading DRAM.

MPU Notification:
DMAC interrupt (when enabled).

Status:
The DLPE bit is set in the DMAC Status Register (address $FFF40048 bit 5).

Comments:
If the TBL bit is set (address $FFF40048 bit 2) the error occurred during a command table access, otherwise the error occurred during a data access.

DMAC Off-board 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 time-out (LTO) occurred while the DMAC was local bus master.

MPU Notification:
DMAC interrupt (when enabled).

Status:
The DLTO bit is set in the DMAC Status Register (address $FFF40048
bit 3).

Comments:
This indicates the DMAC attempted to access a local bus address at which there was no resource. If the TBL bit is set (address $FFF40048 bit 2) the error occurred during a command table access, otherwise the error occurred during a data access.

DMAC TEA - Cause Unidentified

Description:
An error occurred while the DMAC was local bus master and additional status was not provided.

MPU Notification:
DMAC interrupt (when enabled).

Status:
The DLBE bit is set in the DMAC Status Register (address $FFF40048 bit 6).

Comments:
An 8- or 16-bit write to the LCSR in the VMEchip2 causes this error. If the TBL bit is set (address $FFF40048 bit 2) the error occurred during a command table access, otherwise the error occurred during a data access.

LAN Parity Error

Note The present MVME162FX models do not contain parity DRAM.

Description:
Parity error while the LANCE was reading DRAM MPU.

Notification:
MC2 chip Interrupt (LAN ERROR IRQ).

Status:
MC2 chip 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 MC2 chip. Control for the interrupt is in the MC2 chip LAN Error Interrupt Control Register ($FFF4202B).

LAN Off-board Error

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

MPU Notification:
MC2 chip Interrupt (LAN ERROR IRQ).

Status:
MC2 chip 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 MC2 chip. Control for the interrupt is in the MC2 chip LAN Error Interrupt Control Register ($FFF4202B).

LAN LTO Error

Description:
Local Bus Time-out occurred while the LANCE was local bus master.

MPU Notification:
MC2 chip Interrupt (LAN ERROR IRQ).

Status:
MC2 chip 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 MC2 chip. Control for the interrupt is in the MC2 chip LAN Error Interrupt Control Register ($FFF4202B).

SCSI Parity Error

Note The present MVME162FX models do not contain parity DRAM.

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

MPU Notification:
53C710 Interrupt.

Status:
53C710 DMA Status Register 53C710 DMA Interrupt Status Register MC2 chip SCSI Error Status Register ($FFF4202C).

Comments:
53C710 interrupt enables are controlled in the 53C710 and in the MC2 chip SCSI Interrupt Control Register ($FFF4202F).

SCSI Off-board 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 MC2 chip SCSI Error Status Register ($FFF4202C).

Comments:
53C710 interrupt enables are controlled in the 53C710 and in the MC2 chip SCSI Interrupt Control Register ($FFF4202F).

SCSI LTO Error

Description:
Local Bus Time-out occurred while the 53C710 was local bus master.

MPU Notification:
53C710 Interrupt.

Status:
53C710 DMA Status Register 53C710 DMA Interrupt Status Register MC2 chip SCSI Error Status Register ($FFF4202C).

Comments:
53C710 interrupt enables are controlled in the 53C710 and in the MC2 chip 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 MVME162FX accesses the local bus memory on another MVME162FX 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 MVME162FX accessing another MVME162FX 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 µsec.

The next timer to take over when one MVME162FX 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 time-out 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 sec. This timer can also be disabled for debug purposes. Before an MVME162FX access to another MVME162FX can complete, however, the VMEchip2 on the accessed MVME162FX must decode a slave access and request the local bus of the second MVME162FX. When the local bus is granted (any in-process onboard transfers have completed) then the local bus timer of the accessed MVME162FX starts. Normally, this is also set to 256 µsec. 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.

MVME162FX 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 off-board resources. The MVME162FX does not fully support all RMW operations in all possible cases.

The MVME162FX makes the following assumptions and supports a limited subset of RMW instructions. The MVME162FX 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 MVME162FX 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.

VMEchip2 2

Introduction

This chapter defines the VMEchip2 ASIC, 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), and
    Release-On-Request (ROR); and
    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, two four-bytes in non-BLT mode).
  - An eight four-byte read ahead buffer (BLT mode only).
- 32-bit Local to VMEbus DMA Controller:
  - Programmable 16-bit, 32-bit, and 64-bit VMEbus data width.
  - Programmable short, standard, and extended VMEbus addressing.
  - Programmable AM code.
  - Programmable local bus snoop enable.
  - 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), and

Release-On-End-Of-Data (ROEOD);
Software-configured BR0-BR3 request levels; and
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), and
    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

The following sections provide an overview of the functions provided by the VMEchip2. See Figure 2-1 for a block diagram of the VMEchip2. A detailed programming model for the local control and status registers (LCSR) is provided in the following section. A detailed programming model for the global control and status registers (GCSR) is provided in the next section.

Local Bus to VMEbus Interface

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

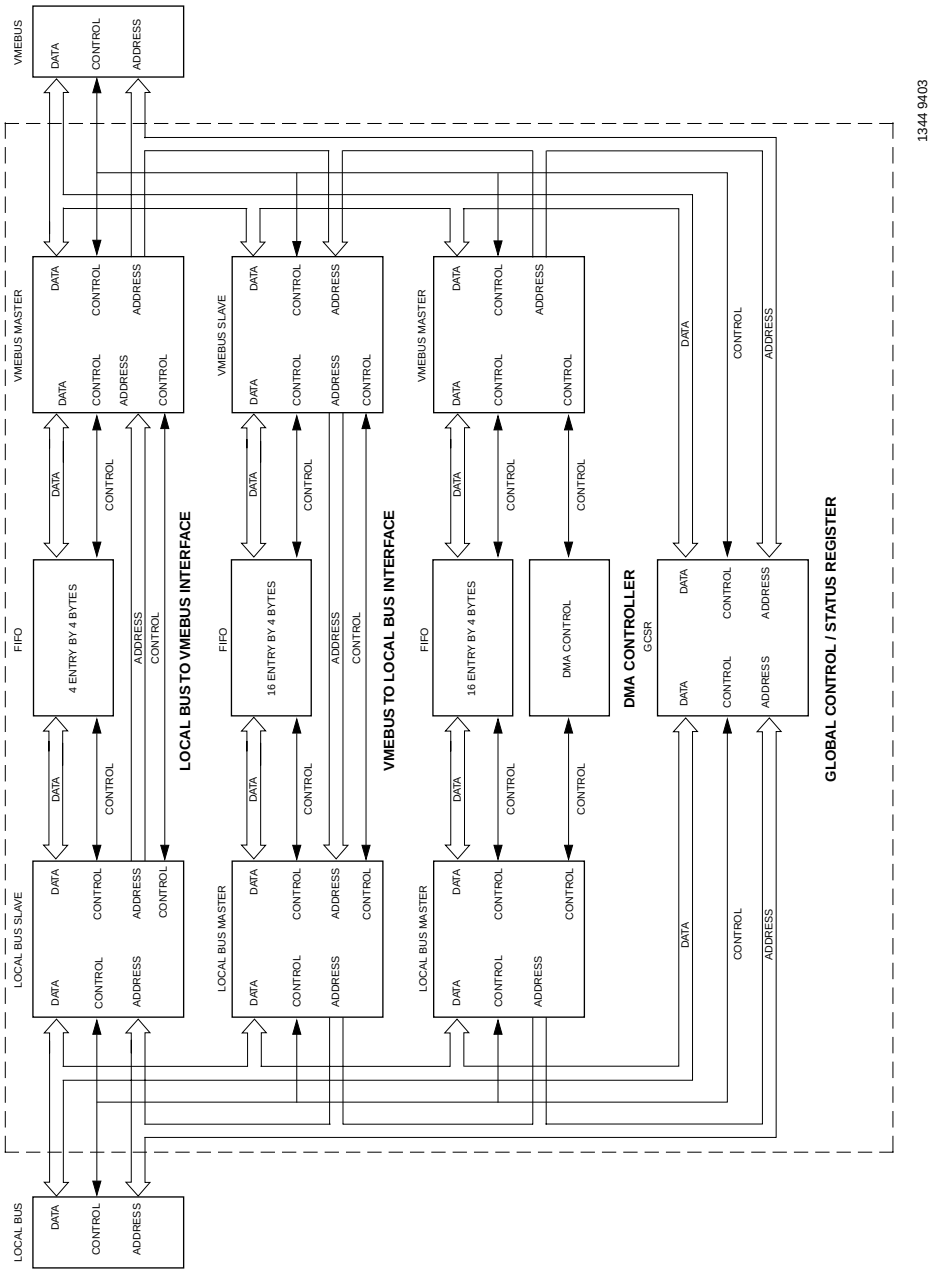

Figure 2-1. VMEchip2 Block Diagram

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

Addressing capabilities: A16, A24, A32
Data transfer capabilities: D08, D16, D32

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

The first four map decoders compare local bus address lines A31 through A16 with a 16-bit start address and a 16-bit end address. When an address in the selected range is detected, a VMEbus select is generated to the VMEbus master. Each map decoder also has eight attribute bits and an enable bit. The attribute bits are for VMEbus AM 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 $FF7FFFFF.

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

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

The write post buffer stores 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.

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 µs, 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 off-board VMEbus master access to onboard resources. The VMEbus to local bus interface includes the V*MEbus 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 MVME162FX 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 onchip 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 Timers

The VMEchip2 includes two general purpose tick timers. These timers can be used to generate interrupts at various rates or the counters can be read at various times for interval timing. The timers have a resolution of 1 µs 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 1µs. When the counter is enabled in the clear-on-compare mode, it increments every 1µs 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 that the one being serviced.

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

IACK Daisy-Chain Driver

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

Bus Timer

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

In addition to the VMEbus timer, the chip contains a local bus timer. This timer asserts the local TEA when the local bus cycle maintained in its asserted state for longer that the programmed time-out period. This timer can be enabled or disabled under software control. The time-out period can be programmed for 8, 64, or 256 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 time-out, or by a control bit in the LCSR. SYSRESET remains asserted for at least 200 msec, as required by the VMEbus specification.

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

The local reset driver is enabled even when the chip is not the system controller. A local reset may be generated by the RESET switch, a power up reset, a watch dog time-out, a VMEbus SYSRESET, 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 by 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 shown in 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					ADDER 2	SNP 2		WP 2	SUP 2	USR 2	A32 2	A24 2	BLK D64 2	BLK 2	PRGM 2	DATA2
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
14	MASTER ENDING ADDRESS 1															
18	MASTER ENDING ADDRESS 2															
1C	MASTER ENDING ADDRESS 3															
20	MASTER ENDING ADDRESS 4															
24	MASTER ADDRESS TRANSLATION ADDRESS 4															
28	MAST D16 EN	MAST WP EN	MASTER AM 4						MAST D16 EN	MAST WP EN	MASTER AM 3					
2C	GCSR GROUP SELECT								GCSR BOARD SELECT				MAST 4 EN	MAST 3 EN	MAST 2 EN	MAST 1 EN
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
30											WAIT RMW	ROM ZERO	DMA TB SNP MODE		SRAM SPEED	
34																
38	DMA CONTROLLER															
3C	DMA CONTROLLER															
40	DMA CONTROLLER															
44	DMA CONTROLLER															
48		TICK 2/1	TICK IRQ 1 EN	CLR IRQ	IRQ STAT	VMEBUS INTERRUPT LEVEL			VMEBUS INTERRUPT VECTOR							

This sheet continues on facing page. ⟶

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SLAVE STARTING ADDRESS 1															
SLAVE STARTING ADDRESS 2															
SLAVE ADDRESS TRANSLATION SELECT 1															
SLAVE ADDRESS TRANSLATION SELECT 2															
				ADDER 1	SNP 1		WP 1	SUP 1	USR 1	A32 1	A24 1	BLK D64 1	BLK 1	PRGM 1	DATA1

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 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								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		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																
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		AC FAIL IRQ LEVEL				ABORT IRQ LEVEL				SYS FAIL IRQ LEVEL				MST WP ERROR IRQ LEVEL		
7C		VME IACK IRQ LEVEL				DMA IRQ LEVEL				SIG 3 IRQ LEVEL				SIG 2 IRQ LEVEL		
80		SW7 IRQ LEVEL				SW6 IRQ LEVEL				SW5 IRQ LEVEL				SW4 IRQ LEVEL		
84		SPARE IRQ LEVEL				VME IRQ 7 IRQ LEVEL				VME IRQ 6 IRQ LEVEL				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																

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.

Programming the VMEbus Slave Map Decoders

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

The VMEbus to local bus interface allows off-board VMEbus masters access to local 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 the following registers: *address translation address register*, *address translation select register*, *starting address register*, *ending address register*, *address modifier select register*, and *attribute register*. The addresses and bit definitions of these registers are shown in the following tables.

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

and starting and ending address registers should be programmed first, and then the map decoders should be enabled by programming the address modifier select registers.

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 accesses 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 VMEchip2 used on the MVME166/167/187 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	$FFF40000 (16 bits of 32)		
BIT	31	. . .	16
NAME	Ending Address Register 1		
OPER	R/W		
RESET	0 PS		

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

VMEbus Slave Starting Address Register 1

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

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

VMEbus Slave Ending Address Register 2

ADR/SIZ	$FFF40004 (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

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

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

VMEbus Slave Address Translation Address Offset Register 1

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

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

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

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

VMEbus Slave Address Translation Address Offset Register 2

ADR/SIZ	$FFF4000C (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	$FFF4000C (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	Segment Size	Address Translation Select Value
64KB	FFFF	32MB	FE00
128KB	FFFE	64MB	FC00
256KB	FFFC	128MB	F800
512KB	FFF8	256MB	F000
1MB	FFF0	512MB	E000
2MB	FFE0	1GB	C000
4MB	FFC0	2GB	8000
8MB	FF80	4GB	0000
16MB	FF00		

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					ADDER2	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 responded to VMEbus user access cycles.

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

VMEbus Slave Write Post and Snoop Control Register 1

ADR/SIZ	$FFF40010 (8 bits [4 used] of 32)							
BIT	15	14	13	12	11	10	9	8
NAME					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	$FFF40010 (8 bits of 32)							
BIT	7	6	5	4	3	2	1	0
NAME	SUP	USR	A32	A24	D64	BLK	PGM	DAT
OPER	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RESET	0 PSL	0 PSL	0 PSL	0 PSL	0 PSL	0 PSL	0 PSL	0 PSL

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

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

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

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

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

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

A32 When this bit is high, the first map decoder responds to VMEbus A32 (extended) access cycles. When this bit is low, the first map decoder does not respond to VMEbus A32 access cycles.

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

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

Programming the Local Bus to VMEbus Map Decoders

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

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

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

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

A local bus slave map decoder is programmed by loading the starting address of the segment into the *starting address register* and the ending address of the segment into the *ending address register*. The address modifier code is programmed 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 $FFFFFFFF to the A16 (short I/O) map of the VMEbus. This segment may be enabled using the enable bit. Write posting may be enabled for this segment using the write post enable bit. The transfer size may be D16 or D32 as defined by the D16 bit in the control register.

The second I/O map decoder provides support for the other I/O map of the VMEbus. This decoder maps the local bus address range $F0000000 to $F0FFFFFF to the A24 map of the VMEbus and the address range $F1000000 to $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

ADR/SIZ	$FFF40014 (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 local bus to VMEbus map decoder.

Local Bus Slave (VMEbus Master) Starting Address Register 1

ADR/SIZ	$FFF40014 (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 local bus to VMEbus map decoder.

Local Bus Slave (VMEbus Master) Ending Address Register 2

ADR/SIZ	$FFF40018 (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 local bus to VMEbus map decoder.

Local Bus Slave (VMEbus Master) Starting Address Register 2

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

ADR/SIZ	$FFF4001C (16 bits of 32)		
BIT	31	. . .	16
NAME	Ending Address Register 3		
OPER	R/W		
RESET	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

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

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

Local Bus Slave (VMEbus Master) Ending Address Register 4

ADR/SIZ	$FFF40020 (16 bits of 32)		
BIT	31	. . .	16
NAME	Ending Address Register 4		
OPER	R/W		
RESET	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

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

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

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

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

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

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

Local Bus Slave (VMEbus Master) Attribute Register 4

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

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

AM These bits define the VMEbus address modifier codes 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	O 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

ADR/SIZ	$FFF40028 (8 bits of 32)							
BIT	15	14	13	12	11	10	9	8
NAME	D16	WP	AM					
OPER	R/W	R/W	R/W					
RESET	0 PS	0 PS	O 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	O 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 4		
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 MVME162FX 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)							
BIT					19	18	17	16
NAME					EN4	EN3	EN2	EN1
OPER					R/W	R/W	R/W	R/W
RESET					0 PSL	0 PSL	0 PSL	0 PSL

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

EN1 When this bit is high, the first local bus to VMEbus map decoder is enabled. When this bit is low, the first local bus to VMEbus map decoder is disabled.

EN2 When this bit is high, the second local bus to VMEbus map decoder is enabled. When this bit is low, the second local bus to VMEbus map decoder is disabled.

EN3 When this bit is high, the third local bus to VMEbus map decoder is enabled. When this bit is low, the third local bus to VMEbus map decoder is disabled.

EN4 When this bit is high, the fourth local bus to VMEbus map decoder is enabled. When this bit is low, the fourth local bus to VMEbus map decoder is disabled.

Local Bus to VMEbus I/O Control Register

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

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

This function is not used on the MVME162FX.

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 MVME162FX.

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 MVME162FX. Its function is performed by the ROM0 bit in the PROM Access Time Control Register in the MC2 chip. Refer to Chapter 3.

WAIT RMW This function is not used on the MVME162FX.

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 re-requested at the old level. The requester always requests the VMEbus at level 3 the first time following a SYSRESET.

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

LVRWD When this bit is high, the requester operates in 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	$FFF40030 (8 bits of 32)							
BIT	7	6	5	4	3	2	1	0
NAME	DHALT	DEN	DTBL	DFAIR	DRELM		DREQL	
OPER	S	S	R/W	R/W	R/W		R/W	
RESET	0 PS	0 PS	0 PS	0 PS	0 PS		0 PS	

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

DREQL These bits define the VMEbus request level for the DMAC requester. The request 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 re-requested at the old level. The requester always requests the VMEbus at level 3 the first time following a SYSRESET.

0 VMEbus request level 0
1 VMEbus request level 1
2 VMEbus request level 2
3 VMEbus request level 3

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

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

DFAIR When this bit is high, the DMAC requester operates in 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	$FFF40038 (32 bits)		
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	$FFF4003C (32 bits)		
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

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

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

Table Address Counter

ADR/SIZ	$FFF40044 (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	$FFF40048 (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	$FFF40048 (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 off-board. 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. This bit is not defined for MVME162FX implementation.

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

This is the DMAC status register.

DONE This bit is set when the DMAC has finished executing commands 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 time-out. This bit is cleared when the DMAC is enabled.

DLOB When this bit is set, the DMAC received a TEA and the status indicated off-board. 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. This bit is not defined for MVME162FX implementation.

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 time-out. 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 Time-out Control Register

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

This register controls the VMEbus arbiter time-out timer.

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

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

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

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

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

0	8 µs
1	64 µs
2	256 µs
3	The timer is disabled

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

0	16 µs	**4**	256 µs
1	32 µs	**5**	512 µs
2	64 µs	**6**	1024 µs
3	128 µs	**7**	When done (or no data)

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

0	0 µs	**4**	128 µs
1	16 µs	**5**	256 µs
2	32 µs	**6**	512 µs
3	64 µs	**7**	1024 µs

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

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

WDTO These bits define the watchdog time-out period:

Bit Encoding	Time-out	Bit Encoding	Time-out
0	512 µs	8	128 ms
1	1 ms	9	256 ms
2	2 ms	10	512 ms
3	4 ms	11	1 s
4	8 ms	12	4 s
5	16 ms	13	16 s
6	32 ms	14	32 s
7	64 ms	15	64 s

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

0	8 µs
1	64 µs
2	256 µs
3	The timer is disabled

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

0	64 µs
1	1 ms
2	32 ms
3	The timer is disabled

Prescaler Control Register

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

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

prescaler register = 256 - *B clock* (MHz)

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

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

tick timer clock = *B clock* / (256 - *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	$FFF40050 (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).

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

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	$FFF40054 (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	$FFF40058 (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.

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

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	$FFF4005C (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.

Tick Timer 2 Compare Register

ADR/SIZ	$FFF40058 (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.

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

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	$FFF4005C (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	$FFF40060 (8 bits [7 used] of 32)							
BIT	31	30	29	28	27	26	25	24
NAME		SCON	SFFL	BRFLI	PURS	CPURS	BDFLO	RSWE
OPER		R	R	R	R	C	R/W	R/W
RESET		X	X	1 PSL	1 P	0 PS	1 PSL	1 P

RSWE The RESET switch enable bit is used with the ''no VMEbus interface'' option. This bit is duplicated at the same bit position in the MC2 chip at location $FFF42044. When this bit or the duplicate bit in the MC2 chip is high, the RESET switch is enabled. When both bits are low, the RESET switch is disabled.

BDFLO When this bit is set high, the VMEchip2 asserts the BRDFAIL signal pin. When set low, this bit does not contribute to the BRDFAIL signal on the VMEchip2.

CPURS When this bit is set high, the powerup reset status bit is cleared. This bit is always read zero.

PURS This bit is set by a powerup reset. It is cleared by a write to the CPURS bit.

BRFLI When this status bit is set high, the BRDFAIL signal pin on the VMEchip2 is asserted. When set low, the BRDFAIL signal pin on the VMEchip2 is not asserted. The BRDFAIL pin may be asserted by an external device, the BDFLO bit in this register, or a watchdog time-out.

SFFL When this status bit is set high, the SYSFAIL signal line on the VMEbus is asserted. When set low, the SYSFAIL signal line on the VMEbus is not asserted.

SCON When this status bit is set high, the VMEchip2 is configured as system controller. When set 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	1 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 time-out occurs, a SYSRESET or LRESET is generated. The WDS/L bit in this register selects the reset. When this bit is low, a watchdog time-out does not cause a reset.

WDS/L When this bit is high and the watchdog timer has timed out and the watchdog reset enable (WDRSE bit in this register) is high, a SYSRESET signal is generated on the VMEbus which in turn causes LRESET to be asserted. When this bit is low and the watchdog timer has timed out and the watchdog reset enable (WDRSE bit in this register) is high, an LRESET signal is generated on the local bus.

WDBFE When this bit is high and the watchdog timer has timed out, the VMEchip2 asserts the BRDFAIL signal pin. When this bit is low, the watchdog timer does not contribute to the BRDFAIL signal on the VMEchip2.

WDTO When this status bit is high, a watchdog time-out has occurred. When this status bit is low, a watchdog time-out has not occurred. This bit is cleared by writing a 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 time-out period or a watchdog time-out occurs.

WDCS When this bit is set high, the watchdog time-out status bit (WDTO bit in this register) is cleared.

SRST When this bit is set high, a SYSRESET signal is generated on the VMEbus. SYSRESET resets the VMEchip2 and clears this bit.

Tick Timer 2 Control Register

ADR/SIZ	$FFF40060 (8 bits [7 used] of 32)							
BIT	15	14	13	12	11	10	9	8
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 of 32)							
BIT	7	6	5	4	3	2	1	0
NAME	OVF					COVF	COC	EN
OPER	R					C	R/W	R/W
RESET	0 PS					0 PS	0 PS	0 PS

EN When this bit is high, the counter increments. When this bit is low, the counter does not increment.

COC When this bit is high, the counter is reset to 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.

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.

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	

Table 2-3. Local Bus Interrupter Summary (Continued)

Interrupt	Vector	Priority for Simultaneous Interrupts
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	
(Not used on MVME162FX)	$XB	
VMEbus Master Write Post Error	$XC	
VMEbus SYSFAIL	$XD	
(Not used on MVME162FX)	$XE	
VMEbus ACFAIL	$XF	Highest

Notes 1. X = The contents of vector base register 0.

2. Y = The contents of vector base register 1.

3. Refer to the Vector Base Register description later in this chapter for recommended Vector Base Register values.

Local Bus Interrupter Status Register (bits 24-31)

ADR/SIZ	$FFF40068 (8 bits of 32)							
BIT	31	30	29	28	27	26	25	24
NAME	ACF	AB	SYSF	MWP	PE	VI1E	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:

TIC1 Tick timer 1 interrupt

TIC2 Tick timer 2 interrupt

VI1E VMEbus IRQ1 edge-sensitive interrupt

PE Not used on MVME162FX

MWP VMEbus master write post error interrupt

SYSF VMEbus SYSFAIL interrupt

AB Not used on MVME162FX

ACF VMEbus ACFAIL interrupt

Local Bus Interrupter Status Register (bits 16-23)

ADR/SIZ	$FFF40068 (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:

LM0 GCSR LM0 interrupt

LM1 GCSR LM1 interrupt

SIG0 GCSR SIG0 interrupt

SIG1 GCSR SIG1 interrupt

SIG2 GCSR SIG2 interrupt

SIG3 GCSR SIG3 interrupt

DMA DMAC interrupt

VIA VMEbus interrupter acknowledge interrupt

Local Bus Interrupter Status Register (bits 8-15)

ADR/SIZ	$FFF40068 (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:

SW0 Software 0 interrupt

SW1 Software 1 interrupt

SW2 Software 2 interrupt

SW3 Software 3 interrupt

SW4 Software 4 interrupt

SW5 Software 5 interrupt

SW6 Software 6 interrupt

SW7 Software 7 interrupt

Local Bus Interrupter Status Register (bits 0-7)

ADR/SIZ	$FFF40068 (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:

VME1 VMEbus IRQ1 Interrupt

VME2 VMEbus IRQ2 Interrupt

VME3 VMEbus IRQ3 Interrupt

VME4 VMEbus IRQ4 Interrupt

VME5 VMEbus IRQ5 Interrupt

VME6 VMEbus IRQ6 Interrupt

VME7 VMEbus IRQ7 Interrupt

SPARE Not used

Local Bus Interrupter Enable Register (bits 24-31)

ADR/SIZ	$FFF4006C (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.

ETIC1 Enable tick timer 1 interrupt

ETIC2 Enable tick timer 2 interrupt

EVI1E Enable VMEbus IRQ1 edge-sensitive interrupt

EPE Not used on MVME162FX

EMWP Enable VMEbus master write post error interrupt

ESYSF Enable VMEbus SYSFAIL interrupt

EAB Not used on MVME162FX

EACF Enable VMEbus ACFAIL interrupt

Local Bus Interrupter Enable Register (bits 16-23)

ADR/SIZ	$FFF4006C (8 bits of 32)							
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.

