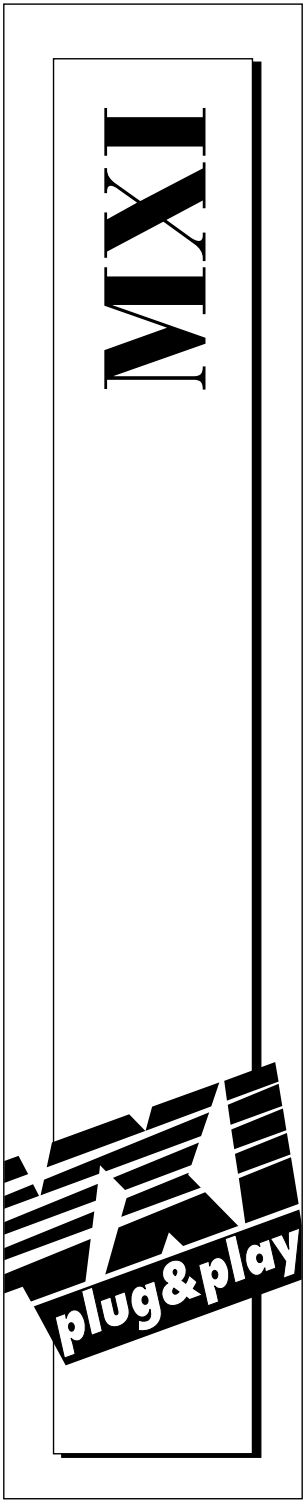


VME-MXI-2
User Manual

January 1996 Edition
Part Number 321071A-01

Internet Support

GPIB: gpib.support@natinst.com
DAQ: daq.support@natinst.com
VXI: vxi.support@natinst.com
LabVIEW: lv.support@natinst.com
LabWindows: lw.support@natinst.com
HiQ: hiq.support@natinst.com
VISA: visa.support@natinst.com

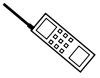
FTP Site: ftp.natinst.com
Web Address: www.natinst.com

Bulletin Board Support

BBS United States: (512) 794-5422 or (800) 327-3077
BBS United Kingdom: 01635 551422
BBS France: 1 48 65 15 59

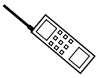
FaxBack Support

(512) 418-1111 or (800) 329-7177



Telephone Support (U.S.)

Tel: (512) 795-8248
Fax: (512) 794-5678 or (800) 328-2203



International Offices

Australia 03 9 879 9422, Austria 0662 45 79 90 0, Belgium 02 757 00 20,
Canada (Ontario) 519 622 9310, Canada (Québec) 514 694 8521, Denmark 45 76 26 00,
Finland 90 527 2321, France 1 48 14 24 24, Germany 089 741 31 30, Hong Kong 2645 3186,
Italy 02 48301892, Japan 03 5472 2970, Korea 02 596 7456, Mexico 95 800 010 0793,
Netherlands 0348 433466, Norway 32 84 84 00, Singapore 2265886, Spain 91 640 0085,
Sweden 08 730 49 70, Switzerland 056 200 51 51, Taiwan 02 377 1200, U.K. 01635 523545

National Instruments Corporate Headquarters

6504 Bridge Point Parkway Austin, TX 78730-5039 Tel: (512) 794-0100

Important Information

Warranty

The VME-MXI-2 is warranted against defects in materials and workmanship for a period of one year from the date of shipment, as evidenced by receipts or other documentation. National Instruments will, at its option, repair or replace equipment that proves to be defective during the warranty period. This warranty includes parts and labor.

The media on which you receive National Instruments software are warranted not to fail to execute programming instructions, due to defects in materials and workmanship, for a period of 90 days from date of shipment, as evidenced by receipts or other documentation. National Instruments will, at its option, repair or replace software media that do not execute programming instructions if National Instruments receives notice of such defects during the warranty period. National Instruments does not warrant that the operation of the software shall be uninterrupted or error free.

A Return Material Authorization (RMA) number must be obtained from the factory and clearly marked on the outside of the package before any equipment will be accepted for warranty work. National Instruments will pay the shipping costs of returning to the owner parts which are covered by warranty.

National Instruments believes that the information in this manual is accurate. The document has been carefully reviewed for technical accuracy. In the event that technical or typographical errors exist, National Instruments reserves the right to make changes to subsequent editions of this document without prior notice to holders of this edition. The reader should consult National Instruments if errors are suspected. In no event shall National Instruments be liable for any damages arising out of or related to this document or the information contained in it.

EXCEPT AS SPECIFIED HEREIN, NATIONAL INSTRUMENTS MAKES NO WARRANTIES, EXPRESS OR IMPLIED, AND SPECIFICALLY DISCLAIMS ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. CUSTOMER'S RIGHT TO RECOVER DAMAGES CAUSED BY FAULT OR NEGLIGENCE ON THE PART OF NATIONAL INSTRUMENTS SHALL BE LIMITED TO THE AMOUNT THERETOFORE PAID BY THE CUSTOMER. NATIONAL INSTRUMENTS WILL NOT BE LIABLE FOR DAMAGES RESULTING FROM LOSS OF DATA, PROFITS, USE OF PRODUCTS, OR INCIDENTAL OR CONSEQUENTIAL DAMAGES, EVEN IF ADVISED OF THE POSSIBILITY THEREOF. This limitation of the liability of National Instruments will apply regardless of the form of action, whether in contract or tort, including negligence. Any action against National Instruments must be brought within one year after the cause of action accrues. National Instruments shall not be liable for any delay in performance due to causes beyond its reasonable control. The warranty provided herein does not cover damages, defects, malfunctions, or service failures caused by owner's failure to follow the National Instruments installation, operation, or maintenance instructions; owner's modification of the product; owner's abuse, misuse, or negligent acts; and power failure or surges, fire, flood, accident, actions of third parties, or other events outside reasonable control.

Copyright

Trademarks

NI-VXI™ is a trademark of National Instruments Corporation.

Product and company names listed are trademarks or trade names of their respective companies.

WARNING REGARDING MEDICAL AND CLINICAL USE OF NATIONAL INSTRUMENTS PRODUCTS

National Instruments products are not designed with components and testing intended to ensure a level of reliability suitable for use in treatment and diagnosis of humans. Applications of National Instruments products involving medical or clinical treatment can create a potential for accidental injury caused by product failure, or by errors on the part of the user or application designer. Any use or application of National Instruments products for or involving medical or clinical treatment must be performed by properly trained and qualified medical personnel, and all traditional medical safeguards, equipment, and procedures that are appropriate in the particular situation to prevent serious injury or death should always continue to be used when National Instruments products are being used. National Instruments products are NOT intended to be a substitute for any form of established process, procedure, or equipment used to monitor or safeguard human health and safety in medical or clinical treatment.

FCC/DOC Radio Frequency Interference Compliance

This equipment generates and uses radio frequency energy and, if not installed and used in strict accordance with the instructions in this manual, may cause interference to radio and television reception. This equipment has been tested and found to comply with the following two regulatory agencies:

Federal Communications Commission

This device complies with Part 15 of the Federal Communications Commission (FCC) Rules for a Class A digital device. Operation is subject to the following two conditions:

1. This device may not cause harmful interference in commercial environments.
2. This device must accept any interference received, including interference that may cause undesired operation.

Canadian Department of Communications

This device complies with the limits for radio noise emissions from digital apparatus set out in the Radio Interference Regulations of the Canadian Department of Communications (DOC).

Le présent appareil numérique n'émet pas de bruits radioélectriques dépassant les limites applicables aux appareils numériques de classe A prescrites dans le règlement sur le brouillage radioélectrique édicté par le ministère des communications du Canada.

Instructions to Users

These regulations are designed to provide reasonable protection against harmful interference from the equipment to radio reception in commercial areas. Operation of this equipment in a residential area is likely to cause harmful interference, in which case the user will be required to correct the interference at his own expense.

There is no guarantee that interference will not occur in a particular installation. However, the chances of interference are much less if the equipment is installed and used according to this instruction manual.

If the equipment does cause interference to radio or television reception, which can be determined by turning the equipment on and off, one or more of the following suggestions may reduce or eliminate the problem.

- Operate the equipment and the receiver on different branches of your AC electrical system.
- Move the equipment away from the receiver with which it is interfering.
- Reorient or relocate the receiver's antenna.
- Be sure that the equipment is plugged into a grounded outlet and that the grounding has not been defeated with a cheater plug.

Notice to user: ***Changes or modifications not expressly approved by National Instruments could void the user's authority to operate the equipment under the FCC Rules.***

If necessary, consult National Instruments or an experienced radio/television technician for additional suggestions. The following booklet prepared by the FCC may also be helpful: *How to Identify and Resolve Radio-TV Interference Problems.* This booklet is available from the U.S. Government Printing Office, Washington, DC 20402, Stock Number 004-000-00345-4.

Table of Contents

About This Manual

Chapter 1
Introduction

Chapter 2
Functional Overview

Chapter 3
VME-MXI-2 Configuration and Installation

Chapter 4
Register Descriptions

Chapter 5
System Configuration

Chapter 6
VXI*plug&play* for the VME-MXI-2

Figures

Tables

About This Manual

The *VME-MXI-2 User Manual* describes the functional, physical, and electrical aspects of the VME-MXI-2 and contains information concerning its operation and programming.

Organization of This Manual

The *VME-MXI-2 User Manual* is organized as follows:

- Chapter 1, *Introduction*, describes the VME-MXI-2, lists what you need to get started, lists optional equipment, and introduces the concepts of MXI-2.
- Chapter 2, *Functional Overview*, contains functional descriptions of each major logic block on the VME-MXI-2.
- Chapter 3, *VME-MXI-2 Configuration and Installation*, contains the instructions to configure and install the VME-MXI-2 module.
- Chapter 4, *Register Descriptions*, contains detailed information on some of the VME-MXI-2 registers, which you can use to configure and control the module's operation.
- Chapter 5, *System Configuration*, explains important considerations for programming and configuring a VMEbus/MXIbus system using VME-MXI-2 mainframe extenders.
- Chapter 6, *VXIplug&play for the VME*-MXI-2, describes the contents of the VXI*plug&play* disk that came with your VME-MXI-2 kit. The disk contains a VXI*plug&play* soft front panel and a VXI*plug&play* knowledge base file.

- Appendix A, *Specifications*, lists various module specifications of the VME-MXI-2, such as physical dimensions and power requirements.

- Appendix B, *Programmable Configurations*, describes some features of the VME-MXI-2 that are configured by programming an onboard EEPROM through software rather than by onboard switches or jumpers.

- Appendix C, *VME-MXI-2 Front Panel Configuration*, describes the front panel and connectors on the VME-MXI-2 interface module. This material contains the information relevant to VXI*plug&play* Specification VPP-8, *VXI Module/Mainframe to Receiver Interconnection*.

- Appendix D, *Differences and Incompatibilities between the VME-MXI and the VME-MXI-2*, describes the differences and incompatibilities between the first-generation MXIbus-to-VMEbus interface, the VME-MXI, and the VME-MXI-2. This information may be helpful for users of the VME-MXI who are moving to the VME-MXI-2.

- Appendix E, *Configuring a Two-Frame System*, describes how to configure a system containing two mainframes linked by VME-MXI-2 mainframe extenders.

- Appendix F, *DMA Programming Examples*, contains two example programs for using the DMA controllers on the VME-MXI-2. If you are using a version of the National Instruments NI-VXI software that has remote DMA controller functionality, this information is not necessary because you can make use of the VME-MXI-2 module's DMA controllers from the NI-VXI high-level function calls.

- Appendix G, *Mnemonics Key*, contains an alphabetical listing of all mnemonics used in this manual to describe signals and terminology specific to MXIbus, VMEbus, VXIbus, and register bits. Refer also to the *Glossary*.

- Appendix H, *Customer Communication*, contains forms you can use to request help from National Instruments or to comment on our products and manuals.

- The *Glossary* contains an alphabetical list and description of terms used in this manual, including abbreviations, acronyms, metric prefixes, and symbols.
- The *Index* contains an alphabetical list of key terms and topics in this manual, including the page where you can find each one.

Conventions Used in This Manual

The following conventions are used in this manual:

italic	Italic text denotes emphasis, a cross reference, or an introduction to a key concept.
bold italic	Bold italic text denotes a note, caution, or warning.
`monospace`	Lowercase text in this font denotes text or characters that are to be literally input from the keyboard, sections of code, programming examples, and syntax examples. This font is also used for the proper names of disk drives, paths, directories, programs, subprograms, subroutines, device names, functions, variables, filenames, and extensions, and for statements and comments taken from program code.
`bold monospace`	Bold text in this font denotes the messages and responses that the computer automatically prints to the screen.
<>	Angle brackets enclose the name of a key on the keyboard—for example, <PageDown>.
-	A hyphen between two or more key names enclosed in angle brackets denotes that you should simultaneously press the named keys—for example, <Control-Alt-Delete>.
	Abbreviations, acronyms, metric prefixes, mnemonics, symbols, and terms are listed in the *Glossary*.

How to Use This Manual

If you will be installing your VME-MXI-2 into a system with a VXIbus Multiframe Resource Manager, you only need to read Chapters 1 through 3 of this manual. If you have more than two VME-MXI-2 modules extending your system, you will find useful system configuration information in Chapter 5. Appendix E is a quick

reference for users who have a system containing two mainframes linked by VME-MXI-2 modules. If your system does not have a VXIbus Resource Manager, you can find programming information and descriptions of the VME-MXI-2 hardware in Chapters 4 and 5.

Related Documentation

The following documents contain information that you may find helpful as you read this manual:

- ANSI/IEEE Standard 1014-1987, *IEEE Standard for a Versatile Backplane Bus: VMEbus*
- ANSI/IEEE Standard 1155-1993, *IEEE VMEbus Extensions for Instrumentation: VXIbus*
- ANSI/VITA 1-1994, *VME64*
- *Multisystem Extension Interface Bus Specification*, Version 2.0 (available from National Instruments Corporation)
- VXI-6, *VXIbus Mainframe Extender Specification*, Rev. 1.0, VXIbus Consortium

Customer Communication

National Instruments wants to receive your comments on our products and manuals. We are interested in the applications you develop with our products, and we want to help if you have problems with them. To make it easy for you to contact us, this manual contains comment and configuration forms for you to complete. These forms are in Appendix H, *Customer Communication*, at the end of this manual.

Chapter 1

Introduction

This chapter describes the VME-MXI-2, lists what you need to get started, lists optional equipment, and introduces the concepts of MXI-2.

VME-MXI-2 Overview

The VME-MXI-2 interface is a mainframe extender for the VMEbus. It extends the VMEbus architecture outside a VMEbus mainframe via MXI-2, the second-generation MXIbus (Multisystem eXtension Interface bus). A VMEbus mainframe equipped with a VME-MXI-2 can be connected to other MXIbus devices such as other VMEbus mainframes, MXIbus instruments, or MXIbus-equipped personal computers. The VME-MXI-2 interface module uses address mapping to transparently translate bus cycles on the VMEbus system bus (VMEbus) to the MXIbus and vice versa.

What You Need to Get Started

- ❑ VMEbus mainframe
- ❑ VME-MXI-2 interface module
- ❑ MXI-2 cable
- ❑ VXI*plug&play* diskette

MXI-2 Description

MXI-2 is the second generation of the National Instruments MXIbus product line. The MXIbus is a general-purpose, 32-bit, multimaster system bus on a cable. MXI-2 expands the number of signals on a standard MXI cable by including all VMEbus interrupts, VXIbus triggers, VXIbus CLK10, and all of the VMEbus utility bus signals (SYSFAIL*, SYSRESET*, and ACFAIL*).

Because MXI-2 incorporates all of these new signals into a single connector, the standard VME-MXI-2 can extend the interrupts and utility signals not only to other mainframes, but also to any computers and devices equipped with MXI-2.

In addition, MXI-2 surpasses the data throughput of previous-generation MXIbus products by defining new high-performance protocols. MXI-2 is a superset of MXI. All accesses initiated by MXIbus devices will work with MXI-2 devices. However, MXI-2 defines synchronous MXI block data transfers that surpass previous block data throughput benchmarks. The new synchronous MXI block protocol increases MXI-2 throughput to a maximum of 33 MB/s between two MXI-2 devices. All National Instruments MXI-2 boards can initiate and respond to synchronous MXI block cycles.

☞ **Note:** ***In the remainder of this manual, the term* MXIbus *refers to* MXI-2.**

VME-MXI-2 Description

The VME-MXI-2 module is a double-height, single-width VMEbus device with optional VMEbus System Controller functions. The VME-MXI-2 can automatically determine if it is located in the first slot of a VMEbus chassis and if it is the MXIbus System Controller.

The VME-MXI-2 module's register set is based on VXI-6, *VXIbus Mainframe Extender Specification*, Rev. 1.0. As a result, you can use the VME-MXI-2 with a VXI-MXI-2 to connect VXI mainframes to VMEbus mainframes.

The VME-MXI-2 converts A32, A24, A16, D64, D32, D16, and D08(EO) VMEbus bus cycles into MXIbus bus cycles and vice versa. The VME-MXI-2 has four address windows that map into and out of the VMEbus mainframe. These four windows represent the three

VMEbus address spaces (A32, A24, and A16) plus a dedicated window for mapping the VXIbus configuration space (the upper 16 KB of A16 space).

The MXIbus is a multidrop system bus that connects multiple devices at the hardware bus level in a software-transparent manner. Multiple VMEbus mainframes with VME-MXI-2 interfaces can be connected to form a single multiframe VMEbus system. An external PC with a MXIbus interface can also be connected to a VMEbus mainframe with a VME-MXI-2. This configuration makes the PC function as though it were an embedded VMEbus controller that is plugged into the VMEbus mainframe.

Multiple MXIbus devices are tightly coupled by mapping together portions of each device's address space and locking the internal hardware bus cycles to the MXIbus. The window address circuitry on each MXIbus device monitors internal local bus cycles to detect bus cycles that map across the MXIbus. Similarly, external MXIbus cycles are monitored to detect MXIbus cycles that map into the VMEbus system. MXIbus devices can operate in parallel at full speed over their local system bus and need to synchronize operation with another device only when addressing or being addressed by a resource located on another MXIbus device. The MXIbus device originating the transaction must gain ownership of both the MXIbus and the local bus in the target MXIbus device. All hardware bus cycles are then coupled across the MXIbus and local buses before the transfer completes.

The VME-MXI-2 has the following features:

- Interfaces the VMEbus to the MXIbus (32-bit Multisystem eXtension Interface bus)
- Extends VMEbus to multiple mainframes, external MXIbus-equipped instruments, and external MXIbus-equipped PCs
- Allows multiple VMEbus mainframes to operate as a single VMEbus system
- Supports the VME *RETRY** signal to resolve deadlock conditions
- Supports D64, block, and synchronous MXI cycles for high-performance data transfers
- Two independent DMA controllers for data transfer
- Can extend VMEbus interrupt levels and utility signals to the MXIbus
- Can operate in either one of two modes: parallel or interlocked

- Allows for optional or user-installable onboard DRAM up to 64 MB, which can be shared with the VMEbus and MXIbus
- Conforms to the VMEbus specification
- Conforms to VXI-6, the *VXIbus Mainframe Extender Specification*
- Conforms to the MXI-2 specification
- Supports automatic first slot detection
- Supports automatic MXIbus System Controller detection
- Supports automatic MXIbus termination
- Has no restrictions on physical location of devices

The VME-MXI-2 generates all the support signals required by the VMEbus:

- VMEbus System Controller functions:
  - 16 MHz system clock driver
  - Data transfer bus arbiter (PRI or RR ARBITER)
  - Interrupt acknowledge daisy-chain driver
- VMEbus miscellaneous services:
  - VMEbus timeout (BTO)
  - Pushbutton system reset switch
- VMEbus master capabilities:
  - Access to A16, A24, and A32 address space
  - D08(EO), D16, D32, and D64 accesses
  - Release-on-Request bus requester (programmable bus request level)
  - Optional FAIR VMEbus requester
- VMEbus slave capabilities:
  - A16, A24, and A32 address space
  - D08(EO), D16, D32, and D64 accesses
- VMEbus interrupter
  - ROAK or RORA (programmable)
  - Responds to D16 or D32 IACK cycles

The VME-MXI-2 does not have support for the serial clock driver or power monitor VMEbus modules.

All integrated circuit drivers and receivers used on the VME-MXI-2 meet the requirements of both the VMEbus specification and the MXIbus specification.

Front Panel Features

The VME-MXI-2 has the following front panel features:

- Three front panel LEDs
  - **SYSFAIL** LED indicates that the VMEbus SYSFAIL line is asserted.
  - **MXI** LED indicates when the VME-MXI-2 is accessed from the MXIbus.
  - **VME** LED indicates when the VME-MXI-2 is accessed from the VMEbus.
- MXIbus connector
- System reset pushbutton

Optional Equipment

- Type M1 MXI-2 Cables—
  Straight-point connector to straight-point connector; available in lengths of 1, 2, 4, 8, or 20 m
- Type M2 MXI-2 Cables—
  Straight-point connector to right-angle daisy-chain connector; available in lengths of 1, 2, 4, 8, or 20 m
- Type M3 MXI-2 Cables—
  Right-angle point connector to right-angle daisy-chain connector; available in lengths of 1, 2, 4, 8, or 20 m
- Type M4 MXI-2 Cables—
  Straight-point connector to reverse right-angle daisy-chain connector; available in lengths of 1, 2, 4, 8, or 20 m
- Onboard DRAM options of 4, 8, 16, 32, or 64 MB

Chapter 2

Functional Overview

This chapter contains functional descriptions of each major logic block on the VME-MXI-2.

VME-MXI-2 Functional Description

In the simplest terms, you can think of the VME-MXI-2 as a bus translator that converts VMEbus signals into appropriate MXIbus signals. From the perspective of the MXIbus, the VME-MXI-2 implements a MXIbus interface to communicate with other MXIbus devices. From the perspective of the VMEbus, the VME-MXI-2 is an interface to the outside world.

Figure 2-1 is a functional block diagram of the VME-MXI-2. Following the diagram is a description of each logic block shown.

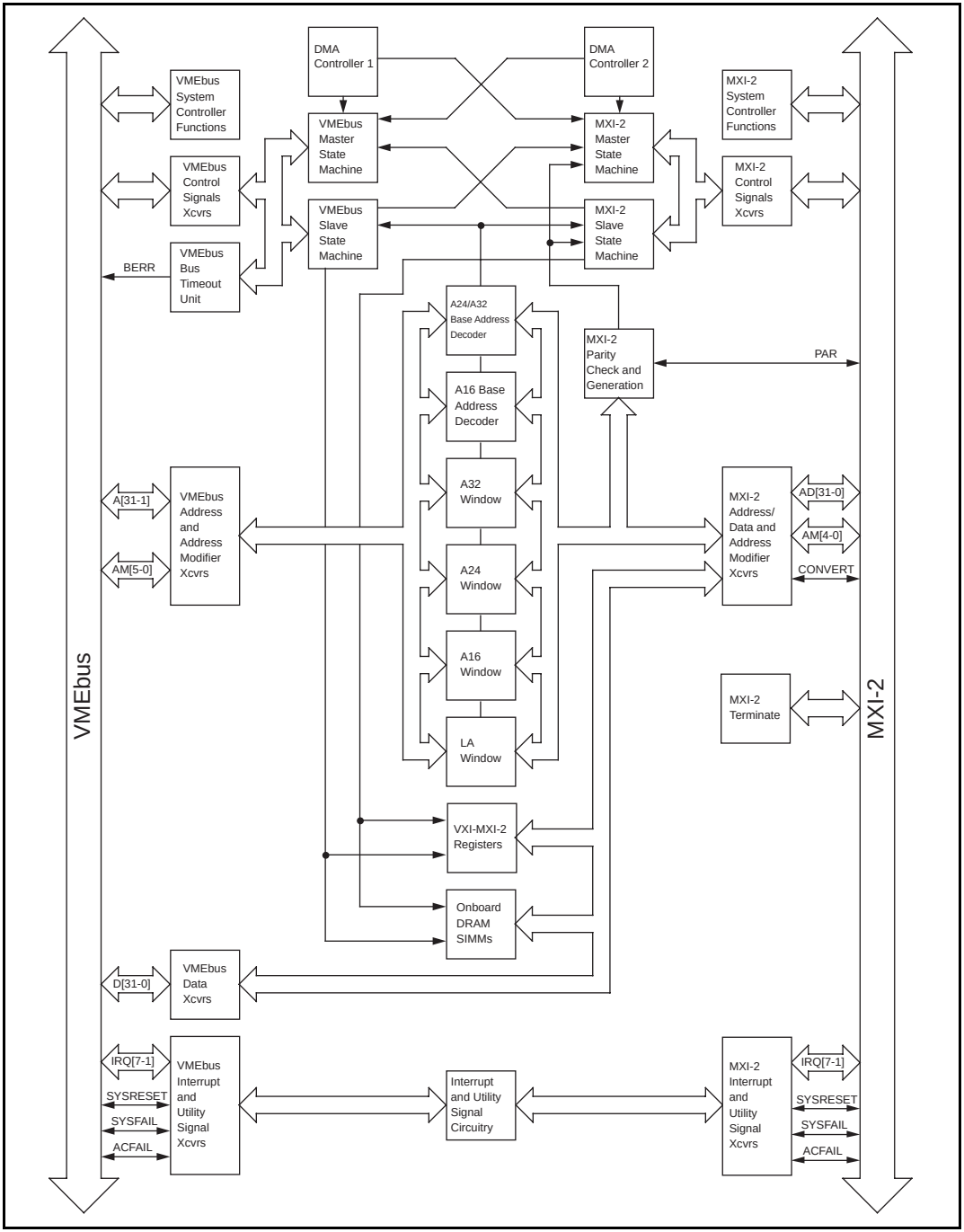


Figure 2-1. VME-MXI-2 Block Diagram

- **VMEbus System Controller Functions** — When the VME-MXI-2 is installed in the first slot of a VMEbus mainframe it assumes the System Controller responsibilities defined in the VMEbus specification. These are the VMEbus 16 MHz system clock driver, VMEbus arbiter, and VMEbus IACK daisy-chain driver. All of these functions are disabled when the VME-MXI-2 is not acting as the VMEbus System Controller. The VME-MXI-2 has the ability to automatically detect if it is installed in the first slot of a VMEbus mainframe. The VME-MXI-2 does not provide a power monitor or serial clock driver.
- **DMA Controllers 1 and 2** — The VME-MXI-2 has two DMA controllers, which operate independently of each other. Each DMA controller can be programmed to move data from any source to any destination. The source and destination can be located on the VMEbus, MXIbus, or the VME-MXI-2 module's onboard DRAM. The DMA controllers will direct the MXIbus and VMEbus master state machines to initiate data transfer cycles on their respective bus and can access the onboard DRAM directly. The DMA controllers allow different cycle types and even different data widths between the source and destination during the DMA transfer.
- **MXI-2 System Controller Functions** — The VME-MXI-2 has the ability to act as the MXI-2 system controller. When acting as the system controller, the VME-MXI-2 provides the MXIbus arbiter, priority-selection daisy-chain driver, and bus timeout unit. The VME-MXI-2 can automatically detect from the MXI-2 cable if it is the system controller.
- **VMEbus Control Signals Transceivers** — These transceivers ensure that the VME-MXI-2 meets the loading, driving, and timing requirements of the VMEbus specification for the various control signals.
- **VMEbus Master State Machine** — This state machine generates VMEbus master data transfer cycles when directed to do so by the MXI-2 slave state machine, thus allowing MXIbus cycles to map to the VMEbus. This state machine will also generate VMEbus master data transfer cycles when instructed to do so by one of the DMA controllers. The VME-MXI-2 can generate D64, D32, D16, and D08(EO) single, block, and RMW cycles on the VMEbus in A32 and A24 space. All data transfers can also be generated in A16 space with the exception of D64 and block transfers. Two consecutive MXIbus slave cycles are required to generate a single D64 data transfer cycle. The VME-MXI-2 will not generate unaligned VMEbus data transfers.