ELM0 Enable GCSR LM0 interrupt

ELM1 Enable GCSR LM1 interrupt

ESIG0 Enable GCSR SIG0 interrupt

ESIG1 Enable GCSR SIG1 interrupt

ESIG2 Enable GCSR SIG2 interrupt

ESIG3 Enable GCSR SIG3 interrupt

EDMA Enable DMAC interrupt

EVIA VMEbus interrupter acknowledge interrupt

Local Bus Interrupter Enable Register (bits 8-15)

ADR/SIZ	$FFF4006C (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.

ESW0 Enable software 0 interrupt

ESW1 Enable software 1 interrupt

ESW2 Enable software 2 interrupt

ESW3 Enable software 3 interrupt

ESW4 Enable software 4 interrupt

ESW5 Enable software 5 interrupt

ESW6 Enable software 6 interrupt

ESW7 Enable software 7 interrupt

Local Bus Interrupter Enable Register (bits 0-7)

ADR/SIZ	$FFF4006C (8 bits of 32)							
BIT	7	6	5	4	3	2	1	0
NAME	SPARE	EIRQ7	EIRQ6	EIRQ5	EIRIQ4	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.

EIRQ1 Enable VMEbus IRQ1 interrupt

EIRQ2 Enable VMEbus IRQ2 interrupt

EIRQ3 Enable VMEbus IRQ3 interrupt

EIRQ4 Enable VMEbus IRQ4 interrupt

EIRQ5 Enable VMEbus IRQ5 interrupt

EIRQ6 Enable VMEbus IRQ6 interrupt

EIRQ7 Enable VMEbus IRQ7 interrupt

SPARE SPARE

Software Interrupt Set Register (bits 8-15)

ADR/SIZ	$FFF40070 (8 bits of 32)							
BIT	15	14	13	12	11	10	9	8
NAME	SSW7	SSW6	SSW5	SSW4	SSW3	SSW2	SSW1	SSW0
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:

SSW0 Set software 0 interrupt

SSW1 Set software 1 interrupt

SSW2 Set software 2 interrupt

SSW3 Set software 3 interrupt

SSW4 Set software 4 interrupt

SSW5 Set software 5 interrupt

SSW6 Set software 6 interrupt

SSW7 Set software 7 interrupt

Interrupt Clear Register (bits 24-31)

ADR/SIZ	$FFF40074 (8 bits of 32)							
BIT	31	30	29	28	27	26	25	24
NAME	CACF	CAB	CSYSF	CMWP	CPE	CVI1E	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.

CTIC1 Clear tick timer 1 interrupt

CTIC2 Clear tick timer 2 interrupt

CVI1E Clear VMEbus IRQ1 edge-sensitive interrupt

CPE Not used on MVME162FX

CMWP Clear VMEbus master write post error interrupt

CSYSF Clear VMEbus SYSFAIL interrupt

CAB Not used on MVME162FX

CACF Clear VMEbus ACFAIL interrupt

Interrupt Clear Register (bits 16-23)

ADR/SIZ	$FFF40074 (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.

CLM0 Clear GCSR LM0 interrupt

CLM1 Clear GCSR LM1 interrupt

CSIG0 Clear GCSR SIG0 interrupt

CSIG1 Clear GCSR SIG1 interrupt

CSIG2 Clear GCSR SIG2 interrupt

CSIG3 Clear GCSR SIG3 interrupt

CDMA Clear DMA controller interrupt

CVIA Clear VMEbus interrupter acknowledge interrupt

Interrupt Clear Register (bits 8-15)

ADR/SIZ	$FFF40074 (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:

CSW0 Clear software 0 interrupt

CSW1 Clear software 1 interrupt

CSW2 Clear software 2 interrupt

CSW3 Clear software 3 interrupt

CSW4 Clear software 4 interrupt

CSW5 Clear software 5 interrupt

CSW6 Clear software 6 interrupt

CSW7 Clear software 7 interrupt

Interrupt Level Register 1 (bits 24-31)

ADR/SIZ	$FFF40078 (8 bits [6 used] of 32)							
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 MVME162FX

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

Interrupt Level Register 1 (bits 16-23)

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

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

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

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

Interrupt Level Register 1 (bits 8-15)

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

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

IRQ1E LEVEL These bits define the level of the VMEbus IRQ1 edge-sensitive interrupt.

PE LEVEL Not used on MVME162FX

Interrupt Level Register 1 (bits 0-7)

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

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

TICK1 LEVEL These bits define the level of the tick timer 1 interrupt.

TICK2 LEVEL These bits define the level of the tick timer 2 interrupt.

Interrupt Level Register 2 (bits 24-31)

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

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

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

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

Interrupt Level Register 2 (bits 16-23)

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

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

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

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

Interrupt Level Register 2 (bits 8-15)

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

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

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

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

Interrupt Level Register 2 (bits 0-7)

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

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

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

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

Interrupt Level Register 3 (bits 24-31)

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

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

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

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

Interrupt Level Register 3 (bits 16-23)

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

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

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

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

Interrupt Level Register 3 (bits 8-15)

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

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

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

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

Interrupt Level Register 3 (bits 0-7)

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

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

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

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

Interrupt Level Register 4 (bits 24-31)

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

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

VIRQ7 LEVEL These bits define the level of the VMEbus IRQ7 interrupt.

SPARE LEVEL Not used on the MVME162FX

Interrupt Level Register 4 (bits 16-23)

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

This register is used to define the level of the VMEbus 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				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				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	$FFF40088 (8 bits of 32)							
BIT	**31**	**30**	**29**	**28**	**27**	**26**	25	24
NAME	VBR 0				VBR 1			
OPER	R/W				R/W			
RESET	0 PSL				0 PSL			

This register is used to define the interrupt base vectors.

VBR 1 These bits define the interrupt base vector 1.

VBR 0 These bits define the interrupt base vector 0.

Note Refer to Table 2-3, *Local Bus Interrupter Summary*, 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	GPOEN3	GPOEN2	GPOEN1	GPOEN0
OPER	R/W	R	R	R	R/W	R/W	R/W	R/W
RESET	0 PSL	X	X	X	0 PS	0 PS	0 PS	0 PS

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

Bits 16-19 control the direction of the four General Purpose I/O pins (GPIO0-3).

Note The General Purpose I/O pins are not used on the MVME162FX.

GPOEN0 Not used on MVME162FX

GPOEN1 Not used on MVME162FX

GPOEN2 Not used on MVME162FX

GPOEN3 Not used on MVME162FX

ABRTL Not used on MVME162FX

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 function is not used on the MVME162FX.

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	GPI40	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 MVME162FX.

Miscellaneous Control Register

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

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

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

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

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

NOELBBSY When this bit is high, the early release feature of bus busy feature on the VMEbus is disabled. The VMEchip2 drives BBSY* low whenever VMEbus AS* is low. When this bit is low, the early release feature of bus busy feature on the VMEbus is not disabled.

DISMST When this bit is high, the VME LED on the MVME162FX 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 MVME162FX is lit when local bus reset is asserted, the VMEchip2 is driving local bus busy, or the VMEchip2 is driving the VMEbus address strobe.

DISSRAM When this bit is high, the SRAM decoder in the VMEchip2 is disabled. When this bit is low, the SRAM decoder in the VMEchip2 is enabled. Because the SRAM decoder in the VMEchip2 is not used on the MVME162FX, this bit must be set.

REVEROM This function is not used on the MVME162FX. This bit must not be set.

MPIRQEN This function is not used on the MVME162FX. This bit must not be set.

GCSR Programming Model

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

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

The GCSR section of the VMEchip2 contains the following registers: a *chip ID register*, a *chip revision register*, a *location monitor status registe*r, 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.

Caution

The GCSR allows a VMEbus master to reset the local bus. This feature is very dangerous and should be used with caution. The local reset feature is a partial system reset, not a complete system reset such as powerup reset or SYSRESET. When the local bus reset signal is

asserted, a local bus cycle may be aborted. The VMEchip2 is connected to both the local bus and the VMEbus and if the aborted cycle is bound for the VMEbus, erratic operation may result. Communications between the local processor and a VMEbus master should use interrupts or mailbox locations; reset should not be used in normal communications. Reset should be used only when the local processor is halted or the local bus is hung and reset is the last resort.

Programming the GCSR

A complete description of the GCSR is provided in the following tables. Each register definition includes a table with five 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 This bit is affected by power-up reset.

  S The bit is affected by SYSRESET.

  L The bit is affected by local bus reset.

  X The bit is not affected by reset.

A summary of the GCSR is shown in Table 2-4.

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	SIG3	SIG2	SIG1	SIG0	RST	ISF	BF	SCON	SYSFL	X	X	X
4	8	GENERAL PURPOSE CONTROL AND STATUS REGISTER 0															
6	C	GENERAL PURPOSE CONTROL AND STATUS REGISTER 1															
8	10	GENERAL PURPOSE CONTROL AND STATUS REGISTER 2															
A	14	GENERAL PURPOSE CONTROL AND STATUS REGISTER 3															
C	18	GENERAL PURPOSE CONTROL AND STATUS REGISTER 4															
E	1C	GENERAL PURPOSE CONTROL AND STATUS REGISTER 5															

VMEchip2 Revision Register

ADR/SIZ	Local Bus: $FFF40100/VMEbus: $XXY0 (8 bits)		
BIT	15	. . .	8
NAME	VMEchip2 Revision Register		
OPER	R		
RESET	01 PS		

This register is the VMEchip2 revision register. The revision level for the VMEchip2 starts at zero and is incremented if mask changes are required. The VMEchip2 used on the MVME162FX is revision $01 or greater.

VMEchip2 ID Register

ADR/SIZ	Local Bus: $FFF40100/VMEbus: $XXY0 (8 bits)		
BIT	7	. . .	0
NAME	VMEchip2 ID Register		
OPER	R		
RESET	10 PS		

This register is the VMEchip2 ID register. The ID for the VMEchip2 is 10.

VMEchip2 LM/SIG Register

ADR/SIZ	Local Bus: $FFF40104/VMEbus: $XXY2 (8 bits)							
BIT	15	14	13	12	11	10	9	8
NAME	LM3	LM2	LM1	LM0	SIG3	SIG2	SIG1	SIG0
OPER	R	R	R	R	S/R	S/R	S/R	S/R
RESET	1 PS	1 PS	1 PS	1 PS	0 PS	0 PS	0 PS	0 PS

This register is the VMEchip2 location monitor register and the interrupt register.

SIG0 The SIG0 bit is set when a VMEbus master writes a 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: $FFF40104/VMEbus: $XXY2 (8 bits [5 used])							
BIT	7	6	5	4	3	2	1	0
NAME	RST	ISF	BF	SCON	SYSFL			
OPER	S/R	R/W	R	R	R			
RESET	0 PSL	0 PSL	1 PS	X	1 PSL			

This register is the VMEchip2 board status/control register.

SYSFL This bit is set when the VMEchip2 is driving the SYSFAIL signal.

SCON This bit is set if the VMEchip2 is system controller.

BF When this bit is high, the Board Fail signal is active. When this bit is low, the Board Fail signal is inactive. When this bit is set, the VMEchip2 drives SYSFAIL if the inhibit SYSFAIL bit is not set.

ISF When this bit is set, the VMEchip2 is prevented from driving the VMEbus SYSFAIL signal line. When this bit is cleared, the VMEchip2 is allowed to drive the VMEbus SYSFAIL signal line.

RST This bit allows a VMEbus master to reset the local bus. Refer to the note on local reset in the ***GCSR Programming Model*** section, earlier in this chapter. When this bit is set, a local bus reset is generated. This bit is cleared by the local bus reset.

General Purpose Register 0

ADR/SIZ	Local Bus: $FFF40108/VMEbus: $XXY4 (16 bits)		
BIT	15	. . .	0
NAME	General Purpose Register 0		
OPER	R/W		
RESET	0 PS		

This register is a general purpose register that allows a local bus master to communicate with a VMEbus master. The function of this register is not defined by the hardware specification.

General Purpose Register 1

ADR/SIZ	Local Bus: $FFF4010C/VMEbus: $XXY6 (16 bits)		
BIT	15	. . .	0
NAME	General Purpose Register 1		
OPER	R/W		
RESET	0 PS		

This register is a general purpose register that allows a local bus master to communicate with a VMEbus master. The function of this register is not defined by the hardware specification.

General Purpose Register 2

ADR/SIZ	Local Bus: $FFF40110/VMEbus: $XXY8 (16 bits)		
BIT	15	. . .	0
NAME	General Purpose Register 2		
OPER	R/W		
RESET	0 PS		

This register is a general purpose register that allows a local bus master to communicate with a VMEbus master. The function of this register is not defined by the hardware specification.

General Purpose Register 3

ADR/SIZ	Local Bus: $FFF40114/VMEbus: $XXYA (16 bits)		
BIT	15	. . .	0
NAME	General Purpose Register 3		
OPER	R/W		
RESET	0 PS		

This register is a general purpose register that allows a local bus master to communicate with a VMEbus master. The function of this register is not defined by the hardware specification.

General Purpose Register 4

ADR/SIZ	Local Bus: $FFF40118/VMEbus: $XXYC (16 bits)		
BIT	15	. . .	0
NAME	General Purpose Register 4		
OPER	R/W		
RESET	0 PS		

This register is a general purpose register that allows a local bus master to communicate with a VMEbus master. The function of this register is not defined by the hardware specification.

General Purpose Register 5

ADR/SIZ	Local Bus: $FFF4011C/VMEbus: $XXYE (16 bits)		
BIT	15	. . .	0
NAME	General Purpose Register 5		
OPER	R/W		
RESET	0 PS		

This register is a general purpose register that allows a local bus master to communicate with a VMEbus master. The function of this register is not defined by the hardware specification.

MC2 Chip 3

Introduction

The Memory Controller ASIC (MC2 chip) is one of three ASICs that are part of the MVME162FX hardware set. The MC2 chip is designed to operate synchronously with the MC68040 local bus clock at 25MHz or 32MHz.

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.
- DRAM controller supporting several configurations.
- Four Zilog Z85230 SCC interface.
- NCR 53C710 SCSI Coprocessor interface.
- Intel 82596CA LAN Coprocessor interface.
- Four 32-bit tick timers.
- Interrupt support for ABORT switch, LAN, SCSI, SCC, DRAM, and Timers.
- Local bus access timer.
- Watchdog timer.

Functional Description

The following sections provide an overview of the functions provided by the MC2 chip. A detailed programming model for the MC2 chip control and status registers is provided in a later section.

MC2 chip Initialization

The MC2 chip ASIC is designed to accommodate several memory configurations and MVME162FX population versions. A factory-programmed resident device is used to initialize the MVME162FX Version Register, General Purpose Inputs Register, and DRAM/SRAM Options Register (read only). A different initialization device is used for each version of the MVME162FX. Refer to the *Programming Model* on page 3-11.

Flash and EPROM Interface

The MC2 chip interfaces the MC68040 local bus to one 1M x 8 Intel 28F008SA and a 32-pin PLCC JEDEC standard EPROM. The Flash and EPROM memory map locations can be swapped based upon a jumper (J22, pins 9 and 10, GPIO3) input to the initialization PAL. (The initialization device was discussed in the previous section.) This enables the MVME162FX to execute reset code from either the EPROM or Flash.

Note that MVME162FX 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 J22 is configured for the EPROM memory map. Refer to Chapters 3 and 4 of the *MVME162FX Installation and Use* documentation, V162FXA, for further details.

The MC2 chip executes multiple cycles to the eight-bit Flash/EPROM devices so that byte, word, or longword accesses are allowed. Burst accesses to Flash/EPROM are inhibited by the interface so that they are broken into four longword accesses.

The 28F008SA has a ready/busy pin to interrupt the processor when certain commands have completed. The MC2 chip does not utilize this feature. Software has to poll the status register to determine device availability.

The MC2 chip ASIC supports write cycles to EPROM memory space with a normal cycle termination by asserting transfer acknowledge. Data is not changed. The MC2 chip allows the write cycle to time out.

The Flash memory has a write-protect feature. A CSR bit in the Flash Parameter Register (FWEN, bit 11) inhibits write cycles to Flash.

BBRAM Interface

The MC2 chip 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 BBRAM 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 MC2 chip.

82596CA LAN Interface

The LAN controller interface is described in the following sections.

MPU Port and MPU Channel Attention

The MC2 chip 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 MC2 chip only supports (decodes) MPU Port writes. It does not decode MPU Port reads. (Nor does the 82596CA support MPU Port reads.)

MPU Channel Attention access is used to cause the 82596CA to begin executing memory resident Command blocks. To execute an MPU Channel Attention, the 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 MC2 chip 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 MC2 chip is updated and a LANC bus error interrupt is generated if it is enabled in the MC2 chip. 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 MC2 chip provides an interrupt control register for normal LANC termination and another register for bus error termination of LANC operation. The MC2 chip requests an interrupt at the level programmed in the LANC interrupt control registers if the interrupt is enabled and a positive edge is detected on the 82596CA INT* pin or if the LANC bus error condition is detected.

53C710 SCSI Controller Interface

The MC2 chip provides a map decoder and an interrupt handler for the NCR 53C710 SCSI I/O Processor. The base address for the 53C710 is $FFF47000. The MC2 chip requests an interrupt at the level programmed in the SCSI interrupt control register if the interrupt is enabled and a low level is detected on the 53C710 IRQ* pin.

SRAM Memory Controller

The SRAM base address and size are programmable. The SRAM controller is designed to operate with 100 ns devices. The size of the SRAM is initialized in the DRAM/SRAM Options Register when the MVME162FX is reset. SRAM performance at 25MHz is 5,3,3,3 for read and write cycle. SRAM performance at 32MHz is 6,4,4,4 for read cycles and 6,3,3,3 for write cycles.

DRAM Memory Controller

The DRAM base address, DRAM array size, and DRAM device size are programmable. The DRAM controller assumes an interleaved architecture if the DRAM size requires eight physical devices. (That is, there are two memory configurations which allow interleaved DRAM architecture: when memory array size is 4MB and DRAM technology is 4-Mbits per device; and when memory array size is 16MB and DRAM technology is 16-Mbits per device.)

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

Note The present MVME162FX models do not contain parity DRAM.

Table 3-1. DRAM Performance

Clock Budget	Operating Conditions
4,2,2,2	Non-interleaved, read, 25MHz, without TEA on parity error
4,1,1,1	Interleaved, read, 25MHz, without TEA on parity error
5,3,3,3	Non-interleaved, read, 25MHz, with TEA on parity error
5,2,2,2	Interleaved, read, 25MHz, with TEA on parity error
3,2,2,2	Write, 25MHz
5,3,3,3	Non-interleaved, read, 32MHz, without TEA on parity error

Table 3-1. DRAM Performance

Clock Budget	Operating Conditions
5,2,2,2	Interleaved, read, 32MHz, without TEA on parity error
6,4,4,4	Non-interleaved, read, 32MHz, with TEA on parity error
6,3,3,3	Interleaved, read, 32MHz, with TEA on parity error
4,2,2,2	Write, 32MHz

Note TEA is the MC68040 bus error transaction signal. "With TEA" indicates that a bus error cycle occurs if a DRAM parity error was detected.

Z85230 SCC Interface

The MC2 chip provides a map decoder and an interrupt handler for the Zilog Z85230. The base address is $FFF45000. The MC2 chip 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.

The MC2 chip supports as many as four Z85230 devices. (There is only one Z85230 on the MVME162FX. Refer to the *Board Level Hardware Description* in the *MVME162FX Embedded Controller Installation and Use* manual.) The addresses for the 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
$FFF45400 - $FFF457FF	1
$FFF45800 - $FFF45BFF	2
$FFF45C00 - $FFF45FFF	3

Tick Timers

The MC2 chip implements four 32-bit tick timers. These timers are identical to the timers in the VMEchip2. The timers run on a 1MHz clock which is derived from the processor clock by a prescaler.

Each timer has a 32-bit counter, a 32-bit compare register, and a clear-on-compare enable bit. The counter is readable and writable at any time. 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 µs 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 µs 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 MC2 chip. The watchdog timer implemented in the MC2 chip 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 MC2 chip timer cannot generate a system reset because it is not connected to the VMEbus.

Local Bus Timer

The MVME162FX provides a time-out function for the local bus. When the timer is enabled and a local bus access times out, a Transfer Error Acknowledge (TEA) signal is sent to the local bus master. The time-out value is selectable by software for 8 µsec, 64 µsec, 256 µsec, 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 Example of the Proper Use of Bus Timers in Chapter 1 for more information.

The access timer logic is duplicated in the VMEchip2 and MC2 chip ASIC. Because the local bus timer in the VMEchip2 can detect an off-board access and the MC2 chip 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 MC2 chip Registers

The register map and address of the memory controller ASIC (MC2 chip) 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. MC2 chip Register Map

MC2 chip Base Address = $FFF42000

Offset	D31-D24	D23-D16	D15-D8	D7-D0
$00	MC2 chip ID	MC2 chip Revision	General Control	Interrupt Vector Base Register
$04	Tick Timer 1 Compare Register			
$08	Tick Timer 1 Counter Register			
$0C	Tick Timer 2 Compare Register			
$10	Tick Timer 2 Counter Register			
$14	LSB Prescaler Count Register	Prescaler Clock Adjust	Tick Timer 2 Control	Tick Timer 1 Control
$18	Tick Timer 4 Interrupt Control	Tick Timer 3 Interrupt Control	Tick Timer 2 Interrupt Control	Tick Timer 1 Interrupt Control
$1C	DRAM Parity Error Interrupt Control	SCC Interrupt Control	Tick Timer 4 Control	Tick Timer 3 Control
$20	DRAM Space Base Address Register		SRAM Space Base Address Register	
$24	DRAM Space Size	DRAM/SRAM Options	SRAM Space Size	(reserved)
$28	LANC Error Status	(reserved)	LANC Interrupt Control	LANC Bus Error Interrupt Control
$2C	SCSI Error Status	General Purpose Inputs	MVME162FX Version	SCSI Interrupt Control
$30	Tick Timer 3 Compare Register			
$34	Tick Timer 3 Counter Register			
$38	Tick Timer 4 Compare Register			
$3C	Tick Timer 4 Counter Register			
$40	Bus Clock	EPROM Access Time Control	Flash Parameter Control	ABORT Switch Interrupt Control
$44	RESET Switch Control	Watchdog Timer Control	Access & Watchdog Time Base Select	(reserved)
48	DRAM Control	(reserved)	MPU Status	(reserved)
$4C	32-bit Prescaler Count Register			

Programming Model

This section defines the programming model for the control and status registers (CSR) in the MC2 chip. 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 powerup reset are as defined below.

P The bit is affected by powerup reset.
L The bit is affected by local reset.
X The bit is not affected by reset.
0 The bit is always 0.
1 The bit is always 1.

MC2 chip ID Register

ADR/SIZ	$FFF42000 (8 bits)							
BIT	31	30	29	28	27	26	25	24
NAME	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
OPER	R	R	R	R	R	R	R	R
RESET	1 PL	0 PL	0 PL	0 PL	0 PL	1 PL	0 PL	0 PL

ID7-ID0 The chip ID number is $84. This register is read only. It ignores a write but ends the cycle with TA*, i.e., the cycle terminates without exceptions.

MC2 chip 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	1 PL

RV7-RV0 The current value of the chip revision is $01. This register is read only. It ignores a write but ends the cycle with TA*, i.e., the cycle terminates without exceptions.

General Control Register

ADR/SIZ	$FFF42000 (8 bits)							
BIT	15	14	13	12	11	10	9	8
NAME				SCCIT1	SCCIT0	PPC	MIEN	FAST
OPER	R	R	R	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 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.

MIEN Master Interrupt Enable. When this bit is high, interrupts from and via the MC2 chip are allowed to reach the MPU. When it is low, all interrupts from the MC2 chip are disabled. Also, when the bit is low, all interrupt acknowledge cycles to the MC2 chip are passed on, via the IACKOUT* pin. This bit is cleared by a reset.

PPC PowerPC interrupt mode. When this bit is high, the IPL<2> signal output is compatible with the int signal on the PowerPC. When this bit is low, the IPL signal outputs are compatible with the MC68040.

Caution

This bit is low for the MVME162FX boards. Do not change it. If it is changed, the board will not operate properly.

SCCIT<1:0> These bits define the IACK daisy chain time for the SCC chips. They must be set based on the number of SCC devices.

SCCIT<1:0>	Number of Z85230s
00	1
01	2
10	3
11	4

Caution

These bits are low for the MVME162FX boards because they contain only one Z85230 chip. Do not change them. If they are changed, the board will not operate properly.

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 MC68*xx*040 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	$FFF42000 (8 bits)							
BIT	7	6	5	4	3	2	1	0
NAME	IV7	IV6	IV5	IV4	IV3	IV2	IV1	IV0
OPER	R/W	R/W	R/W	R/W	R	R	R	R
RESET	0 PL	0 PL	0 PL	0 PL	1 PL	1 PL	1 PL	1 PL

The encoding for the interrupt sources is shown in the next table, where IV3-IV0 refer to bits 3-0 of the vector passed during the IACK cycle:

The priority referenced in the following table is established in the MC2 chip 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 MC2 chip is wired to have the highest priority followed by the IndustryPack interface ASIC (IP2 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 (Z85230) 0-3	Note 1	Next Highest
ABORT Switch	$E	Highest
Unused	$F	. . .

Note The Z85230 controller has an integrated interrupt vector register which is separate from the vector generation found on the MC2 chip. The Z85230 also supports a scheme where the base register value is changed based upon the interrupt requested.

Programming the Tick Timers

There are four programmable tick timers in the MC2 chip. These timers are identical in function to the timers implemented in the PCCchip2 and the VMEchip2.

Tick Timer 1 and 2 Compare and Counter Registers

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

$$T\ (\mu s) = \textit{Compare Register}$$

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	$FFF42004 (32 bits)		
BIT	31	. . .	0
NAME	Tick timer 1 Compare Register		
OPER	R/W		
RESET	0 P		

Tick Timer 1 Counter

ADR/SIZ	$FFF42008 (32 bits)		
BIT	31	. . .	0
NAME	Tick timer 1 Counter		
OPER	R/W		
RESET	X		

Tick Timer 2 Compare Register

ADR/SIZ	$FFF4200C (32 bits)		
BIT	31	. . .	0
NAME	Tick timer 2 Compare Register		
OPER	R/W		
RESET	0 P		

Tick Timer 2 Counter

ADR/SIZ	$FFF42010 (32 bits)		
BIT	31	. . .	0
NAME	Tick timer 2 Counter		
OPER	R/W		
RESET	0 P		

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	$FFF42014 (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:

Prescaler Clock Adjust Register = 256 - *processor clock* (MHz)

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

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:

Tick clock = *processor clock* / (256 - *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	$FFF42014 (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	$FFF42018 (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	$FFF42018 (8 bits)							
BIT	7	6	5	4	3	2	1	0
NAME			INT	IEN	ICLR	IL2	IL1	IL0
OPER	R	R	R	R/W	C	R/W	R/W	R/W
RESET	0	0	0 PL	0 PL	0 PL	0 PL	0 PL	0 PL

IL2-IL0 These three bits select the interrupt level for the tick timers. Level 0 does not generate an interrupt.

ICLR Writing a logic 1 to this bit clears the tick timer interrupt (i.e., INT bit in this register). This bit is always read as zero.

IEN When this bit is set high, the interrupt is enabled. The interrupt is disabled when this bit is low.

INT When this bit is high, a tick timer interrupt is being generated at the level programmed in IL2-IL0 (if nonzero). This bit is edge-sensitive and can be cleared by writing a logic 1 into the ICLR control bit.

DRAM Parity Error Interrupt Control Register

The DRAM Parity Error Interrupt Control Register controls the interrupt logic for parity error interrupts. In the MVME162FX, the parity control and interrupt logic is contained in the DRAM Parity Error Interrupt Control Register and the DRAM Control Register located at $FFF4201C and $FFF42048 respectively.

ADR/SIZ	$FFF4201C (8 bits)							
BIT	31	30	29	28	27	26	25	24
NAME			INT	IEN	ICLR	IL2	IL1	IL0
OPER	R	R	R	R/W	C	R/W	R/W	R/W
RESET	0	0	0 PL	0 PL	0 PL	0 PL	0 PL	0 PL

IL2-IL0 These three bits select the interrupt level for the DRAM parity error detection. Level 0 does not generate an interrupt.

ICLR Writing a logic 1 to this bit clears the DRAM parity error detection interrupt. This clears the INT bit in this register. This bit is always read as zero.

Note Do not set the IEN bit unless parity is supported by the DRAM mezzanine. The present MVME162FX models do not contain parity DRAM.

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.

INT When this bit is high, a interrupt is being generated due to a DRAM parity error. The interrupt is at the level programmed in IL2-IL0.

SCC Interrupt Control Register

ADR/SIZ	$FFF4201C (8 bits)							
BIT	23	22	21	20	19	18	17	16
NAME			INT	IEN		IL2	IL1	IL0
OPER	R	R	R	R/W	R	R/W	R/W	R/W
RESET	0	0	0 PL	0 PL	0	0 PL	0 PL	0 PL

IL2-IL0 These three bits select the interrupt level for the SCC controller. Level 0 does not generate an interrupt.

IEN When this bit is set high, the interrupt is enabled. The interrupt is disabled when this bit is low.

INT This bit reflects the state of the INT pin from the Z85230 controller (qualified by the IEN bit). When this bit is high, an SCC controller interrupt is being generated at the level programmed in IL2-IL0. When the interrupt is cleared in the Z85230, INT returns to zero.

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.

DRAM and SRAM Memory Controller Registers

The DRAM decode logic consists of a base register, a size register, and an options register. The SRAM decode logic consists of a similar set of registers.

The reset logic initializes the DRAM and SRAM Base registers so that DRAM space starts at address 0 and SRAM space starts at $FFE00000. DRAM and SRAM are inhibited by reset. Software can examine the MVME162FX DRAM/SRAM Options Register at address $FFF42024 bits 20-16 to determine the size of the SRAM and DRAM.

DRAM Space Base Address Register

ADR/SIZ	$FFF42020 16 bits)					
BIT	31	...	20	19	...	16
NAME	B31-B20					
OPER	R/W			R		
RESET	0 PL			0		

B31-B20 B31 - B20 are compared to local bus address signals A31 - A20 for memory reference cycles. If they compare, a DRAM cycle is initiated. Note that there is linkage between the Base Address Register and its associated Size Register. The Size Register masks the least significant address signals for the comparison. Therefore, the Base Address Register contents must be set to a multiple of the Size Register. For example, if the size is set for 4096KB, the Base Register must be set to 0, or 4096KB, or 8192KB, or 12288KB, etc.

SRAM Space Base Address Register

ADR/SIZ	$FFF42020 16 bits)	
BIT	15-1	0
NAME	B31-B17	
OPER	R/W	R
RESET	$FFE0 PL	

B31-B17 B31 - B17 are compared to local bus address signals A31 - A17 for memory reference cycles. If they compare, an SRAM cycle is initiated. Note that the same linkage that exists between the DRAM Base and Size Registers also exists for the SRAM decode logic. Refer to *DRAM Space Base Address Register* above.

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 The size bits configure the DRAM decoder for a particular memory size. The following table defines their encoding. Note that the table specifies the allowed bit combinations for DZ2 - DZ0. Any other combinations generate unpredictable results.

DZ2 - DZ0 are set equal to the DZ2 - DZ0 bits of the DRAM/SRAM Options Register. Note that changing DZ2-DZ0 so that the DRAM architecture changes between interleaved and non-interleaved relocates the data. DZ2 - DZ0 are programmable to facilitate diagnostic software.

Table 3-4. DRAM Size Control Bit Encoding

DZ2 - DZ0	Memory Size
$0	1MB with 4-Mbit DRAMs
$1	2MB with 4-Mbit DRAMs (Memory mezzanine is not available for this setting)
$3	4MB with 4-Mbit DRAMs (interleaved)
$4	4MB with 16-Mbit DRAMs (Memory mezzanine is not available for this setting)
$5	8MB with 16-Mbit DRAMs
$6	DRAM mezzanine is not present.
$7	16MB with 16-Mbit DRAMs (interleaved) (Memory mezzanine is not available for this setting)

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	BEQ1	F1	F0	SZ1	SZ0	DZ2	DZ1	DZ0
OPER	R	R	R	R	R	R	R	R
RESET	Application Specific							

DZ2-DZ0 DZ*x* bits indicate the size and architecture of the DRAM array. Software must initialize the DRAM Space Size Register ($FFF42024 bits 26 - 24) based on the value of DZ2 - DZ0. DZ2 -DZ0 are initialized at reset to a value which is determined by the contents of a factory-programmed resident device.

Table 3-5. DRAM Size Control Bit Encoding

DZ2 - DZ0	DRAM Configuration
$0	1MB with 4-Mbit DRAMs
$1	2MB with 4-Mbit DRAMs (Memory mezzanine is not available for this setting)
$3	4MB with 4-Mbit DRAMs (interleaved)
$4	4MB with 16-Mbit DRAMs (Memory mezzanine is not available for this setting)
$5	8MB with 16-Mbit DRAMs
$6	DRAM mezzanine is not present
$7	16MB with 16-Mbit DRAMs (interleaved) (Memory mezzanine is not available for this setting)

SZ1-SZ0 SZ*x* bits indicate the size of the SRAM array. Software must initialize the SRAM Space Size Register ($FFF42024 bits 9 - 8) based on the value of SZ1-SZ0. SZ1-SZ0 are initialized at reset to a value which is determine by the contents of a factory-programmed resident device.

Table 3-6. SRAM Size Control Bit Encoding

SZ1 - SZ0	SRAM Configuration
$0	Reserved
$1	512KB
$2	1MB
$3	2MB

F0 F0 is a status bit indicating the Flash population option of the MVME162FX. 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.

F1 F1 bit controls the internal data path of the MC2 chip ASIC. If it is set to a 0, the Flash data path is eight bits and passes through the ASIC. If it is set to a 1, the path is 32 bits and is external to the ASIC.

BEQ1 BEQ1 set to a 0 indicates that the base board population of the MVME162FX is 4 MB. When it is set to a 1, the population option is 1 MB. This is a read-only bit.

SRAM Space Size Register

ADR/SIZ	$FFF42024 (8 bits)							
BIT	15	14	13	12	11	10	9	8
NAME						SEN	SZ1	SZ0
OPER	R	R	R	R/W	R/W	R/W	R/W	R/W
RESET	0 PL							1 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 DRAM/SRAM Options Register. SZ1 - SZ0 are programmable to facilitate diagnostic software.

Table 3-7. SRAM Size Control Bit Encoding

SZ1 - SZ0	Memory Size
$0	Reserved
$1	512KB
$2	1MB
$3	2MB

LANC Error Status Register

ADR/SIZ	$FFF42028 (8 bits)							
BIT	31	30	29	28	27	26	25	24
NAME					PRTY	EXT	LTO	SCLR
OPER	R	R	R	R	R	R	R	C
RESET	0	0	0	0	0 PL	0 PL	0 PL	0 PL

SCLR Writing a 1 to this bit clears bits LTO,EXT, and PRTY. Reading this bit always yields 0.

LTO,EXT, PRTY These bits indicate the status of the last local bus error condition encountered by the LANC while performing DMA accesses to the local bus. A local bus error condition is flagged by the assertion of TEA*. When the LANC receives TEA* if the source of the error is local time-out, then LTO is set and EXT and PRTY are cleared. If the source of the TEA* is due to an error in going to the VMEbus, then EXT is set and the other two status bits are cleared. If the source of the error is DRAM parity check error, then PRTY is set and the other two status bits are cleared. If the source of the error is none of the above conditions, then all three bits are cleared. Writing a 1 to bit 24 (SCLR) also clears all three bits.

82596CA LANC Interrupt Control Register

ADR/SIZ	$FFF42028 (8 bits)							
BIT	15	14	13	12	11	10	9	8
NAME	PLTY	E/L*	INT	IEN	IICLR	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 MC2 chip 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/W	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 SCSI processor while performing DMA accesses to the local bus. A local bus error condition is flagged by the assertion of TEA*. When the SCSI processor receives TEA*, if the source of the error is local time-out, then LTO is set and EXT and PRTY are cleared. If the source of the TEA* is due to an error in going to the VMEbus, then EXT is set and the other two status bits are cleared. If the source of the error is DRAM parity check error, then PRTY is set and the other two status bits are cleared. If the source of the error is none of the above conditions, then all three bits are cleared. Writing a 1 to bit 24 (SCLR) also clears all three bits.

General Purpose Inputs Register

The contents of a PAL and the state of an 8-position jumper block are translated to bit settings of the General Purpose Inputs Register, Version Register and DRAM/SRAM Options Register when the MC2 chip 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 MVME162FX board. Refer to the *MVME162FX Embedded Controller Installation and Use* manual for jumper pin definitions. 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 J22-12/J22-11 through J22-16/J22-15 (for GPI2, GPI1, and GPI0), respectively.

V11 Refer to Table 1-2. *Local Bus Memory Map*, Notes 5 and 6. The jumper for V11 is located at J22-10/J22-9 (for GPI3). Refer to Chapter 1 for jumper pin definitions.

Caution

Do not remove the jumper from J22, pins 9 and 10.
Doing so will cause the board to become inoperable.

V15-V12 V15 - V12 are general purpose inputs. Refer to the description for V10 - V8. The jumpers for V15 - V12 are located at J22-2/J22-1 through J22-8/J22-7 (for GPI7, GPI6, GPI5, and GPI4), respectively.

MVME162FX 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 DRAM/SRAM Options Register when the MC2 chip 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.

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 is set to a zero, the MC2 chip 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 is present. V4 set to a zero indicates that a MC68LC040 is present.

V5 This bit is not functional on the MC2 chip.

V6 Reserved for internal use only. (V6 is set to a 0 indicating that the IP2 chip #2 is not present.)

V7 Reserved for internal use only. (V7 is set to a 1 indicating that the IP2 chip #1 is present.)

SCSI Interrupt Control Register

ADR/SIZ	$FFF4202C (8 bits)							
BIT	7	6	5	4	3	2	1	0
NAME			INT	IEN		IL2	IL1	IL0
OPER	R	R	R	R/W	R	R/W	R/W	R/W
RESET	0	0	R	0 PL	0	0 PL	0 PL	0 PL

IL2-IL0 Interrupt Level. These three bits select the interrupt level for the SCSI processor. Level 0 does not generate an interrupt.

IEN Interrupt Enable. When this bit is set high, the interrupt is enabled. The interrupt is disabled when this bit is low.

INT Interrupt Status. This status bit reflects the state of the INT pin from the SCSI processor (qualified by the IEN bit). When this bit is high, a SCSI processor interrupt is being generated at the level programmed in IL2-IL0. This status bit does not need to be cleared, because it is level sensitive.

Tick Timer 3 and 4 Compare and Counter Registers

Tick timers 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	$FFF42030 (32 bits)		
BIT	31	. . .	0
NAME	Tick Timer 3 Compare Register		
OPER	R/W		
RESET	0 P		

Tick Timer 3 Counter

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

Tick Timer 4 Compare Register

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

Tick Timer 4 Counter

ADR/SIZ	$FFF4203C (32 bits)		
BIT	31	. . .	0
NAME	Tick Timer 4 Counter		
OPER	R/W		
RESET	X		

Bus Clock Register

The Bus Clock Register should be programmed with the hexadecimal value of the operating clock frequency in MHz (i.e., $20 for 32 MHz). The MC2 chip uses the value programmed in this register to control the refresh timer so that the DRAMs are refreshed every 15.6 microseconds. After powerup, 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:

*Refresh Rate = BCK/BUS CLOCK * 16*

where BCK is the value programmed in the Bus Clock Register, and BUS CLOCK is the MC68*xx*040 bus clock frequency.

EPROM Access Time Control Register

The MVME162FX 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 <= N at 25 MHz where N =	EPROM ACCESS <= 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

ROM0 Refer to Table 1-2. *Local Bus Memory Map*, Note 1, in Chapter 1.

Flash Parameter Register

The MVME162FX is populated with a 120ns Flash memory device. Due to the wide range of Flash speeds, the contents can be changed by software to adjust for a specific speed.

ADR/SIZ	$FFF42040 (8 bits)							
BIT	15	14	13	12	11	10	9	8
NAME					FWEN	FT2	FT1	FT0
OPER	R	R	R	R	R	R/W	R/W	R/W
RESET	0	0	0	0	0	1 PL	1 PL	1 PL

FWEN Flash write enable function is internal to the ASIC for the MC2 chip. FWEN set to a 1 enables writes to the Flash memory space. FWEN set to a 0 inhibits writes to the Flash memory but the cycle completes without exception.

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 <= N at 25 MHz where N =	Flash Access <= 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

ABORT Switch Interrupt Control Register

The following table describes the ABORT switch interrupt logic in the MC2 chip.

ADR/SIZ	$FFF42040 (8 bits)							
BIT	7	6	5	4	3	2	1	0
NAME		ABS	INT	IEN	ICLR	IL2	IL1	IL0
OPER	R	R	R	R/W	C	R/W	R/W	R/W
RESET	0	0 PL	0 PL	0 PL	0 PL	0 PL	0 PL	0 PL

IL2-IL0 These three bits select the interrupt level for the ABORT switch. Level 0 does not generate an interrupt.

ICLR Writing a logic 1 to this bit clears the abort interrupt (i.e., the INT bit in this register). This bit is always read as 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 MVME162FX 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 position in the VMEchip2 at location $FFF40060. When this bit is high, or the duplicate bit in the VMEchip2 is high, the RESET switch is enabled. When both bits are low, the RESET switch is disabled.

BDFLO When this bit is high, the MC2 chip 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 MVME162FX.

CPURS When this bit is set high, the powerup reset status bit is cleared. This bit is always read zero.

PURS This bit is set by a powerup reset. It is cleared by a write to the CPURS bit.

BRFLI When this status bit is high, the BRDFAIL signal pin on the MC2 chip is asserted. When this status bit is low, the BRDFAIL signal pin on the MC2 chip is not asserted. The BRDFAIL pin may be asserted by an external device, the BDFLO bit in this register, or a watchdog time-out.

Watchdog Timer Control Register

The watchdog timer control logic in the MC2 chip 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 MC2 chip.

ADR/SIZ	$FFF42044 (8 bits)							
BIT	23	22	21	20	19	18	17	16
NAME		WDCS	WDCC	WDTO	WDBFE		WDRSE	WDEN
OPER	R	C	C	R	R/W	R	R/W	R/W
RESET	0	0 P	0 P	0 P	0 PL	0	0 PL	0 PL

WDEN When this bit is high, the watchdog timer is enabled. When this bit is low, the watchdog timer is not enabled.

WDRSE When this bit is high, and a watchdog time-out occurs, a LRESET is generated. When this bit is low, a watchdog time-out does not cause a reset.

WDBFE When this bit is high and the watchdog timer has timed out, the MC2 chip asserts the BRDFAIL signal pin. When this bit is low, the watchdog timer does not contribute to the BRDFAIL signal on the MC2 chip.

WDTO When this status bit is high, a watchdog time-out has occurred.
When this status bit is low, a watchdog time-out has not occurred. This bit is cleared by writing a 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 time-out period or a watchdog time-out occurs.

WDCS When this bit is set high, the watchdog time-out status bit (WDTO bit in this register) is cleared.

Access and Watchdog Time Base Select Register

The watchdog timer control logic in the MC2 chip 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 MC2 chip.

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		0 PL		0 PL			

WDTO These bits define the watchdog time-out period:

Bit Encoding	Time-out	Bit Encoding	Time-out
0	512 µs	8	128 ms
1	1 ms	9	256 ms
2	2 ms	10	512 ms
3	4 ms	11	1 s
4	8 ms	12	4 s
5	16 ms	13	16 s
6	32 ms	14	32 s
7	64 ms	15	64 s

LBTO These bits define the local bus time-out value. The timer begins timing when TS is asserted on the local bus. If TA or TAE is not asserted before the timer times out, a TEA signal is sent to the local bus. Note that the Version Register bit V1 must be set to a 1 to enable the MC2 chip access timer (i.e., it must be a "No VMEbus Interface" option).

0	8 µs
1	64 µs
2	256 µs
3	The timer is disabled.

DRAM Control Register

This register controls the parity checking mode and DRAM enable.

Note Do not enable parity unless it is supported by the DRAM mezzanine. The present MVME162FX models do not contain parity DRAM.

ADR/SIZ	$FFF42048 (8 bits)							
BIT	31	30	29	28	27	26	25	24
NAME					WWP	PARINT	PAREN	RAMEN
OPER	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RESET	0	0	0	0	0 PL	0 PL	0 PL	0 PL

RAMEN This bit enables the access of the DRAM. The DRAM should be enabled after the DRAM Space Base Address Register is enabled and the ROM0 bit has been cleared. The DRAM Space Base Address Register is located at $FFF42020 bits 31-16 and the ROM0 bit is located at $FFF42040 bit 20.

PAREN-PARINT

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

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 DRAM accesses by one clock tick.

WWP 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. Note that the MPU Status Register in the MC2 chip is functional independent of the “No VMEbus Interface” option.

ADR/SIZ	$FFF42048 (8 bits)							
BIT	15	14	13	12	11	10	9	8
NAME					MCLR	MLBE	MLPE	MLTO
OPER	R	R	R	R	C	R	R	R
RESET	0	0	0	0	0 PL	0 PL	0 PL	0 PL

MLTO When this bit is set, the MPU received a TEA and the status indicated a local bus time-out. This bit is cleared by 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 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. This bit is used with the “No VMEbus Interface” option and is duplicated in the VMEchip2 at address $FFF40048 bit 9. However, the MVME162FX does not have parity and this bit is not implemented.

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 MC2 chip. 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 3-18. 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	$FFF4204C (32 bits)	
BIT	31 . . . 8	7-0
NAME	MSB	LSB
OPER	R/W	R
RESET	0 P	

LSB7-0 The least significant bits of the 32-bit prescaler. These bits are read only. They are duplicated in the memory map in the MC2 chip 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 MVME162FX with the VMEchip2 as well as an MVME162FX without the VMEchip2. This feature is provided for those applications which require a Prescaler Count Register to run on all MVME162FX versions.

IP2 Chip 4

Introduction

This chapter describes the IndustryPack Interface Controller (IP2 chip) ASIC for the MC68040 bus. The IP2 chip is designed for the MVME162FX board and interfaces to up to four IndustryPacks (IPs).

Summary of Major Features

- Provides all logic required to interface 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.
- Supports four DMA channels, one per IndustryPack interface, or two channels on IP_a and IP_c.
- Supports a pacer clock for strobe generation to the IndustryPack interface.
- 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.
- Thirteen Interrupt Handler Control Registers, two for each IndustryPack, one per DMA controller (DMAC) and pacer clock.

- 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 IP2 chip. A detailed programming model for the IP2 chip control and status registers is provided in a later section of this chapter.

General Description

The IP2 chip converts IP-bound MC68040 read/write/interrupt acknowledge cycles to IndustryPack cycles. Control registers within the IP2 chip control the assumed width of the IndustryPack that is being accessed. The IP2 chip interfaces to four 16-bit IndustryPack positions. The naming convention for single size IndustryPack population of each of these positions is: IndustryPack-a (IP_a), IndustryPack-b (IP_b), IndustryPack-c (IP_c), and IndustryPack-d (IP_d). The naming convention for double size IndustryPack population of these positions is IndustryPack-a/b (IP_ab) and IndustryPack-c/d (IP_cd). (A double size IndustryPack can occupy positions A and B, or it can occupy positions C and D.)

Cache Coherency

The IP2 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 IP2 chip pair ignores the memory inhibit (MI*) signal line. When SC1, SC0 do not indicate that snooping is inhibited, the IP2 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 IP2 chip assumes that the cycle is over and that it is not to participate.

Local Bus to IndustryPack DMA Controllers

The IP2 supports two basic types of DMA cycles: “standard DMA” (sDMA) and “addressed DMA” (aDMA). sDMA cycles are requested by the IP. When the DMA controller (DMAC) detects a DMA request and if that DMA controller is enabled, it will acknowledge the request by transferring data between the local bus and the I/O space of the requesting IP device. Alternatively, aDMA transfers are not linked to a request/acknowledge protocol. aDMA cycles are initiated by the DMA controller as soon as its control registers have been initialized by the MC68040. It will transfer data between the local bus and a selected IP module memory space. The IP2 chip implements four DMA controllers which can operate in the sDMA or aDMA mode. The DMA controllers can be configured so one is controller attached to each of the four possible IndustryPack interfaces or so that DMA controllers a and b are attached to IP_a and controllers c and d are attached to IP_c. The DMA controllers support 8-, 16-, and 32-bit IndustryPack widths. The four DMA channels can operate concurrently.

Each DMA controller has a 32-bit local address counter, a 32-bit table address counter, a 24-bit byte counter, control registers, status registers, and a 24-bit IP address counter. The data path for each DMA controller passes through a FIFO which is eight locations deep and four bytes wide.

sDMA transfers and byte count parameters must accommodate the I/O port width. If the port width is 16 bits, then the byte count must be initialized to an even value; if the width is 32 bits, then the byte count must be set to a value which is a multiple of four. This implies that transfer to I/O space under DMA control will always be the same size as the port width. The IP address register must be initialized to 0 before sDMA is enabled. This counter is used to align data in the IP2 Chip.

The data has no alignment restrictions as it is moved to or from the memory on the local bus. This would typically be DRAM on the MC68040 local bus, but it could also be memory on the VMEbus.

To optimize local bus use when the IndustryPack is less than 32 bits wide, the FIFO converts 8-bit and 16-bit IP transfers to 32-bit local bus transfers. The FIFO data path logic also aligns data if the source

and destination addresses are not aligned, so the local bus and the IndustryPack can operate at their maximum data transfer sizes. The FIFO also buffers enough data so that accesses to the local bus are in the burst mode.

Each DMAC also supports command chaining through the use of a singly-linked list built in local (not IP) memory. Each entry in the list includes an IP address, a local bus address, a byte count, a control word, and a pointer to the next entry. When the command chaining mode is enabled, the DMAC reads and executes commands from the list in local memory until all commands are executed.

Each DMAC can be programmed to generate an interrupt request when any specific table entry has completed, when the byte count reaches zero, when an error condition occurs, or when the DMAEND* signal is asserted by the IndustryPack.

The DMA arbiter has two modes of operation. One mode is to implement a round robin type of arbitration, which guarantees equal access to the local bus. The other method is to set the arbitration priority to one of four states. In this case, the priority is constant with one DMA channel having the highest priority, and the other three having the second, third, and fourth highest priority.

Note that the IP specification supports a DMA burst where the DMA cycles can be executed back to back. The DMA arbiter logic will not release a DMA channel until a burst of IP cycles are completed, if the burst protocol is observed. However, if the burst protocol is not observed, the arbiter is released and the next DMA request/grant is resolved. This takes two clock cycles due to the asynchronous clocks controlling the MC68040 local bus and the IP busses. The IP designer should take this into consideration if maximum DMA bandwidth availability is required.

Note also that when the DMA register context is updated for the next command packet, a DMA function is used. The state of the snoop control signals for this DMA function is determined by the settings of jumper J26 (as is the state of the snoop control signals for all other DMA cycle types). Refer to the *Hardware Preparation* section of the *Installation and Use* manual, V162FXA/IH.

Clocking Environments and Performance

The IP2 chip has two clock domains. The majority of the logic is controlled by the MC68040 local bus clock which can be 25 MHz or 32 MHz. The IndustryPack interface is controlled by the IndustryPack clock. The IndustryPack clock can be 8 MHz or set equal to the local bus clock. When logic signals cross from one clock domain to another, they must be synchronized to the new clock frequency. The latency time due this synchronization is generally hidden due to the FIFOs in the data path. However, there are two functions where the latency time affects performance. One of them is when a local bus master such at the MC68040 accesses an IndustryPack resource, such as reading back to back memory locations. One to two IP clock cycles of overhead is associated with this function. The other is when arbitration logic must resolve inputs from both clock domains to determine which IndustryPack will be granted DMA service. There are two IP clock cycles of overhead associated with this function. The following table explains the effect of this latency for given clocking environments.

The bandwidth which is specified in the following table is the available bandwidth to the IndustryPack bus. This bandwidth can be split between one, two, three, or four IP modules.

Table 4-1. IP2 chip Clock Cycles

Bus Frequency		Period and Bandwidth to 32-Bit IP Space		
MC68040	**IP**	**Back to Back Examine** (Note 1)	**Four Cycle DMA Burst** (Note 2)	**Single Cycle DMA** (Note 3)
25 MHz	8 MHz	4 IP clocks 8 MB/sec	10 IP clocks 12.8 MB/sec	4 IP clocks 8 MB/sec
32 MHz	8 MHz	3 IP clocks 10.6 MB/sec	10 IP clocks 12.8 MB/sec	4 IP clocks 8 MB/sec
32 MHz	32 MHz (Note 5)	6 IP clocks 21 MB/sec	12 IP clocks 42 MB/sec (Note 4)	6 IP clocks 21 MB/sec

Notes 1. This column is a measure of IndustryPack bandwidth for back to back cycles for a local bus master which is accessing a memory or I/O space location on an IndustryPack. It assumes a zero wait state acknowledge reply from the IndustryPack.