- **MXI-2 Master State Machine** This state machine generates MXIbus master data transfer cycles when directed to do so by the VMEbus slave state machine, thus allowing VMEbus cycles to map to the MXIbus. This state machine will also generate MXIbus master data transfer cycles when instructed to do so by one of the DMA controllers. The VME-MXI-2 can generate D64, D32, D16, and D08(EO) single, block, RMW, and synchronous burst cycles on the MXIbus in A32 and A24 space. All data transfers can also be generated in A16 space with the exception of D64, block, and synchronous burst transfers. A single VMEbus D64 data transfer is converted to two consecutive MXIbus data transfers. Synchronous burst MXIbus cycles can be generated only by the DMA controllers, not the VMEbus slave state machine. The MXI-2 master state machine also checks MXIbus parity on read data received and either returns a BERR* to the VMEbus cycle or stores an error status when a parity error is detected.
- **MXI-2 Control Signals Transceivers** These transceivers ensure that the VME-MXI-2 meets the loading, driving, and timing requirements of the MXI-2 specification for the various control signals.
- **VMEbus Slave State Machine** This state machine monitors the output of the address decoders and extender window decoders and responds to VMEbus cycles that are intended for the VME-MXI-2. Cycles that map to the A16 Base Address Decoder access the VME-MXI-2 registers, while cycles that map to the A24/A32 Base Address Decoder access either the VME-MXI-2 registers or the onboard DRAM SIMMs. Cycles that map through an extender window decoder are directed to the MXI-2 master state machine, effectively mapping the VMEbus cycle to the MXIbus. The VME-MXI-2 can accept D32, D16, and D08(EO) single and RMW VMEbus cycles in A32, A24, and A16 space. The VME-MXI-2 can also accept D64 and block VME cycles in A32 and A24 space. Unaligned VMEbus data transfers are treated as D32 data transfers.

- **MXI-2 Slave State Machine** — This state machine monitors the output of the address decoders and extender window decoders and responds to MXIbus cycles that are intended for the VME-MXI-2. Cycles that map to the A16 Base Address Decoder access the VME-MXI-2 registers, while cycles that map to the A24/A32 Base Address Decoder access either the VME-MXI-2 registers or the onboard DRAM SIMMs. Cycles that map through an extender window decoder are directed to the VMEbus master state machine, effectively mapping the MXIbus cycle to the VMEbus. The VME-MXI-2 can accept D32, D16, and D08(EO) single and RMW MXIbus cycles in A32, A24, and A16 space. The VME-MXI-2 can also accept synchronous, D64, and block MXIbus cycles in A32 and A24 space. The MXI-2 slave state machine can also convert MXIbus synchronous and block cycles into single VMEbus cycles for cases where the destination VMEbus slave device does not support VMEbus block cycles. The MXI-2 slave state machine checks for MXIbus parity errors. If a parity error is detected during the address phase of a cycle, the VME-MXI-2 ignores the cycle. If a parity error is detected during the data phase of write cycle, the MXI-2 slave state machine responds with a BERR* on the MXIbus.
- **VMEbus Bus Timeout Unit** — The VME-MXI-2 has a VMEbus Bus Timeout Unit, which terminates (with BERR*) any VMEbus cycle in which DTACK* or BERR* are not asserted in a prescribed amount of time after DS* is asserted. The duration of the timeout is programmably selectable in the range of 15 µs to 256 ms. The VME-MXI-2 must be the sole bus timer of its VMEbus chassis even when not acting as the System Controller. This is because the bus timer should not terminate VMEbus cycles that map to the MXIbus. The MXI-2 bus timer is responsible for timing these cycles.
- **A24/A32 Base Address Decoder** — This address decoder monitors the VMEbus and MXIbus for access to the VME-MXI-2 A24/A32 memory space. All resources located on the VME-MXI-2 are accessible in this region. The lowest 4 KB are directed to the VME-MXI-2 registers while the remainder maps to the onboard DRAM SIMMs.
- **A16 Base Address Decoder** — This address decoder monitors the VMEbus and MXIbus for A16 accesses to the VME-MXI-2 configuration registers. A subset of the VME-MXI-2 registers are accessible in this region, which conforms to VXI-6, the *VXIbus Mainframe Extender Specification*.

- **MXI-2 Parity Check and Generation** The MXI-2 parity check/generation circuitry checks for even parity at any time that the VME-MXI-2 is receiving the AD[31–0] signals. If parity is not even, the appropriate MXI-2 state machine is signaled. The MXI-2 master state machine is signaled for a parity error during the data phase of a MXIbus master read cycle while the MXI-2 slave state machine is signaled for a parity error during the address phase of any MXIbus slave cycle and the data phase of a MXIbus slave write cycle. Even parity is also generated and sent to the MXIbus with master address and write data as well as slave read data.
- **VMEbus Address and Address Modifier Transceivers** These transceivers ensure that the VME-MXI-2 meets the loading, driving, and timing requirements of the VMEbus specification for the A[31–1] and AM[5–0] signals.
- **MXI-2 Address/Data and Address Modifier Transceivers** These transceivers ensure that the VME-MXI-2 meets the loading, driving, and timing requirements of the MXI-2 specification for the AD[31–0], AM[4–0], and CONVERT* signals.
- **A32 Window** This address decoder monitors the VMEbus and MXIbus for A32 accesses that map to the opposite bus, and alerts the appropriate state machines when one occurs. This window behaves as defined in VXI-6, the *VXIbus Mainframe Extender Specification*.
- **A24 Window** This address decoder monitors the VMEbus and MXIbus for A24 accesses that map to the opposite bus, and alerts the appropriate state machines when one occurs. This window behaves as defined in VXI-6, the *VXIbus Mainframe Extender Specification*.
- **A16 Window** This address decoder monitors the VMEbus and MXIbus for A16 accesses that map to the opposite bus, and alerts the appropriate state machines when one occurs. This window accepts cycles only within the lower 48 KB of A16 space. The upper 16 KB (VXIbus configuration space) must be mapped through the LA window. This window behaves as defined in VXI-6, the *VXIbus Mainframe Extender Specification*.
- **LA Window** This address decoder monitors the VMEbus and MXIbus for VXIbus configuration accesses (the upper 16 KB of A16 space) that map to the opposite bus and alerts the appropriate state machines when one occurs. This window behaves as defined in VXI-6, the *VXIbus Mainframe Extender Specification*.
- **MXI-2 Terminate** The VME-MXI-2 has onboard MXI-2 termination circuitry that automatically detects if it is at either cable end to terminate the MXIbus signals. The MXI-2 cable is designed to allow this. If the VME-MXI-2 is a middle device on the MXIbus, the termination is disabled.

- **VME-MXI-2 Registers** This logic block represents all registers on the VME-MXI-2. The registers are accessible from either the VMEbus or MXIbus. All registers are available in the first 4 KB of the VME-MXI-2 A24/A32 memory space, while a subset is accessible in the VME-MXI-2 VXIbus A16 configuration area.
- **Onboard DRAM SIMMs** This logic block represents the two DRAM SIMM sockets on the VME-MXI-2. If DRAM is installed, it will be accessible in the VME-MXI-2 A24/A32 memory space that is not mapped to registers (above 4 KB).
- **VMEbus Data Transceivers** These transceivers ensure that the VME-MXI-2 meets the loading, driving, and timing requirements of the VMEbus specification for the D[31–0] signals.
- **VMEbus Interrupt and Utility Signal Transceivers** These transceivers ensure that the VME-MXI-2 meets the loading, driving, and timing requirements of the VMEbus specification for the IRQ*[7–1], SYSRESET*, SYSFAIL*, and ACFAIL* signals.
- **Interrupt and Utility Signal Circuitry** This circuitry handles mapping of the interrupt and utility signals between the VMEbus and MXIbus. The utility signals include SYSRESET*, SYSFAIL*, and ACFAIL*. This circuitry also generates interrupts from other conditions on the VME-MXI-2 and allows generation of the utility signals.
- **MXI-2 Interrupt and Utility Signal Transceivers** These transceivers ensure that the VME-MXI-2 meets the loading, driving, and timing requirements of the MXI-2 specification for the IRQ*[7–1], SYSRESET*, SYSFAIL*, and ACFAIL* signals.

Chapter 3

VME-MXI-2 Configuration and Installation

This chapter contains the instructions to configure and install the VME-MXI-2 module.

Some features of the VME-MXI-2 are not configurable with onboard switches or jumpers but are instead programmable. Refer to Chapter 6, *VXIplug&play for the VME-MXI-2*, or Appendix B, *Programmable Configurations*, for a description of the programmable features.

Warning: ***Electrostatic discharge can damage several components on your VME-MXI-2 module. To avoid such damage in handling the module, touch the antistatic plastic package to a metal part of your VME chassis before removing the VME-MXI-2 from the package.***

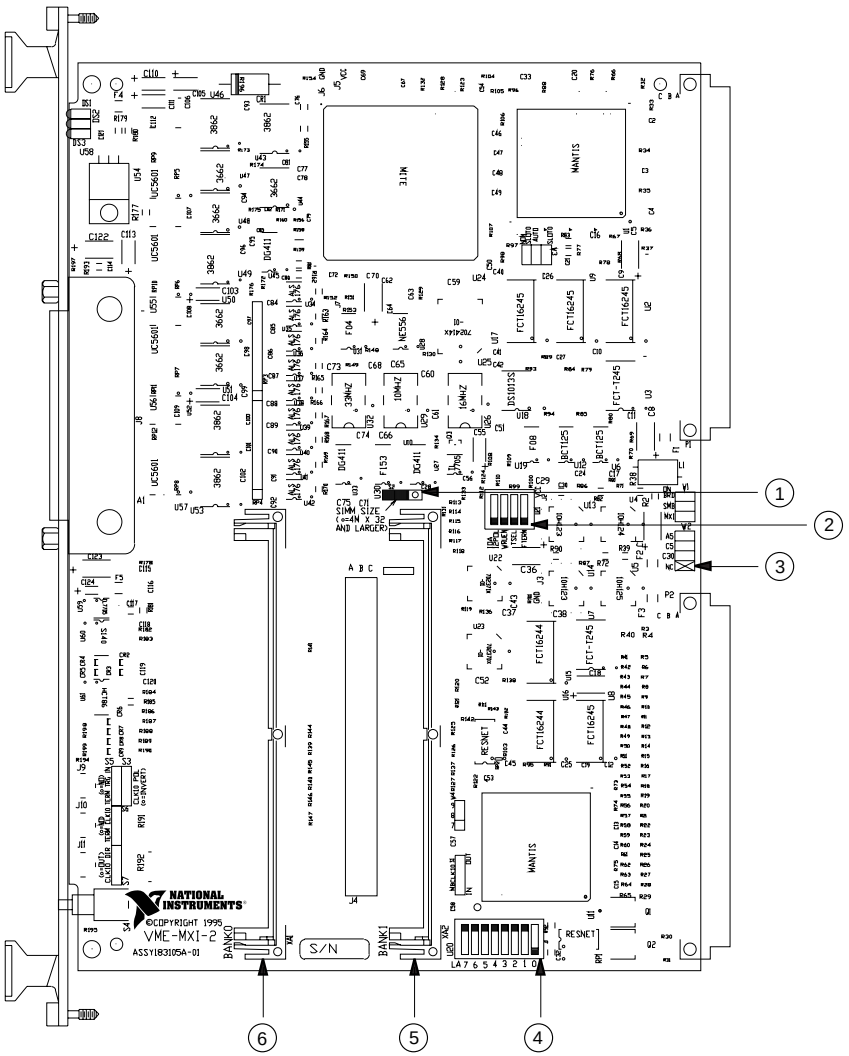
Configure the VME-MXI-2

This section describes how to configure the following options on the VME-MXI-2.

- VMEbus A16 base address
- VME-MXI-2 intermodule signaling
- MXIbus termination
- Configuration EEPROM
- Onboard DRAM

The VME-MXI-2 automatically detects if it is located in the first slot of the chassis to perform the VMEbus System Controller functions. It is not necessary to configure the VME-MXI-2 System Controller option. The module can be installed in any double-height slot of a VMEbus chassis.

Figure 3-1 shows the VME-MXI-2. The drawing shows the location and factory-default settings of the configuration switches and jumpers on the module.



1	S2	3	W2	5	DRAM
2	U21	4	U20	6	DRAM

Figure 3-1. VME-MXI-2 Parts Locator Diagram

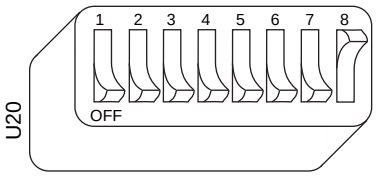
VMEbus A16 Base Address

The VME-MXI-2 requires 64 bytes of A16 space for its configuration registers. It uses the *logical address* scheme of the VXIbus specification, in which each device is assigned an 8-bit value called the logical address. This logical address allocates 64 bytes of space to the device within the upper quarter of A16 space. The VME-MXI-2 cannot be configured to locate its registers in the lower three quarters of A16 space. The A16 base address of the VME-MXI-2 will be address lines 15 and 14 high with address lines 13 through 6 matching the logical address of the VME-MXI-2, and address lines 5 through 0 low. In other words, the A16 base address of the VME-MXI-2 module's 64-byte register set is as calculated below:

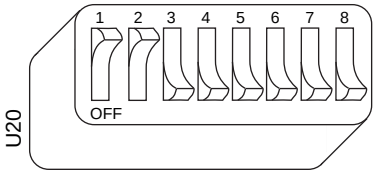
*base address = C000 hex + (logical address) * 40 hex*

The factory-default logical address for the VME-MXI-2 is 1, which locates the registers in the range C040 hex to C07F hex. You can change the logical address of the VME-MXI-2 by changing the setting of the 8-bit DIP switch at location designator U20. The ON position of the DIP switch corresponds to a logic value of 0, and the OFF position corresponds to a logic value of 1. Allowable logical addresses for the VME-MXI-2 range from 1 to 254 (FE hex). Verify that no other devices in your system use the A16 address space for the VME-MXI-2. If possible, configure all other VMEbus A16 devices to be located within the lower three quarters of A16 space. Also, when setting base addresses, keep in mind the grouping requirements set by the system hierarchy. See either Chapter 5, *System Configuration*, or VXI-6, the *VXIbus Mainframe Extender Specification*, for more information on setting base addresses on a multimainframe hierarchy.

Figure 3-2 shows switch settings for A16 base address C040 hex and F000 hex.



a. Switch Set to A16 Base Address C040 Hex (Default)



b. Switch Set to A16 Base Address F000 Hex

Figure 3-2. A16 Base Address Selection

VME-MXI-2 Intermodule Signaling

If you will be installing more than one VME-MXI-2 in a single VMEbus chassis, you must select a user-defined pin for use by the VME-MXI-2. The VME-MXI-2 modules use this signal to disable the bus timeout unit(s) on the other VME-MXI-2 modules during VMEbus accesses that map to the MXIbus. This is done because the MXIbus bus timeout unit should be the sole timer of any MXIbus access. Since bus timeout units on other VMEbus modules cannot monitor this signal, they should be permanently disabled. If it is not possible to disable a module's bus timeout unit, it should be configured to the highest setting to give MXIbus accesses as much time as possible to complete.

You can choose from three user-defined pins on J2/P2. The pin you select must be bused on the VMEbus backplane between all slots that will have a VME-MXI-2 installed. Use jumper W2 to select pin A5, C5, or C30 of J2/P2, as shown in Figure 3-3.

Notice that a fourth position is also available on the jumper. This is the factory-default setting, which does not connect the VME-MXI-2 to any user-defined pin. You would use this option only if you are installing a single VME-MXI-2 in a chassis.

Figure 3-3 shows the four intermodule signaling settings.

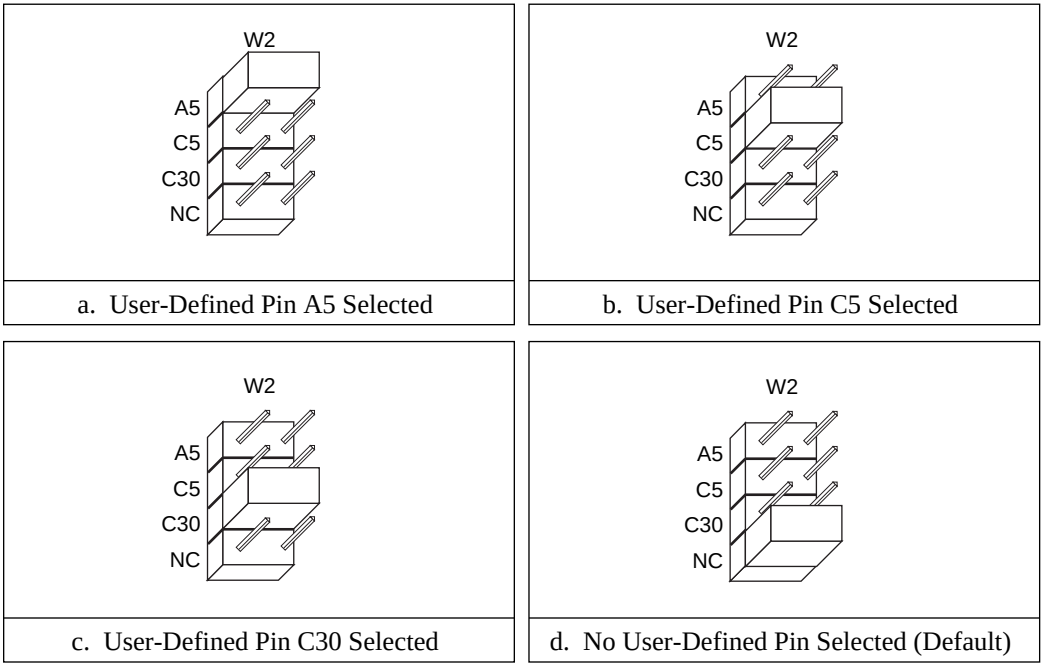


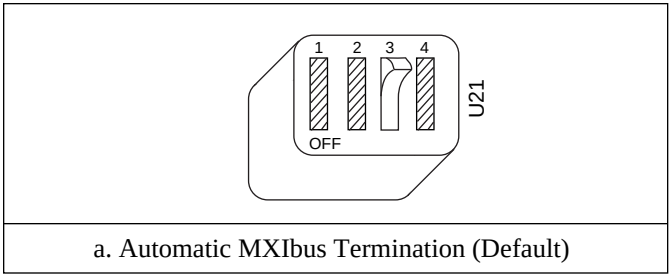
Figure 3-3. VME-MXI-2 Intermodule Signaling Settings

MXIbus Termination

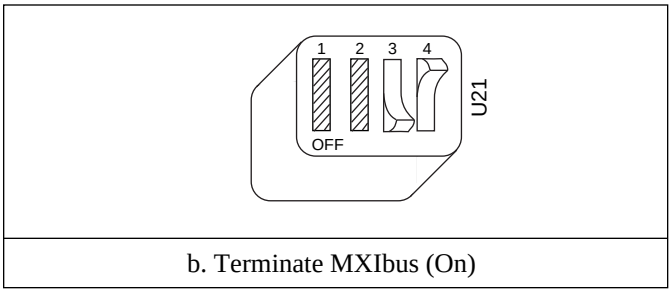
The first and last MXIbus devices connected to the MXIbus—whether it is a single MXI-2 cable or daisy-chained MXI-2 cables—must terminate the MXIbus. Any MXIbus devices in the middle of a MXIbus daisy chain must *not* terminate the MXIbus.

The VME-MXI-2 automatically senses if it is at either end of the MXIbus cable to terminate the MXIbus. You can manually control MXIbus termination by defeating the automatic circuitry. Use switches 3 and 4 of the four-position switch at location U21 to control whether MXIbus termination is automatic (Figure 3-4a), on (Figure 3-4b), or off (Figure 3-4c). The settings of switches 1 and 2 have no effect on MXIbus termination.

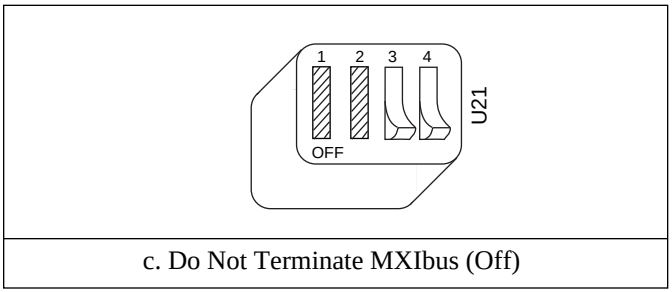
Use switch 3 to select whether you want the VME-MXI-2 to automatically control termination of the MXIbus. Switch 4 lets you manually control whether to terminate the MXIbus when automatic termination is turned off. Switch 4 has no effect when switch 3 is set for automatic MXIbus termination; you must turn off automatic termination if you want to manually control termination.



a. Automatic MXIbus Termination (Default)



b. Terminate MXIbus (On)



c. Do Not Terminate MXIbus (Off)

Figure 3-4. MXIbus Termination

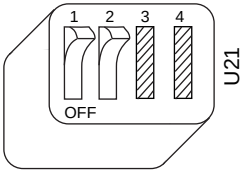
Configuration EEPROM

The VME-MXI-2 has an onboard EEPROM, which stores default register values that are loaded at power-on. The EEPROM is divided into two halves—a factory-configuration half, and a user-configuration half. Both halves were factory configured with the same configuration values so you can modify the user-configurable half, while the factory-configured half stores a back-up of the default user settings.

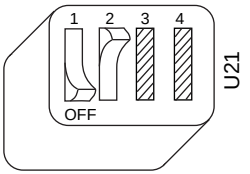
Use switches 1 and 2 of the four-position switch at location U21 to control the operation of the EEPROM. The Restore Factory Configuration switch (switch 1) causes the VME-MXI-2 to boot off the factory-configured half instead of the user-modified settings. This is useful in the event that the user-configured half of the EEPROM becomes corrupted in such a way that the VME-MXI-2 boots to an unusable state.

The Change Factory Configuration switch (switch 2 of U21) lets you change the factory-default configuration settings by permitting writes to the factory settings section of the EEPROM. This switch serves as a safety measure and should not be needed under normal circumstances. When this switch is off (its default setting) the factory configuration of the EEPROM is protected so any writes to the factory area will be ignored. The factory area is protected regardless of the setting of switch 1 of U21.

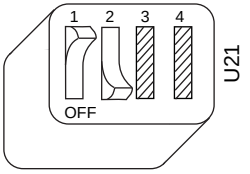
Figure 3-5 shows the configuration settings for EEPROM operation. The settings of switches 3 and 4 have no effect on EEPROM configuration.



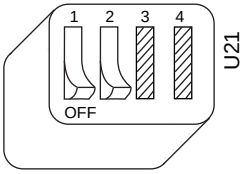
a. Boot from User Configuration (Factory Configuration Protected) (Default)



b. Boot from Factory Configuration (Factory Configuration Protected)



c. Boot from User Configuration (Factory Configuration Unprotected)



d. Boot from Factory Configuration (Factory Configuration Unprotected)

Figure 3-5. EEPROM Operation

Onboard DRAM

The VME-MXI-2 can accommodate up to two 1.35 in. DRAM SIMMs. Table 3-1 lists the SIMMS you can use. You can use 32-bit or 36-bit SIMMS since DRAM parity is not required. Because the VME-MXI-2 supports only one organization at a time, all SIMMs installed must be of the same type. Use Bank 0 first when installing SIMMs. This allows you to install up to 64 MB. The VME-MXI-2 supports DRAM speeds of 80 ns or faster.

Switch S2 is used to select the size of each SIMM. If the SIMMs are 4 M x 32 or larger, S2 should be in the OFF setting as shown in Figure 3-6a. For SIMMs *smaller* than 4 M x 32, use the ON setting as shown in Figure 3-6b.

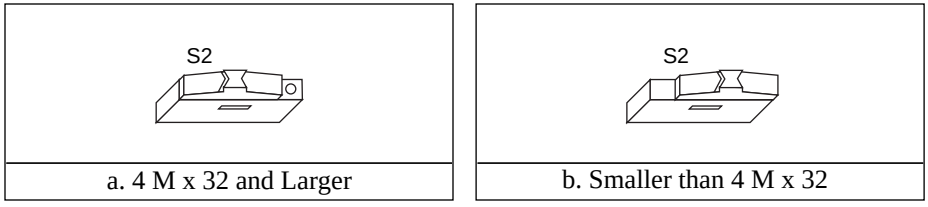


Figure 3-6. SIMM Size Configuration

Refer to Table 3-1 for how to adjust the switch (ON or OFF) for all supported DRAM configurations. Many of the DRAM options are available from National Instruments.