2. This column is a measure of IndustryPack bandwidth for DMA burst cycles between a local bus slave and a memory or I/O space location on an IndustryPack. It assumes a zero wait state acknowledge reply from the IndustryPack.

3. This column is a measure of IndustryPack bandwidth for DMA single cycles between a local bus slave and a memory or I/O space location on an IndustryPack. It assumes a zero wait state acknowledge reply from the IndustryPack.

4. Burst mode sDMA is not supported when both bus frequencies are 32 MHz.

5.: Because the specified band width assumes a zero wait state IndustryPack cycle, it would be difficult to achieve the stated bandwidths for an IP bus frequency of 32 MHz.

Pacer Clock

The IP2 chip implements a general purpose pacer clock output for external connection to the IndustryPacks. This feature complies with the STROBE function defined in the IndustryPack specification. The pacer clock's clock source is the MC68040 bus clock. This clock input is fed through an 8-bit programmable pre-scaling counter whose output is fed to a 16-bit counter. The 16-bit counter increments at rising edges of the output of the pre-scale logic and clears every time it reaches the value programmed into the 16-bit pacer timer register. Depending on its programmed mode, the pacer clock output either pulses or toggles each time the 16-bit counter matches and clears. Additional control bits in the pacer clock control register allow software to stop, start, clear, and

reverse the polarity of the pacer clock output. The pacer clock output's programmable frequency range is from approximately 4 Hz to 16 MHz. The pacer clock logic also includes local bus interrupt control.

Error Reporting

The following paragraphs describe the IP2 chip error reporting.

Error Reporting as a Local Bus Slave

The IP2 chip does not have the ability to assert the TEA* signal as a local bus slave. Because of this, the only bus error cycles that should ever be encountered when accessing to or through the IP2 chip are those that come from an external local bus timer due to no response from an IndustryPack. Note that any external local bus timer should be set for no less than 5 microseconds to allow for normal accesses to the slowest IndustryPack.

Error Reporting as a Local Bus Master

The IP2 chip does not connect to the ST1 and ST0 signal lines. Consequently, when it receives a TEA* termination to any cycle for which it is local bus master, no status will be available to indicate the source of the bus error. There is a status bit in each DMAC status register which indicates that a local bus error did occur as a consequence of a DMA operation. The contents of the local bus address counter can be examined for a valid address if a bus error does occur.

IndustryPack Error Reporting

Each IndustryPack interface has an error pin. The error status from the four interfaces are available in the General Control Registers.

Interrupts

The IP2 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 IP2 chip determines that it is the source of the interrupt being acknowledged, it waits for IACKIN* to be asserted, then it performs an interrupt acknowledge cycle to the appropriate IndustryPack in order to obtain the vector number. It then passes the vector number on to the local bus master and asserts TA* to terminate the cycle.

When the local bus master acknowledges an interrupt, if the IP2 chip determines that it is not the source of the interrupt being acknowledged, it waits for IACKIN* to be asserted, then it passes the acknowledge on down the daisy-chain by asserting IACKOUT*.

The interrupter also provides interrupt capability for the pacer clock and for each of the four DMA controllers. Additionally, interrupts from the pacer clock can be programmed for rising and/or falling edge sensitivity. The vector passed to the local bus master during an interrupt acknowledge for the pacer clock and DMAC interrupts is from the vector base register in the IP2 chip. Part of the vector is programmable; the other part encodes the source of the interrupt.

Overall Memory Map

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

Table 4-2. IP2 chip Overall Memory Map

Address Range	Selected Device	Port Width	Size
Programmable	IP_a/IP_ab Memory Space	D32-D8	64KB-16MB
Programmable	IP_b Memory Space	D16-D8	64KB-8MB
Programmable	IP_c/IP_cd Memory Space	D32-D8	64KB-16MB
Programmable	IP_d Memory Space	D16-D8	64KB-8MB
$FFF58000-$FFF5807F	IP_a I/O Space	D16	128B
$FFF58080-$FFF580BF	IP_a ID Space	D16	64B
$FFF580C0-$FFF580FF	IP_a ID Space Repeated	D16	64B
$FFF58100-$FFF5817F	IP_b I/O Space	D16	128B
$FFF58180-$FFF581BF	IP_b ID Space	D16	64B
$FFF581C0-$FFF581FF	IP_b ID Space Repeated	D16	64B
$FFF58200-$FFF5827F	IP_c I/O Space	D16	128B
$FFF58280-$FFF582BF	IP_c ID Space	D16	64B
$FFF582C0-$FFF582FF	IP_c ID Space Repeated	D16	64B
$FFF58300-$FFF5837F	IP_d I/O Space	D16	128B
$FFF58380-$FFF583BF	IP_d ID Space	D16	64B
$FFF583C0-$FFF583FF	IP_d ID Space Repeated	D16	64B
$FFF58400-$FFF584FF	IP_ab I/O Space	D32-D16	256B
$FFF58500-$FFF585FF	IP_cd I/O Space	D32-D16	256B
$FFF58600-$FFF586FF	IP_ab I/O Space Repeated	D32-D16	256B
$FFF58700-$FFF587FF	IP_cd I/O Space Repeated	D32-D16	256B
$FFFBC000-$FFFBC083	Control/Status Registers	D32-D8	32B

Programming Model

This section defines the programming model for the control and status registers (CSRs) in the IP2 chip. The base address of the CSRs is hardwired to $FFFBC000.

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

R	This bit is a read-only status bit.
R/W	This bit is readable and writable.
R/C	This status bit is cleared by writing a 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 IP2 chip CSR registers is shown in Table 4-3. The CSR registers can be accessed as bytes, words, or longwords. They should not be accessed as lines. They are shown in the table as bytes, and the bits in most of the following register descriptions are labeled as bits 7 through 0.

Table 4-3. IP2 chip Memory Map - Control and Status Registers

IP2 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	1
$02	RESERVED	0	0	0	0	0	0	0	0
$03	VECTOR BASE	IV7	IV6	IV5	IV4	IV3	IV2	IV1	IV0
$04	IP_a MEM BASE UPPER	a_BASE31	a_BASE30	a_BASE29	a_BASE28	a_BASE27	a_BASE26	a_BASE25	a_BASE24
$05	IP_a MEM BASE LOWER	a_BASE23	a_BASE22	a_BASE21	a_BASE20	a_BASE19	a_BASE18	a_BASE17	a_BASE16
$06	IP_b MEM BASE UPPER	b_BASE31	b_BASE30	b_BASE29	b_BASE28	b_BASE27	b_BASE26	b_BASE25	b_BASE24
$07	IP_b MEM BASE LOWER	b_BASE23	b_BASE22	b_BASE21	b_BASE20	b_BASE19	b_BASE18	b_BASE17	b_BASE16
$08	IP_c MEM BASE UPPER	c_BASE31	c_BASE30	c_BASE29	c_BASE28	c_BASE27	c_BASE26	c_BASE25	c_BASE24
$09	IP_c MEM BASE LOWER	c_BASE23	c_BASE22	c_BASE21	c_BASE20	c_BASE19	c_BASE18	c_BASE17	c_BASE16
$0A	IP_d MEM BASE UPPER	d_BASE31	d_BASE30	d_BASE29	d_BASE28	d_BASE27	d_BASE26	d_BASE25	d_BASE24
$0B	IP_d MEM BASE LOWER	d_BASE23	d_BASE22	d_BASE21	d_BASE20	d_BASE19	d_BASE18	d_BASE17	d_BASE16
$0C	IP_a MEM SIZE	a_SIZE23	a_SIZE22	a_SIZE21	a_SIZE20	a_SIZE19	a_SIZE18	a_SIZE17	a_SIZE16
$0D	IP_b MEM SIZE	b_SIZE23	b_SIZE22	b_SIZE21	b_SIZE20	b_SIZE19	b_SIZE18	b_SIZE17	b_SIZE16
$0E	IP_c MEM SIZE	c_cSIZE23	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

4

Table 4-3. IP2 chip Memory Map - Control and Status Registers (Continued)
IP2 chip Base Address = $FFFBC000

Register Offset	Register Name	Register Bit Names							
		D7	D6	D5	D4	D3	D2	D1	D0
$18	IP_a GENERAL CONTROL	a_ERR	0	a_RT1	a_RT0	a_WIDTH1	a_WIDTH0	a_BTD	a_MEN
$19	IP_b GENERAL CONTROL	b_ERR	0	b_RT1	b_RT0	b_WIDTH1	b_WIDTH0	b_BTD	b_MEN
$1A	IP_c GENERAL CONTROL	c_ERR	0	c_RT1	c_RT0	c_WIDTH1	c_WIDTH0	c_BTD	c_MEN
$1B	IP_d GENERAL CONTROL	d_ERR	0	d_RT1	d_RT0	d_WIDTH1	d_WIDTH0	d_BTD	d_MEN
$1C	RESERVED	0	0	0	0	0	0	0	0
$1D	IP CLOCK	0	0	0	0	0	0	0	IP32
$1E	DMA ARBITRATION CONTROL	0	0	0	0	0	ROTAT	PRI1	PRI0
$1F	IP RESET	0	0	0	0	0	0	0	RES

Table 4-3. IP2 chip Memory Map - Control and Status Registers (Continued)
IP2 chip Base Address = $FFFBC000

Register Offset	Register Name	Register Bit Names							
		D7	D6	D5	D4	D3	D2	D1	D0
DMAC for IndustryPack a, request 0. This register set is referred to as DMACa in the text.									
$20	DMA_a STATUS	0	DLBE	0	IPEND	CHANI	TBL	IPTO	DONE
$21	DMA_a INT CTRL	0	0	DINT	DIEN	DICLR	DIL2	DIL1	DIL0
$22	DMA ENABLE	0	0	0	0	0	0	0	DEN
$23	RESERVED	0	0	0	0	0	0	0	0
$24	DMA_a CONTROL 1	DHALT	0	DTBL	ADMA	WIDTH1	WIDTH0	0	XXX
$25	DMA_a CONTROL 2	INTE	0	DMAEI	DMAEO	ENTO	TOIP	0	0
$26	RESERVED	0	0	0	0	0	0	0	0
$27	RESERVED	0	0	0	0	0	0	0	0
$28	DMA_a LB ADDR	LBA31	LBA30	LBA29	LBA28	LBA27	LBA26	LBA25	LBA24
$29	DMA_a LB ADDR	LBA23	LBA22	LBA21	LBA20	LBA19	LBA18	LBA17	LBA16
$2A	DMA_a LB ADDR	LBA15	LBA14	LBA13	LBA12	LBA11	LBA10	LBA9	LBA8
$2B	DMA_a LB ADDR	LBA7	LBA6	LBA5	LBA4	LBA3	LBA2	LBA1	LBA0
$2C	DMA_a IP ADDR	0	0	0	0	0	0	0	0
$2D	DMA_a IP ADDR	IPA23	IPA22	IPA21	IPA20	IPA19	IPA18	IPA17	IPA16
$2E	DMA_a IP ADDR	IPA15	IPA14	IPA13	IPA12	IPA11	IPA10	IPA9	IPA8
$2F	DMA_a IP ADDR	IPA7	IPA6	IPA5	IPA4	IPA3	IPA2	IPA1	IPA0
$30	DMA_a BYTE CNT	0	0	0	0	0	0	0	0
$31	DMA_a BYTE CNT	BCNT23	BCNT22	BCNT21	BCNT20	BCNT19	BCNT18	BCNT17	BCNT16
$32	DMA_a BYTE CNT	BCNT15	BCNT14	BCNT13	BCNT12	BCNT11	BCNT10	BCNT9	BCN8
$33	DMA_a BYTE CNT	BCNT7	BCNT6	BCNT5	BCNT4	BCNT3	BCNT2	BCNT1	BCNT0
$34	DMA_a TBL ADDR	TA31	TA30	TA29	TA28	TA27	TA26	TA25	TA24
$35	DMA_a TBL ADDR	TA23	TA22	TA21	TA20	TA19	TA18	TA17	TA16
$36	DMA_a TBL ADDR	TA15	TA14	TA13	TA12	TA11	TA10	TA9	TA8
$37	DMA_a TBL ADDR	TA7	TA6	TA5	TA4	TA3	TA2	TA1	TA0

Table 4-3. IP2 chip Memory Map - Control and Status Registers (Continued)
IP2 chip Base Address = $FFFBC000

Register Offset	Register Name	Register Bit Names							
		D7	D6	D5	D4	D3	D2	D1	D0
DMAC for IndustryPack b, request 0 or for IndustryPack a, request 1. This register set is referred to as DMACb in the text.									
$38	DMA_b STATUS	0	DLBE	0	IPEND	CHANI	TBL	IPTO	DONE
$39	DMA_b INT CTRL	0	0	DINT	DIEN	DICLR	DIL2	DIL1	DIL0
$3a	DMA ENABLE	0	0	0	0	0	0	0	DEN
$3b	RESERVED	0	0	0	0	0	0	0	0
$3c	DMA_b CONTROL 1	DHALT	0	DTBL	ADMA	WIDTH1	WIDTH0	A_CH1	XXX
$3d	DMA_b CONTROL 2	INTE	0	DMAEI	DMAEO	ENTO	TOIP	0	0
$3e	RESERVED	0	0	0	0	0	0	0	0
$3f	RESERVED	0	0	0	0	0	0	0	0
$40	DMA_b LB ADDR	LBA31	LBA30	LBA29	LBA28	LBA27	LBA26	LBA25	LBA24
$41	DMA_b LB ADDR	LBA23	LBA22	LBA21	LBA20	LBA19	LBA18	LBA17	LBA16
$42	DMA_b LB ADDR	LBA15	LBA14	LBA13	LBA12	LBA11	LBA10	LBA9	LBA8
$43	DMA_b LB ADDR	LBA7	LBA6	LBA5	LBA4	LBA3	LBA2	LBA1	LBA0
$44	DMA_b IP ADDR	0	0	0	0	0	0	0	0
$45	DMA_b IP ADDR	IPA23	IPA22	IPA21	IPA20	IPA19	IPA18	IPA17	IPA16
$46	DMA_b IP ADDR	IPA15	IPA14	IPA13	IPA12	IPA11	IPA10	IPA9	IPA8
$47	DMA_b IP ADDR	IPA7	IPA6	IPA5	IPA4	IPA3	IPA2	IPA1	IPA0
$48	DMA_b BYTE CNT	0	0	0	0	0	0	0	0
$49	DMA_b BYTE CNT	BCNT23	BCNT22	BCNT21	BCNT20	BCNT19	BCNT18	BCNT17	BCNT16
$4a	DMA_b BYTE CNT	BCNT15	BCNT14	BCNT13	BCNT12	BCNT11	BCNT10	BCNT9	BCN8
$4b	DMA_b BYTE CNT	BCNT7	BCNT6	BCNT5	BCNT4	BCNT3	BCNT2	BCNT1	BCNT0
$4c	DMA_b TBL ADDR	TA31	TA30	TA29	TA28	TA27	TA26	TA25	TA24
$4d	DMA_b TBL ADDR	TA23	TA22	TA21	TA20	TA19	TA18	TA17	TA16
$4e	DMA_b TBL ADDR	TA15	TA14	TA13	TA12	TA11	TA10	TA9	TA8
$4f	DMA_b TBL ADDR	TA7	TA6	TA5	TA4	TA3	TA2	TA1	TA0

Table 4-3. IP2 chip Memory Map - Control and Status Registers (Continued)
IP2 chip Base Address = $FFFBC000

Register Offset	Register Name	Register Bit Names							
		D7	D6	D5	D4	D3	D2	D1	D0
DMAC for IndustryPack c, request 0. This register set is referred to as DMACc in the text.									
$50	DMA_c STATUS	0	DLBE	0	IPEND	CHANI	TBL	IPTO	DONE
$51	DMA_c INT CTRL	0	0	DINT	DIEN	DICLR	DIL2	DIL1	DIL0
$52	DMA ENABLE	0	0	0	0	0	0	0	DEN
$53	RESERVED	0	0	0	0	0	0	0	0
$54	DMA_c CONTROL 1	DHALT	0	DTBL	ADMA	WIDTH1	WIDTH0	0	XXX
$55	DMA_c CONTROL 2	INTE	0	DMAEI	DMAEO	ENTO	TOIP	0	0
$56	RESERVED	0	0	0	0	0	0	0	0
$57	RESERVED	0	0	0	0	0	0	0	0
$58	DMA_c LB ADDR	LBA31	LBA30	LBA29	LBA28	LBA27	LBA26	LBA25	LBA24
$59	DMA_c LB ADDR	LBA23	LBA22	LBA21	LBA20	LBA19	LBA18	LBA17	LBA16
$5A	DMA_c LB ADDR	LBA15	LBA14	LBA13	LBA12	LBA11	LBA10	LBA9	LBA8
$5B	DMA_c LB ADDR	LBA7	LBA6	LBA5	LBA4	LBA3	LBA2	LBA1	LBA0
$5C	DMA_c IP ADDR	0	0	0	0	0	0	0	0
$5D	DMA_c IP ADDR	IPA23	IPA22	IPA21	IPA20	IPA19	IPA18	IPA17	IPA16
$5E	DMA_c IP ADDR	IPA15	IPA14	IPA13	IPA12	IPA11	IPA10	IPA9	IPA8
$5F	DMA_c IP ADDR	IPA7	IPA6	IPA5	IPA4	IPA3	IPA2	IPA1	IPA0
60	DMA_c BYTE CNT	0	0	0	0	0	0	0	0
$61	DMA_c BYTE CNT	BCNT23	BCNT22	BCNT21	BCNT20	BCNT19	BCNT18	BCNT17	BCNT16
$62	DMA_c BYTE CNT	BCNT15	BCNT14	BCNT13	BCNT12	BCNT11	BCNT10	BCNT9	BCN8
$63	DMA_c BYTE CNT	BCNT7	BCNT6	BCNT5	BCNT4	BCNT3	BCNT2	BCNT1	BCNT0
$64	DMA_c TBL ADDR	TA31	TA30	TA29	TA28	TA27	TA26	TA25	TA24
$65	DMA_c TBL ADDR	TA23	TA22	TA21	TA20	TA19	TA18	TA17	TA16
$66	DMA_c TBL ADDR	TA15	TA14	TA13	TA12	TA11	TA10	TA9	TA8
$67	DMA_c TBL ADDR	TA7	TA6	TA5	TA4	TA3	TA2	TA1	TA0

4

Table 4-3. IP2 chip Memory Map - Control and Status Registers (Continued)

IP2 chip Base Address = $FFFBC000

Register Offset	Register Name	Register Bit Names							
		D7	D6	D5	D4	D3	D2	D1	D0
DMAC for IndustryPack d, request 0 or for IndustryPack c, request 1, and for PACER CLOCK. This register set, not including the Pacer Clock, is referred to as DMACd in the text.									
$68	DMA_d STATUS	0	DLBE	0	IPEND	CHANI	TBL	IPTO	DONE
$69	DMA_d INT CTRL	0	0	DINT	DIEN	DICLR	DIL2	DIL1	DIL0
$6a	DMA ENABLE	0	0	0	0	0	0	0	DEN
$6b	RESERVED	0	0	0	0	0	0	0	0
$6c	DMA_d CONTROL 1	DHALT	0	DTBL	ADMA	WIDTH1	WIDTH0	C_CH1	XXX
$6d	DMA_d CONTROL 2	INTE	0	DMAEI	DMAEO	ENTO	TOIP	0	0
$6e	RESERVED	0	0	0	0	0	0	0	0
$6f	RESERVED	0	0	0	0	0	0	0	0
$70	DMA_d LB ADDR	LBA31	LBA30	LBA29	LBA28	LBA27	LBA26	LBA25	LBA24
$71	DMA_d LB ADDR	LBA23	LBA22	LBA21	LBA20	LBA19	LBA18	LBA17	LBA16
$72	DMA_d LB ADDR	LBA15	LBA14	LBA13	LBA12	LBA11	LBA10	LBA9	LBA8
$73	DMA_d LB ADDR	LBA7	LBA6	LBA5	LBA4	LBA3	LBA2	LBA1	LBA0
$74	DMA_d IP ADDR	0	0	0	0	0	0	0	0
$75	DMA_d IP ADDR	IPA23	IPA22	IPA21	IPA20	IPA19	IPA18	IPA17	IPA16
$76	DMA_d IP ADDR	IPA15	IPA14	IPA13	IPA12	IPA11	IPA10	IPA9	IPA8
$77	DMA_d IP ADDR	IPA7	IPA6	IPA5	IPA4	IPA3	IPA2	IPA1	IPA0
$78	DMA_d BYTE CNT	0	0	0	0	0	0	0	0
$79	DMA_d BYTE CNT	BCNT23	BCNT22	BCNT21	BCNT20	BCNT19	BCNT18	BCNT17	BCNT16
$7a	DMA_d BYTE CNT	BCNT15	BCNT14	BCNT13	BCNT12	BCNT11	BCNT10	BCNT9	BCN8
$7b	DMA_d BYTE CNT	BCNT7	BCNT6	BCNT5	BCNT4	BCNT3	BCNT2	BCNT1	BCNT0
$7c	DMA_d TBL ADDR	TA31	TA30	TA29	TA28	TA27	TA26	TA25	TA24
$7d	DMA_d TBL ADDR	TA23	TA22	TA21	TA20	TA19	TA18	TA17	TA16
$7e	DMA_d TBL ADDR	TA15	TA14	TA13	TA12	TA11	TA10	TA9	TA8
$7f	DMA_d TBL ADDR	TA7	TA6	TA5	TA4	TA3	TA2	TA1	TA0

Table 4-3. IP2 chip Memory Map - Control and Status Registers (Continued)
IP2 chip Base Address = $FFFBC000

Register Offset	Register Name	Register Bit Names							
		D7	D6	D5	D4	D3	D2	D1	D0
$80	PACER INT CONTROL	0	IRE	INT	IEN	ICLR	IL2	IL1	IL0
$81	PACER GEN CONTROL	PLTY	PLS	0	EN	CLR	PS2	PS1	PS0
$82	PACER TIMER	T15	T14	T13	T12	T11	T10	T9	T8
$83	PACER TIMER	T7	T6	T5	T4	T3	T2	T1	T0

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 IP2 chip terminates them normally with TA*.

ADR/SIZ	$FFFBC000 (8 bits)							
BIT	7	6	5	4	3	2	1	0
NAME	CID7	CID6	CID5	CID4	CID3	CID2	CID1	CID0
OPER	R	R	R	R	R	R	R	R
RESET	0	0	1	0	0	0	1	1

Chip Revision Register

The read-only Chip Revision Register is hard-wired to reflect the revision level of the IP2 chip ASIC. The current value of this register is $01. Writes to this register do nothing, however the IP2 chip terminates them normally with TA*.

Caution

This register reads zero on some IP2 chips. It should read 1. The workaround for this is to test the MC2 chip Revision Register.

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	1

Vector Base Register

ADR/SIZ	$FFFBC003 (8 bits)							
BIT	7	6	5	4	3	2	1	0
NAME	IV7	IV6	IV5	IV4	IV3	IV2	IV1	IV0
OPER	R/W	R/W	R/W	R/W	R/W	R	R	R
RESET	0	0	0	0	1	1	1	1

The interrupt Vector Base Register is an 8-bit read/write register that is used to supply the vector to the CPU during an interrupt acknowledge cycle for the four DMA controller interrupts and for the pacer clock interrupt. Only the most significant five bits are used. The least significant three bits encode the interrupt source during the acknowledge cycle. The exception to this is that after reset occurs, the interrupt vector passed is $07, which remains in effect until a write is generated to the Vector Base Register.

Note Note that this register does not affect the vector supplied during an interrupt acknowledge cycle for any of the eight IndustryPack IRQ*s.

Caution For some versions of the IP2 chip, this register is write only. There is NO known workaround for this error. This register does return the correct value for the interrupt acknowledge cycle.

A normal read access to the Vector Base Register yields the value $0F if the read happens before it has been initialized. A normal read access yields all 0's on bits 0-2, and the value that was last written on bits 3-7, if the read happens after the Vector Base Register was initialized.

The encoding for the interrupt sources is shown below, where IV2-IV0 refer to bits 2-0 of the vector passed during the IACK cycle:

IV2-0	Interrupt Source
0	DMA_a
$1	DMA_b
$2	DMA_c
$3	DMA_d
$4	Pacer Clock

IP_a, IP_b, IP_c, IP_d Memory Base Address Registers

The memory base address registers define the base address at which the IP2 chip initiates memory cycles for their corresponding IndustryPacks. If a 32-bit, double size IndustryPack is used, then the memory base address and memory size registers for IP_a control access for double size ab and those for IP_c control accesses for double size cd.

For each of the four sets of registers, BASE31-BASE16 are compared to MC68040 address signals 31-16 respectively. The IP2 chip 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 Note that 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.

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

As with the memory base address registers, the IP_a size register is also used to control accesses to double size IP_ab and the IP_c size register is used to control accesses to double size IP_cd.

ADR/SIZ	$FFFBC00C through $FFFBC00F (8 bits each)							
BIT	7	6	5	4	3	2	1	0
NAME($0C)	a_SIZE23	a_SIZE22	a_SIZE21	a_SIZE20	a_SIZE19	a_SIZE18	a_SIZE17	a_SIZE16
NAME($0D)	b_SIZE23	b_SIZE22	b_SIZE21	b_SIZE20	b_SIZE19	b_SIZE18	b_SIZE17	b_SIZE16
NAME($0E)	c_SIZE23	c_SIZE22	c_SIZE21	c_SIZE20	c_SIZE19	c_SIZE18	c_SIZE17	c_SIZE16
NAME($0F)	d_SIZE23	d_SIZE22	d_SIZE21	d_SIZE20	d_SIZE19	d_SIZE18	d_SIZE17	d_SIZE16
OPER	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RESET	0 R	0 R	0 R	0 R	0 R	0 R	0 R	0 R

SIZE23-16 A, B, C, D SIZE should be programmed to match the size of the corresponding IndustryPack memory space. The IP2 chip 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 that Are Compared	Resulting Memory Size
23	22	21	20	19	18	17	16		
0	0	0	0	0	0	0	0	A31-A16	64KB
0	0	0	0	0	0	0	1	A31-A17	128KB
0	0	0	0	0	0	1	1	A31-A18	256KB
0	0	0	0	0	1	1	1	A31-A19	512KB
0	0	0	0	1	1	1	1	A31-A20	1MB
0	0	0	1	1	1	1	1	A31-A21	2MB
0	0	1	1	1	1	1	1	A31-A22	4MB
0	1	1	1	1	1	1	1	A31-A23	8MB
1	1	1	1	1	1	1	1	A31-A24	16MB

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

IP_a, IP_b, IP_c, and IP_d; IRQ0 and IRQ1 Interrupt Control Registers

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

ADR/SIZ	$FFFBC018 through $FFFBC01B (8 bits each)							
BIT	7	6	5	4	3	2	1	0
NAME($18)	a_ERR	0	a_RT1	a_RT0	a_WIDTH1	a_WIDTH0	a_BTD	a_MEN
NAME($19)	b_ERR	0	b_RT1	b_RT0	b_WIDTH1	b_WIDTH0	b_BTD	b_MEN
NAME($1A)	c_ERR	0	c_RT1	c_RT0	c_WIDTH1	c_WIDTH0	c_BTD0	c_MEN
NAME($1B)	d_ERR	0	d_RT1	d_RT0	d_WIDTH1	d_WIDTH0	d_BTD	d_MEN6
OPER	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RESET	? R	0 R	0 R	0 R	0 R	0 R	1 R	0 R

MEN a_MEN/b_MEN/c_MEN/d_ MEN enable the local bus to perform read/write accesses to their corresponding IndustryPack memory space when set, and disable such accesses when cleared. When a double size IndustryPack is used in ab, a_MEN should be set and the WIDTH and MEN control bits in the IP_b General Control Register should be cleared. When a double size IndustryPack is used in cd, c_MEN should be set, and the WIDTH and MEN control bits in the IP_d General Control Register should be cleared.

BTD Setting BTD (bus turn around delay) to a one will insert one inactive clock period following a read cycle on the IP bus. This idle cycle is to eliminate bus contention which would occur if the time to assert a valid address by the IP2 chip is less than the time required by the Industry Pack to put the bus in a high impedance state following the read cycle. The IP2 chip will drive the bus valid following the positive edge of the IP clock in typically seven nanoseconds, and worse case in 15. Note that IndustryPack modules which were designed to meet the 0.7 or earlier revision of the GreenSprings IndustryPack Specification were allowed 40 ns to three-state the bus following a read cycle. The state of BTD affects IP bus cycles which are a result of the

DMA function because they are the only cycles which can occur back to back. When BTD is set to a zero, the IndustryPack interface will start the next cycle as soon as possible.

Note The default BTD setting is to insert the additional one clock period delay between read cycles.

WIDTH1, WIDTH0 The IP2 chip assumes the memory space data-bus width of each of IP_a, IP_b, IP_c, and IP_d to be the value decoded from its control bits WIDTH1 and WIDTH0. Note that the width bits control the assumed memory width for the load-stored (programmed I/O) data path. There is a similar set of bits for the DMA logic memory width control. The following table shows widths inferred by these bits. When a double size IndustryPack is used in ab, then IP_a should be programmed for 32 bit width, and the WIDTH and MEN control bits in the IP_b General Control Register should be cleared. When a double size IndustryPack is used at cd, then IP_c should both be programmed for 32 bit width, and the WIDTH and MEN control bits in the IP_d General Control Register should be cleared.

WIDTH1	WIDTH0	Memory Space Data Width
0	0	32 bits
0	1	8 bits
1	0	16 bits
1	1	Reserved

Note When programming b_WIDTH1-b_WIDTH0 for either 8-bits or 16-bits, a_WIDTH1-a_WIDTH0 must be programmed for 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 IP2 chip 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 with IP = 8 MHz	Recovery Time with IP = 32 MHz
0	0	0 microseconds	0 microseconds
0	1	2 microseconds	0.5 microsecond
1	0	4 microseconds	1 microsecond
1	1	8 microseconds	2 microseconds

There are some restrictions 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 behavior.

a_ERR This bit reflects the state of the ERROR* signal from the IP_a interface.

b_ERR This bit reflects the state of the ERROR* signal from the IP_b interface.

c_ERR This bit reflects the state of the ERROR* signal from the IP_c interface.

d_ERR This bit reflects the state of the ERROR* signal from the IP_d interface.

IP Clock Register

ADR/SIZ	$FFFBC01D (8 bits)							
BIT	7	6	5	4	3	2	1	0
NAME	0	0	0	0	0	0	0	IP32
OPER	R	R	R	R	R	R	R	S
RESET	0 R	0 R	0 R	0 R	0 R	0 R	0 R	0 R

IP32 Setting IP32 to a 1 enables the IndustryPack bus to operate synchronously with the MC68040 local bus clock, 32MHz (on MVME162-5xx boards only). Setting it to a zero enables 8MHz operation (on both MVME162-4*xx* and -5*xx* boards). In this mode, the IndustryPack bus cycles and MC68040 local bus cycles operate asynchronously.

The IP32 bit controls clock synchronization logic. It does not change the clock frequency on the bus. Jumper J24 on the MVME162FX printed circuit board controls the IP bus clock source. If J24 pins 1 and 2 are jumpered, then the IP clock source is set to 8MHz. For this setting, the IP32 control bit must be a 0. If J24 pins 3 and 2 are jumpered, then the IP clock source is set to be synchronous with a 32MHz MC68040 local bus clock. For this setting, the IP32 control bit must be a 1. Whether the setting is 8MHz or 32MHz, all IP ports operate at the same speed.

Note Some versions of the MVME162FX (those identified with assembly number 01-W3960B*xxx*) may have J24 factory-hardwired in the 8MHz position with a staple between J24 pins 1 and 2, hidden beneath an IP module.

Changing the factory setting to the 32MHz setting requires removal of the staple between pins 1 and 2, and installation of a jumper between pins 2 and 3.

DMA Arbitration Control Register

The DMA arbitration control register contents determine whether a fixed or fair arbitration algorithm is used to determine how the MC68040 local bus is attached to the internal DMA data paths.

ADR/SIZ	$FFFBC01E (8 bits)							
BIT	7	6	5	4	3	2	1	0
NAME	0	0	0	0	0	ROTAT	PRI1	PRI0
OPER	R	R	R	R	R	R/W	R/W	R/W
RESET	0 R	0 R	0 R	0 R	0 R	0 R	0 R	0 R

ROTAT ROTAT set to a zero enables a rotating arbitration method where each DMAC has equal access to the MC68040 local bus. If ROTAT is set to a one, the priority is fixed according to the following table

PRI1,PRI0 Fixed priority assignment is defined by the following tables.

PRI1 - PRI0	Priority with one DMA channel at IP sockets a, b, c, & d			
	Highest	Next Highest	Next Lowest	Lowest
00	DMA_a	DMA_b	DMA_c	DMA_d
01	DMA_b	DMA_c	DMA_d	DMA_a
10	DMA_c	DMA_d	DMA_a	DMA_b
11	DMA_d	DMA_a	DMA_b	DMA_c

Programming Model

PRI1 - PRI0	Priority with two DMA channel at IP sockets a and c			
	Highest	Next Highest	Next Lowest	Lowest
00	DMA_a chan 0	DMA_a chan 1	DMA_c chan 0	DMA_c chan 1
01	DMA_a chan 1	DMA_c chan 0	DMA_c chan 1	DMA_a chan 0
10	DMA_c chan 0	DMA_c chan 1	DMA_a chan 0	DMA_a chan 1
11	DMA_c chan 1	DMA_a chan 0	DMA_a chan 1	DMA_c chan 0

IP RESET Register

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

RES Setting RES to a one asserts the IP2 chip IPRESET* signal. IPRESET* is intended to be connected to the Reset* signal on all four IndustryPacks. When software sets the RES bit, IPRESET* stays asserted until software clears RES.

Unlike the IPIC used on earlier versions of the MVME162, the IPRESET* is not asserted when the RESET* in to the ASIC (board level reset) is asserted.

For board artwork marked 84-W90B01E or later, the Reset signal to the IP bus is always driven with the power-up Reset. The power-up Reset is combined (ORed together) with the IPRESET* signal from the IP2 ASIC.

Note The MVME162FX does not comply with the IP specification regarding reset. The MVME162FX does not monitor Vcc and assert reset if Vcc is below a certain threshold.

Programming the DMA Controllers

The IP2 chip implements four DMA channels. They can operate in the standard or addressed mode. sDMA transfers must accommodate the I/O port width. If the port width is 16 bits, then the byte count must be even; if the width is 32 bits, then the byte count must be a multiple of four bytes. The IP address counter must be initialized to zero before an sDMA transfer is enabled. There are no other restrictions placed on DMA operations.

Each DMAC has two modes of operation: command chaining, and direct. In the direct mode, the local bus address, the IndustryPack address, the byte count, and the control register of a DMAC are programmed and the DMAC is enabled. The DMAC transfers data, as programmed, until the byte count is zero, DMAEND is detected true as an input, or an error is detected. When the DMAC stops, the status bits in the DMAC status register are set and an interrupt is sent to the local bus master (if the DMACs interrupts are enabled). Multiple DMAC commands can be automatically invoked using the command chaining mode.

In the command chaining mode, a singly-linked command list is built in local memory and the table address register in the DMAC is programmed with the starting address of the list. DMAC control register 1 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 master (if the DMAC interrupts are enabled). Additionally, when the DMAC finishes processing a command in the list, and interrupts are enabled for that command, the DMAC sends an interrupt to the local bus master if its interrupts are enabled.

Note that when the DMA register context is updated for the next command packet, a DMA function is used. The state of the snoop control signals for this DMA function is determined by the settings of jumper J26 (as is the state of the snoop control signals for all other DMA cycle types). Refer to the *Hardware Preparation* section of the *Installation and Use* manual, V162FXA/IH.

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

There is a case when the byte count for a DMA (to local bus) operation is initially set larger than the actual received data. In this case, data can be stuck in the internal data paths of the DMA controller. To flush the data, set the byte count register to 0. A normal termination of the DMA occurs after the byte count register has been initialized to zero.

Once a DMAC is enabled, its 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 (not IP), 32-bit memory and the entries must be aligned to a 16-byte boundary. That is, the address which is loaded into the DMA table address counter must have bits three through zero set to a zero. This is true for the initial value which is loaded by the processor or the subsequent values which are loaded by the command chaining logic. If they are not set to zero, the command chaining process will halt.

A DMAC command packet includes a control word that defines: single command interrupt enable, DMA transfer direction, and DMAEND direction and usage. The format of the control word is the same as control register 2. The command packet also includes a local bus address, an IP address, a byte count, and a pointer to the next command packet in the list. The end of a command is indicated by setting bit 0 or 1 of the next command address. The command packet format is shown in the following table.

Entry	Function	
0(bits 23-16)	Address of Next Command Packet	
1(Bits 31-0)	Local Bus Address	
2(Bits 23-0)	Control Word	Byte Count
3(Bits 23-0)		IndustryPack Address DMA

DMA Enable Function

There are certain DMA channel contexts which are illegal. If an attempt is made to program the DMA control register 1 for each channel a and b or c and d to an illegal state, the DEN (DMA enable control bit) will not set when it is loaded to a one via a processor store instruction. This condition can be tested by writing the DEN bit to a one and reading a zero. Refer to the description of the DMA Enable Register for the required programming sequence of the control registers and enable bits.

The following are legal contexts for DMA channel configurations. Note that configuration rules for DMA controllers for IP_a and IP_b are defined. The same relationships exist for IP_c and IP_d.

- If IP_a data bus is 8 or 16 bits, there are not any restrictions placed on IP_b.
- If IP_a data bus is 32 bits and the ADMA mode is selected, then the DMA controller associated with IP_b cannot be used.
- If A_CH1 bit is set in the DMA controller register associated with IP_b and both channel A and B operate in the SDMA mode, then the DMA channels associated with IP_a and IP_b can both be used if the data width for channel A and B are set equal. This case allows the DMA channel that normally responds to IP_b-DMAreq_0 pin to respond to IP_a-DMAreq_1 pin. This enables full duplex communications operation at IP_a.

DMA Control and Status Register Set Definition

The four sets of DMA controller CSRs are almost identical in functionality. Each register set is grouped as DMACa, DMACb, DMACc, and DMACd. These register sets are shown pictorially in the CSR register summary section. Only one register set is defined except that the offset is noted for the four possible values. Refer to the definitions of bit 1 of the DMA Control Register 1 for a description of how the register sets are associated with the physical DMA request from the Industry Packs.

DMA Status Register

ADR/SIZ	$FFFBC020, $38, $50, $68 (8 bits each)							
BIT	7	6	5	4	3	2	1	0
NAME	0	DLBE	0	IPEND	CHANI	TBL	IPTO	DONE
OPER	R	R	R	R	R	R	R	R
RESET	0 R	0 R	0 R	0 R	0 R	0 R	0 R	1 R

DONE This bit is set when DMAC has finished executing commands and there were no errors, or DMAC has finished executing commands because the DHALT bit was set. This bit is cleared when DMAC is enabled. A DMAC interrupt will be generated if interrupts are enabled.

IPTO When this bit is set, a DMAC access to an IndustryPack timed out. This bit is cleared when DMA is enabled. A DMAC interrupt will be generated if interrupts are enabled.

TBL When this bit is set, DMAC received an error on the local bus while it was reading commands from the command packet. Additional information is provided in bit 6 (DLBE).This bit is cleared when DMAC is enabled.

DLBE When this bit is set, DMAC received a TEA. (TEA is transfer error acknowledge signal on the MC68040 local bus. It indicates that a timeout occurred.) This bit is cleared when DMAC is enabled. A DMAC interrupt will be generated if interrupts are enabled.

CHANI When this bit is set, the INTE bit in the DMA Control Register 2 was detected. This bit is cleared when DMA is enabled or the interrupt status bit is cleared in the DMA interrupt control register or the DHALT

bit was detected in the DMA Control Register 1. A DMAC interrupt will be generated if interrupts are enabled.

IPEND When this bit is set, the DMA process was terminated if the DMAEND signal was asserted by the Industry Pack and the DMAEI bit is set in the DMA Control Register 2.This bit is cleared when DMA is enabled. A DMAC interrupt will be generated if interrupts are enabled

DMA Interrupt Control Register

ADR/SIZ	$FFFBC021, $39, $51, $69 (8 bits each)							
BIT	7	6	5	4	3	2	1	0
NAME	0	0	DINT	DIEN	DICLR	DIL2	DIL1	DIL0
OPER	R	R	R	R/W	C	R/W	R/W	R/W
RESET	0 R	0 R	1 R	0 R	0 R	0 R	0 R	0 R

DIL2-DIL0 These three bits select the interrupt level for DMA. Level 0 does not generate an interrupt.

DICLR Writing a logic 1 to this bit clears the DINT status bit.

DIEN When DIEN is set, the interrupt is enabled. When DIEN is cleared, the interrupt is disabled.

DINT When this bit is high, an interrupt will be generated for a DMAC if the DIEN bit is set to a one. The interrupt is at the level programmed in DL2-DL0. The DINT bit is set when one of the following bits are set in the Status Register: DLBE, IPEND, CHANI, IPTO, and DONE.

DMA Enable Register

ADR/SIZ	$FFFBC022, $3A, $52, $6A (8 bits each)							
BIT	7	6	5	4	3	2	1	0
NAME	0	0	0	0	0	0	0	DEN
OPER	R	R	R	R	R	R	R	S
RESET	0 R	0 R	0 R	0 R	0 R	0 R	0 R	0 R

DEN Setting the DEN bit to a one will enable the DMA function. Software should not write to the DMA control registers between the time the DEN bit is set and the DMA process is completed. In general, this is true on a per channel basis but there are inter-relationships between the DMA channels/board architecture which require the initialization of certain bits for each pair of DMA channels before the DEN bits can be set. That is, if DMA channels a and b are going to be used concurrently, DMA Control Register 1 should be initialized for both channels before ether channel is enabled. This is also true for DMA channels c and d. Refer to the section on the DMA Enable Function.

DMA Control Register 1

ADR/SIZ	$FFFBC024, $3C, $54, $6C (8 bits each)							
BIT	7	6	5	4	3	2	1	0
NAME	DHALT	0	DTBL	ADMA	WIDTH1	WIDTH0	A_CH1 or C_CH1	XXX
OPER	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RESET	0 R	0 R	0 R	0 R	0 R	0 R	0 R	0R

XXX This bit must remain cleared. If it is set to a one, the IP2 chip ASIC will not function correctly.

A_CH1, C_CH1 When A_CH1 is set to a zero, DMA request 0 from Industry Pack b is associated with DMACb register set. When it is set to a one, DMA request 1 from Industry Pack a is associated with DMACb register set. When C_CH1 is set to a zero, DMA request 0 from Industry Pack d is associated with DMACd register set. When it is set to a one, DMA request 1 from Industry Pack c is associated with DMACd register set. Note that DMACa register set is always associated with DMA request 0 from Industry Pack a and DMACc register set is always associated with DMA request 0 from Industry Pack c. Therefore these bit positions are not defined for these two register sets. Refer to the section on the Enable DMA Function for information and restrictions on the operation of A_CH1 and C_CH1.

WIDTH1-WIDTH0 WIDTH bits specify the width of the IndustryPack interface at position a or position a_b. The following table defines the bit encoding. Note that these width control bits are independent of the width control bits in the General Control Registers. Also note that

unlike the width control bits in the General Control Registers, these width control bits define the width of both the memory and I/O interface.

WIDTH1	WIDTH0	Assumed Data Bus Width
0	0	32 bits
0	1	8 bits
1	0	16 bits
1	1	RESERVED

ADMA Setting ADMA to a one will enable the address mode DMA operation. Setting it to a zero will enable the standard mode of DMA operation. Refer to the section on the DMA Enable Function for information and restrictions on the operation of the ADMA control bit.

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

DHALT When this bit is high, DMA halts at the end of a command when DMA is operating in the command chaining mode. When this bit is low, DMA executes the next command in the list. Software must be careful not to change the state of bits 0 through 6 of this control register when the DHALT bit is set.

DMA Control Register 2

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

ADR/SIZ	$FFFBC025, $3D, $55, $6D (8 bits each)							
BIT	7	6	5	4	3	2	1	0
NAME	INTE	0	DMAEI	DMAEO	ENTO	TOIP	XXX	XXX
OPER	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RESET	0 R	0 R	0 R	0 R	0 R	0 R	0 R	0 R

XXX,XXX These bits are not under software control. Reading or writing to them has no effect on the operation of the IP2 chip ASIC.

TOIP This bit defines the direction in which DMAC transfers data. When this bit is high, data is transferred to the IndustryPack. When it is low, data is transferred from the IndustryPack.

ENTO ENTO set to a one will enable the watchdog timeout function for DMA cycles on the IP bus. The timeout period is fixed at approximately 1 msec. If a timeout does occur, the IP bus cycle is terminated and the IPTO bit is set in the DMA Status Register. Note that the IndustryPack interface in the IP2 chip ASIC will wait indefinitely if the ENTO bit is cleared and a DMA cycle on the IP bus is not acknowledged. The IP2 chip ASIC must be reset to clear this condition. It is recommended that ENTO be set to a one.

DMAEO When DMAEO is set, DMA drives DMAEND and asserts it during the DMA IP cycle in which the byte count expires. When DMAEO is cleared, DMA's DMAEND driver is disabled.

DMAEI When DMAEI is set, DMA terminates if the assertion of DMAEND is detected and the sDMA function is enabled That is, the ADMA control bit in the DMA Control Register 1 must be set to a zero. If the assertion of DMAEND is not detected, DMA will terminate according to the byte count value and the command chaining mode.

Note that the DMAEI function does not operate correctly for DMA channels B or D if the associated A_CH1 or C_CH1 bit is set in the DMA control register for channel B or D. Do not use DMAEI to halt the DMA operation under these conditions.

INTE This bit is used only in the command chaining mode and it is only modified when DMA loads its control register from the control word in the command packet. When this bit in the command packets is set, an interrupt is sent to the local bus interrupter when the command in the packet has been executed. The local bus is interrupted if DMA's interrupt is enabled.

DMA Local Bus Address Counter

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

ADR/SIZ	$FFFBC028, $40, $58, $70 (32 bits each)		
BIT	31	...	0
NAME	DMA Local bus Address Counter		
OPER	R/W		
RESET	0 R		

DMA IndustryPack Address Counter

In the direct mode, this counter is programmed with the starting address of the data buffer in IndustryPack memory. The value programed in the IndustryPack address counter is the address that would be used when referencing the IndustryPack memory space from the local bus. Refer to the addressing mapping from the local bus to the IndustryPack bus for different IndustryPack memory widths, described later in this chapter.

For sDMA operations, the IndustryPack Counter must be cleared before the DMAC is enabled.

ADR/SIZ	$FFFBC02C, $44, $5C, $74 (32 bits each)			
BIT	31...23	22		0
NAME	0	DMA Industry Pack Address Counter		
OPER	R	R/W		
RESET	0 R			

DMA Byte Counter

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

For sDMA operations, if the port width is 16 bits, then the byte count must be initialized to an even value; if the width is 32 bits, then the byte count must be initialized to a multiple of four bytes.

If the port width is 8 bits, the byte count value does not have restrictions.

ADR/SIZ	$FFFBC030, $48, $60, $78 (32 bits each)			
BIT	31...24	23		0
NAME	0	DMA_a Byte Counter		
OPER	R	R/W		
RESET	0 R			

DMA Table Address Counter

In the command chaining mode, this counter should be loaded by the processor with the starting address of the list of commands. Note that the command packets in local bus memory must always be 16-byte aligned. That is, the starting address of any command packet must have the least significant nibble of the address set to a zero. If the Table Address Counter is initialized with a value where the four least significant bits are not a zero, the logic will interpret it as a halt condition and the command chaining process will not start. Therefore the entry in the last command packet which is loaded into the Table Address Counter should have one or more of these address bits set to a one to halt the command chaining process.

ADR/SIZ	$FFFBC034, $4C, $64, $7C (32 bits each)		
BIT	31	...	0
NAME	DMA Table Address Counter		
OPER	R/W		
RESET	0 R		

Programming the Pacer Clock

Pacer clock registers are defined in the following paragraphs.

Pacer Clock Interrupt Control Register

ADR/SIZ	$FFFBC080 (8 bits)							
BIT	7	6	5	4	3	2	1	0
NAME	0	IRE	INT	IEN	ICLR	IL2	IL1	IL0
OPER	R	R/W	R	R/W	C	R/W	R/W	R/W
RESET	0 R	0 R	0 R	0 R	0 R	0 R	0 R	0 R

IL2-0 These three bits select the interrupt level for the pacer clock interrupt. Level 0 does not generate an interrupt.

ICLR Writing a logic 1 to this bit clears the INT status bit. This bit always reads as 0.

IEN When IEN is set, the pacer clock interrupt is enabled. When IEN is cleared, the interrupt is disabled.

INT When this bit is high, an interrupt is being generated for the pacer clock at the level programmed in IL2-IL0.

IRE This bit controls which action of the pacer clock output causes interrupts.

IRE	Pacer Clock Action That Causes Interrupts
0	Rising Edge
1	Falling Edge

Pacer Clock General Control Register

ADR/SIZ	$FFFBC081 (8 bits)							
BIT	7	6	5	4	3	2	1	0
NAME	PLTY	PLS	0	EN	CLR	PS2	PS1	PS0
OPER	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W
RESET	0 R	0 R	0 R	0 R	0 R	0 R	0 R	0 R

PS2-0 These three bits select the frequency of the pre-scale logic output The MC68040 bus clock (BCK) is used as the input to the pre-scale logic. BCK is ether 25 MHz or 32 MHz. BCK frequency can be determined by examining the Version Register in the MC2 chip ASIC.

PS2-PS0	Pre-scaler Output Frequency	
	PLS = 0	PLS = 1
0	BCK/2	No Output
1	BCK/4	BCK/2
2	BCK/8	BCK/4
3	BCK/16	BCK/8
4	BCK/32	BCK/16
5	BCK/64	BCK/32
6	BCK/128	BCK/64
7	BCK/256	BCK/128

CLR Setting this bit forces the pacer clock's internal registers (except for the interrupt and general control registers) to zero. These registers include the pre-scaler and timer counters. Note that these registers will remain cleared until the CLR bit is set to a zero.

EN When the EN bit is set, the pacer clock is enabled. When it is cleared, the pacer clock is suspended. EN performs its function by enabling/disabling the pre-scaler's counter. Note that clearing EN does not clear any of the pacer clock's registers.

PLS When PLS is set, the pacer clock output is asserted for one BCK period. When PLS is cleared, the pacer clock output toggles creating a square wave.

PLTY PLTY controls the polarity of the pacer clock output. When PLTY is cleared, the negated (and cleared) state of the output is a logic 0, and the asserted state is a logic 1. When PLTY is set, the opposite is true.

Pacer Clock Timer Register

ADR/SIZ	$FFFBC082 (16 bits)		
BIT	15	...	0
NAME	Pacer Clock Timer Register		
OPER	R/W		
RESET	0 R		

When enabled, the pacer clock timer counter increments until it matches the value contained in this register, at which time it rolls over and resumes counting. The effect is that the frequency of the pacer clock output is the frequency of the (pre-scaler output)/(the-value-in-this-register + 1). For example, if the PLS bit is cleared, PLS2-0 are %000, and the timer register contains $0001, then the pacer clock output frequency is BCK/4 = 8 MHz if BCK = 32 MHz. For the pulsed output, the formula for the period of the frequency of the recurring pulse is 1/((pre-scaler output)/ (the-value-in-this-register + 1)). For example, if the PLS bit is set, PLS2-0 are %001, and the timer register contains $0001, then the pacer clock frequency of the pulsed output is BCK/4 = 8 MHz if BCK = 32 MHz.

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.

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.

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<22-1> = LBA<23-2>, and IPA<0> = LBA<0>.

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
|	|	|
|	|	
|	|	|
|	|	
|	|	|
|	|	
$00FFFFFB	$7FFFFD	IP_a
$00FFFFFC	$7FFFFE	IP_b or ab
$00FFFFFD	$7FFFFF	IP_b
$00FFFFFE	$7FFFFE	IP_a
$00FFFFFF	$7FFFFF	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<6-0> = LBA<6-0>. Note that IPA<22-7> 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<6-1> = LBA<7-2> and IPA<0> = LBA<0>. Note that IPA<22-7> do not pertain to I/O space.

LBA	IPA<6-0>	Comments
$FFF58400	%000000	IP_b or ab
$FFF58401	%000001	IP_b
$FFF58402	%000000	IP_a
$FFF58403	%000001	IP_a
$FFF58404	%000010	IP_b or ab
$FFF58405	%000011	IP_b
|	|	|
|	|	
|	|	|
|	|	
|	|	|
|	|	
$FFF584FC	%111110	IP_b or ab
$FFF584FD	%111111	IP_b
$FFF584FE	%111110	IP_a
$FFF584FF	%111111	IP_a

IP_a ID Space

This example is for IP_a ID space. The relationship of the IndustryPack address to the local bus address is: IPA<5-0> = LBA<5-0>. Note that IPA<22-6> do not pertain to ID space.