Table 3-1. VME-MXI-2 DRAM Configurations

Bank 0	Bank 1	Total DRAM	National Instruments Option?	Switch Setting of S2
—	—	0	—	—
256 K x 32 or 256 K x 36	—	1 MB	—	ON
256 K x 32 or 256 K x 36	256 K x 32 or 256 K x 36	2 MB	—	ON
512 K x 32 or 512 K x 36	—	2 MB	—	ON
512 K x 32 or 512 K x 36	512 K x 32 or 512 K x 36	4 MB	—	ON
1 M x 32 or 1 M x 36	—	4 MB	YES	ON
1 M x 32 or 1 M x 36	1 M x 32 or 1 M x 36	8 MB	—	ON
2 M x 32 or 2 M x 36	—	8 MB	YES	ON
2 M x 32 or 2 M x 36	2 M x 32 or 2 M x 36	16 MB	—	ON
4 M x 32 or 4 M x 36	—	16 MB	YES	OFF
4 M x 32 or 4 M x 36	4 M x 32 or 4 M x 36	32 MB	—	OFF
8 M x 32 or 8 M x 36	—	32 MB	YES	OFF
8 M x 32 or 8 M x 36	8 M x 32 or 8 M x 36	64 MB	YES	OFF

Install the VME-MXI-2

This section contains general installation instructions for the VME-MXI-2. Consult your VMEbus mainframe user manual or technical reference manual for specific instructions and warnings.

1. Plug in your mainframe before installing the VME-MXI-2. The power cord grounds the mainframe and protects it from electrical damage while you are installing the module.

Warning: ***To protect both yourself and the mainframe from electrical hazards, the mainframe should remain off until you are finished installing the VME-MXI-2 module.***

2. Remove or open any doors or covers blocking access to the mainframe slots.
3. Insert the VME-MXI-2 in the slot you have selected by aligning the top and bottom of the board with the card-edge guides inside the mainframe. Slowly push the VME-MXI-2 straight into the slot until its plug connectors are resting on the backplane receptacle connectors. Using slow, evenly distributed pressure, press the VME-MXI-2 straight in until it seats in the expansion slot. The front panel of the VME-MXI-2 should be even with the front panel of the mainframe.
4. Tighten the retaining screws on the top and bottom edges of the front panel.
5. Check the installation.
6. Connect the cables as described in the following section before restoring power.
7. Replace or close any doors or covers to the mainframe.

Connect the MXIbus Cable

There are two basic types of MXI-2 cables. MXI-2 cables can have either a single connector on each end or a single connector on one end and a double connector on the other end.

Connect the labeled end of the cable to the MXI-2 device that will be the MXIbus System Controller. Connect the other end of the cable to the other device. Be sure to tighten the screw locks to ensure proper pin connection.

Figure 3-7 shows a VME system containing a VME-MXI-2 module residing in Slot 1 of a VMEbus mainframe cabled to a device acting as the MXIbus System Controller. Notice that you can expand your system to include other devices by using an additional MXI-2 cable. However, in such a case the first cable needs to have a double connector on one end. You can then use a cable with a single connector on each end to connect the last device on the MXIbus.

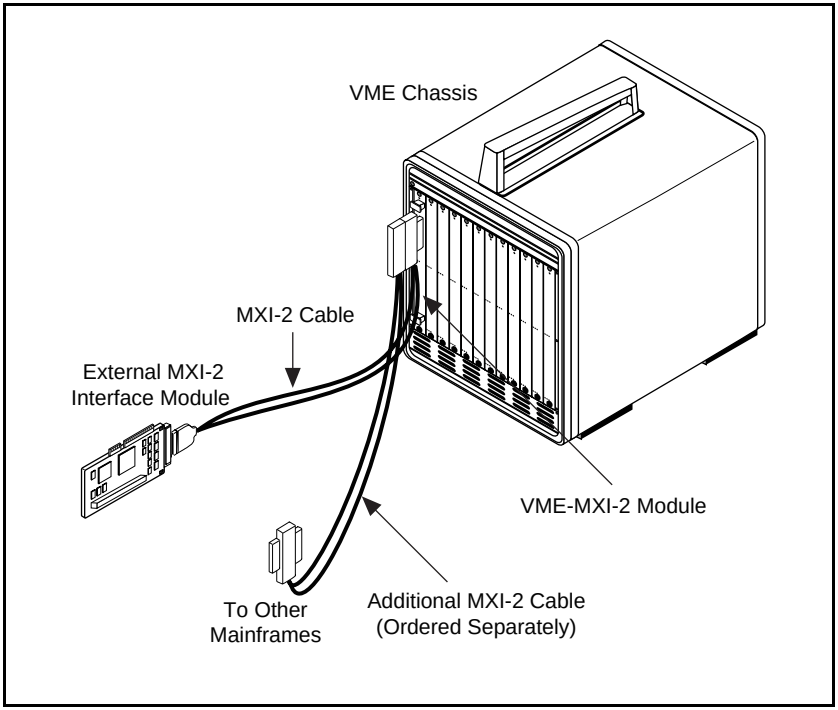


Figure 3-7. MXI-2 Cable Configuration Using an External Device and a VME-MXI-2

Chapter 4

Register Descriptions

This chapter contains detailed information on some of the VME-MXI-2 registers, which you can use to configure and control the module's operation. Some of these registers are a subset of the VME-MXI-2 register set, which is accessible in VXIbus configuration (A16) space, while others are accessible only in the lower 4 KB of the VME-MXI-2 module's A24/A32 memory space. All registers are accessible from either the MXIbus or VMEbus.

If you are using a multiframe VXIbus Resource Manager application, you may not need the information provided in this chapter.

Hard and Soft Reset

Each register description in this chapter indicates which bits are affected by a hard and/or soft reset. A *hard* reset occurs when the mainframe is powered on and when the VMEbus SYSRESET* signal is asserted. A *soft* reset occurs when the RESET bit in the VXIbus Control Register (VCR) is written with a 1 while the VME-MXI-2 is not in the PASSED state. The VME-MXI-2 enters the PASSED state shortly after a hard reset and cannot be put into the soft reset state afterwards. The PASSED bit in the VXIbus Status Register (VSR) indicates when the VME-MXI-2 is in the PASSED state.

Register Description Format

A detailed description of each register follows. Each register description shows a diagram of the register with the most significant bit (bit 31 for 32-bit registers, or bit 15 for 16-bit registers) shown on the upper left, and the least significant bit (bit 0) at the lower right.

The upper 16 bits of a 32-bit register are accessed during a 16-bit cycle to the offset of the register, while the lower 16 bits are accessed during a 16-bit cycle to the offset of the register plus 2. During 8-bit cycles to a 32-bit register, the upper eight bits are accessible at the offset of the register, and the lower eight bits are accessible at the offset of the register plus 3, with the two middle bytes accessible at the offset of the register plus 1 and 2, respectively. The upper eight bits of a 16-bit register are accessed during an 8-bit cycle

to the offset of the register, while the lower eight bits are accessed during an 8-bit cycle to the offset of the register plus 1.

A square is used to represent each bit. Each bit is labeled with a name inside its square. An asterisk (*) after a bit name indicates that the bit is active low.

VXIbus Configuration Registers

Table 4-1 is a register map of the VME-MXI-2 register subset, which is accessible in A16 space. The table gives the mnemonic, offset from the base address, access type (read only, write only, or read/write), access size, and register name.

To access a register in A16 space, the offset given must be added to a base address which can be derived from the following equation:

$$base\ address = C000\ hex + (40\ hex * LA)$$

where *LA* is the logical address of the VME-MXI-2 being accessed. See the *VMEbus A16 Base Address* section in Chapter 3, *VME-MXI-2 Configuration and Installation*, for information on setting the logical address of the VME-MXI-2.

For example, to access the VDTR (VXIbus Device Type Register) on a VME-MXI-2 configured to be Logical Address 1, the base address would be C040 hex and the VDTR would be located at C042 hex, since the VDTR is at offset 2.

Use only the access sizes given in Table 4-1 when accessing each register. For convenience, the access size is repeated in each register description. Because the table is organized with 16 bits per row, accessing any register with a 32-bit access will actually access two of the registers (except in the case of the VIARx registers). Check the **Access Size** column in Table 4-1 to see which registers allow 32-bit accesses.

Table 4-1. VME-MXI-2 VXIbus Configuration Register Map

Mnemonic	Offset (Hex)	Access Type	Access Size	Register Name
VIDR	0	Read Only	32/16/8 bit	VXIbus ID
VDTR	2	Read Only	16/8 bit	VXIbus Device Type
VSR/ VCR	4	Read Only/ Write Only	32/16/8 bit	VXIbus Status/ VXIbus Control
VOR	6	Read/Write	16/8 bit	VXIbus Offset
	8			Reserved
VWR0	A	Read/Write	16/8 bit	Extender LA Window

Table 4-1. VME-MXI-2 VMEbus Configuration Register Map (Continued)

Mnemonic	Offset (Hex)	Access Type	Access Size	Register Name
VWR1	C	Read/Write	32/16/8 bit	Extender A16 Window
VWR2	E	Read/Write	16/8 bit	Extender A24 Window
VWR3	10	Read/Write	16/8 bit	Extender A32 Window
VICR	12	Read/Write	16/8 bit	VXIbus Interrupt Configuration
	14			Reserved
	16			Reserved
VUCR	18	Read/Write	16/8 bit	VXIbus Utility Configuration
	1A			Reserved
	1C			Reserved
VSCR	1E	Read Only	16/8 bit	VXIbus Subclass
VMSR/ VMCR	20	Read Only/ Write Only	16/8 bit	VME-MXI-2 Status/ VME-MXI-2 Control
VLR	22	Read/Write	16/8 bit	VMEbus Lock
	24			Reserved
VLAR	26	Read Only	16/8 bit	VME-MXI-2 Logical Address
	28			Reserved
VISTR/ VICTR	2A	Read Only/ Write Only	16/8 bit	VMEbus Interrupt Status/ VMEbus Interrupt Control
VSIDR	2C	Read/Write	16/8 bit	VMEbus Status ID Register
	2E			Reserved
	30			Reserved
VIAR1	32	Read Only	16/8 bit	VMEbus IACK 1
VIAR2	34	Read Only	32/16/8 bit	VMEbus IACK 2
VIAR3	36	Read Only	16/8 bit	VMEbus IACK 3
VIAR4	38	Read Only	32/16/8 bit	VMEbus IACK 4
VIAR5	3A	Read Only	16/8 bit	VMEbus IACK 5
VIAR6	3C	Read Only	32/16/8 bit	VMEbus IACK 6
VIAR7	3E	Read Only	16/8 bit	VMEbus IACK 7

VXIbus ID Register (VIDR)

VMEbus A16 Offset: 0 (hex)

Attributes: Read Only 32, 16, 8-bit accessible

15	14	13	12	11	10	9	8
DEVCLASS[1]	DEVCLASS[0]	ADSPC[1]	ADSPC[0]	MANID[11]	MANID[10]	MANID[9]	MANID[8]
7	6	5	4	3	2	1	0
MANID[7]	MANID[6]	MANID[5]	MANID[4]	MANID[3]	MANID[2]	MANID[1]	MANID[0]

This register contains information about the VME-MXI-2. You can determine the device class, the address spaces in which the VME-MXI-2 has operational registers, and the manufacturer ID of the VME-MXI-2. This register conforms to the VXIbus specification. When accessed with a 32-bit cycle, the bits of this register appear on bits 31 to 16 along with the VXIbus Device Type Register (VDTR) on bits 15 to 0. Hard and soft resets have no effect on this register.

Bit	Mnemonic	Description
15-14	DEVCLASS[1:0]	Device Class These bits return 01 (binary) to indicate that the VME-MXI-2 is an Extended Class device.
13-12	ADSPC[1:0]	Address Space These bits indicate the address spaces in which the VME-MXI-2 has operational registers. These bits return 00 (binary) when the VME-MXI-2 is configured for A16/A24 space or 01 (binary) when configured for A16/A32 space. Refer to Chapter 6, *VXIplug&play for the VME-MXI-2*, or Appendix B, *Programmable Configurations*, for information on configuring the address space of the VME-MXI-2.
11-0	MANID[11:0]	Manufacturer ID These bits return FF6 (hex) to indicate that the manufacturer of the VME-MXI-2 is National Instruments.

VXIbus Device Type Register (VDTR)

VMEbus A16 Offset: 2 (hex)

Attributes: Read Only 16, 8-bit accessible

15	14	13	12	11	10	9	8
REQMEM[3]	REQMEM[2]	REQMEM[1]	REQMEM[0]	MODEL[11]	MODEL[10]	MODEL[9]	MODEL[8]
7	6	5	4	3	2	1	0
MODEL[7]	MODEL[6]	MODEL[5]	MODEL[4]	MODEL[3]	MODEL[2]	MODEL[1]	MODEL[0]

This register contains information about the VME-MXI-2 that indicates the amount of required address space and identifies the model code of the VME-MXI-2. This register conforms to the VXIbus specification. Hard and soft resets have no effect on this register.

Bit	Mnemonic	Description
15-12	REQMEM[3:0]	Required Memory These bits determine the amount of memory space that will be requested by the VME-MXI-2 in either A24 or A32 address space as determined by the Address Space bits (ADSPC[1:0]) in the VXIbus ID Register (VIDR). The amount of space requested will be $256^{ADSPC[1:0]} * 2^{(23 - REQMEM[3:0])}$ bytes. Refer to Chapter 6, *VXIplug&play for the VME-MXI-2*, or Appendix B, *Programmable Configurations*, for information on configuring the required memory of the VME-MXI-2.
11-0	MODEL[11:0]	Model Code These bits return FE9 hex. National Instruments has assigned this unique code to identify the VME-MXI-2.

VXIbus Status Register (VSR)

VMEbus A16 Offset: 4 (hex)

Attributes: Read Only 32, 16, 8-bit accessible

15	14	13	12	11	10	9	8
A24/A32 ACTIVE	MODID*	EDTYPE[3]	EDTYPE[2]	EDTYPE[1]	EDTYPE[0]	X	ACCDIR
7	6	5	4	3	2	1	0
VERSION[3]	VERSION[2]	VERSION[1]	VERSION[0]	READY	PASSED	SFINH	RESET

This register contains status information about the VME-MXI-2. This register conforms to the VXIbus specification. When accessed with a 32-bit cycle, the bits of this register appear on bits 31 to 16 along with the VXIbus Offset Register (VOR) on bits 15 to 0.

Bit	Mnemonic	Description
15	A24/A32 ACTIVE	A24/A32 Active This bit reflects the state of the A24/A32 ENABLE bit in the VXIbus Control Register (VCR). A 1 indicates that the local A24/A32 registers and memory can be accessed.
14	MODID*	MODID Line Status This bit is hard wired to 1 since the VME-MXI-2 cannot be selected using the VXIbus MODID protocol.
13-10	EDTYPE[3:0]	Extended Device Type Class These bits provide information about the additional capabilities of the VME-MXI-2 as determined by optional configurations or daughter boards. These bits return E (hex) on the standard VME-MXI-2 and are not affected by hard or soft resets.
9	X	Reserved This is a reserved bit. The value it returns is meaningless.

8	ACCDIR	Access Direction This bit returns a 1 when it is read from the MXIbus. When this bit is read from the VMEbus it returns a 0.
7-4	VERSION[3:0]	Version Number These bits indicate the revision of the VME-MXI-2. They return hex F to indicate Revision A. These bits are not affected by hard or soft resets.
3	READY	Ready This bit becomes 1 shortly after a hard reset to indicate that the VME-MXI-2 is ready to execute all of its functionality. This bit is not affected by a soft reset.
2	PASSED	Passed This bit becomes 1 shortly after a hard reset to indicate that the VME-MXI-2 has completed its power-on initialization sequence. The VME-MXI-2 asserts the SYSFAIL* line on the VMEbus after a hard reset until this bit becomes 1. This bit is not affected by a soft reset.
1	SFINH	Sysfail Inhibit This bit reflects the state of the SFINH bit in the VXIbus Control Register (VCR).
0	RESET	Soft Reset This bit reflects the state of the RESET bit in the VXIbus Control Register (VCR).

VXIbus Control Register (VCR)

VMEbus A16 Offset: 4 (hex)

Attributes: Write Only 32, 16, 8-bit accessible

15	14	13	12	11	10	9	8
A24/A32 ENABLE	X	X	X	X	X	X	X

7	6	5	4	3	2	1	0
X	X	X	X	X	X	SFINH	RESET

This register provides various control bits for the VME-MXI-2. This register conforms to the VXIbus specification. When accessed with a 32-bit cycle, the bits of this register appear on bits 31 to 16 along with the VXIbus Offset Register (VOR) on bits 15 to 0.

Bit	Mnemonic	Description
15	A24/A32 ENABLE	A24/A32 Enable Writing a 1 to this bit enables the A24/A32 address decoding on the VME-MXI-2. When this bit is 0 the VME-MXI-2 does not respond to accesses to its onboard A24/A32 resources. This bit is cleared on a hard reset and is not affected by a soft reset.
14-2	X	Reserved These bits are reserved. Write each of these bits with 1 when writing to the VCR.
1	SFINH	Sysfail Inhibit Writing a 1 to this bit disables the VME-MXI-2 from asserting the SYSFAIL* line due to its PASSED bit in the VXIbus Status Register (VSR) being clear. The VME-MXI-2 is still able to assert SYSFAIL* if the DSYSFAIL bit in the VME-MXI-2 Control Register (VMCR) is set or if SYSFAIL* is mapped from the MXIbus to the VMEbus regardless of the state of this bit. This bit is cleared on a hard reset and is not affected by a soft reset.

0	RESET	Reset Writing a 1 to this bit while the PASSED bit in the VXIbus Status Register (VSR) is clear forces the VME-MXI-2 into the Soft Reset state. The VME-MXI-2 cannot be put in the Soft Reset state once the PASSED bit becomes 1. When this bit is 0, the VME-MXI-2 is in the normal operation state. This bit is cleared on a hard reset.

VXIbus Offset Register (VOR)

VMEbus A16 Offset: 6 (hex)

Attributes: Read/Write 16, 8-bit accessible

15	14	13	12	11	10	9	8
OFFSET[15]	OFFSET[14]	OFFSET[13]	OFFSET[12]	OFFSET[11]	OFFSET[10]	OFFSET[9]	OFFSET[8]
7	6	5	4	3	2	1	0
OFFSET[7]	OFFSET[6]	OFFSET[5]	OFFSET[4]	OFFSET[3]	OFFSET[2]	OFFSET[1]	OFFSET[0]

This register determines the base address on the VMEbus and the MXIbus at which to locate the VME-MXI-2 module's A24/A32 resources. This register conforms to the VXIbus specification.

Bit	Mnemonic	Description
15-0	OFFSET[15:0]	VMEbus Offset These bits define the A24 or A32 base address at which the VME-MXI-2 will locate its registers and memory. These bits correspond to VMEbus address lines 23 through 8 when the VME-MXI-2 is configured for A24, and address lines 31 through 16 when configured for A32. The REQMEM[3:0] bits in the VXIbus Device Type Register (VDTR) determine the size of the VME-MXI-2 module's VMEbus memory space by controlling how many bits of OFFSET[15:0] are used. The VME-MXI-2 module's A24/A32 Decoder compares the REQMEM[3:0] + 1 most significant bits of OFFSET[15:0] to their corresponding address lines and responds to cycles that match. The remainder of the OFFSET[15:0] bits are ignored. These bits are cleared by a hard reset and are not affected by a soft reset.

Extender Logical Address Window Register (VWR0)

VMEbus A16 Offset: A (hex)

Attributes: Read/Write 16, 8-bit accessible

15	14	13	12	11	10	9	8
0	LAEN	LADIR	1	1	LASIZE[2]	LASIZE[1]	LASIZE[0]
7	6	5	4	3	2	1	0
LABASE[7]	LABASE[6]	LABASE[5]	LABASE[4]	LABASE[3]	LABASE[2]	LABASE[1]	LABASE[0]

You can use this register to control the mapping of VXIbus configuration space between the VMEbus and the MXIbus. When programming this register, you do not need to consider the VXIbus configuration space that the VME-MXI-2 itself requires. This is because the A16 Base Address Decoder has a higher priority than VWR0 and the VME-MXI-2 will respond to its configuration accesses from both the VMEbus and the MXIbus. This register conforms to the VXIbus Mainframe Extender specification.

This register takes on a different form when the CMODE bit in the VME-MXI-2 Control Register (VMCR) is set. This different form does not comply with the VXIbus Mainframe Extender specification and the CMODE bit should not be set when using a VXIbus multiframe Resource Manager. For more information on the CMODE bit, refer to the VMCR register description.

To accommodate 8-bit masters that write to this register, the window is not enabled until the lower byte of the register is written. Therefore, 8-bit masters should write the upper byte first, followed by the lower byte.

Bit	Mnemonic	Description
15	0	Reserved This bit is reserved and returns 0 when read. This bit can be written with any value.
14	LAEN	Extender Logical Address Window Enable Writing a 1 to this bit enables mapping of VMEbus configuration space through the Extender Logical Address Window. When this bit is cleared, no VXIbus configuration accesses are mapped between the VMEbus and the MXIbus. This bit is cleared by a hard reset and is not affected by a soft reset.

13	LADIR	Extender Logical Address Window Direction When this bit is set, the address range defined by LASIZE[2:0] and LABASE[7:0] applies to MXIbus cycles that are mapped in to VMEbus cycles (inward cycles). When this bit is cleared, the range applies to VMEbus cycles that are mapped out to MXIbus cycles (outward cycles). The complement of the defined range is mapped in the opposite direction. This bit is cleared by a hard reset and is not affected by a soft reset.
12-11	1	Reserved These bits are reserved. They return 11 (binary) when the VWR0 is read. These bits can be written with any value.
10-8	LASIZE[2:0]	Extender Logical Address Window Size These bits define the size of the range of logical addresses that map through the Extender Logical Address Window. They specify the number of address lines that are compared to the LABASE[7:0] bits when determining if a VXIbus configuration access is in the mapped range. The LASIZE[2:0] most significant bits of LABASE[7:0] are compared, while the remaining bits are ignored. Thus, the number of logical addresses in the range mapped is $2^{8\text{-LASIZE[2:0]}}$. These bits are cleared by a hard reset and are not affected by a soft reset.
7-0	LABASE[7:0]	Extender Logical Address Window Base These bits define the base address of the range of logical addresses that map through the Extender Logical Address Window. They correspond to address lines 13 through 6, which effectively makes them the logical address lines. These bits can be thought of as the *base logical address* of the range that maps through the VME-MXI-2. These bits are cleared by a hard reset and are not affected by a soft reset.

Extender A16 Window Register (VWR1)

VMEbus A16 Offset: C (hex)

Attributes: Read/Write 32, 16, 8-bit accessible

15	14	13	12	11	10	9	8
0	A16EN	A16DIR	1	1	A16SIZE[2]	A16SIZE[1]	A16SIZE[0]
7	6	5	4	3	2	1	0
A16BASE[7]	A16BASE[6]	A16BASE[5]	A16BASE[4]	A16BASE[3]	A16BASE[2]	A16BASE[1]	A16BASE[0]

You can use this register to control the mapping of VMEbus A16 space between the VMEbus and the MXIbus. Only the lower three quarters of A16 space can be mapped using this register, because the upper one quarter is VXIbus configuration space, which must be mapped through VWR0. This register conforms to the VXIbus Mainframe Extender specification.

This register takes on a different form when the CMODE bit in the VME-MXI-2 Control Register (VMCR) is set. This different form does not comply with the VXIbus Mainframe Extender specification, and the CMODE bit should not be set when using a VXIbus multiframe Resource Manager. For more information on the CMODE bit, refer to the VMCR register description.

When accessed with a 32-bit cycle, the bits of this register appear on bits 31 to 16 along with the Extender A24 Window Register (VWR2) on bits 15 to 0. To accommodate 8-bit masters that write to this register, the window is not enabled until the lower byte of the register is written. Therefore, 8-bit masters should write the upper byte first, followed by the lower byte.

Bit	Mnemonic	Description
15	0	Reserved This bit is reserved and returns 0 when read. This bit can be written with any value.
14	A16EN	Extender A16 Window Enable Writing a 1 to this bit enables mapping of VMEbus A16 space through the Extender A16 Window. When this bit is cleared, no VMEbus A16 accesses are mapped between the VMEbus and the MXIbus. This bit is cleared by a hard reset and is not affected by a soft reset.

13	A16DIR	Extender A16 Window Direction When this bit is set, the address range defined by A16SIZE[2:0] and A16BASE[7:0] applies to MXIbus cycles that are mapped in to VMEbus cycles (inward cycles). When this bit is cleared, the range applies to VMEbus cycles that are mapped out to MXIbus cycles (outward cycles). The complement of the defined range is mapped in the opposite direction. This bit is cleared by a hard reset and is not affected by a soft reset.
12-11	1	Reserved These bits are reserved. They return 11 (binary) when the VWR1 is read. These bits can be written with any value.
10-8	A16SIZE[2:0]	Extender A16 Window Size These bits define the size of the range of A16 addresses that map through the Extender A16 Window. They specify the number of address lines that are compared to the A16BASE[7:0] bits when determining if a VMEbus A16 access is in the mapped range. The A16SIZE[2:0] most significant bits of A16BASE[7:0] are compared, while the remaining bits are ignored. Thus, the number of A16 addresses in the range mapped is $256 * 2^{8-A16SIZE[2:0]}$. These bits are cleared by a hard reset and are not affected by a soft reset.
7-0	A16BASE[7:0]	Extender A16 Window Base These bits define the base address of the range of A16 addresses that map through the Extender A16 Window. They correspond to address lines 15 through 8 (the eight most significant address lines used in VMEbus A16 space). No part of the upper one quarter of A16 space will be mapped through the Extender A16 Window regardless of the size and base programmed. These bits are cleared by a hard reset and are not affected by a soft reset.