LBA	IPA<5-0>	Comments
$FFF58080	%000000	
$FFF58081	%000001	
$FFF58082	%000010	
$FFF58083	%000011	
|	|	|
|	|	
|	|	|
|	|	
|	|	|
|	|	
$FFF580BC	%111100	
$FFF580BD	%111101	
$FFF580BE	%111110	
$FFF580BF	%111111	

IP to Local Bus Data Routing

This section shows data routing from an IP to the local bus.

Memory Space Accesses

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

IPWIDTH refers to the memory space width that has been programmed into the general control register for the IndustryPack being accessed.

LBSIZE refers to local bus transfer size.

LBA refers to local bus address signals 1 and 0.

LD refers to the local data bus.

IPA refers to IndustryPack address signals 2,1,0. The IP2 chip implements dynamic bus sizing for memory space accesses whose local bus size is greater than the port width of the IndustryPack that is being accessed. Because of this, the IP2 chip 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

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>
	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>

Serial Port Connections

5

Introduction

This chapter contains connection diagrams for Serial Ports 1 and 2 on the MVME162FX. (Serial Port 2 uses Serial Interface Modules (SIMs) to establish its port.) These ports are connected to external devices through the MVME712 series of transition modules. The table below lists the connection diagrams:

Figure Number	MVME162FX		MVME712M		MVME712A, AM, -12, -13	
	Serial Port	Type	Serial Port	Type	Serial Port	Type
5-1	1	EIA-232 DCE	2	EIA-232 DTE		
5-2	1	EIA-232 DCE	2	EIA-232 DCE		
5-3	2	EIA-232 DTE	4	EIA-232 DTE		
5-4	2	EIA-232 DCE	4	EIA-232 DCE		
5-5	2	EIA-232 DTE	4	EIA-232 DCE		
5-6	2	EIA-232 DCE	4	EIA-232 DTE		
5-7	2	EIA-530 DTE	n/a	n/a	n/a	n/a
5-8	2	EIA-530 DCE	n/a	n/a	n/a	n/a
5-9	1	EIA-232 DCE			2	EIA-232 DTE
5-10	1	EIA-232 DCE			2	Modem
5-11	2	EIA-232 DTE			4	EIA-232 DTE
5-12	2	EIA-232 DCE			4	EIA-232 DTE
5-13	2	EIA-485/-422	n/a	n/a	n/a	n/a

The configuration of the serial ports as EIA-232-D Data Terminal Equipment (DTE) or Data Circuit-terminating Equipment (DCE), or as a modem port in one case, is accomplished by jumpers on the MVME712 series of transition modules. For more information, refer to the *MVME712-12, MVME712-13, MVME712A, MVME712AM,*

and MVME712B Transition Modules and LCP2 Adapter Board User's Manual or the *MVME712M Transition Module and P2 Adapter Board User's Manual* for more information.

Note Refer to the *Serial Communications Interface* section in the *MVME162FX Embedded Controller Installation and Use* manual for more details of the use of the MVME712 series modules with the MVME162FX.