Extender A24 Window Register (VWR2)

VMEbus A16 Offset: E (hex)

Attributes: Read/Write 16, 8-bit accessible

15	14	13	12	11	10	9	8
0	A24EN	A24DIR	1	1	A24SIZE[2]	A24SIZE[1]	A24SIZE[0]
7	6	5	4	3	2	1	0
A24BASE[7]	A24BASE[6]	A24BASE[5]	A24BASE[4]	A24BASE[3]	A24BASE[2]	A24BASE[1]	A24BASE[0]

You can use this register to control the mapping of VMEbus A24 space between the VMEbus and the MXIbus. When programming this register, you do not have to consider any VMEbus A24 space that the VME-MXI-2 itself requires. This is because the A24/A32 Base Address Decoder has a higher priority than VWR2, and the VME-MXI-2 will respond to its A24 accesses from both the VMEbus and the MXIbus. This register conforms to the VXIbus Mainframe Extender specification.

This register takes on a different form when the CMODE bit in the VME-MXI-2 Control Register (VMCR) is set. This different form does not comply with the VXIbus Mainframe Extender specification, and the CMODE bit should not be set when using a VXIbus multiframe Resource Manager. For more information on the CMODE bit, refer to the VMCR register description.

To accommodate 8-bit masters that write to this register, the window is not enabled until the lower byte of the register is written. Therefore, 8-bit masters should write the upper byte first, followed by the lower byte.

Bit	Mnemonic	Description
15	0	Reserved This bit is reserved and returns 0 when read. This bit can be written with any value.
14	A24EN	Extender A24 Window Enable Writing a 1 to this bit enables mapping of VMEbus A24 space through the Extender A24 Window. When this bit is cleared, no VMEbus A24 accesses are mapped between the VMEbus and the MXIbus. This bit is cleared by a hard reset and is not affected by a soft reset.

13	A24DIR	Extender A24 Window Direction When this bit is set, the address range defined by A24SIZE[2:0] and A24BASE[7:0] applies to MXIbus cycles that are mapped in to VMEbus cycles (inward cycles). When this bit is cleared, the range applies to VMEbus cycles that are mapped out to MXIbus cycles (outward cycles). The complement of the defined range is mapped in the opposite direction. This bit is cleared by a hard reset and is not affected by a soft reset.
12-11	1	Reserved These bits are reserved. They return 11 (binary) when the VWR2 is read. These bits can be written with any value.
10-8	A24SIZE[2:0]	Extender A24 Window Size These bits define the size of the range of A24 addresses that map through the Extender A24 Window. They specify the number of address lines that are compared to the A24BASE[7:0] bits when determining if a VMEbus A24 access is in the mapped range. The A24SIZE[2:0] most significant bits of A24BASE[7:0] are compared, while the remaining bits are ignored. Thus, the number of A24 addresses in the range mapped is $65536 * 2^{8-A24SIZE[2:0]}$. These bits are cleared by a hard reset and are not affected by a soft reset.
7-0	A24BASE[7:0]	Extender A24 Window Base These bits define the base address of the range of A24 addresses that map through the Extender A24 Window. They correspond to address lines 23 through 16 (the eight most significant address lines used in VMEbus A24 space). These bits are cleared by a hard reset and are not affected by a soft reset.

Extender A32 Window Register (VWR3)

VMEbus A16 Offset: 10 (hex)

Attributes: Read/Write 16, 8-bit accessible

15	14	13	12	11	10	9	8
0	A32EN	A32DIR	1	1	A32SIZE[2]	A32SIZE[1]	A32SIZE[0]
7	6	5	4	3	2	1	0
A32BASE[7]	A32BASE[6]	A32BASE[5]	A32BASE[4]	A32BASE[3]	A32BASE[2]	A32BASE[1]	A32BASE[0]

You can use this register to control the mapping of VMEbus A32 space between the VMEbus and the MXIbus. When programming this register, you do not need to consider any VMEbus A32 space that the VME-MXI-2 itself requires. This is because the A24/A32 Base Address Decoder has a higher priority than VWR3, and the VME-MXI-2 will respond to its A32 accesses from both the VMEbus and the MXIbus. This register conforms to the VXIbus Mainframe Extender specification.

This register takes on a different form when the CMODE bit in the VME-MXI-2 Control Register (VMCR) is set. This different form does not comply with the VXIbus Mainframe Extender specification, and the CMODE bit should not be set when using a VXIbus multiframe Resource Manager. For more information on the CMODE bit, refer to the VMCR register description.

To accommodate 8-bit masters that write to this register, the window is not enabled until the lower byte of the register is written. Therefore, 8-bit masters should write the upper byte first, followed by the lower byte.

Bit	Mnemonic	Description
15	0	Reserved This bit is reserved and returns 0 when read. This bit can be written with any value.
14	A32EN	Extender A32 Window Enable Writing a 1 to this bit enables mapping of VMEbus A32 space through the Extender A32 Window. When this bit is cleared, no VMEbus A32 accesses are mapped between the VMEbus and the MXIbus. This bit is cleared by a hard reset and is not affected by a soft reset.

13	A32DIR	Extender A32 Window Direction When this bit is set, the address range defined by A32SIZE[2:0] and A32BASE[7:0] applies to MXIbus cycles that are mapped in to VMEbus cycles (inward cycles). When this bit is cleared, the range applies to VMEbus cycles that are mapped out to MXIbus cycles (outward cycles). The complement of the defined range is mapped in the opposite direction. This bit is cleared by a hard reset and is not affected by a soft reset.
12-11	1	Reserved These bits are reserved. They return 11 (binary) when the VWR3 is read. These bits can be written with any value.
10-8	A32SIZE[2:0]	Extender A32 Window Size These bits define the size of the range of A32 addresses that map through the Extender A32 Window. They specify the number of address lines that are compared to the A32BASE[7:0] bits when determining if a VMEbus A32 access is in the mapped range. The A32SIZE[2:0] most significant bits of A32BASE[7:0] are compared, while the remaining bits are ignored. Thus, the number of A32 addresses in the range mapped is $16777216 * 2^{8-A32SIZE[2:0]}$. These bits are cleared by a hard reset and are not affected by a soft reset.
7-0	A32BASE[7:0]	Extender A32 Window Base These bits define the base address of the range of A32 addresses that map through the Extender A32 Window. They correspond to address lines 31 through 24 (the eight most significant address lines in VMEbus A32 space). These bits are cleared by a hard reset and are not affected by a soft reset.

VXIbus Interrupt Configuration Register (VICR)

VMEbus A16 Offset: 12 (hex)

Attributes: Read/Write 16, 8-bit accessible

15	14	13	12	11	10	9	8
0	INTEN[7]	INTEN[6]	INTEN[5]	INTEN[4]	INTEN[3]	INTEN[2]	INTEN[1]
7	6	5	4	3	2	1	0
0	INTDIR[7]	INTDIR[6]	INTDIR[5]	INTDIR[4]	INTDIR[3]	INTDIR[2]	INTDIR[1]

You can use this register to control the routing of the seven VMEbus interrupt lines between the VMEbus and the MXIbus. Any interrupts that the VME-MXI-2 itself generates will be driven on the VMEbus and must be routed to the MXIbus through this register if the handler for the interrupt is located on the MXIbus. Interrupt Acknowledge cycles are mapped in the opposite direction of the corresponding interrupt, which allows the handler to transparently reach the interrupter when acknowledging an interrupt. More than one VME-MXI-2 can route the same interrupt level to the same bus (the VMEbus or MXIbus). This register conforms to the VXIbus Mainframe Extender specification.

Bit	Mnemonic	Description
15	0	Reserved This bit is reserved. It returns 0 when read. This bit can be written with any value.
14-8	INTEN[7:1]	Interrupt Enable Setting these bits individually enables routing of the seven VMEbus interrupt lines between the VMEbus and the MXIbus. Any interrupt line whose corresponding INTEN[7:1] bit is clear is not routed. These bits are cleared by a hard reset and are not affected by a soft reset.
7	0	Reserved This bit is reserved. It returns 0 when read. This bit can be written with any value.

6-0	INTDIR[7:1]	Interrupt Direction When the corresponding INTEN[7:1] bit is clear, these bits are ignored. When the corresponding INTEN[7:1] bit is set, these bits control the direction that the interrupt is routed. The interrupt is routed from the VMEbus to the MXIbus when its INTDIR[7:1] bit is 0 (outward), and from the MXIbus to the VMEbus when its INTDIR[7:1] bit is 1 (inward). These bits are cleared by a hard reset and are not affected by a soft reset.

VXIbus Utility Configuration Register (VUCR)

VMEbus A16 Offset: 18 (hex)

Attributes: Read/Write 16, 8-bit accessible

15	14	13	12	11	10	9	8
1	1	1	UTIL*	1	1	1	1
7	6	5	4	3	2	1	0
1	1	ACFIN	ACFOUT	SFIN	SFOUT	SRIN	SROUT

You can use this register to control the routing of the VMEbus utility signals between the VMEbus and the MXIbus. The VMEbus utility signals are ACFAIL*, SYSFAIL*, and SYSRESET*. Any utility signals that the VME-MXI-2 itself generates are driven on the VMEbus and must be routed to the MXIbus through this register if the destination for the signal is located on the MXIbus. Likewise, the VME-MXI-2 can sense the three utility signals only from the VMEbus, so any signal originating on the MXIbus that the VME-MXI-2 must sense should be routed through this register to the VMEbus. There are no restrictions on either the number of VME-MXI-2 modules routing the utility signals or the directions in which they are routed. Also, the VME-MXI-2 can route any utility signal in both directions simultaneously. This register conforms to the VXIbus Mainframe Extender specification.

Bit	Mnemonic	Description
15-13	1	Reserved These bits are reserved. They each return 1 when read. These bits can be written with any value.
12	UTIL*	Utility Signal Support This read-only bit returns a 0 to indicate that the VME-MXI-2 supports routing of the VMEbus utility signals ACFAIL*, SYSFAIL*, and SYSRESET*. The value written to this bit is irrelevant.
11-6	1	Reserved These bits are reserved. They each return 1 when read. Write each of these bits with 1 when writing to the VUCR.

5	ACFIN	ACFAIL* In Setting this bit causes the VME-MXI-2 to route the ACFAIL* signal from the MXIbus to the VMEbus. When this bit is clear, ACFAIL* is ignored on the MXIbus. This bit is cleared by a hard reset and is not affected by a soft reset.
4	ACFOUT	ACFAIL* Out Setting this bit causes the VME-MXI-2 to route the ACFAIL* signal from the VMEbus to the MXIbus. When this bit is clear, ACFAIL* is ignored on the VMEbus. You can route ACFAIL* in both directions simultaneously. This bit is cleared by a hard reset and is not affected by a soft reset.
3	SFIN	SYSFAIL* In Setting this bit causes the VME-MXI-2 to route the SYSFAIL* signal from the MXIbus to the VMEbus. When this bit is clear, SYSFAIL* is ignored on the MXIbus. This bit is cleared by a hard reset and is not affected by a soft reset.
2	SFOUT	SYSFAIL* Out Setting this bit causes the VME-MXI-2 to route the SYSFAIL* signal from the VMEbus to the MXIbus. When this bit is clear, SYSFAIL* is ignored on the VMEbus. You can route SYSFAIL* in both directions simultaneously. This bit is cleared by a hard reset and is not affected by a soft reset.
1	SRIN	SYSRESET* In Setting this bit causes the VME-MXI-2 to route the SYSRESET* signal from the MXIbus to the VMEbus. When this bit is clear, SYSRESET* is ignored on the MXIbus. This bit is cleared by a hard reset and is not affected by a soft reset.

0	SROUT	SYSRESET* Out Setting this bit causes the VME-MXI-2 to route the SYSRESET* signal from the VMEbus to the MXIbus. When this bit is clear, SYSRESET* is ignored on the VMEbus. You can route SYSRESET* in both directions simultaneously. This bit is cleared by a hard reset and is not affected by a soft reset.

VXIbus Subclass Register (VSCR)

VMEbus A16 Offset: 1E (hex)

Attributes: Read Only 16, 8-bit accessible

15	14	13	12	11	10	9	8
SC[15]	SC[14]	SC[13]	SC[12]	SC[11]	SC[10]	SC[9]	SC[8]
7	6	5	4	3	2	1	0
SC[7]	SC[6]	SC[5]	SC[4]	SC[3]	SC[2]	SC[1]	SC[0]

The VXIbus Subclass Register (VSCR) is used to specify the precise class of a device when it indicates with the DEVCLASS[1:0] bits in the VXIbus ID Register (VIDR) that it is an Extended Class device. The VME-MXI-2 is a Mainframe Extender, which is one of the VXIbus-defined Extended classes. This register contains the VXIbus Mainframe Extender subclass code. This register conforms to the VXIbus Mainframe Extender specification.

Bit	Mnemonic	Description
15-0	SC[15:0]	Subclass These read-only bits return FFFC (hex).

VME-MXI-2 Status Register (VMSR)

VMEbus A16 Offset: 20 (hex)

Attributes: Read Only 16, 8-bit accessible

15	14	13	12	11	10	9	8
0	CMODE	1	POSTERR	MXSCTO	INTLCK	DSYSFAIL	FAIR
7	6	5	4	3	2	1	0
MXISC	0	0	0	SCFG	MBERR	0	PARERR

This VME-MXI-2-specific register provides status bits for various operations.

Bit	Mnemonic	Description
15	0	Reserved This bit is reserved and returns 0 when read.
14	CMODE	Comparison Mode Status This bit reflects the state of the CMODE bit in the VME-MXI-2 Control register (VMCR).
13	1	Reserved This bit is reserved and returns 1 when read.
12	POSTERR	Write Post Error Status This bit returns 1 when a write-posted cycle results in an error. This is actually two bits; one can be read from the MXIbus, the other can be read from the VMEbus. When a VMEbus master reads this bit as a 1, a VMEbus data cycle that mapped to the MXIbus and was posted results in an error. When a MXIbus master reads this bit as a 1, a MXIbus data cycle that mapped to the VMEbus and was posted results in an error. Each bit clears when read and on hard and soft resets. Write posting can be enabled using the VXI*plug&play* soft front panel for the VME-MXI-2.

11	MXSCTO	MXIbus System Controller Timeout Status If the VME-MXI-2 is the MXIbus System Controller, this bit is set when the VME-MXI-2 terminates a MXIbus cycle with a BERR due to a bus timeout. This bit is cleared by hard and soft resets and when read.
10	INTLCK	Interlocked Status This bit reflects the state of the INTLCK bit in the VME-MXI-2 Control Register (VMCR).
9	DSYSFAIL	Drive SYSFAIL* Status This bit reflects the state of the DSYSFAIL bit in the VME-MXI-2 Control Register (VMCR).
8	FAIR	MXIbus Fair Status This bit indicates if the VME-MXI-2 is a fair MXIbus requester. The VME-MXI-2 is fair if this bit returns a 1, and not fair if it returns a 0. Refer to Chapter 6, *VXIplug&play for the VME-MXI-2*, or Appendix B, *Programmable Configurations*, for information on configuring the VME-MXI-2 as a fair MXIbus requester.
7	MXISC	MXIbus System Controller Status This bit returns a 1 if the VME-MXI-2 is the MXIbus System Controller, or a 0 when the VME-MXI-2 is not the MXIbus System Controller.
6-4	0	Reserved These bits are reserved and return 000 (binary) when read.

3	SCFG	Self-Configuration Status After a hard reset, the VME-MXI-2 executes an initialization sequence called *self-configuration*. When this bit returns a 1, self-configuration is in process and the VME-MXI-2 may not be fully initialized. When this bit returns a 0, self-configuration is complete and the VME-MXI-2 is initialized. The PASSED bit in the VXIbus Status Register (VSR) also does not become set until self-configuration is complete; this prevents a VXIbus Resource Manager from attempting to program the VME-MXI-2 before initialization is complete.
2	MBERR	MXIbus Bus Error Status If this bit is set, the VME-MXI-2 terminated the previous MXIbus transfer by driving the MXIbus BERR* line. This indicates that the cycle was terminated because of a bus error or a retry condition. This bit is cleared by hard and soft resets and on successful MXIbus accesses.
1	0	Reserved This bit is reserved and returns 0 when read.
0	PARERR	Parity Error Status If this bit is set, a MXIbus parity error occurred on either the address or the data portion of the last MXIbus transfer. This bit is cleared by hard and soft resets and on MXIbus transfers without a parity error.

VME-MXI-2 Control Register (VMCR)

VMEbus A16 Offset: 20 (hex)

Attributes: Write Only 16, 8-bit accessible

15	14	13	12	11	10	9	8
0	CMODE	0	0	0	0	DSYSFAIL	DSYSRST
7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	INTLCK

This VME-MXI-2 specific register provides control bits for various operations.

Bit	Mnemonic	Description
15	0	Reserved This bit is reserved. Write a 0 when writing to this bit.
14	CMODE	Comparison Mode This bit selects the range comparison mode for the Extender Logical Address (VWR0), A16 (VWR1), A24 (VWR2), and A32 (VWR3) Window Registers. If CMODE is cleared, a Base/Size range comparison is used to determine the range of addresses in the windows, as described in the VWR*x* register descriptions. If CMODE is set, an upper and lower bound is used to determine the range of addresses in the windows. The upper eight bits of each VWR*x* register form the upper bound (HIGH[7:0]), while the lower eight bits form the lower bound (LOW[7:0]). The LOW[7:0] bits define the lower limit of the range of MXIbus addresses that map into the VMEbus, while the HIGH[7:0] bits define the upper limit. As with the normal comparison mode, any address that is not in the range will map in the opposite direction.

		When HIGH[7:0] > [range] ≥ LOW[7:0], a MXIbus cycle within the range maps to the VMEbus, while a VMEbus cycle out of that range maps to the MXIbus. When LOW[7:0] > [range] ≥ HIGH[7:0], a VMEbus cycle within the range maps to the MXIbus, while a MXIbus cycle out of that range maps to the VMEbus. When HIGH[7:0] = LOW[7:0] = 0, the window is disabled. When FF (hex) ≥ (HIGH[7:0] = LOW[7:0]) ≥ 80 (hex), all VMEbus addresses are mapped out to the MXIbus. When 7F (hex) ≥ (HIGH[7:0] = LOW[7:0]) > 0, all MXIbus addresses are mapped in to the VMEbus. To accommodate 8-bit devices that write to the VWRx registers, the window is not enabled until the lower byte is written. Therefore, 8-bit masters should write the upper byte first, then the lower byte. This bit is cleared by hard and soft resets.
13-10	0	Reserved These bits are reserved. Write a 0 to each of these bits when writing the VMCR.
9	DSYSFAIL	Drive SYSFAIL* Writing a 1 to this bit causes the VME-MXI-2 to assert the VMEbus SYSFAIL* line. This bit is cleared by hard and soft resets.
8	DSYSRST	Drive SYSRESET* Writing a 1 to this bit causes the VME-MXI-2 to assert the VMEbus SYSRESET* line for a minimum of 200 ms. This bit is automatically cleared after the assertion of SYSRESET*.
7-1	0	Reserved These bits are reserved. Write each of these bits with 0 when writing to the VMCR.

0	INTLCK	Interlocked Mode Writing a 1 to this bit causes the VME-MXI-2 to interlock arbitration between the VMEbus and the MXIbus. When arbitration is interlocked, the VME-MXI-2 will always own either the VMEbus or the MXIbus. When the VME-MXI-2 must release the bus that it owns, it does not do so until it obtains ownership of the other bus (VMEbus or the MXIbus). If the VME-MXI-2 does not own either bus when this bit is written with a 1, it will arbitrate for the VMEbus. This bit is cleared by a hard reset and is not affected by a soft reset. Refer to Chapter 6, *VXIplug&play for the VME-MXI-2*, for more detailed information on interlocked mode.

VMEbus Lock Register (VLR)

VMEbus A16 Offset: 22 (hex)

Attributes: Read/Write 16, 8-bit accessible

15	14	13	12	11	10	9	8
1	1	1	1	1	1	1	1

7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	LOCKED

This register is used to lock the VMEbus or the MXIbus. This register performs differently depending on whether the register is accessed from the VMEbus or the MXIbus.

Bit	Mnemonic	Description
15-1	1	Reserved These bits are reserved and each returns 1 when read. Write a 0 to each of these bits when writing to the VLR.
0	LOCKED	VMEbus or MXIbus Locked When this bit is set by a VMEbus access, the VME-MXI-2 arbitrates for the MXIbus. Once the VME-MXI-2 wins arbitration, it does not give up ownership of the MXIbus until either this bit is cleared or a reset occurs. This prevents any other MXIbus masters from using the bus so that the VME-MXI-2 can complete indivisible operations. When this bit is set by a MXIbus access, the VMEbus is locked by that device so that indivisible operations to local VMEbus resources can be performed from the MXIbus. Similarly, when a VMEbus device reads this bit as a 1, it indicates that the MXIbus is locked. When a MXIbus device reads this bit as a 1, it indicates that the VMEbus is locked. This bit does not read as a 1 until the VME-MXI-2 has successfully arbitrated for and won the indicated bus. Writing a 0 to this bit unlocks the appropriate bus. This bit is cleared by hard and soft resets.

VME-MXI-2 Logical Address Register (VLAR)

VMEbus A16 Offset: 26 (hex)

Attributes: Read Only 16, 8-bit accessible

15	14	13	12	11	10	9	8
0	0	0	0	0	0	0	0
7	6	5	4	3	2	1	0
LA[7]	LA[6]	LA[5]	LA[4]	LA[3]	LA[2]	LA[1]	LA[0]

This register provides the logical address of the VME-MXI-2.

Bit	Mnemonic	Description
15-8	0	Reserved These bits are reserved. They return 0 when read.
7-0	LA[7:0]	Logical Address Status These bits return the logical address of the VME-MXI-2.

VMEbus Interrupt Status Register (VISTR)

VMEbus A16 Offset: 2A (hex)

Attributes: Read Only 16, 8-bit accessible

15	14	13	12	11	10	9	8
LINT[3]	LINT[2]	LINT[1]	AFINT	BKOFF	0	SYSFAIL	ACFAIL
7	6	5	4	3	2	1	0
SFINT	IRQ[7]	IRQ[6]	IRQ[5]	IRQ[4]	IRQ[3]	IRQ[2]	IRQ[1]

You can use this register to monitor the VMEbus IRQ[7:1] lines and the status of local VME-MXI-2 interrupt conditions. Bits 15 through 8 of this register, along with the logical address of the VME-MXI-2 on bits 7 through 0, are returned during an interrupt acknowledge cycle for the local interrupt condition.

Bit	Mnemonic	Description
15-13	LINT[3:1]	Local Interrupt Level These bits reflect the state of the LINT[3:1] bits in the VMEbus Interrupt Control Register (VICTR).
12	AFINT	VMEbus ACFAIL* Interrupt Status This bit returns 1 when the VME-MXI-2 is driving the VMEbus IRQ[7:1] selected by LINT[3:1] because the ACFAIL* line is asserted. This bit clears after the VME-MXI-2 responds to an interrupt acknowledge cycle for the local interrupt. The ACFAIL* interrupt is enabled with the AFIE bit in the VMEbus Interrupt Control Register (VICTR).
11	BKOFF	Back Off Status This bit is set when the VME-MXI-2 encounters a deadlock condition between the VMEbus and the MXIbus. If the BKOFFIE bit in the VMEbus Interrupt Control Register (VICTR) is set, an interrupt is also generated. This bit stores the deadlock status even when the interrupt is not enabled. This bit clears when read either directly or through an interrupt acknowledge cycle.

10	0	Reserved This bit is reserved and returns 0 when read.
9	SYSFAIL	SYSFAIL* Status This bit returns the current state of the VMEbus SYSFAIL* signal. A 1 indicates that SYSFAIL* is asserted (low), while a 0 indicates it is not asserted (high). If the SFIE bit in the VMEbus Interrupt Control Register (VICTR) is set, an interrupt is also generated when SYSFAIL* asserts.
8	ACFAIL	ACFAIL* Status This bit returns the current state of the VMEbus ACFAIL* signal. A 1 indicates that ACFAIL* is asserted (low), while a 0 indicates it is not asserted (high). If the AFIE bit in the VMEbus Interrupt Control Register (VICTR) is set, an interrupt is also generated when ACFAIL* asserts.
7	SFINT	VMEbus SYSFAIL* Interrupt Status This bit returns 1 when the VME-MXI-2 is driving the VMEbus IRQ[7:1] selected by LINT[3:1] because the SYSFAIL* line is asserted. This bit clears after the VME-MXI-2 responds to an interrupt acknowledge cycle for the local interrupt. The SYSFAIL* interrupt is enabled with the SFIE bit in the VMEbus Interrupt Control Register (VICTR).
6-0	IRQ[7:1]	VMEbus Interrupt Request [7:1] Status These bits return the current state of the seven VMEbus interrupt request lines on the mainframe. If a bit returns a 1, the corresponding IRQ is asserted.

VMEbus Interrupt Control Register (VICTR)

VMEbus A16 Offset: 2A (hex)

Attributes: Write Only 16, 8-bit accessible

15	14	13	12	11	10	9	8
LINT[3]	LINT[2]	LINT[1]	0	BKOFFIE	0	SFIE	AFIE
7	6	5	4	3	2	1	0
0	DIRQ[7]	DIRQ[6]	DIRQ[5]	DIRQ[4]	DIRQ[3]	DIRQ[2]	DIRQ[1]

This register allows the VME-MXI-2 to assert the VMEbus IRQ[7:1] lines and provides enable bits for the various VME-MXI-2 local interrupts.

Bit	Mnemonic	Description
15-13	LINT[3:1]	Local Interrupt Level These bits determine which VMEbus interrupt level the local interrupt conditions will assert. The local interrupt conditions are the BKOFF, SFINT, and AFINT bits of the VMEbus Interrupt Status Register (VISTR). Each condition can be individually enabled in this register. Write a number in the range 1 through 7 to these bits to select the desired interrupt level. Writing a 0 to these bits globally disables the local interrupt. These bits are cleared by a hard reset and are not affected by a soft reset.
12	0	Reserved This bit is reserved. Write a 0 when writing to this bit.
11	BKOFFIE	Back Off Interrupt Enable Writing a 1 to this bit enables the BKOFF interrupt condition in the VMEbus Interrupt Status Register (VISTR) to assert the VMEbus IRQ[7:1] selected by LINT[3:1]. This bit is cleared by a hard reset and is not affected by a soft reset.
10	0	Reserved This bit is reserved. Write a 0 when writing to this bit.

9	SFIE	SYSFAIL* Interrupt Enable Writing a 1 to this bit enables the SFINT interrupt condition in the VMEbus Interrupt Status Register (VISTR) to assert the VMEbus IRQ[7:1] selected by LINT[3:1]. This bit is cleared by a hard reset and is not affected by a soft reset.
8	AFIE	ACFAIL* Interrupt Enable Writing a 1 to this bit enables the AFINT interrupt condition in the VMEbus Interrupt Status Register (VISTR) to assert the VMEbus IRQ[7:1] selected by LINT[3:1]. This bit is cleared by a hard reset and is not affected by a soft reset.
7	0	Reserved This bit is reserved. Write a 0 when writing to this bit.
6-0	DIRQ[7:1]	Drive VMEbus Interrupt Request [7:1] Writing a 1 to one of these bits causes the VME-MXI-2 to assert the corresponding VMEbus interrupt request. When the interrupt driven from these bits is acknowledged, the value in the VMEbus Status ID Register (VSIDR) is returned and the DIRQ[7:1] bit clears, releasing the interrupt. These bits are cleared by a hard reset and are not affected by a soft reset. The state of the VMEbus interrupt request lines can be monitored in the VMEbus Interrupt Status Register (VISTR).

VMEbus Status ID Register (VSIDR)

VMEbus A16 Offset: 2C (hex)

Attributes: Read/Write 16, 8-bit accessible

15	14	13	12	11	10	9	8
S[15]	S[14]	S[13]	S[12]	S[11]	S[10]	S[9]	S[8]
7	6	5	4	3	2	1	0
S[7]	S[6]	S[5]	S[4]	S[3]	S[2]	S[1]	S[0]

This register contains the Status ID value, which is returned during an interrupt acknowledge cycle for an IRQ[7:1] line that is being driven with the DIRQ[7:1] bits in the VMEbus Interrupt Control Register (VICTR).

Bit	Mnemonic	Description
15-0	S[15:0]	Status ID These bits are cleared by a hard reset and are not affected by a soft reset.

VMEbus Interrupt Acknowledge Register 1 (VIAR1)

VMEbus A16 Offset: 32 (hex)

Attributes: Read Only 16, 8-bit accessible

15	14	13	12	11	10	9	8
I1[15}	I1[14]	I1[13]	I1[12]	I1[11]	I1[10]	I1[9]	I1[8]
7	6	5	4	3	2	1	0
I1[7]	I1[6]	I1[5]	I1[4]	I1[3]	I1[2]	I1[1]	I1[0]

This register generates a VMEbus Interrupt Acknowledge (IACK) cycle for interrupt level 1 when read from the MXIbus and returns the Status ID received from the interrupter. It can generate 16-bit or 8-bit IACK cycles. Generating an 8-bit IACK cycle requires reading offset 33 (hex). When read from the VMEbus, this register does not generate an IACK cycle and returns FFFF (hex).

Bit	Mnemonic	Description
15-0	I1[15:0]	Level 1 Interrupter Status ID These bits return the Status ID received during the IACK cycle.

VMEbus Interrupt Acknowledge Register 2 (VIAR2)

VMEbus A16 Offset: 34 (hex)

Attributes: Read Only 32, 16, 8-bit accessible

31	30	29	28	27	26	25	24
I2[31]	I2[30]	I2[29]	I2[28]	I2[27]	I2[26]	I2[25]	I2[24]
23	22	21	20	19	18	17	16
I2[23]	I2[22]	I2[21]	I2[20]	I2[19]	I2[18]	I2[17]	I2[16]
15	14	13	12	11	10	9	8
I2[15]	I2[14]	I2[13]	I2[12]	I2[11]	I2[10]	I2[9]	I2[8]
7	6	5	4	3	2	1	0
I2[7]	I2[6]	I2[5]	I2[4]	I2[3]	I2[2]	I2[1]	I2[0]

This register generates a VMEbus Interrupt Acknowledge (IACK) cycle for interrupt level 2 when read from the MXIbus and returns the Status ID received from the interrupter. It can generate 32-bit, 16-bit, or 8-bit IACK cycles. Generating an 8-bit IACK cycle requires reading offset 35 (hex). When read from the VMEbus, this register does not generate an IACK cycle and returns FFFFFFFF (hex).

Bit	Mnemonic	Description
31-0	I2[31:0]	Level 2 Interrupter Status ID These bits return the Status ID received during the IACK cycle.

VMEbus Interrupt Acknowledge Register 3 (VIAR3)

VMEbus A16 Offset: 36 (hex)

Attributes: Read Only 16, 8-bit accessible

15	14	13	12	11	10	9	8
I3[15]	I3[14]	I3[13]	I3[12]	I3[11]	I3[10]	I3[9]	I3[8]
7	6	5	4	3	2	1	0
I3[7]	I3[6]	I3[5]	I3[4]	I3[3]	I3[2]	I3[1]	I3[0]

This register generates a VMEbus Interrupt Acknowledge (IACK) cycle for interrupt level 3 when read from the MXIbus and returns the Status ID received from the interrupter. It can generate 16-bit or 8-bit IACK cycles. Generating an 8-bit IACK cycle requires reading offset 37 (hex). When read from the VMEbus, this register does not generate an IACK cycle and returns FFFF (hex).

Bit	Mnemonic	Description
15-0	I3[15:0]	Level 3 Interrupter Status ID These bits return the Status ID received during the IACK cycle.

VMEbus Interrupt Acknowledge Register 4 (VIAR4)

VMEbus A16 Offset: 38 (hex)

Attributes: Read Only 32, 16, 8-bit accessible

31	30	29	28	27	26	25	24
I4[31]	I4[30]	I4[29]	I4[28]	I4[27]	I4[26]	I4[25]	I4[24]
23	22	21	20	19	18	17	16
I4[23]	I4[22]	I4[21]	I4[20]	I4[19]	I4[18]	I4[17]	I4[16]
15	14	13	12	11	10	9	8
I4[15]	I4[14]	I4[13]	I4[12]	I4[11]	I4[10]	I4[9]	I4[8]
7	6	5	4	3	2	1	0
I4[7]	I4[6]	I4[5]	I4[4]	I4[3]	I4[2]	I4[1]	I4[0]

This register generates a VMEbus Interrupt Acknowledge (IACK) cycle for interrupt level 4 when read from the MXIbus and returns the Status ID received from the interrupter. It can generate 32-bit, 16-bit, or 8-bit IACK cycles. Generating an 8-bit IACK cycle requires reading offset 39 (hex). When read from the VMEbus, this register does not generate an IACK cycle and returns FFFFFFFF (hex).

Bit	Mnemonic	Description
31-0	I4[31:0]	Level 4 Interrupter Status ID These bits return the Status ID received during the IACK cycle.

VMEbus Interrupt Acknowledge Register 5 (VIAR5)

VMEbus A16 Offset: 3A (hex)

Attributes: Read Only 16, 8-bit accessible

15	14	13	12	11	10	9	8
I5[15]	I5[14]	I5[13]	I5[12]	I5[11]	I5[10]	I5[9]	I5[8]
7	6	5	4	3	2	1	0
I5[7]	I5[6]	I5[5]	I5[4]	I5[3]	I5[2]	I5[1]	I5[0]

This register generates a VMEbus Interrupt Acknowledge (IACK) cycle for interrupt level 5 when read from the MXIbus and returns the Status ID received from the interrupter. It can generate 16-bit or 8-bit IACK cycles. Generating an 8-bit IACK cycle requires reading offset 3B (hex). When read from the VMEbus, this register does not generate an IACK cycle and returns FFFF (hex).

Bit	Mnemonic	Description
15-0	I5[15:0]	Level 5 Interrupter Status ID These bits return the Status ID received during the IACK cycle.

VMEbus Interrupt Acknowledge Register 6 (VIAR6)

VMEbus A16 Offset: 3C (hex)

Attributes: Read Only 32, 16, 8-bit accessible

31	30	29	28	27	26	25	24
I6[31]	I6[30]	I6[29]	I6[28]	I6[27]	I6[26]	I6[25]	I6[24]
23	22	21	20	19	18	17	16
I6[23]	I6[22]	I6[21]	I6[20]	I6[19]	I6[18]	I6[17]	I6[16]
15	14	13	12	11	10	9	8
I6[15]	I6[14]	I6[13]	I6[12]	I6[11]	I6[10]	I6[9]	I6[8]
7	6	5	4	3	2	1	0
I6[7]	I6[6]	I6[5]	I6[4]	I6[3]	I6[2]	I6[1]	I6[0]

This register generates a VMEbus Interrupt Acknowledge (IACK) cycle for interrupt level 6 when read from the MXIbus and returns the Status ID received from the interrupter. It can generate 32-bit, 16-bit, or 8-bit IACK cycles. Generating an 8-bit IACK cycle requires reading offset 3D (hex). When read from the VMEbus, this register does not generate an IACK cycle and returns FFFFFFFF (hex).

Bit	Mnemonic	Description
31-0	I6[31:0]	Level 6 Interrupter Status ID These bits return the Status ID received during the IACK cycle.

VMEbus Interrupt Acknowledge Register 7 (VIAR7)

VMEbus A16 Offset: 3E (hex)

Attributes: Read Only 16, 8-bit accessible

15	14	13	12	11	10	9	8
I7[15]	I7[14]	I7[13]	I7[12]	I7[11]	I7[10]	I7[9]	I7[8]
7	6	5	4	3	2	1	0
I7[7]	I7[6]	I7[5]	I7[4]	I7[3]	I7[2]	I7[1]	I7[0]

This register generates a VMEbus Interrupt Acknowledge (IACK) cycle for interrupt level 7 when read from the MXIbus and returns the Status ID received from the interrupter. It can generate 16-bit or 8-bit IACK cycles. Generating an 8-bit IACK cycle requires reading offset 3F (hex). When read from the VMEbus, this register does not generate an IACK cycle and returns FFFF (hex).

Bit	Mnemonic	Description
15-0	I7[15:0]	Level 7 Interrupter Status ID These bits return the Status ID received during the IACK cycle.

VMEbus A24/A32 Registers

Some of the registers on the VME-MXI-2 are accessible only within the A24 or A32 space that is allocated to the VME-MXI-2. The following are register descriptions of some of these registers. See Table 4-2 for a register map of these registers. The table gives the mnemonic, offset from the base address, access type (read only, write only, or read/write), access size, and register name.

To enable access to the A24 or A32 space on the VME-MXI-2, first write the desired base address to the VXIbus Offset Register (VOR), then set the A24/A32 ENABLE bit in the VXIbus Control Register (VCR). Prior to these actions, the VME-MXI-2 will not respond to any A24 or A32 access—only the A16 registers are available. Read the VXIbus ID Register (VIDR) to determine the address space (either A24 or A32) to which the VME-MXI-2 will respond. To determine the amount of space that the VME-MXI-2 requires, read the REQMEM [3:0] bits in the VXIbus Device Type Register (VDTR). The base address of the VME-MXI-2 A24 or A32 space must be located on a boundary of the size being requested. If you are using a VXIbus Resource Manager, this will be done automatically and you can just read the VIDR and VOR to determine the address space (A24 or A32) and base address at which the VME-MXI-2 is located. See the register descriptions of the VIDR, VDTR, VCR, and VOR for more information.

These registers occupy the first 4 KB of address space allocated to the VME-MXI-2. Any access to the VME-MXI-2 A24/A32 space beyond the first 4 KB (address offsets above FFF hex) will map to the onboard DRAM SIMM sockets. The address offset shown in each register description is the offset from the base A24/A32 address of the VME-MXI-2 as defined by the VIDR and VOR registers.

Most of these registers are used to configure the two onboard DMA controllers. The two DMA controllers are identical to each other but are independent; they can be used simultaneously without affecting the operation of each other. Because the registers for the two DMA controllers are identical, this section describes only one set of registers, but the descriptions apply to both DMA controllers. The registers for DMA Controller 1 begin at offset D00 from the VME-MXI-2 module base A24/A32 address, while the registers for DMA Controller 2 begin at offset E00 as shown in Table 4-2. For an example of how to use the DMA controllers, refer to Appendix F, *DMA Programming Examples*.

Table 4-2. VME-MXI-2 VMEbus A24/A32 Register Map

Mnemonic	Offset (Hex)	Access Type	Access Size	Register Name
DMAICR	8	Read/Write	16/8 bit	DMA Interrupt Configuration
DMAIER	12	Read/Write	16/8 bit	DMA Interrupt Enable
DMAISIDR	20	Read/Write	16/8 bit	DMA Interrupt Status/ID
VMSR2/ VMCR2	758	Read/Write	32/16/8 bit	VME-MXI-2 Status/Control Register 2
SMSR/ SMCR	C40	Read/Write	32/16/8 bit	Shared MXIbus Status/Control Register
CHOR1	D00	Read/Write	32/16/8 bit	DMA Channel 1 Operation
CHCR1	D04	Read/Write	32/16/8 bit	DMA Channel 1 Control
TCR1	D08	Read/Write	32/16/8 bit	DMA Channel 1 Transfer Count
SCR1	D0C	Read/Write	32/16/8 bit	DMA Channel 1 Source Configuration
SAR1	D10	Read/Write	32/16/8 bit	DMA Channel 1 Source Address
DCR1	D14	Read/Write	32/16/8 bit	DMA Channel 1 Destination Configuration
DAR1	D18	Read/Write	32/16/8 bit	DMA Channel 1 Destination Address
CHSR1	D3C	Read Only	32/16/8 bit	DMA Channel 1 Status
FCR1	D40	Read Only	32/16/8 bit	DMA Channel 1 FIFO Count
CHOR2	E00	Read/Write	32/16/8 bit	DMA Channel 2 Operation
CHCR2	E04	Read/Write	32/16/8 bit	DMA Channel 2 Control
TCR2	E08	Read/Write	32/16/8 bit	DMA Channel 2 Transfer Count
SCR2	E0C	Read/Write	32/16/8 bit	DMA Channel 2 Source Configuration
SAR2	E10	Read/Write	32/16/8 bit	DMA Channel 2 Source Address
DCR2	E14	Read/Write	32/16/8 bit	DMA Channel 2 Destination Configuration
DAR2	E18	Read/Write	32/16/8 bit	DMA Channel 2 Destination Address
CHSR2	E3C	Read Only	32/16/8 bit	DMA Channel 2 Status
FCR2	E40	Read Only	32/16/8 bit	DMA Channel 2 FIFO Count

DMA Interrupt Configuration Register (DMAICR)

VMEbus A24 or A32 Offset: 8 (hex)

Attributes: Read/Write 16, 8-bit accessible

15	14	13	12	11	10	9	8
SID8	SIDLA	1	0	1	0	0	0
7	6	5	4	3	2	1	0
ISTAT	0	0	0	0	ILVL[2]	ILVL[1]	ILVL[0]

This register controls aspects of the DMA interrupt that are configurable. Although the two DMA controllers are independent, they share a common interrupt condition.

Bit	Mnemonic	Description
15	SID8	8-bit Status/ID This bit selects between an 8-bit or 16-bit Status/ID when the DMA interrupt is acknowledged. When this bit is set, the VME-MXI-2 responds to IACK cycles of any size and supplies 8 bits of Status/ID information. The information supplied for the 8-bit Status/ID is selected using the SIDLA bit. When this bit is clear, the VME-MXI-2 responds to 16-bit or 32-bit IACK cycles and supplies 16 bits of Status/ID information. The 16 bits of Status/ID are composed of the contents of the DMA Interrupt Status/ID Register (DMAISIDR) and the logical address of the VME-MXI-2. The DMAISIDR appears on the upper 8 bits and the logical address appears on the lower 8 bits during the IACK cycle. When this bit is clear, the VME-MXI-2 does not respond to 8-bit IACK cycles. This bit is cleared on a hard reset and is not affected by a soft reset.

14	SIDLA	Logical Address Status/ID When the SID8 bit is set, this bit selects what information is provided during IACK cycles for the DMA interrupt. This bit should not be set when SID8 is clear. When this bit is set, the logical address of the VME-MXI-2 is used as the Status/ID information. When this bit is clear, the contents of the DMAISIDR are used. This bit is cleared on a hard reset and is not affected by a soft reset.
13	1	Reserved This bit is reserved. It must be initialized to 1 for the DMA interrupt to operate properly. This bit is cleared on a hard reset and is not affected by a soft reset.
12	0	Reserved This bit is reserved. Write this bit with 0 when writing the DMAICR. The value this bit returns when read is meaningless.
11	1	Reserved This bit is reserved. It must be initialized to 1 for the DMA interrupt to operate properly. This bit is cleared on a hard reset and is not affected by a soft reset.
10-8	0	Reserved These bits are reserved. Write each of these bits with 0 when writing the DMAICR. The value these bits return when read is meaningless.
7	ISTAT	DMA Interrupt Status bit This read-only bit indicates the status of the DMA interrupt. When this bit returns 1, it means that the interrupt condition is present. Once the condition is present, it will remain until re-armed. Notice that even though the VME-MXI-2 releases the IRQ* line on the VMEbus during the IACK cycle, the IACK cycle does not clear this status bit. See Appendix F, *DMA Programming Examples*, for more information on re-arming the DMA interrupt.

6-3	0	Reserved These bits are reserved. Write each of these bits with 0 when writing the DMAICR. The value these bits return when read is meaningless.
2-0	ILVL[2:0]	DMA Interrupt Level These bits select the VMEbus interrupt level that the DMA interrupt condition will assert. Write a 7 to these bits for IRQ7*, write a 6 for IRQ6*, and so on. These bits must be initialized to a value between 7 and 1 for the DMA interrupt to operate properly. These bits are cleared on a hard reset and are not affected by a soft reset.

DMA Interrupt Enable Register (DMAIER)

VMEbus A24 or A32 Offset: 12 (hex)

Attributes: Read/Write 16, 8-bit accessible

15	14	13	12	11	10	9	8
0	0	0	0	DMAIEN	0	0	ENABLE
7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0

This register enables mapping of the DMA interrupt to the VMEbus. The interrupt can be asserted only on the VMEbus and must be routed through the VXIbus Interrupt Configuration Register (VICR) if the interrupt handler is located across the MXIbus. This register is also used to re-arm the DMA interrupt after one has occurred by first disabling the interrupt using this register, next clearing the DMA interrupt condition using either the CLRDONE bit in the DMA Channel Operation Register (CHORx) or the CLRDMAIE or CLRDONEIE bit in the DMA Channel Control Register (CHCRx), and then re-enabling the interrupt using this register.

Bit	Mnemonic	Description
15-12	0	Reserved These bits are reserved. Write each of these bits with 0 when writing the DMAIER. The value these bits return when read is meaningless.
11	DMAIEN	DMA Interrupt Enable This bit is used in combination with the ENABLE bit to enable the DMA interrupt to be mapped to the VMEbus. To enable the interrupt write a 1 to both bits. To disable the interrupt write a 1 to this bit and a 0 to the ENABLE bit. This bit returns a 1 when read if the interrupt is enabled and a 0 if the interrupt is disabled. The interrupt is disabled on a hard reset and is not affected by a soft reset.
10-9	0	Reserved These bits are reserved. Write each of these bits with 0 when writing the DMAIER. The value these bits return when read is meaningless.

8	ENABLE	Enable Interrupt This bit controls whether the interrupt is enabled or disabled when writing to the DMAIER. Write this bit with a 1 to enable the interrupt or a 0 to disable the interrupt. The DMAIEN bit should always be written with a 1. This bit always returns a 0 when read.
7-0	0	Reserved These bits are reserved. Write each of these bits with 0 when writing the DMAIER. The value these bits return when read is meaningless.

DMA Interrupt Status/ID Register (DMAISIDR)

VMEbus A24 or A32 Offset: 20 (hex)

Attributes: Read/Write 16, 8-bit accessible

15	14	13	12	11	10	9	8
0	0	0	0	0	0	0	0
7	6	5	4	3	2	1	0
DMASID[7]	DMASID[6]	DMASID[5]	DMASID[4]	DMASID[3]	0	1	1

This register provides the Status/ID information during IACK cycles for the DMA interrupt. If SID8 and SIDLA are both set in the DMA Interrupt Configuration Register (DMAICR), only the VME-MXI-2 module's logical address is provided and this register is not used. If SID8 is clear in the DMAICR (16-bit Status/ID) this register provides the upper 8 bits of the Status/ID and the VME-MXI-2 module's logical address is placed on the lower 8 bits.

Bit	Mnemonic	Description
15-8	0	Reserved These bits are reserved. Write each of these bits with 0 when writing the DMAISIDR. The value these bits return when read is meaningless.
7-3	DMASID[7:3]	DMA Status/ID 7 through 3 These bits can be written with any value to uniquely identify the DMA interrupt during an IACK cycle. When SID8 is clear in the DMAICR (16-bit Status/ID), these bits provide bits 15 through 11 of the Status/ID. When SID8 is set (8-bit Status/ID) and SIDLA is clear in the DMAICR, these bits provide bits 7 through 3 of the Status/ID. These bits are cleared on a hard reset and are not affected by a soft reset.

2-0	‘011’	DMA Status/ID 2 through 0 When SID8 is clear in the DMAICR (16-bit Status/ID), these bits provide bits 10 through 8 of the Status/ID. When SID8 is set (8-bit Status/ID) and SIDLA is clear in the DMAICR, these bits provide bits 2 through 0 of the Status/ID. These bits return 011 (binary) during IACK cycles and 000 (binary) when read directly.

VME-MXI-2 Status/Control Register 2 (VMSR2/VMCR2)

VMEbus A24 or A32 Offset: 758 (hex)

Attributes: Read/Write 32, 16, 8-bit accessible

31	30	29	28	27	26	25	24
0	0	0	0	0	0	0	0
23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8
0	0	0	0	0	0	0	0
7	6	5	4	3	2	1	0
IOCONFIG	0	0	0	0	0	0	1

This register enables access to the VME-MXI-2 onboard EEPROM. For more information on changing configuration settings in the EEPROM, refer to Appendix B, *Programmable Configurations*.

Bit	Mnemonic	Description
31-8	0	Reserved These bits are reserved. Write these bits with 0 when writing the VMCR2.
7	IOCONFIG	I/O Configuration Space Enable This bit controls accesses to I/O configuration space (the onboard EEPROM). A device requesting access to the I/O configuration space must set this bit. When this bit is set, any accesses through the A24/A32 inward window that would normally map to the onboard DRAM (address offsets above FFF hex) instead map to the configuration space, accessing the EEPROM. On completion of configuration activity, the master should then clear this bit. Notice that this bit cannot be locked. The master must ensure that it is the only device accessing VME-MXI-2 address offsets above FFF (hex) while this bit is set. This bit is cleared on a hard reset and is not affected by a soft reset.

6-1	0	Reserved These bits are reserved. Write these bits with 0 when writing to the VMCR2.
0	1	Reserved This bit is reserved. Write this bit with 1 when writing to the VMCR2.

Shared MXIbus Status/Control Register (SMSR/SMCR)

VMEbus A24 or A32 Offset: C40 (hex)

Attributes: Read/Write 32, 16, 8-bit accessible

31	30	29	28	27	26	25	24
0	0	DMA2MBS	DMA1MBS	DMAMB S/N*	0	0	0
23	22	21	20	19	18	17	16
1	1	FAIR	0	0	PAREN	0	1
15	14	13	12	11	10	9	8
0	0	0	0	0	0	0	1
7	6	5	4	3	2	1	0
0	0	0	0	MBTO[3]	MBTO[2]	MBTO[1]	MBTO[0]

This register provides control bits for the configurable features of the MXIbus interface on the VME-MXI-2.

Bit	Mnemonic	Description
31-30	0	Reserved These bits are reserved. Write these bits with 0 when writing to the SMCR.
29	DMA2MBS	DMA Controller 2 MXIbus Block Select This bit, combined with the DMAMB S/N* bit, controls whether block cycles to the MXIbus from DMA Controller 2 are performed as normal MXIbus block cycles or synchronous MXIbus burst cycles. Non-block cycles to the MXIbus are unaffected by this bit. Write a 1 to both DMA2MBS and DMAMB S/N* to cause DMA Controller 2 block cycles to the MXIbus to be synchronous burst cycles. Write a 1 to DMA2MBS and a 0 to DMAMB S/N* to cause DMA Controller 2 block cycles to the MXIbus to be normal block cycles. When DMA2MBS is written with a 0 the bit is unaffected (it remains in whatever state it was in before the write). This bit returns 1 when read if synchronous burst cycles are enabled and 0 when normal block cycles are enabled.

		Notice that synchronous MXIbus burst cycles cannot be used for the source or destination of a DMA operation when both are located on the MXIbus. In such a case, you must either program this bit to use normal MXIbus block cycles, or program the DMA Source Configuration Register 2 (SCR2) and the DMA Destination Configuration Register 2 (DCR2) to both use single (non-block) cycles by clearing the BLOCKEN bit. A hard reset causes block cycles to the MXIbus from DMA Controller 2 to be normal block cycles. This bit is not affected by soft resets.
28	DMA1MBS	DMA Controller 1 MXIbus Block Select This bit performs the same function as DMA2MBS but for DMA Controller 1.
27	DMAMB S/N*	DMA MXIbus Block Synchronous/Normal* When this bit is written with a 1, any DMAxMBS bit that is also being written with a 1 is set (synchronous MXIbus burst cycles). When this bit is written with a 0, any DMAxMBS bit that is being written with a 1 is cleared (normal MXIbus block cycles). The value this bit returns when read is meaningless.
26-24	0	Reserved These bits are reserved. Write these bits with 0 when writing to the SMCR.
23-22	1	Reserved These bits are reserved. Write these bits with 1 when writing to the SMCR.
21	FAIR	MXIbus Fair Requester Setting this bit enables the MXIbus fair requester protocol. When this bit is clear, the VME-MXI-2 is an unfair requester on the MXIbus. Refer to Chapter 6, *VXIplug&play for the VME-MXI-2*, or Appendix B, *Programmable Configurations*, for more information on the Fair MXIbus Requester protocol. On a hard reset, this bit is initialized to the value stored in the onboard EEPROM for this bit.