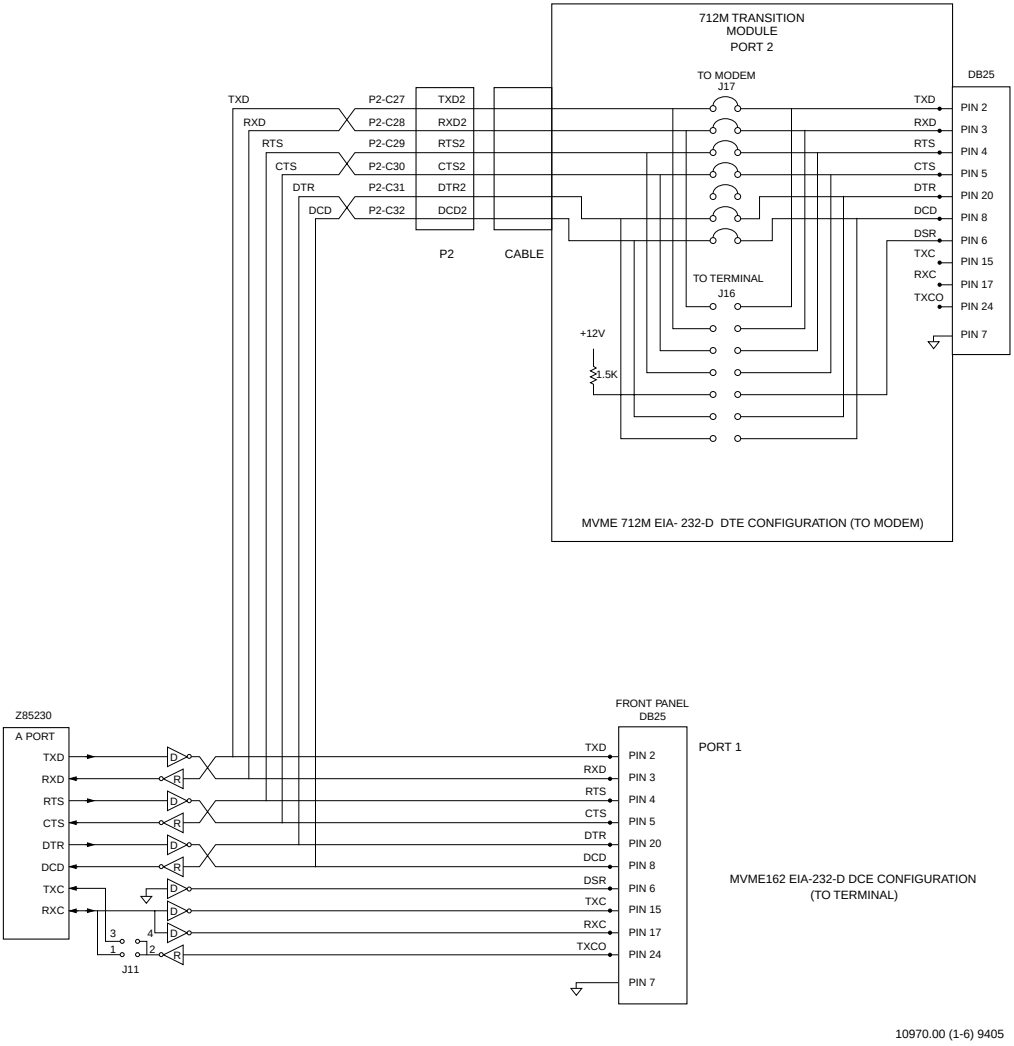

Figure 5-1. MVME162FX Port 1 EIA-232 DCE, MVME712M Port 2 DTE

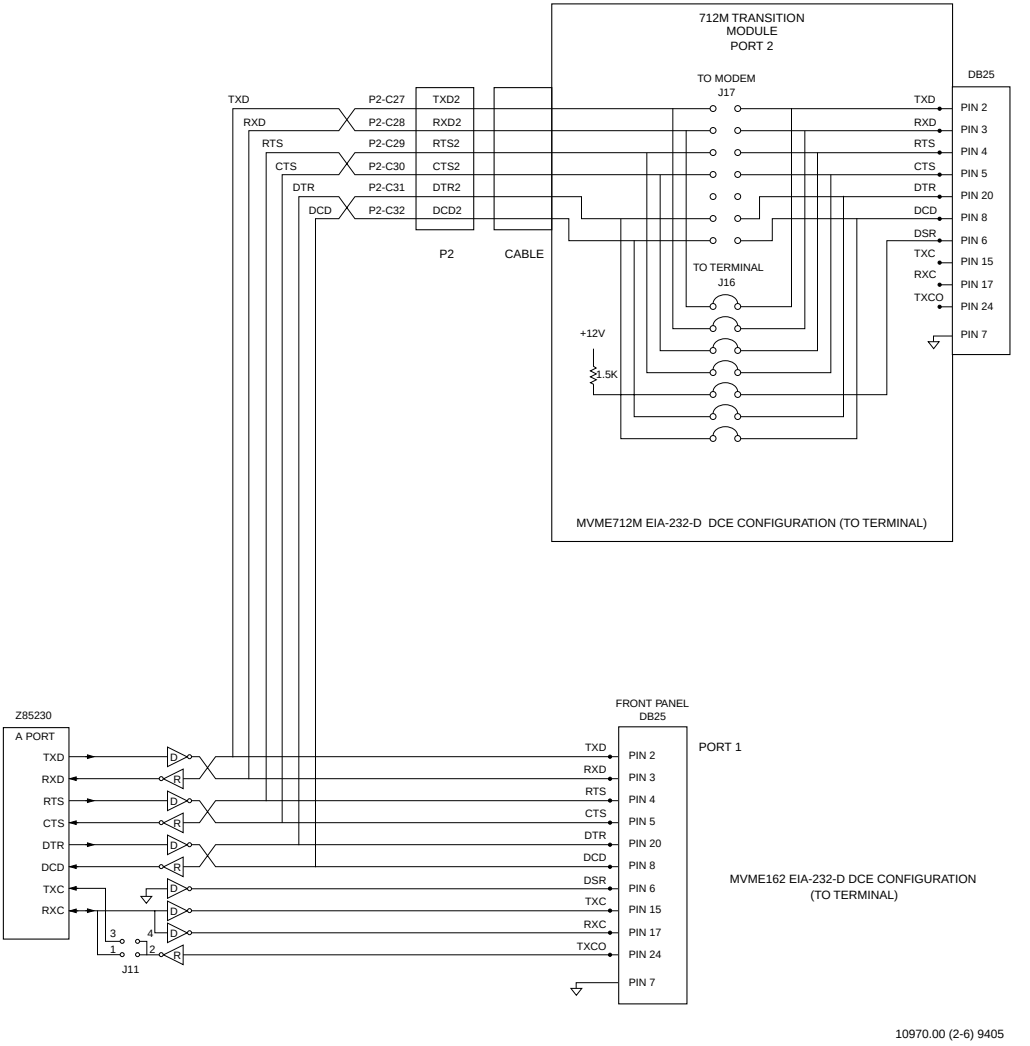

Figure 5-2. MVME162FX Port 1 EIA-232 DCE, MVME712M Port 2 DCE

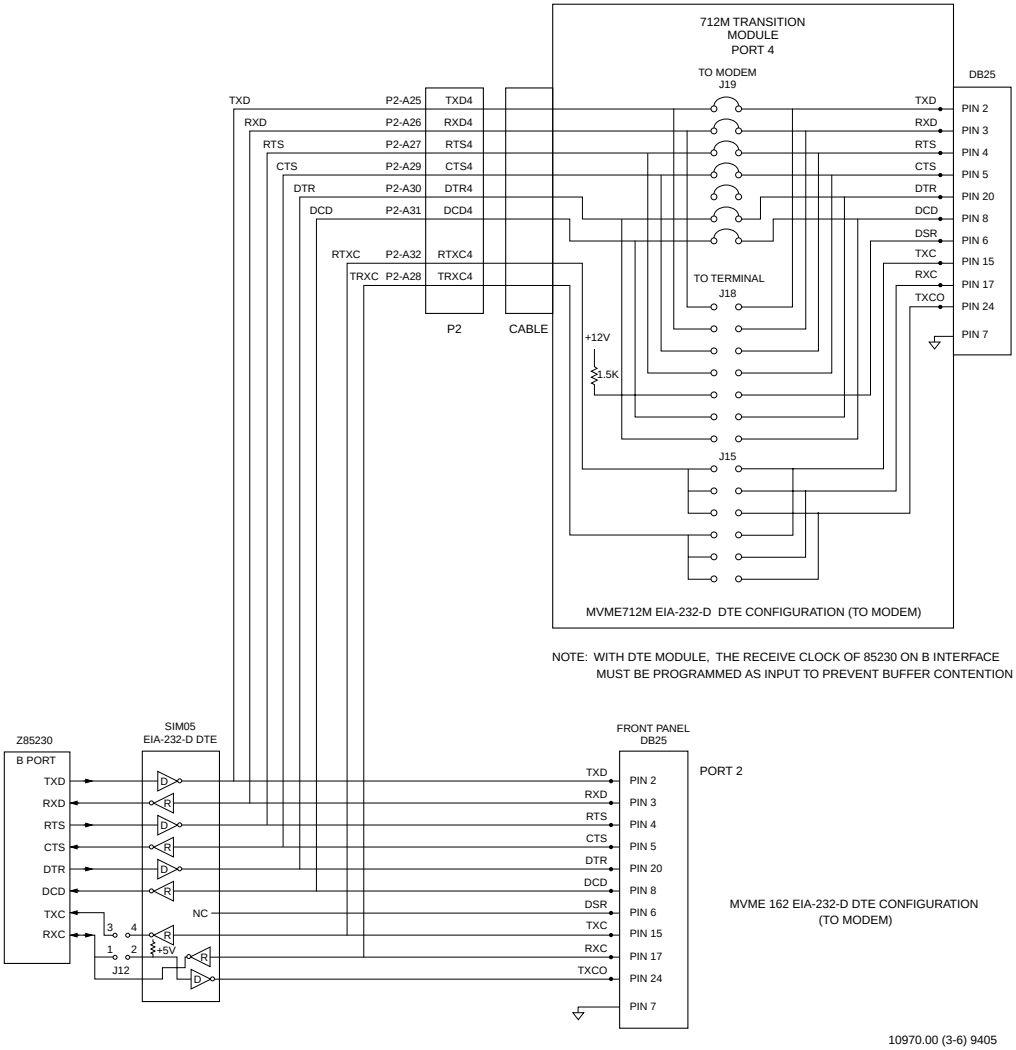

Figure 5-3. MVME162FX Port 2 EIA-232 DTE, MVME712M Port 4 DTE

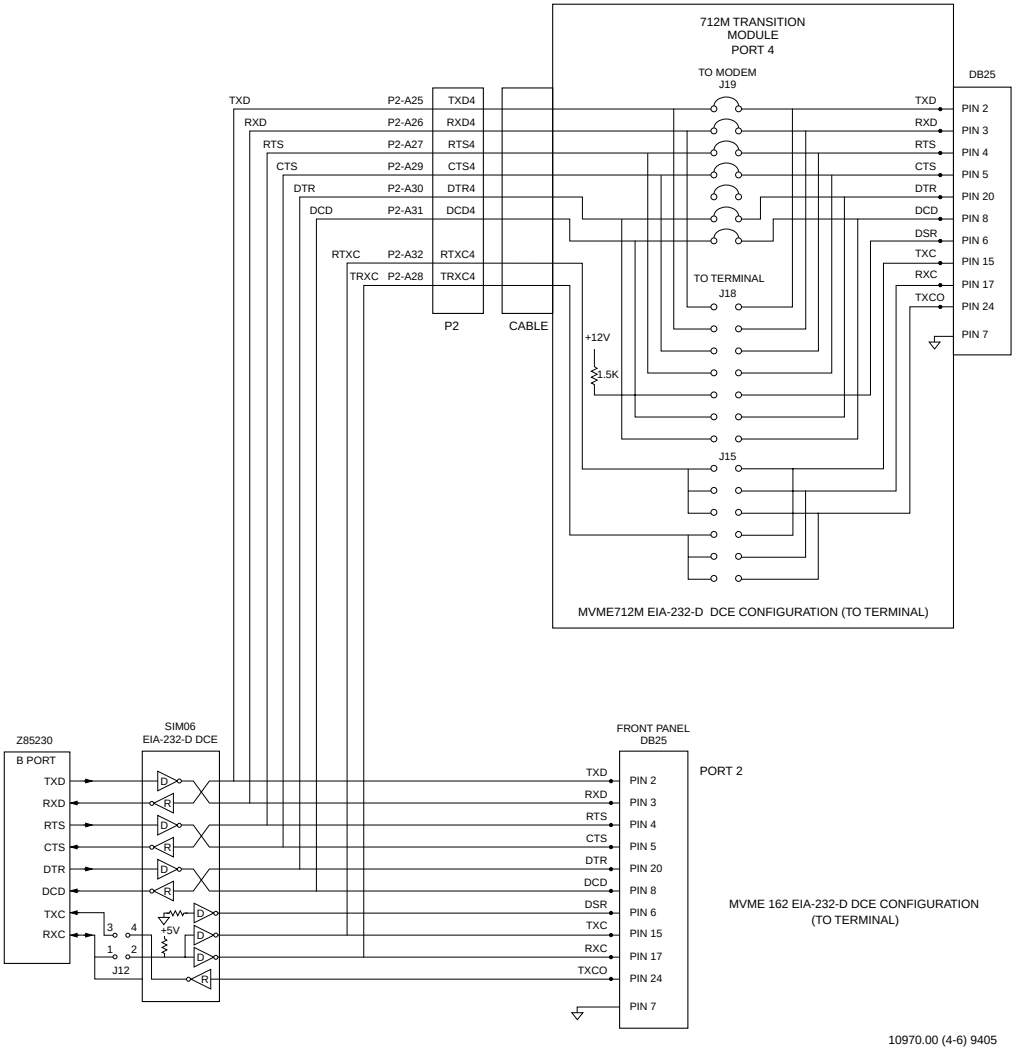

Figure 5-4. MVME162FX Port 2 EIA-232 DCE, MVME712M Port 4 DCE

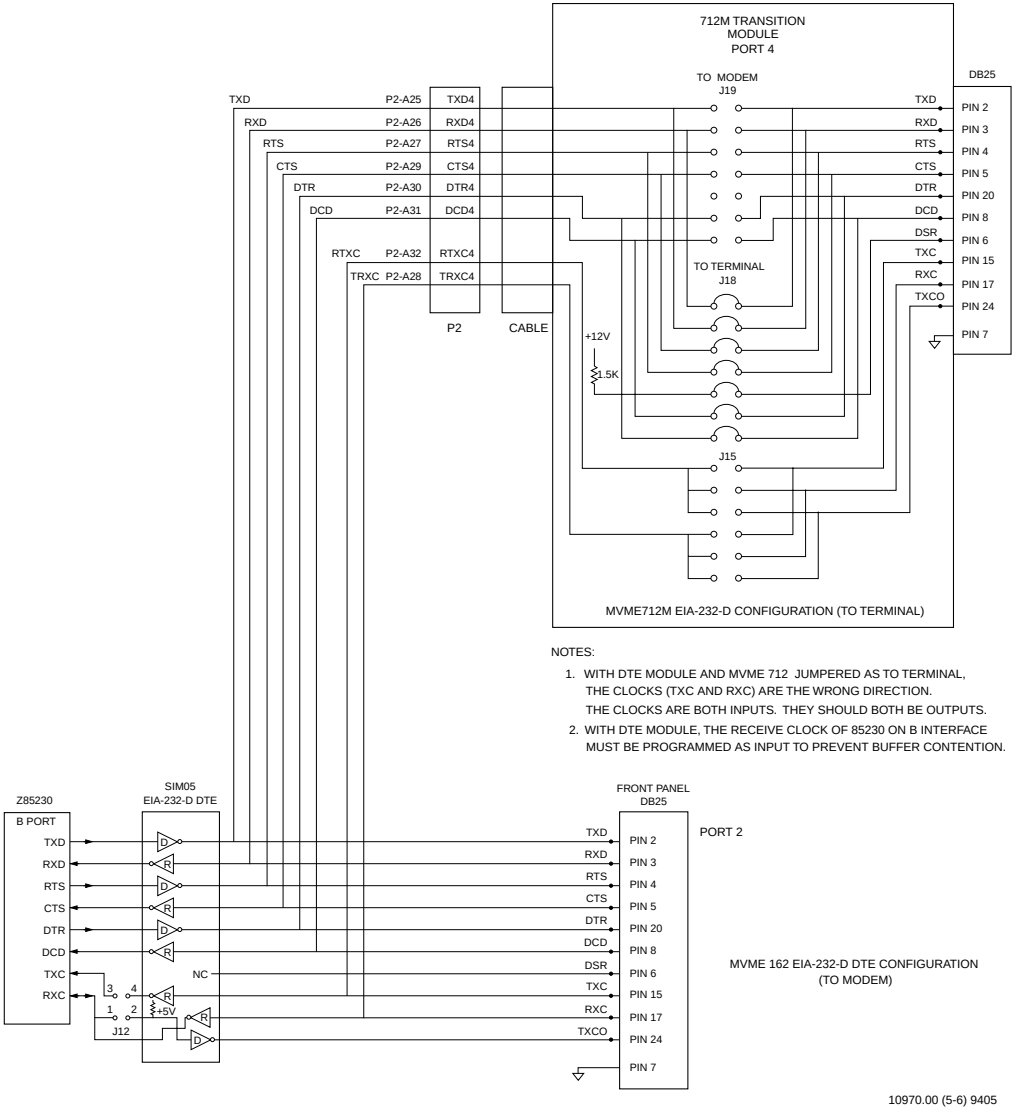

Figure 5-5. MVME162FX Port 2 EIA-232 DTE, MVME712M Port 4 DCE

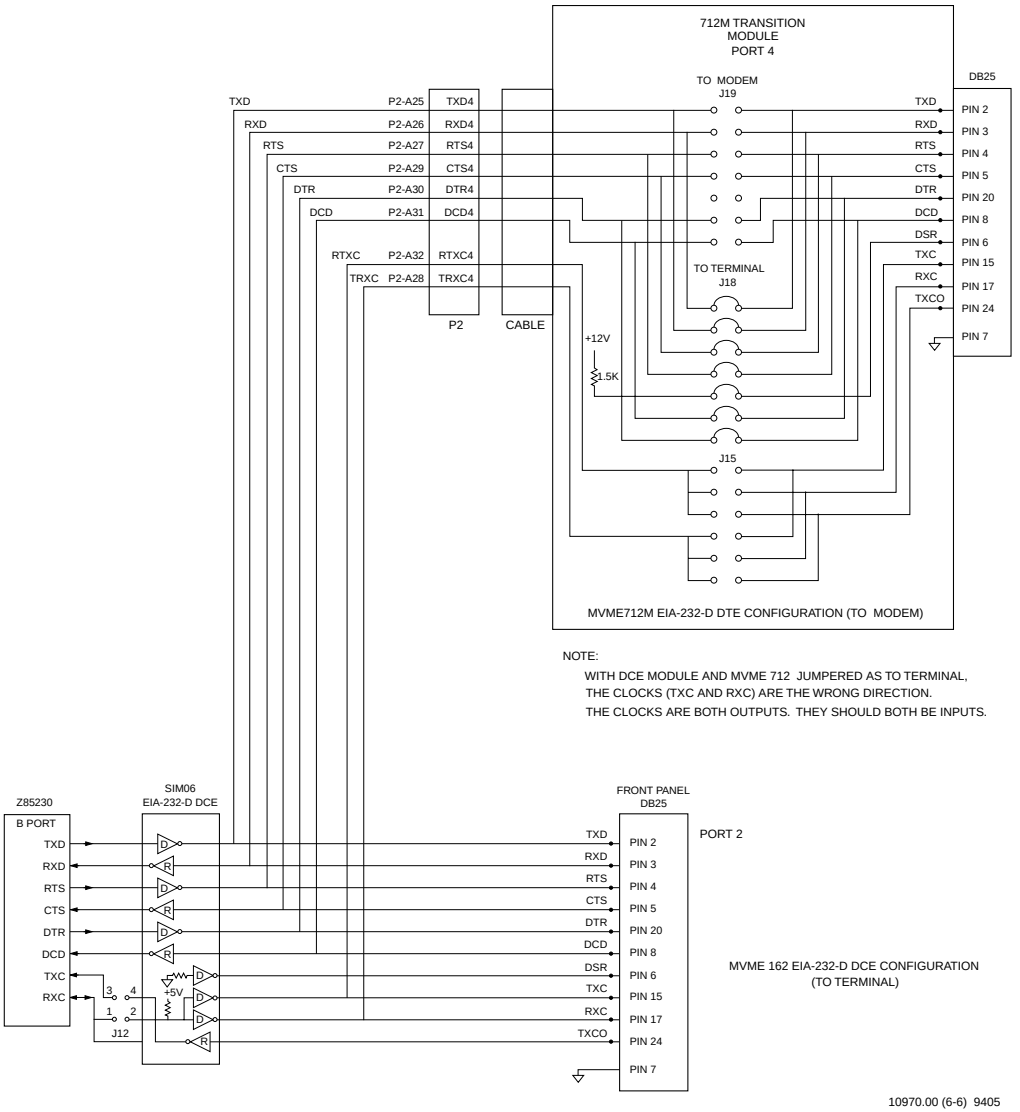

Figure 5-6. MVME162FX Port 2 EIA-232 DCE, MVME712M Port 4 DTE

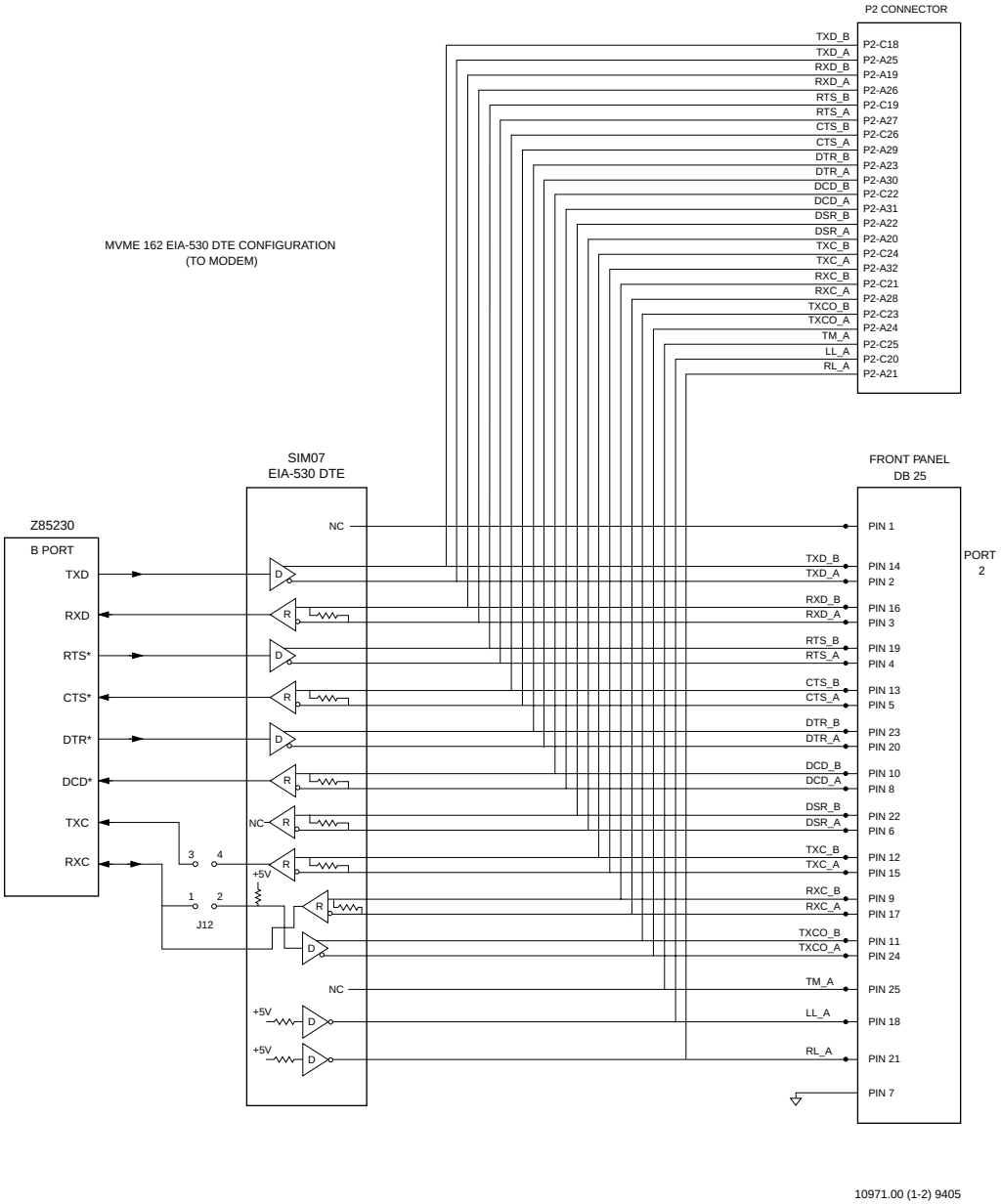

Figure 5-7. MVME162FX Port 2 EIA-530 DTE

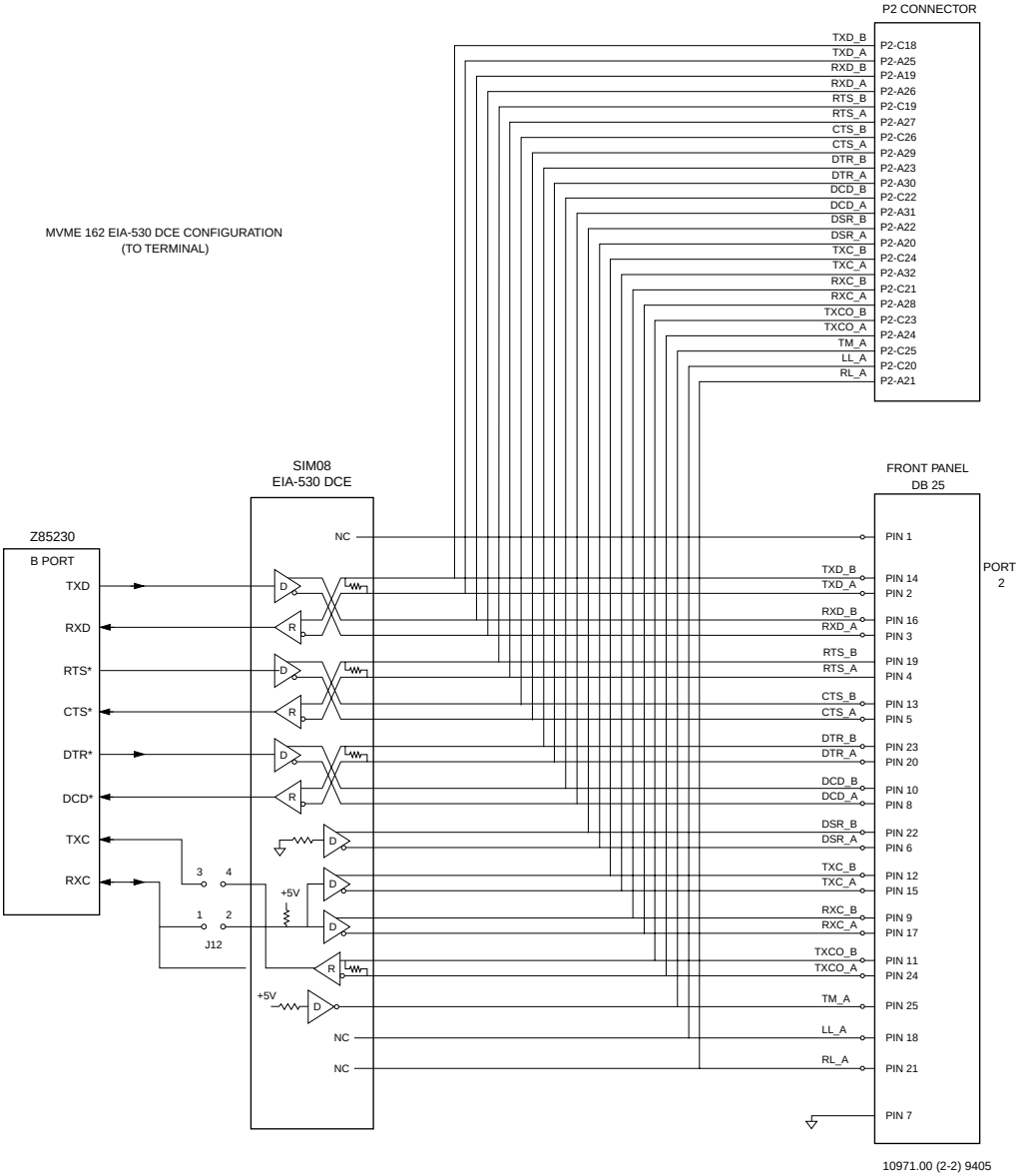

Figure 5-8. MVME162FX Port 2 EIA-530 DCE

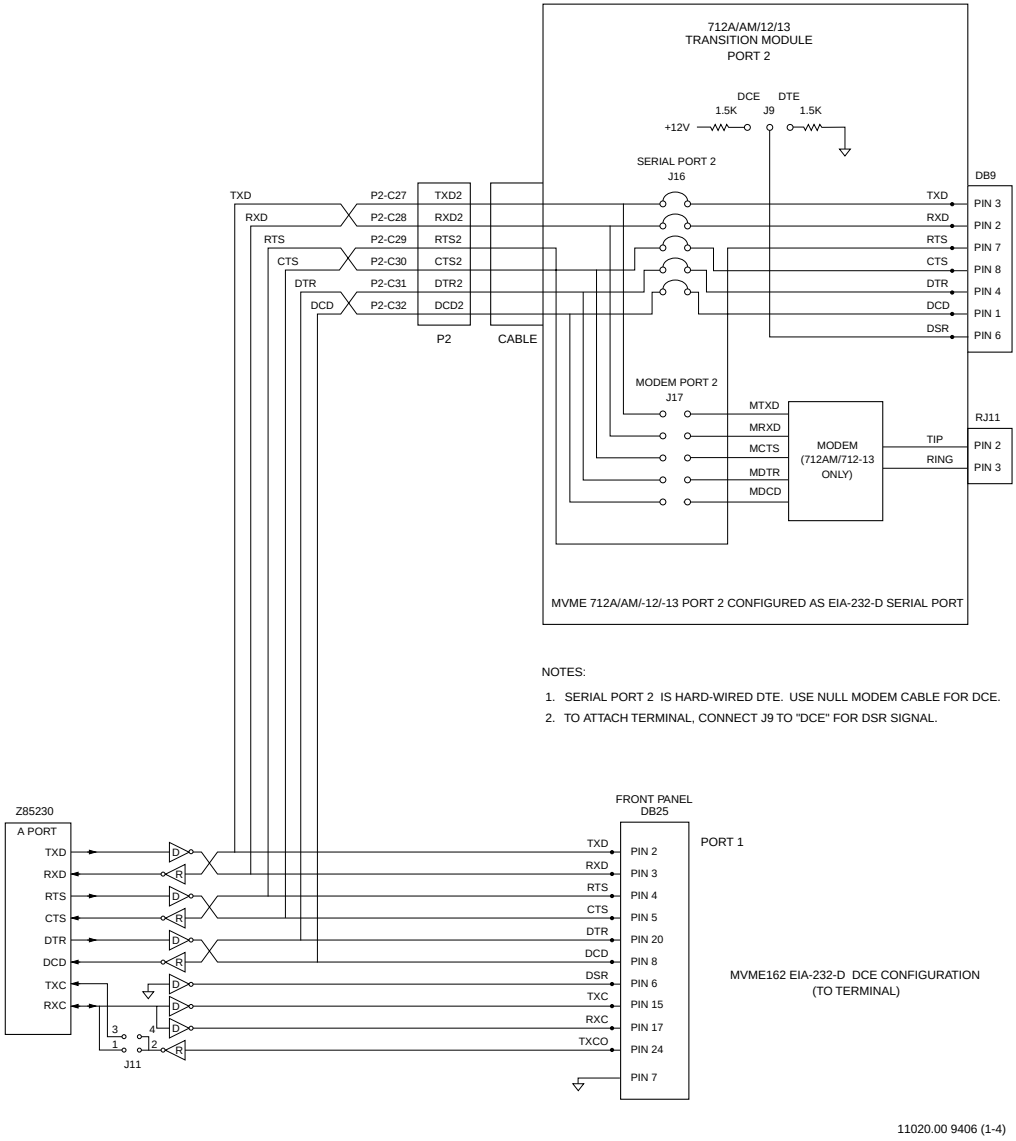

Figure 5-9. MVME162FX Port 1 EIA-232 DCE, MVME712A/AM/-12/-13 Port 2 DTE

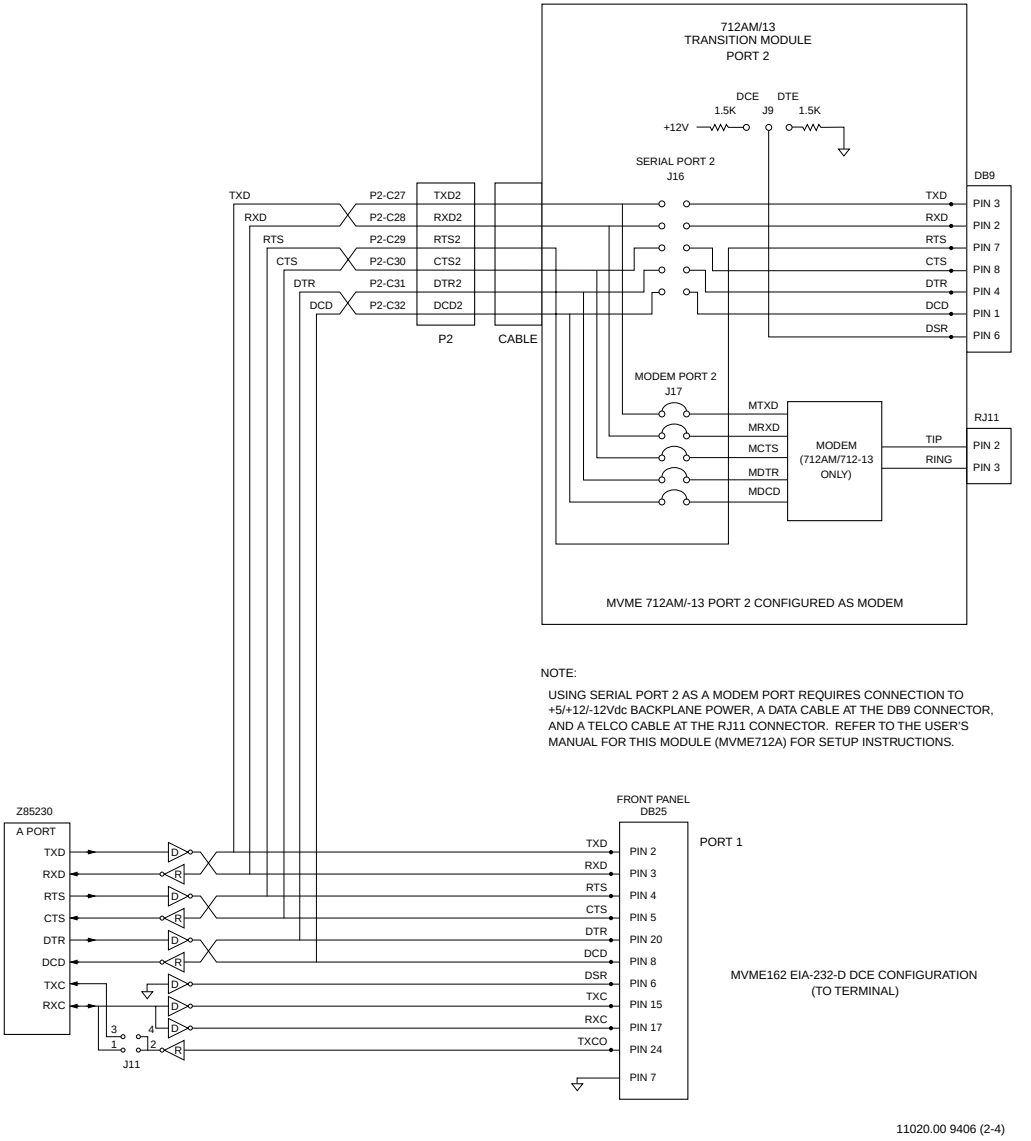

Figure 5-10. MVME162FX Port 1 EIA-232 DCE, MVME712AM/-13 Port 2 as Modem Port

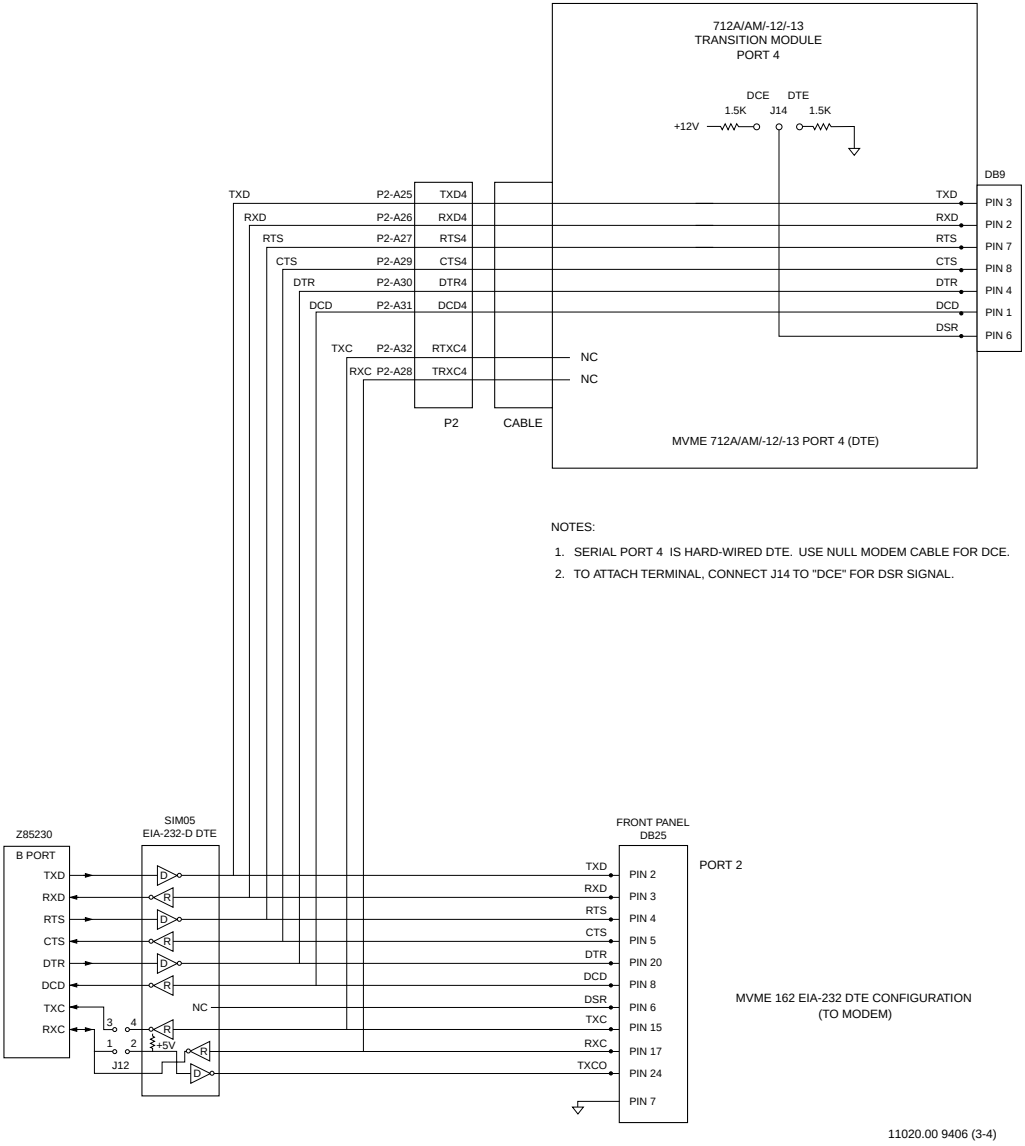

Figure 5-11. MVME162FX Port 2 EIA-232 DTE, MVME712A/AM/-12/-13 Port 4 DTE

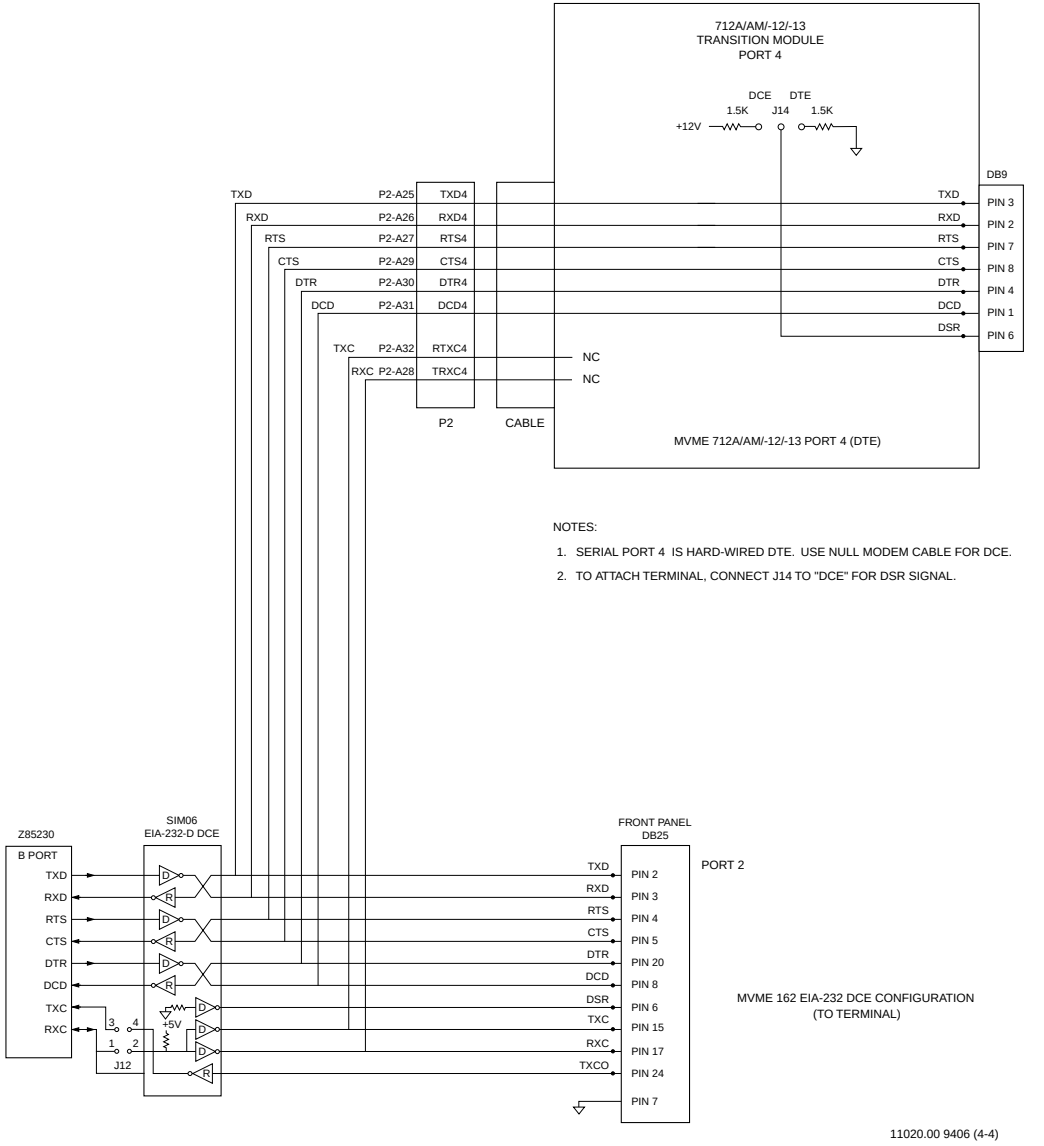

Figure 5-12. MVME162FX Port 2 EIA-232 DCE, MVME712A/AM/-12/-13 Port 4 DTE

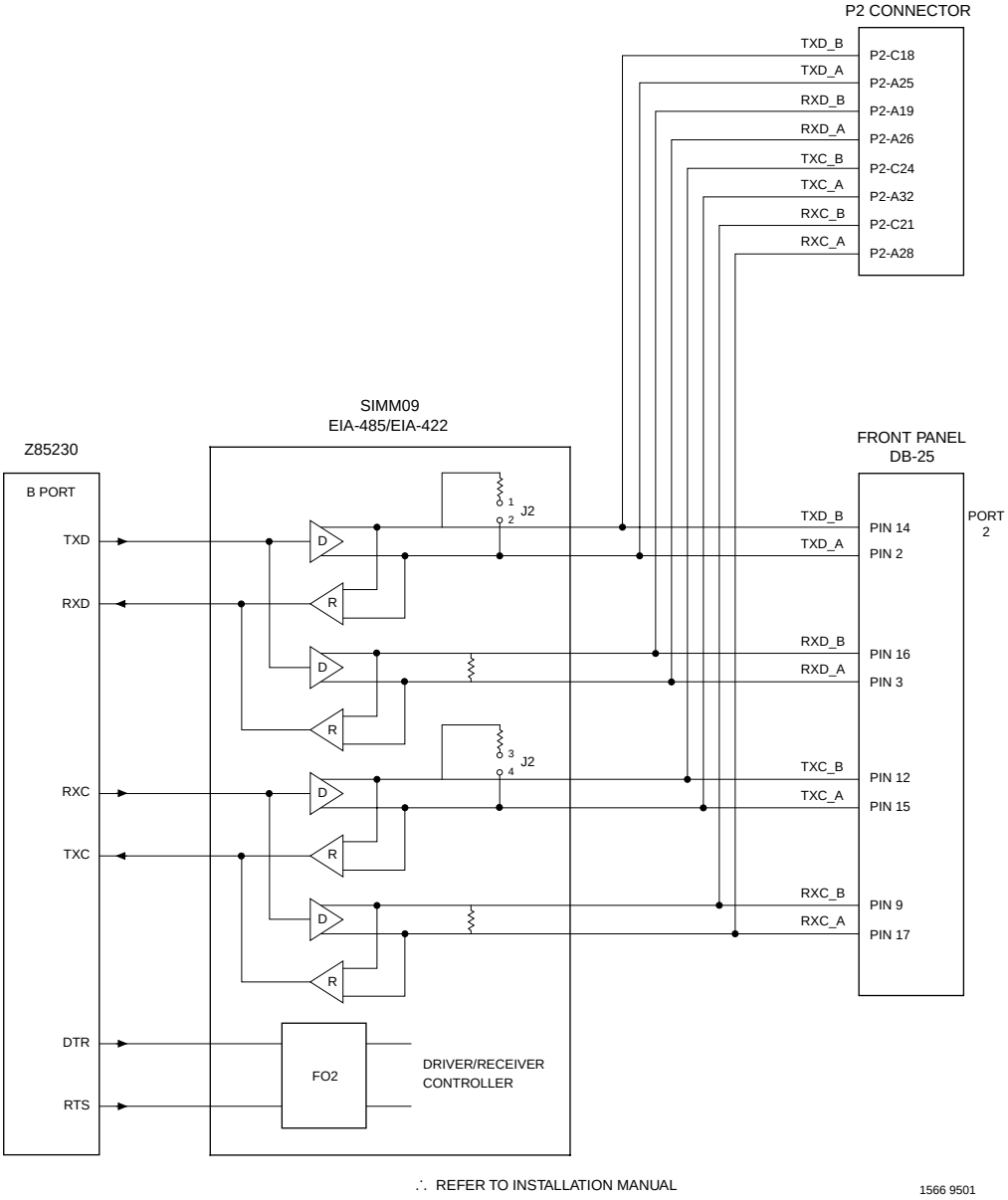

Figure 5-13. MVME162FX EIA-485/EIA-422 Connections

Using Interrupts on the MVME162FX

A

Introduction

This appendix demonstrates how to use interrupts on the MVME162FX. It gives an example of how to generate and handle a VMEchip2 Tick Timer 1 interrupt on a MVME162FX that has a VMEbus connection. Specific values have been given for the register writes.

Read this entire appendix before performing any of these procedures.

VMEchip2 Tick Timer 1 Periodic Interrupt Example

A. 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 - **Bclock** (MHz). This gives a 1 MHz clock to the tick timers. **Bclock** 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 = **Period** (s). For example, if you want an interrupt every millisecond, set the register value to 1000 ($3E8). Refer to the *Tick Timer 1 Compare Register* description in Chapter 2.
3.	Tick Timer 1 Counter Register $FFF40054	Write a zero to clear.

Step	Register and Address	Action and Reference
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.

B. Set up local bus interrupter.

Step	Register and Address	Action and Reference
5.	Vector Base Register $FFF40088 (8 of 32 bits)	If not already initialized by the debugger, set interrupt base register 0 by writing to bits 28-31. Refer to the *Vector Base Register* description and to Table 2-3. *Local Bus Interrupter Summary*, 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 *all* interrupts from the VMEchip2.

Periodic Tick Timer 1 interrupts now occur, so you need an interrupt handler. Section C gives the details, as follows.

C. How to set up an interrupt handler routine.

Step	Action and Reference
	Your interrupt handler should include the following features.
1.	Be sure 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 the proper exception vector location to set from the MC68040 vector base register, the VMEchip2 base register, and Table 2-3, the Local Bus Interrupter Summary, 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 *interrupt handler itself* 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.