20-19	0	Reserved These bits are reserved. Write these bits with 0 when writing to the SMCR.
18	PAREN	MXIbus Parity Enable Setting this bit enables the checking of MXIbus parity. When this bit is clear, the VME-MXI-2 does not check MXIbus parity. Refer to Chapter 6, *VXIplug&play for the VME-MXI-2*, or Appendix B, *Programmable Configurations*, for more information on MXIbus parity checking. On a hard reset, this bit is initialized to the value stored in the onboard EEPROM for this bit.
17	0	Reserved This bit is reserved. Write this bit with 0 when writing to the SMCR.
16	1	Reserved This bit is reserved. Write this bit with 1 when writing to the SMCR.
15-9	0	Reserved These bits are reserved. Write these bits with 0 when writing to the SMCR.
8	1	Reserved This bit is reserved. Write this bit with 1 when writing to the SMCR.
7-4	0	Reserved These bits are reserved. Write these bits with 0 when writing to the SMCR.
3-0	MBTO[3:0]	MXIbus Timeout Value The MBTO[3:0] bits determine the amount of time the VME-MXI-2 will wait before terminating a MXIbus cycle by asserting BERR* when acting as the MXIbus System Controller.

The following table lists the values to write to these bits for all possible times. Refer to Chapter 6, *VXIplug&play for the VME-MXI-2*, or Appendix B, *Programmable Configurations*, for more information on the MXIbus timer. On a hard reset, these bits are initialized to the value stored in the onboard EEPROM for these bits.

Time Limit	Value (hex)
Timer Disabled	0
8 µs	1
15 µs	2
30 µs	3
60 µs	4
125 µs	5
250 µs	6
500 µs	7
1 ms	8 (default)
2 ms	9
4 ms	A
8 ms	B
16 ms	C
32 ms	D
64 ms	E
128 ms	F

DMA Channel Operation Register (CHORx)

CHOR1 VMEbus A24 or A32 Offset: D00 (hex)

CHOR2 VMEbus A24 or A32 Offset: E00 (hex)

Attributes: Read/Write 32, 16, 8-bit accessible

31	30	29	28	27	26	25	24
0	0	0	0	0	0	0	0
23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8
0	0	0	0	0	0	0	0
7	6	5	4	3	2	1	0
CLRDONE	0	0	FRESET	ABORT	STOP	0	START

This register is used to control overall operation of the DMA controller, such as starting a transfer after all the other DMA registers have been programmed.

Bit	Mnemonic	Description
31-8	0	Reserved These bits are reserved. Write each of these bits with 0 when writing the CHORx. The value these bits return when read is meaningless.
7	CLRDONE	Clear DONE This bit can be written with a 1 to clear the DONE bit in the DMA Channel Status Register (CHSRx). The DONE bit also clears automatically when a new DMA operation is started. It is not necessary to clear the CLRDONE bit after writing a 1 to it.
6-5	0	Reserved These bits are reserved. Write each of these bits with 0 when writing the CHORx. The value these bits return when read is meaningless.

4	FRESET	DMA FIFO Reset This bit can be written with a 1 to reset the DMA FIFO. It is necessary to reset the FIFO after an ABORT operation or if a DMA transfer ends due to an error condition. It is not necessary to clear the FRESET bit after writing a 1 to it. The FIFO is reset by a hard reset and is not affected by a soft reset.
3	ABORT	Abort DMA Operation This bit can be written with a 1 to abort the current DMA operation. When a DMA operation is aborted, it is possible that some data will have been read from the source that does not get written to the destination. The ABORT bit automatically clears when a new DMA operation is started. It is not necessary to clear the ABORT bit after writing a 1 to it.
2	STOP	Stop DMA Operation This bit can be written with a 1 to stop the current DMA operation. After the STOP bit is set, the DMA controller immediately stops reading data from the source and stops writing data to the destination as soon as the FIFO is emptied (unlike an ABORT operation, any data already read from the source is written to the destination before the DMA controller stops). After stopping a DMA operation, the same operation can be allowed to finish by writing the START bit with a 1, or a new operation can be started by reprogramming the DMA registers. After setting the STOP bit, the DMA registers should not be reprogrammed until the DONE bit in the DMA Channel Status Register (CHSRx) becomes 1. The STOP bit will automatically clear when the START bit is set. It is not necessary to clear the STOP bit after writing a 1 to it.
1	0	Reserved This bit is reserved. Write this bit with 0 when writing the CHORx. The value this bit returns when read is meaningless.

0	START	Start DMA Operation This bit should be written with a 1 to start a new DMA operation after the other DMA registers are initialized. This bit can also be set after a DMA operation has been stopped with the STOP bit to allow the operation to complete. When restarting a stopped DMA operation, the START bit should not be set until the DONE bit becomes 1 after setting the STOP bit. After setting the START bit, the DONE bit becomes clear and the DMA controller begins performing the operation.

DMA Channel Control Register (CHCRx)

CHCR1 VMEbus A24 or A32 Offset: D04 (hex)

CHCR2 VMEbus A24 or A32 Offset: E04 (hex)

Attributes: Read/Write 32, 16, 8-bit accessible

31	30	29	28	27	26	25	24
SET DMAIE	CLR DMAIE	0	0	0	0	SET DONEIE	CLR DONEIE
23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8
0	1	0	0	0	0	0	0
7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0

This register is used to individually enable the two DMA controllers to assert the DMA interrupt. Because the DMA interrupt is shared between the two DMA controllers, this register allows either DMA controller (or both) to use the interrupt.

Bit	Mnemonic	Description
31	SET DMAIE	Set DMA Interrupt Enable Writing a 1 to this bit enables the corresponding DMA controller to assert the DMA interrupt. Writing a 0 to this bit has no effect. This bit returns a 1 when read if the corresponding DMA controller is enabled to assert the interrupt and a 0 if it is disabled. The interrupt is disabled by a hard reset and is not affected by a soft reset.
30	CLR DMAIE	Clear DMA Interrupt Enable Writing a 1 to this bit disables the corresponding DMA controller from asserting the DMA interrupt. Writing a 0 to this bit has no effect. This bit returns a 0 when read if the corresponding DMA controller is enabled to assert the interrupt and a 1 if it is disabled. The interrupt is disabled by a hard reset and is not affected by a soft reset.

29-26	0	Reserved These bits are reserved. Write each of these bits with 0 when writing the CHCRx. The value these bits return when read is meaningless.
25	SET DONEIE	Set DONE Interrupt Enable Writing a 1 to this bit enables the corresponding DMA controller to interrupt on the DONE condition in the DMA Channel Status Register (CHSRx). Writing a 0 to this bit has no effect. This bit returns a 1 when read if the corresponding DMA controller is enabled to interrupt on the DONE condition and a 0 if it is disabled. The interrupt is disabled by a hard reset and is not affected by a soft reset.
24	CLR DONEIE	Clear DONE Interrupt Enable Writing a 1 to this bit disables the corresponding DMA controller from interrupting on the DONE condition in the DMA Channel Status Register (CHSRx). Writing a 0 to this bit has no effect. This bit returns a 0 when read if the corresponding DMA controller is enabled to interrupt on the DONE condition and a 1 if it is disabled. The interrupt is disabled by a hard reset and is not affected by a soft reset.
23-15	0	Reserved These bits are reserved. Write each of these bits with 0 when writing the CHCRx. The value these bits return when read is meaningless.
14	1	Reserved This bit is reserved. It must be initialized to 1 for the DMA controller to operate properly. This bit is cleared on a hard reset and is not affected by a soft reset.
13-0	0	Reserved These bits are reserved. Write each of these bits with 0 when writing the CHCRx. The value these bits return when read is meaningless.

DMA Transfer Count Register (TCRx)

TCR1 VMEbus A24 or A32 Offset: D08 (hex)

TCR2 VMEbus A24 or A32 Offset: E08 (hex)

Attributes: Read/Write 32, 16, 8-bit accessible

31	30	29	28	27	26	25	24
TC[31]	TC[30]	TC[29]	TC[28]	TC[27]	TC[26]	TC[25]	TC[24]
23	22	21	20	19	18	17	16
TC[23]	TC[22]	TC[21]	TC[20]	TC[19]	TC[18]	TC[17]	TC[16]
15	14	13	12	11	10	9	8
TC[15]	TC[14]	TC[13]	TC[12]	TC[11]	TC[10]	TC[9]	TC[8]
7	6	5	4	3	2	1	0
TC[7]	TC[6]	TC[5]	TC[4]	TC[3]	TC[2]	TC[1]	TC[0]

This register stores the number of bytes to be transferred.

Bit	Mnemonic	Description
31-0	TC[31:0]	Transfer Count

The transfer count is the number of bytes to be transferred from the source to the destination regardless of the width of the data transfers. The transfer count should be programmed before the DMA operation is started. When either the source or destination is using 64-bit data transfers, the transfer count programmed must be divisible by 8. The transfer count is decremented by 1, 2, 4, or 8—depending on the source data transfer width—as data is read from the source. Reading the transfer count will return the number of bytes remaining to be read from the source.

The transfer count has a limit when the *source* of the DMA operation will use synchronous MXIbus burst transfers. This limit does *not* apply when the *destination* uses synchronous MXIbus burst transfers. The limit differs depending on the setting of the MXIbus **Transfer Limit** control in the VXI*plug&play* soft front panel,

which is described in Chapter 6, *VXIplug&play for the VME-MXI-2*. By default, the **Transfer Limit** is set to **Unlimited**; with this setting, the transfer count must not exceed 32 KB (8000 hex) if the source of the DMA operation will use synchronous MXIbus burst transfers. If you changed the setting of the MXIbus **Transfer Limit** control in the soft front panel to something other than **Unlimited**, you must program the transfer count to a multiple of the setting of the **Transfer Limit** control. The only exception is that you can also program the transfer count to be smaller than the **Transfer Limit** control setting. For example, assume the **Transfer Limit** control in the soft front panel is set to 256. In this case the transfer count must be programmed in the range of 1 to 256 or set to one of 512, 768, 1024, and so on.

DMA Source Configuration Register (SCRx)

SCR1 VMEbus A24 or A32 Offset: D0C (hex)

SCR2 VMEbus A24 or A32 Offset: E0C (hex)

Attributes: Read/Write 32, 16, 8-bit accessible

31	30	29	28	27	26	25	24
0	0	0	0	0	0	0	0
23	22	21	20	19	18	17	16
1	1	1	0	0	0	0	0
15	14	13	12	11	10	9	8
0	BLOCKEN	0	0	0	ASCEND	TSIZE[1]	TSIZE[0]
7	6	5	4	3	2	1	0
PORT[1]	PORT[0]	AM[5]	AM[4]	AM[3]	AM[2]	AM[1]	AM[0]

This register is used to configure how the DMA controller will access the source of the data.

Bit	Mnemonic	Description
31-24	0	Reserved These bits are reserved. Write each of these bits with 0 when writing the SCRx. The value these bits return when read is meaningless.
23-21	1	Reserved These bits are reserved. They must be initialized to 111 (binary) for the DMA controller to operate properly. These bits are cleared on a hard reset and are not affected by a soft reset.
20-15	0	Reserved These bits are reserved. Write each of these bits with 0 when writing the SCRx. The value these bits return when read is meaningless.

14	BLOCKEN	Block Mode DMA Write a 1 to this bit to cause the DMA controller to perform block-mode transfers to the source. During block mode, the DMA controller keeps the AS* signal asserted throughout a series of read cycles to the source. The DMA controller automatically deasserts and reasserts the AS* signal when it reaches the appropriate transfer size limit for the bus on which the source is located (for example 256 bytes on the VMEbus). In addition, if the corresponding DMAxMBS bit is set in the Shared MXIbus Control Register (SMCR), any block-mode cycles from the DMA controller to the MXIbus are performed as a synchronous burst cycle. When this bit is clear, the DMA controller performs a series of standard single read cycles to the source deasserting the AS* signal after each cycle. This bit is cleared by a hard reset and is not affected by a soft reset.
13-11	0	Reserved These bits are reserved. Write each of these bits with 0 when writing the SCRx. The value these bits return when read is meaningless.
10	ASCEND	Ascending Addresses Write a 1 to this bit to cause the DMA controller to increment the source address between each data transfer of the DMA operation. The source address is incremented by 1, 2, 4, or 8—depending on the width of the source data transfers—resulting in the DMA controller accessing locations on the source in ascending order. When this bit is clear, the DMA controller does not increment the source address throughout the DMA operation, resulting in all the data coming from the same location on the source. This bit is cleared by a hard reset and is not affected by a soft reset.

9-8	TSIZE[1:0]	Transfer Size These bits control the transfer size to be used to access the source. Write these bits with 01 (binary) to perform 8-bit transfers, 10 (binary) to perform 16-bit transfers, and 11 (binary) to perform 32-bit or 64-bit transfers. The DMA controller can distinguish between 32-bit and 64-bit transfers using the AM[5:0] bits. These bits are cleared by a hard reset and are not affected by a soft reset.
7-6	PORT[1:0]	Port These bits control the bus on which the source is located. Write these bits with 01 (binary) if the source is DRAM onboard the VME-MXI-2, 10 (binary) if the source is on the VMEbus, and 11 (binary) if the source is on the MXIbus. These bits are cleared by a hard reset and are not affected by a soft reset.
5-0	AM[5:0]	Address Modifiers These bits provide the address modifier code used to access the source. Even when the source is on the MXIbus, the VMEbus equivalent address modifier code should be written to these bits (the MXIbus address modifier code should not be written to these bits, because the DMA controller converts VMEbus address modifier codes when the source is on the MXIbus). Table F-1, *Address Modifier Codes*, in Appendix F, *DMA Programming Examples*, describes the address modifier codes that can be written to these bits. When the source is DRAM onboard the VME-MXI-2, these bits must be written with 0. These bits are cleared by a hard reset and are not affected by a soft reset.

DMA Source Address Register (SARx)

SAR1 VMEbus A24 or A32 Offset: D10 (hex)

SAR2 VMEbus A24 or A32 Offset: E10 (hex)

Attributes: Read/Write 32, 16, 8-bit accessible

31	30	29	28	27	26	25	24
SA[31]	SA[30]	SA[29]	SA[28]	SA[27]	SA[26]	SA[25]	SA[24]
23	22	21	20	19	18	17	16
SA[23]	SA[22]	SA[21]	SA[20]	SA[19]	SA[18]	SA[17]	SA[16]
15	14	13	12	11	10	9	8
SA[15]	SA[14]	SA[13]	SA[12]	SA[11]	SA[10]	SA[9]	SA[8]
7	6	5	4	3	2	1	0
SA[7]	SA[6]	SA[5]	SA[4]	SA[3]	SA[2]	SA[1]	SA[0]

This register stores the base address to be used for the source.

Bit	Mnemonic	Description
31-0	SA[31:0]	Source Address These bits store the address used to access the source. The value of these bits is modified after each successful data transfer to the source during the DMA operation, according to the ASCEND bit in the DMA Source Configuration Register (SCRx). If the initial value of these bits is not aligned to the transfer size indicated by the TSIZE[1:0] bits in the SCRx, the DMA controller performs smaller transfers until address alignment occurs. However, if 64-bit data transfers are used for the source, this register must be programmed with an address divisible by 8. In the case that the DMA controller terminates due to an error with the source transfers, these bits would indicate the address that caused the error. When the source is DRAM onboard the VME-MXI-2, these bits must be programmed with the offset of the source location within the VME-MXI-2 module's address space, not the VMEbus address

of the source. To compute this value from the VMEbus address of the source, just subtract the VME-MXI-2 module's A24 or A32 base address.

DMA Destination Configuration Register (DCRx)

DCR1 VMEbus A24 or A32 Offset: D14 (hex)

DCR2 VMEbus A24 or A32 Offset: E14 (hex)

Attributes: Read/Write 32, 16, 8-bit accessible

31	30	29	28	27	26	25	24
0	0	0	0	0	0	0	0
23	22	21	20	19	18	17	16
1	1	1	0	0	0	0	0
15	14	13	12	11	10	9	8
0	BLOCKEN	0	0	0	ASCEND	TSIZE[1]	TSIZE[0]
7	6	5	4	3	2	1	0
PORT[1]	PORT[0]	AM[5]	AM[4]	AM[3]	AM[2]	AM[1]	AM[0]

This register is used to configure how the DMA controller accesses the destination of the data.

Bit	Mnemonic	Description
31-24	0	Reserved These bits are reserved. Write each of these bits with 0 when writing the DCRx. The value these bits return when read is meaningless.
23-21	1	Reserved These bits are reserved. They must be initialized to 111 (binary) for the DMA controller to operate properly. These bits are cleared on a hard reset and are not affected by a soft reset.
20-15	0	Reserved These bits are reserved. Write each of these bits with 0 when writing the DCRx. The value these bits return when read is meaningless.

14	BLOCKEN	Block Mode DMA Write a 1 to this bit to cause the DMA controller to perform block-mode transfers to the destination. During block mode, the DMA controller keeps the AS* signal asserted throughout a series of write cycles to the destination. The DMA controller automatically deasserts and reasserts the AS* signal when it reaches the appropriate transfer size limit for the bus on which the destination is located (for example 256 bytes on the VMEbus). In addition, if the corresponding DMAxMBS bit is set in the Shared MXIbus Control Register (SMCR), any block-mode cycles from the DMA controller to the MXIbus are performed as a synchronous burst cycle. When this bit is clear, the DMA controller performs a series of standard single write cycles to the destination deasserting the AS* signal after each cycle. This bit is cleared by a hard reset and is not affected by a soft reset.
13-11	0	Reserved These bits are reserved. Write each of these bits with 0 when writing the DCRx. The value these bits return when read is meaningless.
10	ASCEND	Ascending Addresses Write a 1 to this bit to cause the DMA controller to increment the destination address between each data transfer of the DMA operation. The destination address is incremented by 1, 2, 4, or 8—depending on the width of the destination data transfers—resulting in the DMA controller accessing locations on the destination in ascending order. When this bit is clear, the DMA controller does not increment the destination address throughout the DMA operation, resulting in all the data going to the same location on the destination. This bit is cleared by a hard reset and is not affected by a soft reset.

9-8	TSIZE[1:0]	Transfer Size These bits control the transfer size to be used to access the destination. Write these bits with 01 (binary) to perform 8-bit transfers, 10 (binary) to perform 16-bit transfers, and 11 (binary) to perform 32-bit or 64-bit transfers. The DMA controller can distinguish between 32-bit and 64-bit transfers using the AM[5:0] bits. These bits are cleared by a hard reset and are not affected by a soft reset.
7-6	PORT[1:0]	Port These bits control the bus on which the destination is located. Write these bits with 01 (binary) if the destination is DRAM onboard the VME-MXI-2, 10 (binary) if the destination is on the VMEbus, and 11 (binary) if the destination is on the MXIbus. These bits are cleared by a hard reset and are not affected by a soft reset.
5-0	AM[5:0]	Address Modifiers These bits provide the address modifier code used to access the destination. Even when the destination is on the MXIbus, the equivalent VMEbus address modifier code should be written to these bits (the MXIbus address modifier code should not be written to these bits because the DMA controller converts VMEbus address modifier codes when the destination is on the MXIbus). Table F-1, *Address Modifier Codes*, in Appendix F, *DMA Programming Examples*, describes the address modifier codes that can be written to these bits. When the destination is DRAM onboard the VME-MXI-2, these bits must be written with 0. These bits are cleared by a hard reset and are not affected by a soft reset.

DMA Destination Address Register (DARx)

DAR1 VMEbus A24 or A32 Offset: D18 (hex)

DAR2 VMEbus A24 or A32 Offset: E18 (hex)

Attributes: Read/Write 32, 16, 8-bit accessible

31	30	29	28	27	26	25	24
DA[31]	DA[30]	DA[29]	DA[28]	DA[27]	DA[26]	DA[25]	DA[24]
23	22	21	20	19	18	17	16
DA[23]	DA[22]	DA[21]	DA[20]	DA[19]	DA[18]	DA[17]	DA[16]
15	14	13	12	11	10	9	8
DA[15]	DA[14]	DA[13]	DA[12]	DA[11]	DA[10]	DA[9]	DA[8]
7	6	5	4	3	2	1	0
DA[7]	DA[6]	DA[5]	DA[4]	DA[3]	DA[2]	DA[1]	DA[0]

This register stores the base address to be used for the destination.

Bit	Mnemonic	Description
31-0	DA[31:0]	Destination Address These bits store the address used to access the destination. The value of these bits is modified after each successful data transfer to the destination during the DMA operation, according to the ASCEND bit in the DMA Destination Configuration Register (DCRx). If the initial value of these bits is not aligned to the transfer size indicated by the TSIZE[1:0] bits in the DCRx, the DMA controller performs smaller transfers until address alignment occurs. However, if 64-bit data transfers are used for the destination, this register must be programmed with an address divisible by 8. In the case that the DMA controller terminates due to an error with the destination transfers, these bits would indicate the address that caused the error. When the destination is DRAM onboard the VME-MXI-2, these bits must be programmed with the offset of the destination location within the VME-MXI-2 module's address space, not the

VMEbus address of the destination. To compute this value from the VMEbus address of the destination, just subtract the VME-MXI-2 module's A24 or A32 base address.

DMA Channel Status Register (CHSRx)

CHSR1 VMEbus A24 or A32 Offset: D3C (hex)

CHSR2 VMEbus A24 or A32 Offset: E3C (hex)

Attributes: Read Only 32, 16, 8-bit accessible

31	30	29	28	27	26	25	24
INT	0	0	0	0	0	DONE	0
23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8
ERROR	SABORT	0	STOPS	0	0	XFERR	0
7	6	5	4	3	2	1	0
0	0	0	0	SERR[1]	SERR[0]	DERR[1]	DERR[0]

This register provides status bits for DMA controller operations and error conditions.

Bit	Mnemonic	Description
31	INT	DMA Interrupt When this bit returns a 1, it indicates that the corresponding DMA controller is asserting the DMA interrupt.
30-26	0	Reserved These bits are reserved. The value these bits return when read is meaningless.
25	DONE	DMA Done bit This status bit is cleared when a DMA operation is started and set when the operation is terminated either successfully or by a stop or error condition. This bit can be either polled or used to generate an interrupt to signal when the operation is complete. See the register descriptions for the DMAICR, DMAIER, DMAISIDR, and CHCRx for more information about generating an interrupt on the DONE bit. Once it is determined that the DMA operation is done, the error condition bits in this register

		(ERROR, SABORT, STOPS, XFERR, SERR[1:0], and DERR[1:0]) should be checked before assuming the transfer completed successfully.
24-16	0	Reserved These bits are reserved. The value these bits return when read is meaningless.
15	ERROR	DMA Error bit When this bit returns a 1 it indicates that the corresponding DMA controller terminated an operation due to an error condition. The other bits in this register can be used to determine the type of error.
14	SABORT	DMA Software Abort bit When this bit returns a 1, it indicates that the corresponding DMA controller terminated an operation because the ABORT bit in the DMA Channel Operation Register (CHORx) was written with a 1.
13	0	Reserved This bit is reserved. The value this bit returns when read is meaningless.
12	STOPS	DMA Stopped Status bit When this bit returns a 1, it indicates that the STOP bit in the DMA Channel Operation Register (CHORx) was written with a 1. This does not indicate that the DMA controller has actually stopped. The DONE bit indicates when the DMA controller has actually stopped the operation.
11-10	0	Reserved These bits are reserved. The value these bits return when read is meaningless.
9	XFERR	Transfer Error When this bit returns a 1, it indicates that the DMA operation has terminated because either the source or destination encountered an error condition. Refer to the SERR[1:0] and DERR[1:0] bit descriptions to determine the type of error.

8-4	0	Reserved These bits are reserved. The value these bits return when read is meaningless.
3-2	SERR[1:0]	Source Error Status These bits indicate the type of error that occurred when accessing the source. When 00 (binary) is returned, no error occurred. When 01 (binary) is returned, a data transfer to the source got a bus error. When 10 (binary) is returned, it indicates that the retry limit was exceeded trying to access the source. The DMA controller retries up to 64 times any data transfer that receives a RETRY* acknowledge. If the data transfer receives a RETRY* acknowledge for the 65th time, the DMA controller terminates the operation and sets the retry limit exceeded status in the SERR[1:0] bits. When 11 (binary) is returned, it indicates that a data cycle to the source got a MXIbus parity error.
1-0	DERR[1:0]	Destination Error Status These bits indicate the type of error that occurred when accessing the destination. When 00 (binary) is returned, no error occurred. When 01 (binary) is returned, a data transfer to the destination got a bus error. When 10 (binary) is returned, it indicates that the retry limit was exceeded trying to access the destination. The DMA controller will retry up to 64 times any data transfer that receives a RETRY* acknowledge. If the data transfer receives a RETRY* acknowledge for the 65th time, the DMA controller terminates the operation and sets the retry limit exceeded status in the DERR[1:0] bits.

DMA FIFO Count Register (FCRx)

FCR1 VMEbus A24 or A32 Offset: D40 (hex)

FCR2 VMEbus A24 or A32 Offset: E40 (hex)

Attributes: Read Only 32, 16, 8-bit accessible

31	30	29	28	27	26	25	24
0	0	0	0	0	0	0	0
23	22	21	20	19	18	17	16
ECR[7]	ECR[6]	ECR[5]	ECR[4]	ECR[3]	ECR[2]	ECR[1]	ECR[0]
15	14	13	12	11	10	9	8
0	0	0	0	0	0	0	0
7	6	5	4	3	2	1	0
FCR[7]	FCR[6]	FCR[5]	FCR[4]	FCR[3]	FCR[2]	FCR[1]	FCR[0]

This register indicates the state of the DMA controller's FIFO buffer.

Bit	Mnemonic	Description
31-24	0	Reserved These bits are reserved. The value these bits return when read is meaningless.
23-16	ECR[7:0]	Empty Count Register These bits indicate the number of empty locations (in bytes) currently in the FIFO.
15-8	0	Reserved These bits are reserved. The value these bits return when read is meaningless.
7-0	FCR[7:0]	Full Count Register These bits indicate the number of bytes of data remaining in the FIFO.

Chapter 5

System Configuration

This chapter explains important considerations for programming and configuring a VMEbus/MXIbus system using VME-MXI-2 mainframe extenders.

☞ **Note:** ***Detailed descriptions of all register bits can be found in Chapter 4,*** *Register Descriptions****.***

In a MXIbus system, MXIbus address space is partitioned between MXIbus devices. A MXIbus device is any device having a MXIbus interface. MXIbus devices can be VMEbus mainframes, PCs, or stand-alone instruments. The MXIbus memory map is the same for all devices in the VMEbus/MXIbus system. You can use a VXIbus Multiframe Resource Manager (RM) application to configure all VME-MXI-2 mainframe extenders in the system to partition the MXIbus address space. If you will not be using a VXIbus Resource Manager, you will have to configure the Extender LA, A16, A24, and A32 window registers on all your VME-VXI-2 modules yourself to allow address mapping in your VMEbus/MXIbus system. If this is the case, consider "RM" in the remainder of this chapter to be your application that is configuring the address mapping on the VME-MXI-2 modules.

You can connect a VMEbus/MXIbus system together to form any arbitrary tree topology. A tree topology has no circular paths. Figures 5-1 and 5-2 show examples of tree topologies. The system in Figure 5-1 would not be a tree structure if a cable were added from the last MXIbus device on Level 1 to the Root PC. Figure 5-2 would also be an illegal and circular system if a cable were added to connect the two MXIbus devices on Level 1. At the root of the tree is the multiframe RM. The root can be a VMEbus mainframe or a stand-alone device, such as a PC with a MXIbus interface, that can operate as the system RM.

All MXIbus devices have address windows that connect them to the MXIbus system address map. MXIbus devices can be assigned space in any of four address spaces: A32, A24, A16, and VXIbus logical address space. Upon initialization, all windows are turned off, isolating all MXIbus devices from each other. The multiframe RM scans the MXIbus links and VMEbus mainframes for devices that conform to the VXIbus-defined register set and configures the window registers on each MXIbus device in order to partition the MXIbus address space among all devices. Most multiframe RMs allow the user to input information about the addressing needs of their VMEbus devices that do not conform to the VXIbus register set. This allows the RM to configure the address mapping on the VME-MXI-2 module to include the address space used by those devices.

Planning a VMEbus/MXIbus System Logical Address Map

The VMEbus/MXIbus system integrator is the person who configures all the VMEbus and MXIbus devices and connects the system together. This chapter assumes that you are the system integrator.

Before you begin setting the logical addresses and base addresses of the devices in your VMEbus/MXIbus system, you must determine the tree configuration of your system. The two basic configurations are the multiframe RM as an external PC with a MXIbus interface, as shown in Figure 5-1, or the multiframe RM in a VMEbus mainframe, as shown in Figure 5-2. The location of the multiframe RM constitutes the root of the system tree. MXIbus links connected to the root of the tree form levels of the tree. Notice that only one MXIbus link can be connected on the first level below a root PC multiframe RM, while multiple MXIbus links can be connected on the first level below a root VMEbus mainframe.

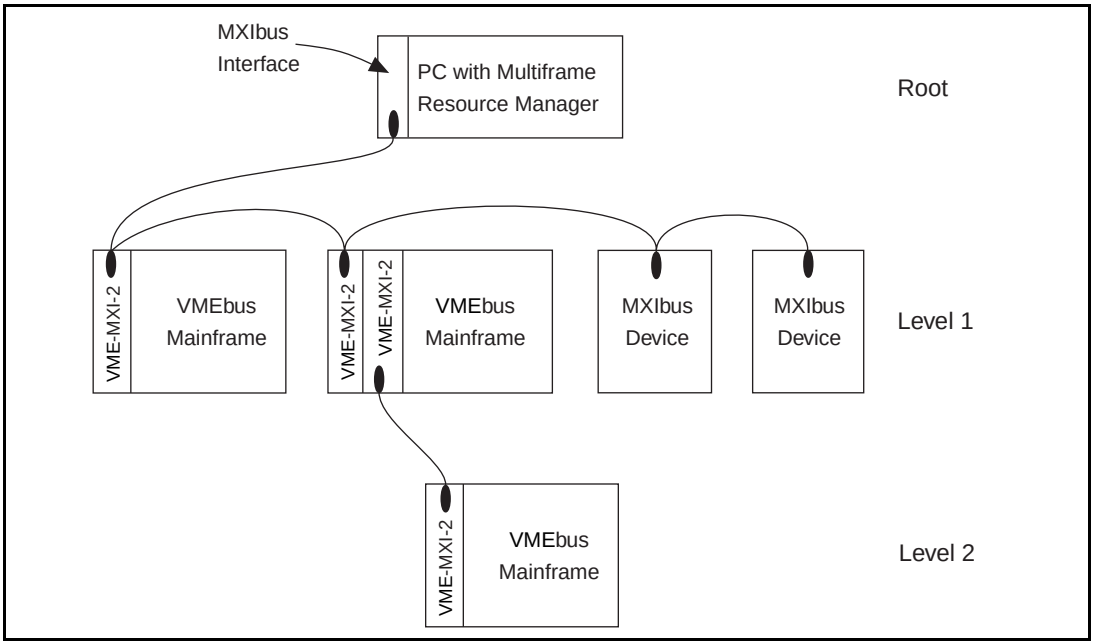


Figure 5-1. VMEbus/MXIbus System with Multiframe RM on a PC

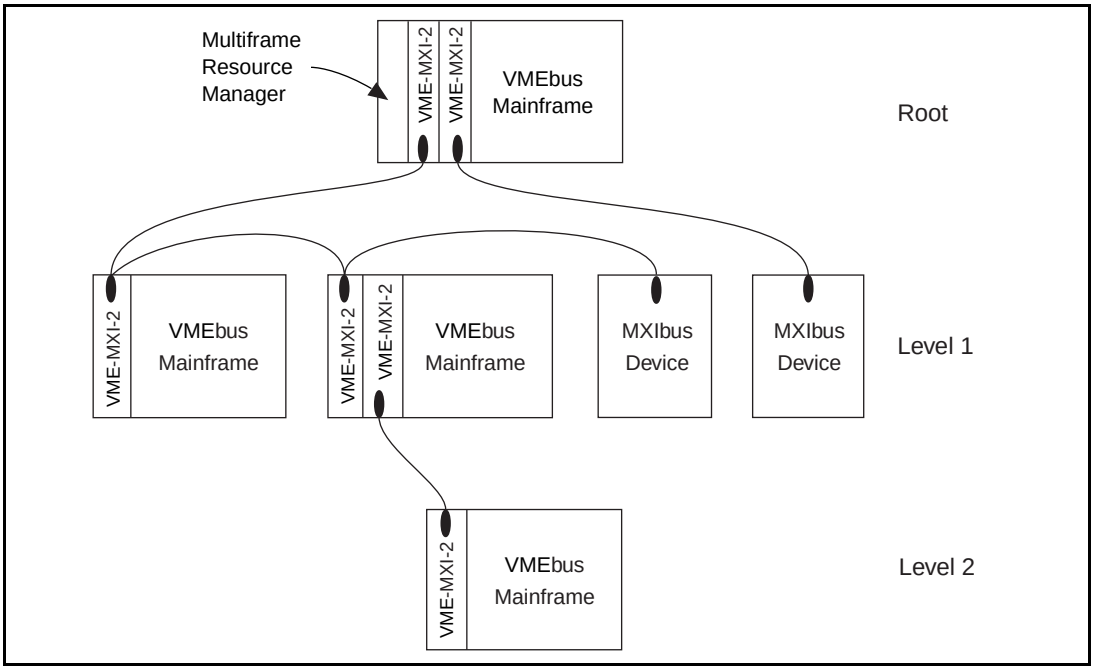


Figure 5-2. VMEbus/MXIbus System with Multiframe RM in a VMEbus Mainframe

The recommended way to set up your system is to fill up Level 1 MXIbus links before adding additional levels. System performance decreases as you increase the number of levels to the system because each level requires additional signal conversion. Also keep in mind these basic rules for VME-MXI-2 installation as you decide where to install your VME-MXI-2 interfaces:

- The VMEbus bus timeout unit must be on a VME-MXI-2.
- Multiple VME-MXI-2 interfaces in a mainframe require a dedicated user-defined pin.

The address mapping windows on the VME-MXI-2 can be configured to have a *Base/Size* format or a *High/Low* format. The CMODE bit in the VME-MXI-2 Control Register (VMCR) selects which format the mapping windows use.

Base/Size Configuration Format

Each address mapping window on a VME-MXI-2 interface has Base and Size parameters associated with it when the CMODE bit in the VME-MXI-2 Control Register (VMCR) is cleared. The Base bits define the base address for the window, and the Size bits indicate the number of Base bits that are significant. Replacing the insignificant bits with zeros gives the actual base address of the window. In other words, the Base and Size define a range of addresses that are in the window. A Direction bit is also included to indicate whether the defined range of addresses are mapped into or out of the VMEbus mainframe.

Table 5-1 shows which bits are compared for each Size setting and the resulting address range in hex if Base is set to 0 and 55 hex. Figure 5-3 further illustrates the number of bits of the Base that are compared for each Size value. Notice that if Size = 0, no bits are compared. Figure 5-4 shows the address range allocation for different Size values.

Table 5-1. Base and Size Combinations

Size	Base7	Base6	Base5	Base4	Base3	Base2	Base1	Base0	Range for 0	Range for 55
7	*	*	*	*	*	*	*		0 to 1	54 to 55
6	*	*	*	*	*	*			0 to 3	54 to 57
5	*	*	*	*	*				0 to 7	50 to 57
4	*	*	*	*					0 to F	50 to 5F
3	*	*	*						0 to 1F	40 to 5F
2	*	*							0 to 3F	40 to 7F
1	*								0 to 7F	00 to 7F
0									0 to FF	00 to FF

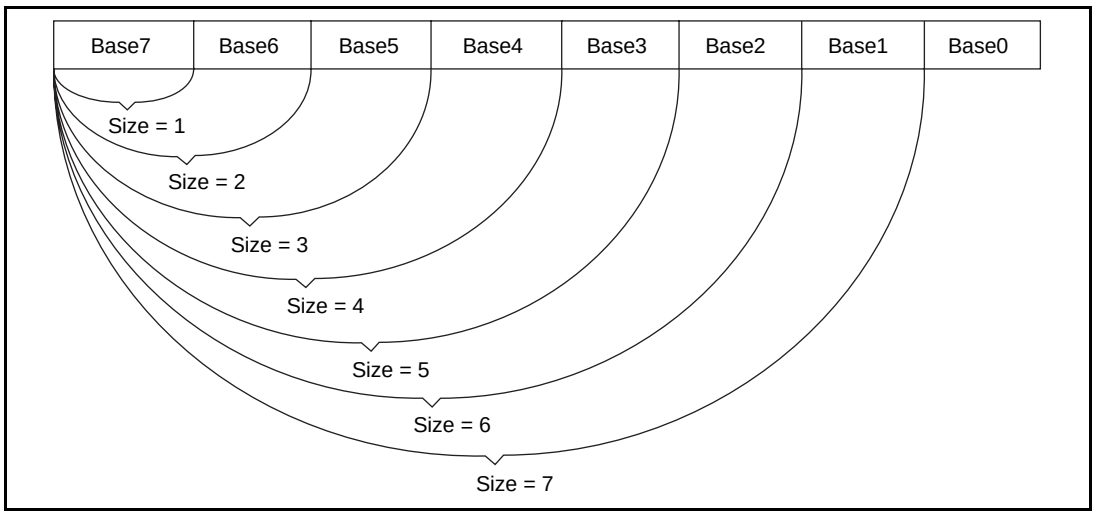


Figure 5-3. Base and Size Combinations

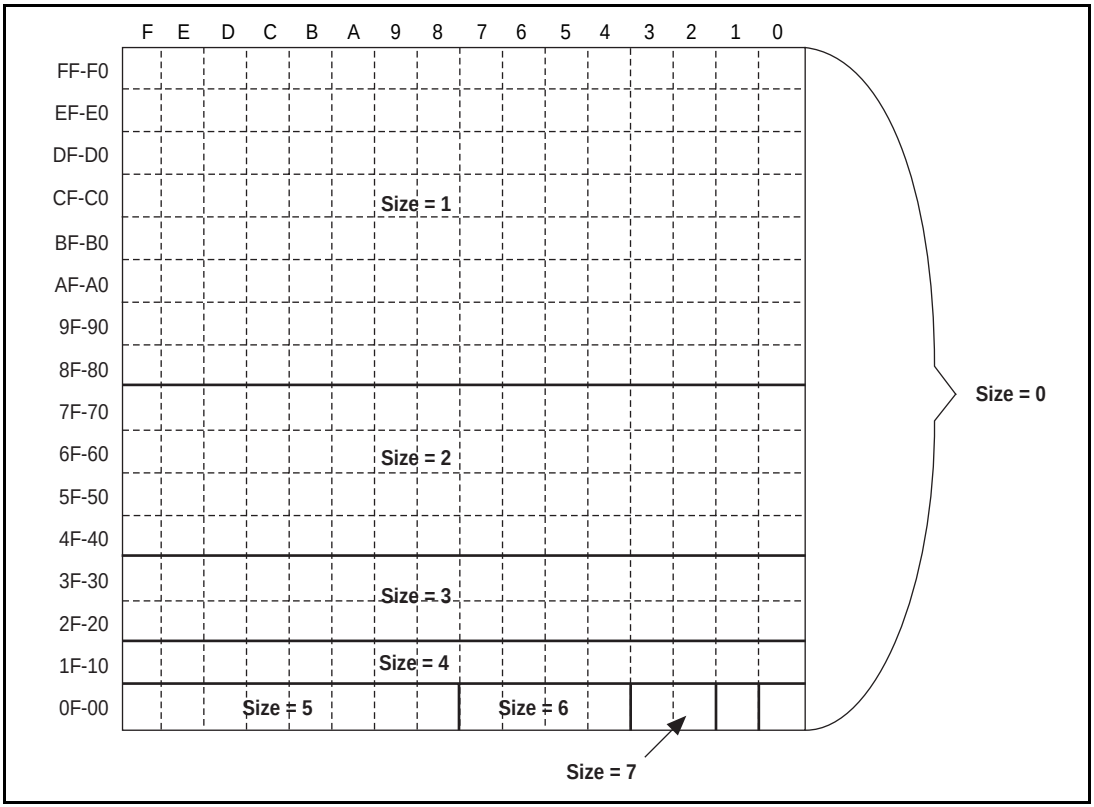


Figure 5-4. Address Range Allocation for Different Size Values

High/Low Configuration Format

Each address mapping window on a VME-MXI-2 interface has High and Low address parameters associated with it when the CMODE bit in the VME-MXI-2 Control Register (VMCR) is set. The High and Low values define the range of MXIbus addresses that map into the VMEbus. The High bits define the upper bound address of the window, and the Low bits indicate the lower bound address of the window. To map a range of addresses from the VMEbus to the MXIbus (out of the mainframe), the RM places the upper bound of the window in the Low field, and the lower bound of the window in the High field. The window is disabled if the upper and lower bound are both equal to 0.

Warning: ***The High/Low configuration format is not defined in*** *VXI-6, VXIbus Mainframe Extender Specification****. This format is a device-specific feature of the VME-MXI-2. Do not design a system architecture based on this format when using a standard VXIbus Resource Manager.***

Steps to Follow When Planning a System Logical Address Map

As system integrator, when installing devices in the VMEbus/MXIbus system, you must assign a range of logical addresses for each VMEbus mainframe and MXIbus link. VME-MXI-2 modules and most other MXIbus devices use the VXIbus logical address scheme to locate their registers in A16 space. However, most VMEbus devices do not adhere to the VXIbus logical address scheme, and their A16 registers should not be placed in the range C000 to FFFF hex if possible. VMEbus devices will not be considered when planning a logical address map. For information on the A16 space required by VMEbus devices, refer to the section *Planning a VMEbus/MXIbus System A16 Address Map*, later in this chapter. The multiframe RM configures the logical address windows of each device to include the static logical addresses it finds in the mainframe, and returns an error if the static logical address assignments prevent assignment of an entire system logical address map. Devices with dynamically configurable logical addresses are assigned logical addresses within the range of addresses defined by the static devices in the mainframe.

The example system in Figure 5-5 has two levels. The RM is in VMEbus Mainframe #1. Use the following steps to develop a logical address map. The example worksheets show numbers for using Base/Size window formats. For High/Low format systems, you do not need to round the range of addresses for each mainframe up to the next power of two. Following the example system are worksheets you can use for analyzing your own system.

1. Lay out your system configuration and determine the number of logical addresses required by each VMEbus mainframe and MXIbus device. See Figure 5-5 and Table 5-2 for examples. Identify the multiframe RM and label its host device as the root of the system. Also identify the levels of the system and the MXIbus links on each level. MXIbus links cannot span across levels.

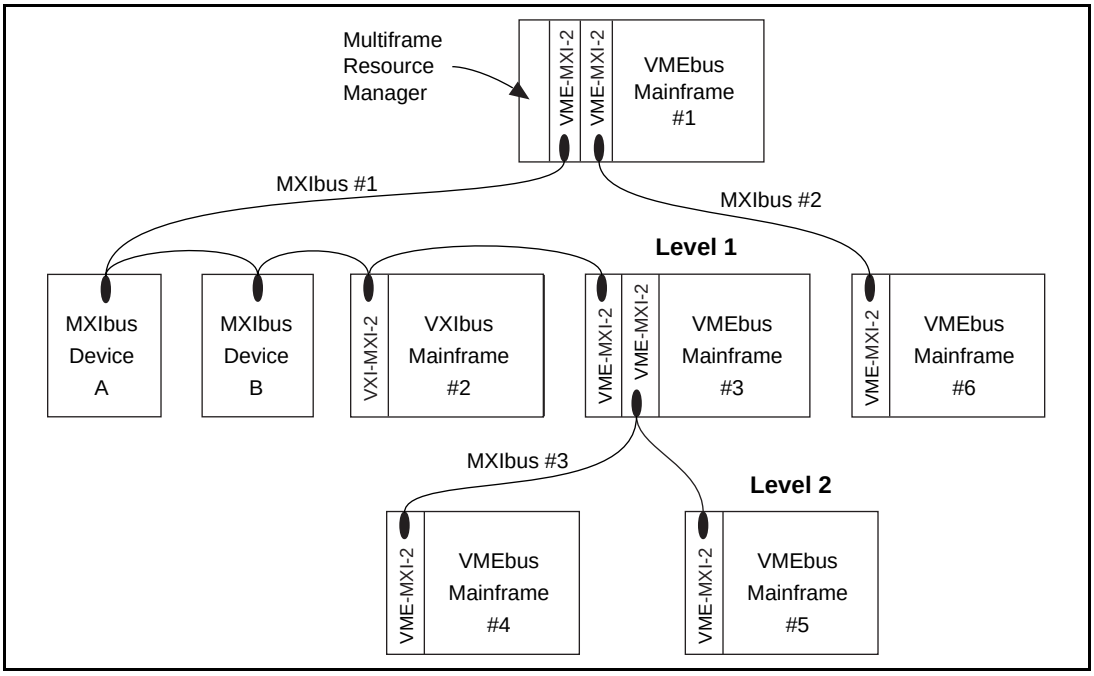


Figure 5-5. Example VMEbus/MXIbus System

Table 5-2. Example VMEbus/MXIbus System Required Logical Addresses

Device	Number of Logical Addresses Required
VMEbus Mainframe #1	2
MXIbus Device A	3
MXIbus Device B	1
VXIbus Mainframe #2	23
VMEbus Mainframe #3	2
VMEbus Mainframe #4	1
VMEbus Mainframe #5	1
VMEbus Mainframe #6	1

2. Determine the number of logical addresses required by the root device. If the RM is a PC with a MXIbus interface, the total number of logical addresses required is 1. If the RM is in a VMEbus mainframe, the total number of logical addresses required is equal to the number of VME-MXI-2 modules plus the number of other VMEbus devices in that frame that adhere to the VXIbus logical address scheme. If the RM is in a VXIbus mainframe, determine the number of logical addresses required by all VXIbus devices in that frame. Fill in that number in the appropriate space in the RM block as shown in Figure 5-7. If you are using the Base/Size format of the windows, round that number to the next highest power of two and place that number in the appropriate space.

☞ **Note:** ***If your RM is a PC with a MXIbus interface and you have more than one VMEbus mainframe on Level 1, you must change the logical addresses of both VME-MXI-2 modules so that they are not at the default of 1. Select a logical address that is greater than or equal to the number of logical addresses required by the mainframe.***

In the example system, the multiframe RM is installed in VMEbus Mainframe #1 and that frame requires two logical addresses. Since the value 2 is already a power of two, we entered the number 2 into the table.

3. Next, fill in the blanks for the number of logical addresses required by the first-level MXIbus devices. Using a separate worksheet for each MXIbus link on Level 1, fill in the blanks for the number of logical addresses required by the devices on each MXIbus link. Remember, you do not need to round numbers to the next power of two if you are using the High/Low format for the windows.

   The example system has two MXIbus links on the first level: MXIbus #1 and MXIbus #2. Figure 5-8 is the worksheet for MXIbus #1 and Figure 5-9 is the worksheet for MXIbus #2. We listed the devices on the MXIbus link and entered the number of logical addresses required by each device into the appropriate spaces. We then rounded the number of logical addresses up to the next power of two and entered this number into the table.

4. Fill out a separate worksheet for second-level MXIbus links and put the results in the appropriate places on the worksheet for the first-level device to which they are connected. Determine the total number of logical addresses required for the first-level device by adding the numbers with "+" next to them. If you are using Base/Size window formats, round this number to the next highest

power of two and place it in the appropriate space on the worksheet.

For the example system, MXIbus #3 is a second-level MXIbus link and it is connected to VMEbus Mainframe #3. We filled out the worksheet in Figure 5-10 for MXIbus #3 and entered the results into the worksheet for MXIbus #1 (Figure 5-8) under the device VMEbus Mainframe #3. MXIbus #3 needs two logical addresses and the devices in VMEbus Mainframe #3 need two logical addresses. The sum of these numbers is 4, which is already a power of two.

5. Determine the total number of logical addresses required by each MXIbus link by adding the numbers adjacent to the "*" symbols and entering that number in the appropriate space at the bottom of the worksheet. If you are using the Base/Size window format, round the number to the next highest power of two and enter it into the appropriate space on the worksheet. Place these numbers in the appropriate spaces on the worksheet for the next highest-level device to which the MXIbus link is connected.

   In the example system, MXIbus #1 requires 41 logical addresses (found at the bottom of Figure 5-8) and MXIbus #2 requires one logical address (found at the bottom of Figure 5-9). We placed these numbers in the corresponding spaces in Figure 5-7.

6. Add up the total number of logical addresses required for the system (at the bottom of Figure 5-7). Round this number up to the highest power of two if you are using Base/Size formats. The result should be equal to or less than 256. If the number is greater than 256, you must reorganize your devices and reconfigure the system. In the example system, this number equals 128, therefore the configuration is acceptable.

7. If you are using Base/Size parameters, determine the Size field of the range for each device and MXIbus link and insert that value in the corresponding locations of the worksheets. When you round up the number of logical addresses required to 2^X, Size = 8 – X.

8. Determine the range of addresses that will be occupied by the root device and each first-level device and MXIbus link. For Base/Size systems, use the Logical Address Map Diagram shown in Figure 5-6 to visualize the logical address map for the system. Each square in this diagram represents one logical address. The maximum number of logical addresses in a system is 256 and address ranges are assigned in blocks divisible by a power of two. Refer to Table 5-1 and Figure 5-4 for example logical address allocations for different Size values.

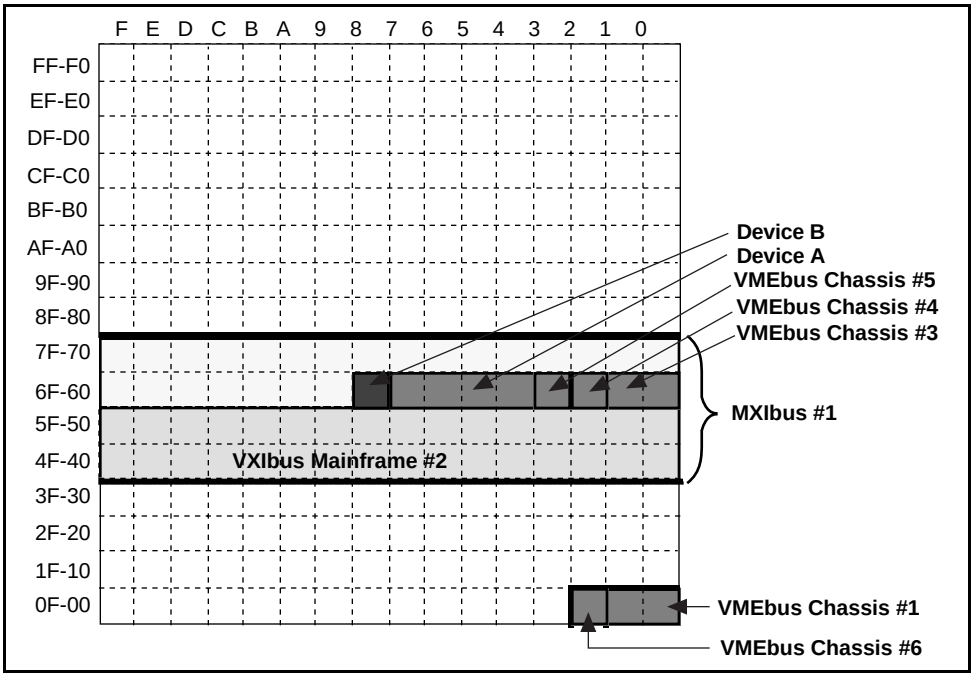


Figure 5-6. Logical Address Map Diagram for Example VMEbus/MXIbus System

The multiframe RM by definition is located at logical address 0; therefore, the host device of the multiframe RM must be assigned a range of logical addresses that includes logical address 0. Starting with the MXIbus link on Level 1, which requires the most logical addresses, assign the lowest available address range of the logical address map and continue with the next largest MXIbus link.

For the example system, VMEbus Mainframe #1, the host to the multiframe RM, requires two logical addresses and must have a range that includes logical address 0. It is assigned address range 0 to 1 hex. The largest first-level MXIbus link is MXIbus #1. It requires 64 logical addresses. The lowest available address range of 64 divisible by a power of two is 40 to 7F hex. The other first-level MXIbus link, MXIbus #2, needs only one logical address. It is assigned the lowest available logical address, 2 hex.

9. Determine the range of addresses that will be occupied by each device in the first-level MXIbus links. Remember that the range of addresses occupied by these devices must be within the range of addresses assigned to the MXIbus link to which it is a member. Start with the largest device in the MXIbus link.