Index

Numerics

28F008SA 3-3
28F008SA data sheet vii
53C710 data sheet vii
53C710 interface, MC2 chip 3-5
82596CA 3-3, 3-33, 3-34
82596CA data sheet vii

A

A16/D16 space 2-6, 2-39
A16/D32 space 2-6, 2-39
A24/D16 space 2-6, 2-40, 2-51
A32/D16 space 2-6, 2-40, 2-51
ABORT Switch Interrupt Control Register, MC2 chip 3-43
ABORT switch interrupt, address 1-8
AC fail interrupter 2-20
access time, Flash memory 3-42
access timer 3-9
  VMEbus 2-7
Access/Watchdog Time Base Select Register, MC2 chip 3-46
ACFAIL signal line 2-20, 2-99
adder 2-34, 2-35, 2-38
adders, VMEchip2 2-29
address
  DRAM, SRAM 3-26
  GCSR, VMEchip2 1-37, 2-103
  LCSR, VMEchip2 2-22
  MC2 chip CSR 3-11
  VMEbus resources 2-39
address counter, VMEbus 2-14
address modifier 2-35, 2-38
  select bits 2-36
address modifier codes 2-45, 2-46, 2-59
address modifier register 2-40
address modifier select bits 2-38
address range
  devices 1-8
  local bus 2-41
address references iv
address space, decoding 1-10
address translation address register 2-29, 2-41
address translation registers 2-29, 2-40
address translation select register 2-29, 2-41
addressing capabilities 2-6, 2-10, 2-11
addressing, local bus to IP 4-47
alternate address register 2-10
ampersand, definition iv
arbiter
  time-out timer, VMEbus 2-66
  VMEbus 2-18
arbitration modes 2-18
assertion, definition iv
asterisk, definition iv
attribute register 2-30
  snoop bits 2-30

B

base address
  MC2 chip registers 3-9
  VMEchip2 GCSR 2-105
base address, LCSR 2-22
battery backup 1-3
BBRAM configuration area memory map 1-32
BBRAM interface, MC2 chip 3-1, 3-3
BBRAM memory map 1-31
BBRAM speed 3-12
BBSY*, VMEbus 2-101
BERR 2-19
BGIN filters, VMEbus 2-100
binary representation iv
block access cycles 2-36, 2-38, 2-39
block diagram
  MVME162FX 1-5
  VMEchip2 2-5
block transfer
  cycles 2-12
  mode 2-10
  modes, DMAC 2-59
board address, GCSR 2-48
Board Control Register 2-73, 2-104

board failure 2-73
board ID 1-35
Board Status/Control Register, VMEchip2 2-109
BRDFAIL signal pin 2-73, 2-74
broadcast interrupt function 2-16
broadcast mode 2-17
BSY signal 2-18
Bus Clock Register, MC2 chip 3-40
bus error 1-39
bus error condition 3-34
bus error handler 1-40
bus error processing 1-40
bus map decoder, LCSR 2-22
bus sizing 2-7
bus timer enable/disable 2-19
bus timers, example of use 1-48
byte counter, DMAC 2-61
byte, definition iv

C

cache coherency 1-38
  IP2 chip 4-2
cache inhibit function 1-8
CAS instruction 1-49
cautions 2-104
checksum 1-37
chip arbiter 2-18
chip ID and revision registers 2-104
Chip ID Register, IP2 chip 4-17
Chip Revision Register, IP2 chip 4-17
clear-on-compare mode 2-16
clock frequency 3-40
clock timing functions 3-50
clocks
  VMEchip2 counters and timers 2-69
command chaining mode, DMAC 2-12, 2-52
command packets, DMAC 2-53
configuration bytes data structure 1-34
connection diagrams, serial ports 5-1
control bit, definition iv
control registers, tick timer 3-19
conventions used in manual iv

D

data access cycles 2-36, 2-38
data sheets, chip vii
data transfer capabilities 2-6, 2-10, 2-12
data transfer size 2-12
data transfers 2-45, 2-46, 2-52
DCE/DTE serial port configuration 5-1
decimal representation iv
decoders
  programmable 2-6
  VMEchip2 2-28
devices, normal address range 1-8
DFAIR bit 2-15
direct mode, DMAC 2-52
DMA Arbitration Control Register, IP2 chip 4-30
DMA Byte Counter, IP2 chip 4-42
DMA control and status register set definition 4-34
DMA Control Register 1, IP2 chip 4-38
DMA Control Register 2, IP2 chip 4-40
DMA Controller (DMAC) 2-11, 2-52
DMA enable function 4-33
DMA Enable Register, IP2 chip 4-37
DMA IndustryPack Address Counter, IP2 chip 4-42
DMA Interrupt Control Register, IP2 chip 4-36
DMA Local Bus Address Counter, IP2 chip 4-41
DMA Status Register, IP2 chip 4-35
DMA Table Address Counter, IP2 chip 4-43
DMA transfers 2-13
DMAC byte counter 2-61
DMAC command packets 2-53
DMAC Control Register 1 (bits 0-7) 2-56
DMAC Control Register 2 (bits 0-7) 2-59
DMAC Control Register 2 (bits 8-15) 2-57
DMAC interrupter 2-21
DMAC local bus address counter 2-60
DMAC LTO error 1-44
DMAC off-board error 1-44
DMAC parity error 1-43
DMAC Registers 2-53
DMAC Status Register 2-65
DMAC TEA, cause unidentified 1-45
DMAC VMEbus address counter 2-61
DMAC VMEbus error 1-43
DMAC VMEbus requester 2-14
documentation, related vi

dollar sign, definition iv
DRAM Control Register, MC2 chip 3-47
DRAM controller 3-1
DRAM enable 3-47
DRAM memory controller, MC2 chip 3-6
DRAM Parity Error Interrupt Control Register, MC2 chip 3-23
DRAM size control bit encoding 3-28, 3-29
DRAM Space Base Address Register 3-26
DRAM Space Size Register 3-28
DRAM, SRAM Memory Controller Registers, MC2 chip 3-26
DRAM/SRAM Options Register 3-29, 3-36, 3-37
DTACK 2-10
DWB pin 2-9

E

edge significant, definition iv
edge-sensitive interrupters 2-20
edge-sensitive interrupts 2-77
EIA-232-D SIMs 5-1
EIA-530 SIMs 5-1
ending address register 2-29, 2-40
EPROM access time 3-41
EPROM Access Time Control Register, MC2 chip 3-41
EPROM interface 3-1
EPROM socket 1-3
errata sheets, chip 1-14
error condition, local bus 3-32
error conditions 1-41
error indication, LANC 3-5
error reporting
  as a local bus slave 4-7
  IndustryPack 4-7
error reporting as a local bus master 4-7
error reporting, IP2 chip 4-7
error sources, local 1-39
error status, SCSI 3-35
Ethernet address 1-35
example
  generating tick timer periodic interrupt A-1
  IP 16-bit memory space 4-48
  IP 32-bit I/O space 4-51
  IP 32-bit memory space 4-49
  IP 8-bit memory space 4-47
  IP I/O space 4-50
  IP ID space 4-52
  setting up interrupt handler routine A-3
  setting up local bus interrupter A-2
  using bus timers 1-48
extended access cycles 2-37, 2-39

F

fair mode, VMEchip2 2-9, 2-15
false, definition iv
features
  IP2 chip 4-1
  MC2 chip 3-1
  MVME162FX 1-3
  VMEchip2 2-1
Flash and EPROM interface, MC2 chip 3-2
Flash and EPROM memory map locations 3-2
Flash Parameter Register, MC2 chip 3-42
functional blocks, VMEchip2 2-4
functional description
  IP2 chip 4-2
  MC2 chip 3-2
  MVME162FX 1-5

G

GCSR base address registers, programming 2-39
GCSR board address 2-48
GCSR group address 2-48
GCSR map decoder 1-37
GCSR programming model 2-103
GCSR SIG3-0 interrupters 2-21
GCSR, programming 2-105
GCSR, VMEchip2 2-21, 2-103
General Control Register, MC2 chip 3-12
General Control Registers, IP2 chip 4-25
general description 4-2
General Purpose I/O pins 2-99
General Purpose Inputs Register, MC2 chip 3-36, 3-37
General Purpose Register 0 2-110
General Purpose Register 1 2-110
General Purpose Register 2 2-111
General Purpose Register 3 2-111
General Purpose Register 4 2-112

General Purpose Register 5 2-112
general purpose registers 2-104
generate clock for tick timers 3-18
Global Control and Status Registers (GCSR) 2-22, 2-103
global reset 2-19
global reset driver 2-19
global time-out timer, VMEbus 2-67
GPI inputs, addresses 1-8
GPI3 1-10, 3-2
group address, GCSR 2-48

H

hexadecimal representation iv

I

I/O and ID space accesses, IP 4-55
I/O Control Register 1 2-99
I/O Control Register 2 2-100
I/O Control Register 3 2-100
I/O interfaces 1-4
I/O map decoders 2-6, 2-39, 2-41
I/O memory maps 1-13
I/O space
  32-bit IP_ab 4-51
  IP_a 4-50
IACK cycle 2-21
IACK daisy-chain 2-17
IACK daisy-chain driver 2-19
ID number, MC2 chip 3-11
ID Register
  MC2 chip 3-11
  VMEchip2 2-107
ID space, IP 4-52
indivisible cycles, MC68040 1-49
IndustryPack addressing 4-47
IndustryPack error reporting 4-7
IndustryPack ID 1-36
IndustryPack interface 4-1
IndustryPack Interface Controller (IP2 chip) 4-1
IndustryPack specification vii
initialization, MC2 chip 3-2
interrupt acknowledge map 1-37
interrupt base vectors 2-98
interrupt control register
  VMEchip2 2-104
interrupt control registers
  tick timer, MC2 chip 3-21
interrupt counter, DMAC 2-64
interrupt handler
  routine, how to set up A-3
  VMEbus 2-17
  VMEchip2 2-20
interrupt level
  LANC 3-33, 3-34
  SCC 3-24
  SCSI 3-38
Interrupt Level Register 4 (bits 0-7) 2-98
interrupt logic, ABORT switch 3-43
interrupt priority, MC2 chip 3-15
interrupt sources, VMEchip2 2-20
interrupt status bit 2-80
Interrupt Vector Base Register, MC2 chip 3-14
interrupt vectors 1-38
interrupter, VMEbus 2-62
interrupts
  broadcast 2-16, 2-18
  edge-sensitive 2-77
  hardware-vectored 1-38
  how to use A-1
  IP2 chip 4-8
  masked 2-99
introduction
  interrupts, MVME162FX A-1
  MC2 chip 3-1
  MVME162FX 1-1
  serial port connections 5-1
  VMEchip2 2-1
IP Clock Register, IP2 chip 4-2, 4-29, 4-30
IP RESET Register, IP2 chip 4-31
IP to local bus data routing 4-53
IP_a/IP_ab Memory Base Address Registers 4-20
IP_b Memory Base Address Registers 4-20
IP_c/IP_cd Memory Base Address Registers 4-21
IP_d Memory Base Address Registers 4-21
IP2 chip 4-1
  features 4-1
  functional description 4-2
  introduction 4-1
  IP to local bus data routing 4-53

local bus to IndustryPack addressing 4-47
overall memory map 4-9
programming model 4-10
IP2 chip overall memory map 1-22
IRQ0, IRQ1 Interrupt Control Registers, IP2 chip 4-23
IRQ1 edge-sensitive interrupter 2-20
IRQ7-1 interrupters 2-21

J

J22 3-2
jumper header J22 1-7, 3-2

L

LAN coprocessor interface 3-1
LAN interface, MC2 chip 3-3
LAN LTO error 1-46
LAN off-board error 1-46
LAN parity error 1-45
LANC
bus error 3-5
interrupt 3-5
termination 3-5
LANC Bus Error Interrupt Control Register, MC2 chip 3-34
LANC Error Status Register, MC2 chip 3-32
LANC Interrupt Control Register, MC2 chip 3-33
LCSR base address 2-22
LCSR memory map 2-24
LCSR programming model 2-22
LCSR, VMEchip2 2-22
LED, VME 2-101
level significant, definition iv
LM/SIG Register, VMEchip2 2-107
local BERR* 1-39
local bus
access timer 3-1
accesses 1-37
address counter, DMAC 2-60
address range 2-41
base address, GCSR 2-103
error condition 3-32, 3-35
interrupt filters 2-100
interrupter 2-13
interrupter summary 2-78
interrupter, how to set up A-2
interrupter, programming 2-77
interrupter, VMEchip2 2-20
map decoder registers 2-40
master 2-9, 2-11
memory map 1-8
reset 2-109
slave 2-6
time-out 1-39, 3-9
time-out value 2-68, 3-46
timer 2-19
timer, MC2 chip 3-9
Local Bus Interrupter Status Register (bits 16-23) 2-81
Local Bus Interrupter Status Register (bits 24-31) 2-80
Local Bus Slave (VMEbus Master) Address Translation Address Register 4 2-44
Local Bus Slave (VMEbus Master) Address Translation Select Register 4 2-45
Local Bus Slave (VMEbus Master) Attribute Register 1 2-47
Local Bus Slave (VMEbus Master) Attribute Register 2 2-46
Local Bus Slave (VMEbus Master) Attribute Register 3 2-46
Local Bus Slave (VMEbus Master) Attribute Register 4 2-45
Local Bus Slave (VMEbus Master) Ending Address Register 1 2-42
Local Bus Slave (VMEbus Master) Ending Address Register 2 2-42
Local Bus Slave (VMEbus Master) Ending Address Register 3 2-43
Local Bus Slave (VMEbus Master) Ending Address Register 4 2-44
Local Bus Slave (VMEbus Master) Starting Address Register 1 2-42
Local Bus Slave (VMEbus Master) Starting Address Register 2 2-43
Local Bus Slave (VMEbus Master) Starting Address Register 3 2-43
Local Bus Slave (VMEbus Master) Starting Address Register 4 2-44
local bus to IndustryPack addressing 4-47
local bus to VMEbus

INDEX

DMA controller, VMEchip2 2-11
interface 2-4
interface, VMEchip2 2-4
map decoders, programming 2-39
requester 2-8
requester register, programming 2-52
Local Bus to VMEbus Enable Control Register 2-49
Local Bus to VMEbus I/O Control Register 2-50
Local Bus to VMEbus Requester Control Register 2-55
Local Control and Status Registers (LCSR) 2-8, 2-22
local DRAM parity error 1-40
local reset 2-19, 2-73
local reset driver 2-19
location monitor interrupters 2-21
location monitor status register 2-104
location monitors LM0-LM3 2-103
longword, definition iv
LSB Prescaler Count Register 3-18
LVFAIR bit 2-9

M

map decoder 2-10
user-programmable 1-37, 2-6
master interrupt enable (MIEN) 2-77, 2-99, 3-13
MC2 chip 3-1
functional description 3-2
introduction 3-1
memory map 3-9
programming model 3-11
MC2 chip Revision Register 3-12
MC2 chip/VMEchip2 redundancies 1-6
MC68040
bus master support for 82596CA 3-4
indivisible cycles 1-49
indivisible RMW memory accesses 1-49
Memory Base Address Registers, IP2 chip 4-19
Memory Controller ASIC (MC2 chip) 3-1
memory map
BBRAM configuration area 1-32
BBRAM, TOD clock 1-31, 1-32
Ethernet LAN 1-30
IP2 chip devices 4-9
IP2 chip, all devices 1-22
IP2 chip, Control and Status Registers 1-23, 4-11
local bus 1-8, 1-9
local bus I/O devices 1-11
MC2 chip 1-21, 3-9
SCSI 1-31
time-of-day clock 1-33
VMEbus 1-37
VMEchip2 GCSR 1-20, 2-106
VMEchip2 LCSR 1-15, 2-24
Z85230 SCC register 1-30
memory maps, MVME162FX 1-8
memory mezzanine board serial number 1-36
memory size 3-28
SRAM 3-31
Memory Size Registers, IP2 chip 4-22
memory space
16-bit IP_a 4-48
32-bit IP_ab 4-49
8-bit IP_a 4-47
memory space accesses, IP 4-53
microprocessor 1-3
MIEN 2-77, 2-99, 3-13
Miscellaneous Control Register 2-100
MK48T08 1-32
MK48T08 data sheet vii
models, MVME162FX iii
MPU local bus time-out 1-42
MPU off-board error 1-41
MPU parity error 1-41
MPU port and MPU channel attention 3-3
MPU Status and DMA Interrupt Count Register 2-64
MPU Status Register
MC2 chip 3-49
VMEchip2 2-52
MPU TEA, cause unidentified 1-42
MVME162Bug 1-7, 3-2
MVME162FX
block diagram 1-5
functional description 1-5
introduction 1-1
memory maps 1-8
models iii
overview 1-1
software support considerations 1-38

MVME162FX Version Register, MC2 chip 3-37
MVME712X 1-2
MVME712x 5-1

N

negation, definition iv
no address increment DMA transfers 2-13
non-privileged access cycles 2-37, 2-39
NVRAM memory map 1-31

O

overflow counter 2-76
  MC2 chip 3-20
  VMEchip2 2-75
overview
  MC2 chip 3-2
  MVME162FX 1-1

P

pacer clock
  programming 4-44
Pacer Clock General Control Register, IP2 chip 4-45
Pacer Clock Interrupt Control Register, IP2 chip 4-44
Pacer Clock Timer Register, IP2 chip 4-46
PAL contents 3-36, 3-37
parity
  checking 3-6
  checking mode 3-47
  control and interrupt logic 3-23
  error interrupts 3-23
percent, definition iv
periodic interrupt example A-1
power monitor 2-18
powerup reset
  MC2 chip 3-44
  VMEchip2 2-73
Prescaler Clock Adjust Register 3-18
Prescaler Control Register 2-69
Prescaler Count Register, MC2 chip 3-50
Prescaler Counter 2-76
prescaler, VMEchip2 2-15
priority
  interrupt 1-38, 3-14
Priority (PRI) mode 2-18
processor clock 3-18
program access cycles 2-36, 2-38
program address modifier code 2-50
programmable map decoders 2-6, 2-39
programming
  DMA controller, VMEchip2 2-52
  GCSR, VMEchip2 2-105
  LCSR, VMEchip2 2-22
  local bus interrupter, VMEchip2 2-77
  local bus to VMEbus map decoders, VMEchip2 2-39
  tick and watchdog timers, VMEchip2 2-66
  tick timers, MC2 chip 3-16
  VMEbus slave map decoders, VMEchip2 2-28
programming model
  IP2 chip 4-10
  MC2 chip 3-11
  VMEchip2 GCSR 2-103
  VMEchip2 LCSR 2-22
programming the DMA controllers 4-31
programming the pacer clock 4-44
PROM Decoder, SRAM and DMA Control Register 2-54
PWB number 1-35

R

redundant functions, VMEchip2/MC2 chip 1-6
refresh rate 3-40
register definitions, LCSR 2-22
register map, MC2 chip 3-10
registers
  local bus map decoders 2-40
  VMEbus slave map decoder 2-28
release-on-acknowledge (ROAK) mode 2-17
release-on-request (ROR) mode 2-9
release-when-done (RWD) mode 2-9
reset drivers 2-19
reset status
  IP2 chip 4-10
  MC2 chip 3-11
RESET switch
  enable/disable 2-73
RESET Switch Control Register, MC2 chip 3-44
Revision Register
  MC2 chip 3-12

INDEX

VMEchip2 2-107
ROM Control Register 2-51
ROM0 bit 2-54
Round Robin Select (RRS) mode 2-18

S

SCC interface, MC2 chip 3-1, 3-7
SCC Interrupt Control Register, MC2 chip 3-24
SCSI controller interface, MC2 chip 3-5
SCSI coprocessor interface 3-1
SCSI Error Status Register, MC2 chip 3-35
SCSI ID 1-35
SCSI Interrupt Control Register, MC2 chip 3-38
SCSI LTO error 1-47
SCSI memory map 1-31
SCSI off-board error 1-47
SCSI parity error 1-46
segment size 2-33, 2-34
SERCLK driver 2-18
Serial Interface Modules (SIMs) 5-1
Serial Port 1, MVME162FX 5-1
Serial Port 2, MVME162FX 5-1
Serial Port 2, MVME712x 5-1
Serial Port 4, MVME712x 5-1
serial port connections 5-1
short I/O area 2-6
short I/O map decoder enable 2-50
short I/O memory map 1-37
short I/O segment 2-50
short I/O space 2-103
short I/O space, VMEbus 2-39
signal interrupts SIG0-SIG3 2-103
Single (SGL) mode 2-18
size and architecture, DRAM 3-29
size of SRAM array 3-30
size, segment 2-33, 2-34
slave map decoder registers 2-28
slave map decoders, VMEbus 2-28
snoop 2-37
snoop control 3-34, 4-2
snoop control bits 2-54
snoop control register 2-35
Snoop Control SC1-SC0 3-4
snoop enable 2-30, 2-35, 2-37
snoop signal lines 2-58
snooping, definition 1-38, 2-11
software 7-0 interrupters 2-21
software support considerations, MVME162FX 1-38
specifications 1-3
speed, board 1-35
speeds
  EPROM 3-41
  Flash 3-42
SRAM 1-3
SRAM controller 3-1
SRAM memory controller, MC2 chip 3-6
SRAM size control bit encoding 3-30, 3-31
SRAM Space Base Address Register 3-27
SRAM Space Size Register 3-31
standard access cycles 2-36, 2-39
starting address register 2-29, 2-40
status bit, definition iv
status LEDs 1-4
status register
  DMAC 2-65
  MPU 2-64
supervisor address modifier code 2-50, 2-51
supervisory access 2-39
supervisory access cycles 2-37
support information vi
switches 1-4
SYS fail interrupter 2-20
SYSFAIL signal line 2-20, 2-73, 2-99
SYSRESET 2-19
SYSRESET function 2-17
SYSRESET signal 2-74
system controller 2-73
  enable/disable 2-18
system reset 2-73

T

table address counter 2-62
TAS cycles 1-49
TEA* signal 1-39
terminology used in manual iv
tick timer
  interrupters 2-20
  periodic interrupt example A-1
Tick Timer 1 and 2 Control Registers 3-19
Tick Timer 1 Compare Register 2-70, 3-17
Tick Timer 1 Control Register 2-76

Tick Timer 1 Counter 2-70, 3-17
Tick Timer 2 Compare Register 2-71, 3-17
Tick Timer 2 Control Register 2-75
Tick Timer 2 Counter 2-71, 2-72, 3-17
Tick Timer 3 and 4 Compare and Counter Registers, MC2 chip 3-39
Tick Timer 3 and 4 Control Registers, MC2 chip 3-25
Tick Timer Interrupt Control Registers 3-21
tick timers 1-3, 3-1
  clock 3-18
  clock frequency 3-18
  MC2 chip 3-8, 3-16
  VMEchip2 2-15
time off / time on timers, DMAC 2-67
time-of-day clock 1-3
  interface 3-1
  memory map 1-31, 1-32, 1-33
time-out
  local bus 1-39, 3-9
  period 2-19
  VMEbus access 1-39
time-out period
  watchdog 2-68, 3-46
timers, VMEbus 2-7
transfer mode, VMEbus 2-13
Transfer Modifiers TM2-TM0 3-4
Transfer Type (TT) signals 1-8
Transfer Types TT1-TT0 3-4
transition board 1-2
transition modules 5-1
true, definition iv

V

Vector Base Register 2-98
Vector Base Register, IP2 chip 4-18
vector base registers 2-77
Version Register 3-36, 3-37
VME Access, Local Bus, and Watchdog Time-out Control Register 2-68
VME LED 2-101
VMEbus access time-out 1-39
VMEbus access time-out value 2-68
VMEbus address counter, DMAC 2-61
VMEbus Arbiter Time-out Control Register 2-66
VMEbus BBSY* 2-101
VMEbus BERR* 1-39
VMEbus capabilities 2-6, 2-10, 2-11
VMEbus global time-out timer 2-67
VMEbus Interface 1-4, 1-7
VMEbus Interface, "no" option 1-6, 3-9, 3-45, 3-49, 3-50
VMEbus interrupter
  acknowledge interrupter 2-20
  programming 2-52
  VMEchip2 2-17
VMEbus Interrupter Control Register 2-62
VMEbus Interrupter Vector Register 2-63
VMEbus IRQ1, IRQ2 interrupt 2-98
VMEbus mapping 1-37
VMEbus maps, creating 2-6
VMEbus master 2-7
VMEbus request 2-56
VMEbus request level 2-55
VMEbus requester, DMAC 2-14
VMEbus slave 2-9
VMEbus Slave Address Modifier Select Register 1 2-38
VMEbus Slave Address Modifier Select Register 2 2-36
VMEbus Slave Address Translation Address Offset Register 1 2-32
VMEbus Slave Address Translation Address Offset Register 2 2-34
VMEbus Slave Address Translation Select Register 1 2-33
VMEbus Slave Address Translation Select Register 2 2-34
VMEbus Slave Ending Address Register 1 2-31
VMEbus Slave Ending Address Register 2 2-31
VMEbus Slave GCSR Group Address Register 2-48
VMEbus slave map decoders 2-28
  programming 2-28
VMEbus Slave Starting Address Register 1 2-31
VMEbus Slave Starting Address Register 2 2-32
VMEbus Slave Write Post and Snoop Control Register 1 2-37
VMEbus Slave Write Post and Snoop Control Register 2 2-35
VMEbus specification vii

VMEbus system controller, VMEchip2 2-18
VMEbus timer 2-19
VMEbus to local bus interface 2-9
VMEchip2
  block diagram 2-5
  functional blocks 2-4
  GCSR programming model 2-103
  introduction 2-1
  local BERR* 1-40
  memory map, LCSR Summary 2-24
  periodic interrupt example A-1
  programming model 2-22
VMEchip2 Board Status/Control
    Register 2-109
VMEchip2 ID Register 2-107
VMEchip2 LM/SIG Register 2-107
VMEchip2 Revision Register 2-107
VMEchip2/MC2 chip redundancies 1-6

W

warnings v
watchdog timer 1-4, 3-1
  control logic, MC2 chip 3-45
  MC2 chip 3-8, 3-46
  VMEchip2 2-15, 2-17
Watchdog Timer Control Register
  MC2 chip 3-45
  VMEchip2 2-71
word, definition iv
write post 2-37
  buffer 2-7, 2-9
  bus error interrupter 2-20
  register 2-35
  timer 2-8
write post enable 2-35, 2-37, 2-41, 2-45, 2-46,
    2-50
write posting 2-7, 2-41
  definition 2-7
  enable 2-51
  operations 2-10
write-protect feature 3-3

Z

Z85230 data sheet vii
Z85230 interface, MC2 chip 3-7
Z85230 SCC Register addresses 1-30