   In the example system, MXIbus #1 has four devices. The largest one is VXIbus Mainframe #2, which requires 32 logical addresses and is assigned the lowest available range of 32 within the MXIbus #1 space—40 to 5F hex. The next largest device, VMEbus Mainframe #3, needs four logical addresses. This device has a second-level MXIbus link that needs two logical addresses, and the mainframe needs two logical addresses for its own VME-MXI-2 modules. First, assign the devices in the mainframe to the lowest available range within the allocated address range of MXIbus #1—60 to 61 hex. Then assign MXIbus #3 the lowest available range of size 2—62 to 63 hex. MXIbus Device A needs four logical addresses and MXIbus Device B needs one logical address. They are assigned 64 to 67, and 68, respectively.

10. Determine the range of logical addresses that will be occupied by each second-level device and MXIbus link. Remember that the range of logical addresses occupied by second-level devices must be within the range of addresses assigned to the device one level above it. Once the first-level MXIbus links have been allocated, assign the MXIbus devices and second-level MXIbus links within the corresponding first-level devices, starting with the largest device.

    In the example system, we assigned MXIbus #3 logical address range 62 to 63 hex. MXIbus #3 has two devices: VMEbus Mainframe #4 and VMEbus Mainframe #5. Each requires one logical address; therefore, we assigned them address 62 hex and 63 hex, respectively.

Resource Manager Mainframe: *VMEbus Mainframe #1*				
Total number of logical addresses required by this device:		*2*	Range =	*0 – 1*
Round total number up to the next power of two:	*	*2 (2^1)*	Size =	*8-1 = 7*
First-Level MXIbus Link: *MXIbus #1*				
(Fill in after completing charts on the following pages)				
Total number of logical addresses required by MXIbus Link:		*41*	Range =	*40 – 7F*
Round total number up to next power of two:	*	*64 (2^6)*	Size =	*8-6 = 2*
First-Level MXIbus Link: *MXIbus #2*				
(Fill in after completing charts on the following pages)				
Total number of logical addresses required by MXIbus Link:		*1*	Range =	*2*
Round total number up to next power of two:	*	*1 (2^0)*	Size =	*8-0 = 8*
First-Level MXIbus Link:				
(Fill in after completing charts on the following pages)				
Total number of logical addresses required by MXIbus Link:			Range =	
Round total number up to next power of two:	*		Size =	
Total Number of Logical Addresses Required:		*67*		
(Add numbers after the "*")			Range =	*0 – 7F*
Round Total Number up to Next Power of Two:		*128 (2^7)*	Size =	*8-7 = 1*
(If this number is greater than 256, you need to reorganize devices and try again.)				

Figure 5-7. Worksheet 1: Summary of Example VMEbus/MXIbus System

MXIbus Link: *MXIbus #1*				
Device: *MXIbus Device A*				
Number of logical addresses required by device:		*3*	Range =	*64 – 67*
Round total number up to the next power of two:		*4 (2^2)*	Size =	*8-2 = 6*
List other MXIbus links to this mainframe:				
Number of logical addresses required by additional MXIbus links:	+	*0*		
Total number of logical addresses required by this device:	=	*3*	Range =	*64 – 67*
Round total number up to the next power of two:	*	*4 (2^2)*	Size =	*8-2 = 6*
Device: *MXIbus Device B*				
Number of logical addresses required by device:		*1*	Range =	*68*
Round total number up to the next power of two:		*1 (2^0)*	Size =	*8-0 = 8*
List other MXIbus links to this mainframe:				
Number of logical addresses required by additional MXIbus links:	+	*0*		
Total number of logical addresses required by this device:	=	*1*	Range =	*68*
Round total number up to the next power of two:	*	*1 (2^0)*	Size =	*8-0 = 8*
Device: *VXIbus Mainframe #2*				
Number of logical addresses required by device:		*23*	Range =	*40 – 5F*
Round total number up to the next power of two:		*32 (2^5)*	Size =	*8-5 = 3*
List other MXIbus links to this mainframe:				
Number of logical addresses required by additional MXIbus links:	+	*0*		
Total number of logical addresses required by this device:	=	*23*	Range =	*40 – 5F*
Round total number up to the next power of two:	*	*32 (2^5)*	Size =	*8-5 = 3*
Device: *VMEbus Mainframe #3*				
Number of logical addresses required by device:		*2*	Range =	*60 – 61*
Round total number up to the next power of two:		*2 (2^1)*	Size =	*8-1 = 7*
List other MXIbus links to this mainframe: *MXIbus #3*				
Number of logical addresses required by additional MXIbus links:	+	*2*		
Total number of logical addresses required by this device:	=	*4*	Range =	*60 – 63*
Round total number up to the next power of two:	*	*4 (2^2)*	Size =	*8-2 = 6*
Device:				
Number of logical addresses required by device:			Range =	
Round total number up to the next power of two:			Size =	
List other MXIbus links to this mainframe:				
Number of logical addresses required by additional MXIbus links:	+			
Total number of logical addresses required by this device:	=		Range =	
Round total number up to the next power of two:	*		Size =	
Total Number of Logical Addresses Required:		*41*		
(Add numbers after the "*")			Range =	*40 – 7F*
Round Total Number up to Next Power of Two:		*64 (2^6)*	Size =	*8-6 = 2*

Figure 5-8. Worksheet 2 for MXIbus #1 of Example VMEbus/MXIbus System

MXIbus Link:	*MXIbus #2*			
Device:	*VMEbus Mainframe #6*			
Number of logical addresses required by device:		*1*	Range =	*2*
Round total number up to the next power of two:		*1 (2^0)*	Size =	*8-0 = 8*
List other MXIbus links to this mainframe:				
Number of logical addresses required by additional MXIbus links:	+	*0*		
Total number of logical addresses required by this device:	=	*1*	Range =	*2*
Round total number up to the next power of two:	*	*1 (2^0)*	Size =	*8-0 = 8*
Device:				
Number of logical addresses required by device:			Range =	
Round total number up to the next power of two:			Size =	
List other MXIbus links to this mainframe:				
Number of logical addresses required by additional MXIbus links:	+			
Total number of logical addresses required by this device:	=		Range =	
Round total number up to the next power of two:	*		Size =	
Total Number of Logical Addresses Required:		*1*		
(Add numbers after the "*")			Range =	*2*
Round Total Number up to Next Power of Two:		*1 (2^0)*	Size =	*8-0 = 8*

Figure 5-9. Worksheet 3 for MXIbus #2 of Example VMEbus/MXIbus System

MXIbus Link:	*MXIbus #3*			
Device:	*VMEbus Mainframe #4*			
Number of logical addresses required by device:		*1*	Range =	*62*
Round total number up to the next power of two:		*1 (2^0)*	Size =	*8-0 = 8*
List other MXIbus links to this mainframe:				
Number of logical addresses required by additional MXIbus links:	+	*0*		
Total number of logical addresses required by this device:	=	*1*	Range =	*62*
Round total number up to the next power of two:	*	*1 (2^0)*	Size =	*8-0 = 8*
Device:	*VMEbus Mainframe #5*			
Number of logical addresses required by device:		*1*	Range =	*63*
Round total number up to the next power of two:		*1 (2^0)*	Size =	*8-0 = 8*
List other MXIbus links to this mainframe:				
Number of logical addresses required by additional MXIbus links:	+	*0*		
Total number of logical addresses required by this device:	=	*1*	Range =	*63*
Round total number up to the next power of two:	*	*1 (2^0)*	Size =	*8-0 = 8*
Total Number of Logical Addresses Required:		*2*		
(Add numbers after the "*")			Range =	*62 – 63*
Round Total Number up to Next Power of Two:		*2 (2^1)*	Size =	*8-1 = 7*

Figure 5-10. Worksheet 4 for MXIbus #3 of Example VMEbus/MXIbus System

Worksheets for Planning Your VMEbus/MXIbus Logical Address Map

Use the worksheets on the following pages for analyzing your own VMEbus/MXIbus system. Follow the procedures used to fill out the worksheets for the example VMEbus/MXIbus system.

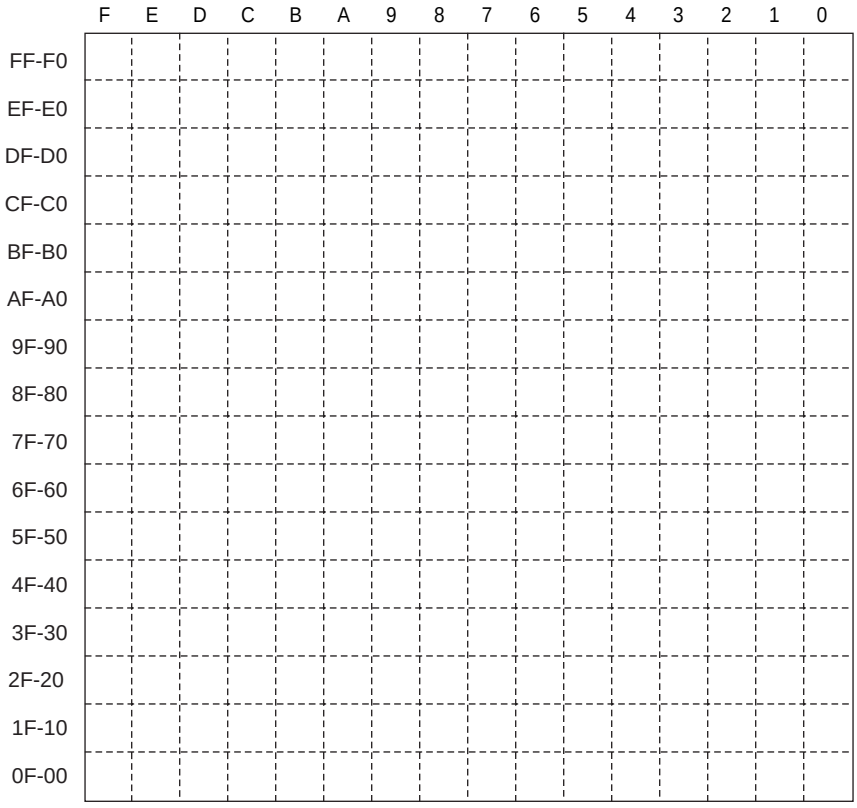


Figure 5-11. Logical Address Map Diagram for Your VMEbus/MXIbus System

Resource Manager Mainframe: ____________________				
Total number of logical addresses required by this device:		__________	Range =	_________
Round total number up to the next power of two:	*	__________	Size =	_________
First-Level MXIbus Link: ____________________				
(Fill in after completing charts on the following pages)				
Total number of logical addresses required by MXIbus Link:		__________	Range =	_________
Round total number up to next power of two:	*	__________	Size =	_________
First-Level MXIbus Link: ____________________				
(Fill in after completing charts on the following pages)				
Total number of logical addresses required by MXIbus Link:		__________	Range =	_________
Round total number up to next power of two:	*	__________	Size =	_________
First-Level MXIbus Link: ____________________				
(Fill in after completing charts on the following pages)				
Total number of logical addresses required by MXIbus Link:		__________	Range =	_________
Round total number up to next power of two:	*	__________	Size =	_________
First-Level MXIbus Link: ____________________				
(Fill in after completing charts on the following pages)				
Total number of logical addresses required by MXIbus Link:		__________	Range =	_________
Round total number up to next power of two:	*	__________	Size =	_________
Total Number of Logical Addresses Required:		__________		
(Add numbers after the "*")			Range =	_________
Round Total Number up to Next Power of Two:		__________	Size =	_________
(If this number is greater than 256, you need to reorganize devices and try again.)				

Figure 5-12. Worksheet 1: Summary of Your VMEbus/MXIbus System

Use Figures 5-13 through 5-15 to show the first three MXIbus links of your VMEbus/MXIbus system.

Note: ***You may need to make more copies of the blank forms if your system consists of more than three MXIbus links.***

MXIbus Link: *MXIbus #1*				
Device: ____				
Number of logical addresses required by device:		____	Range =	____
Round total number up to the next power of two:		____	Size =	____
List other MXIbus links to this mainframe: ____				
Number of logical addresses required by additional MXIbus links:	+	____		
Total number of logical addresses required by this device:	=	____	Range =	____
Round total number up to the next power of two:	*	____	Size =	____
Device: ____				
Number of logical addresses required by device:		____	Range =	____
Round total number up to the next power of two:		____	Size =	____
List other MXIbus links to this mainframe: ____				
Number of logical addresses required by additional MXIbus links:	+	____		
Total number of logical addresses required by this device:	=	____	Range =	____
Round total number up to the next power of two:	*	____	Size =	____
Device: ____				
Number of logical addresses required by device:		____	Range =	____
Round total number up to the next power of two:		____	Size =	____
List other MXIbus links to this mainframe: ____				
Number of logical addresses required by additional MXIbus links:	+	____		
Total number of logical addresses required by this device:	=	____	Range =	____
Round total number up to the next power of two:	*	____	Size =	____
Device: ____				
Number of logical addresses required by device:		____	Range =	____
Round total number up to the next power of two:		____	Size =	____
List other MXIbus links to this mainframe: ____				
Number of logical addresses required by additional MXIbus links:	+	____		
Total number of logical addresses required by this device:	=	____	Range =	____
Round total number up to the next power of two:	*	____	Size =	____
Device: ____				
Number of logical addresses required by device:		____	Range =	____
Round total number up to the next power of two:		____	Size =	____
List other MXIbus links to this mainframe: ____				
Number of logical addresses required by additional MXIbus links:	+	____		
Total number of logical addresses required by this device:	=	____	Range =	____
Round total number up to the next power of two:	*	____	Size =	____
Total Number of Logical Addresses Required:		____		
(Add numbers after the "*")			Range =	____
Round Total Number up to Next Power of Two:		____	Size =	____

Figure 5-13. Worksheet 2 for MXIbus #1 of Your VMEbus/MXIbus System

MXIbus Link: *MXIbus #2*				
Device: ____________				
Number of logical addresses required by device:		______	Range =	______
Round total number up to the next power of two:		______	Size =	______
List other MXIbus links to this mainframe: ______________				
Number of logical addresses required by additional MXIbus links:	+	______		
Total number of logical addresses required by this device:	=	______	Range =	______
Round total number up to the next power of two:	*	______	Size =	______
Device: ____________				
Number of logical addresses required by device:		______	Range =	______
Round total number up to the next power of two:		______	Size =	______
List other MXIbus links to this mainframe: ______________				
Number of logical addresses required by additional MXIbus links:	+	______		
Total number of logical addresses required by this device:	=	______	Range =	______
Round total number up to the next power of two:	*	______	Size =	______
Device: ____________				
Number of logical addresses required by device:		______	Range =	______
Round total number up to the next power of two:		______	Size =	______
List other MXIbus links to this mainframe: ______________				
Number of logical addresses required by additional MXIbus links:	+	______		
Total number of logical addresses required by this device:	=	______	Range =	______
Round total number up to the next power of two:	*	______	Size =	______
Device: ____________				
Number of logical addresses required by device:		______	Range =	______
Round total number up to the next power of two:		______	Size =	______
List other MXIbus links to this mainframe: ______________				
Number of logical addresses required by additional MXIbus links:	+	______		
Total number of logical addresses required by this device:	=	______	Range =	______
Round total number up to the next power of two:	*	______	Size =	______
Device: ____________				
Number of logical addresses required by device:		______	Range =	______
Round total number up to the next power of two:		______	Size =	______
List other MXIbus links to this mainframe: ______________				
Number of logical addresses required by additional MXIbus links:	+	______		
Total number of logical addresses required by this device:	=	______	Range =	______
Round total number up to the next power of two:	*	______	Size =	______
Total Number of Logical Addresses Required:		______		
(Add numbers after the "*")			Range =	______
Round Total Number up to Next Power of Two:		______	Size =	______

Figure 5-14. Worksheet 3 for MXIbus #2 of Your VMEbus/MXIbus System

MXIbus Link: *MXIbus #3*				
Device: ____________				
Number of logical addresses required by device:		________	Range =	________
Round total number up to the next power of two:		________	Size =	________
List other MXIbus links to this mainframe: ____________				
Number of logical addresses required by additional MXIbus links:	+	________		
Total number of logical addresses required by this device:	=	________	Range =	________
Round total number up to the next power of two:	*	________	Size =	________
Device: ____________				
Number of logical addresses required by device:		________	Range =	________
Round total number up to the next power of two:		________	Size =	________
List other MXIbus links to this mainframe: ____________				
Number of logical addresses required by additional MXIbus links:	+	________		
Total number of logical addresses required by this device:	=	________	Range =	________
Round total number up to the next power of two:	*	________	Size =	________
Device: ____________				
Number of logical addresses required by device:		________	Range =	________
Round total number up to the next power of two:		________	Size =	________
List other MXIbus links to this mainframe: ____________				
Number of logical addresses required by additional MXIbus links:	+	________		
Total number of logical addresses required by this device:	=	________	Range =	________
Round total number up to the next power of two:	*	________	Size =	________
Device: ____________				
Number of logical addresses required by device:		________	Range =	________
Round total number up to the next power of two:		________	Size =	________
List other MXIbus links to this mainframe: ____________				
Number of logical addresses required by additional MXIbus links:	+	________		
Total number of logical addresses required by this device:	=	________	Range =	________
Round total number up to the next power of two:	*	________	Size =	________
Device: ____________				
Number of logical addresses required by device:		________	Range =	________
Round total number up to the next power of two:		________	Size =	________
List other MXIbus links to this mainframe: ____________				
Number of logical addresses required by additional MXIbus links:	+	________		
Total number of logical addresses required by this device:	=	________	Range =	________
Round total number up to the next power of two:	*	________	Size =	________
Total Number of Logical Addresses Required:		________		
(Add numbers after the "*")			Range =	________
Round Total Number up to Next Power of Two:		________	Size =	________

Figure 5-15. Worksheet 4 for MXIbus #3 of Your VMEbus/MXIbus System

Alternative Worksheets for Planning Your VMEbus/MXIbus Logical Address Map

For most VMEbus/MXIbus systems, you may find the following worksheet helpful when setting up a system using the High/Low format for window configuration. The entire system can be described on one worksheet. Remember that the High/Low format cannot be used with a standard VXIbus multiframe Resource Manager.

The dotted lines can be used to add additional MXIbus links to Level 1 of the system, or to connect a Level 2 MXIbus link to one of the devices on Level 1.

Figure 5-16 presents one of these worksheets filled out for the example VMEbus/MXIbus system shown in Figure 5-5. Notice that the system does not take up as much of the logical address space as the Base/Size method of configuration because address requirements do not have to occupy blocks in powers of two. With High/Low configuration, you can configure each VME-MXI-2 window for exactly the amount of address space the mainframe needs.

Figure 5-17 is an alternative logical address map worksheet for you to fill out for your VMEbus/MXIbus system.

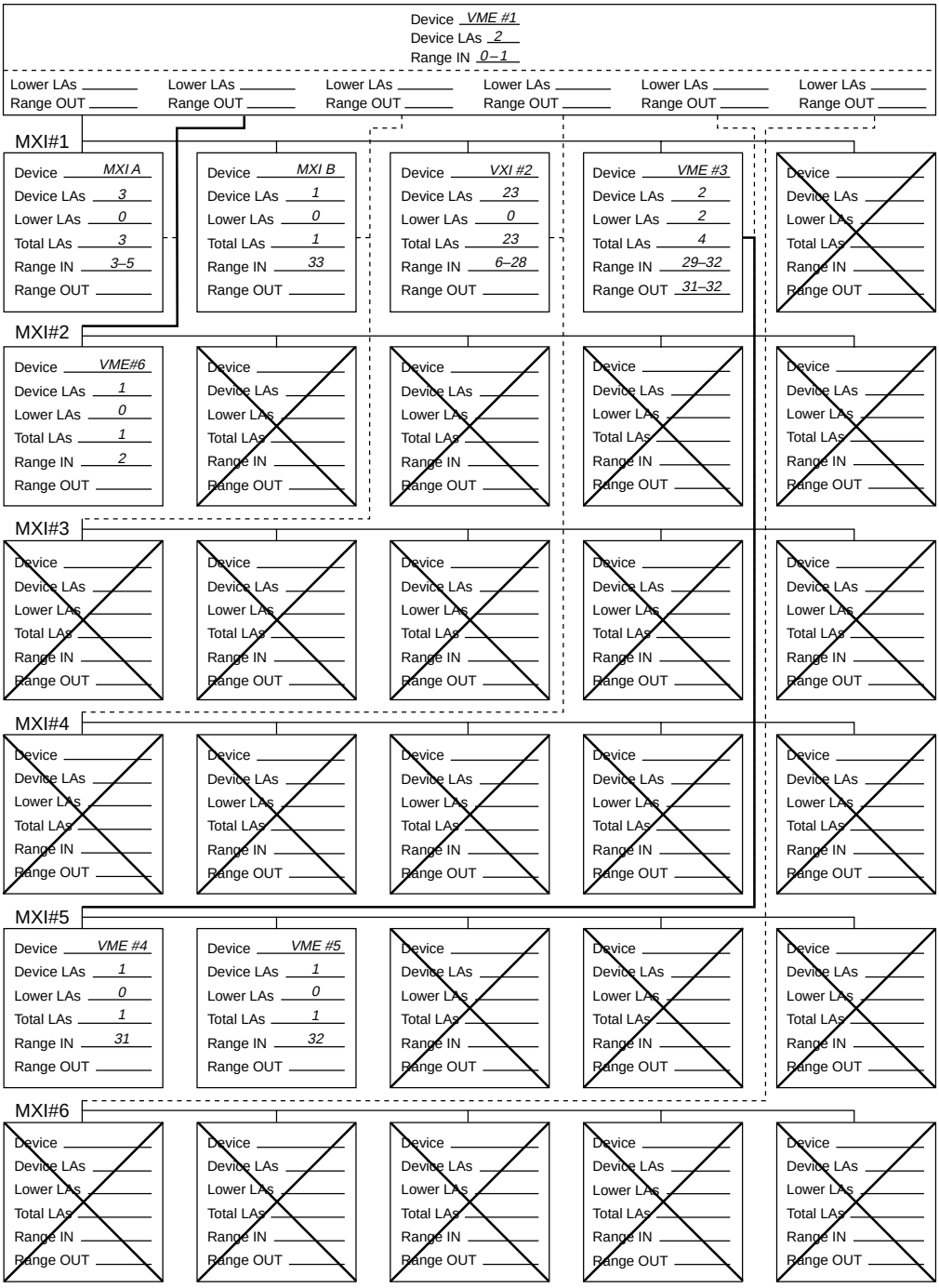


Figure 5-16. Alternative Worksheet: Logical Address Map for Example VMEbus/MXIbus System

Figure 5-17. Alternative Worksheet: Logical Address Map for Your VMEbus/MXIbus System

Planning a VMEbus/MXIbus System A16 Address Map

In a VMEbus/MXIbus system, A16 space is defined as the lower 48 KB of the A16 address space. As system integrator, you must determine the A16 address requirements for your VMEbus/MXIbus system.

You should configure the A16 resources for your VMEbus boards in the lower 48 KB (0000 through BFFF hex) of A16 space, so that you do not interfere with VXIbus configuration space (C000 through FFFF hex). The logical address mapping window is then used for mapping configuration space for VXIbus devices, and the A16 mapping window is used for mapping space for VMEbus devices.

When using Base/Size windowing formats, the minimum size of an A16 window is 512 B and the maximum size is 48 KB (window size = 0). Setting an A16 window address range in the upper 16 KB of A16 space (A15 = 1, A14 = 1) is not allowed, because it would conflict with the logical address space. Table 5-3 shows the A16 allocation sizes used for Base/Size systems.

Table 5-3. Amount of A16 Space Allocated for all Size Values

Size	Amount of A16 Space Allocated (in Bytes)
7	512 B
6	1 KB
5	2 KB
4	4 KB
3	8 KB
2	16 KB
1	32 KB
0	48 KB (All A16 space)

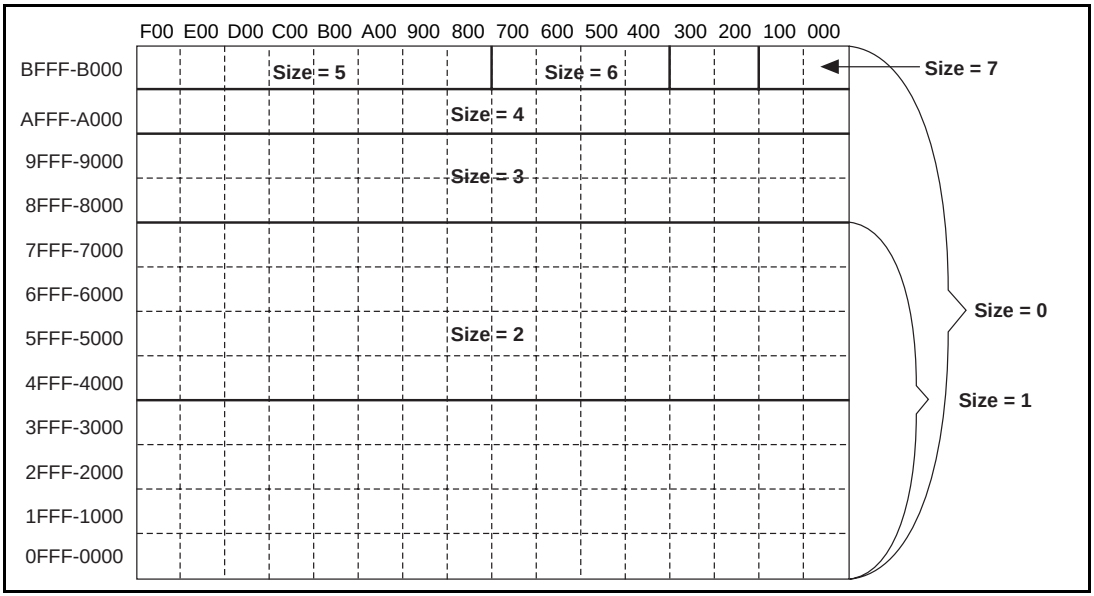


Figure 5-18. A16 Space Allocations for all Size Values

To plan the A16 address map, you will follow procedures similar to those for planning the logical address space address map. Determine the amount of A16 space required by each device; if you are using Base/Size windowing formats, round that amount up to the next address break listed in Table 5-3. Next, assign the A16 space, starting with the root device and working down the VMEbus/MXIbus system tree. To assist you in configuring the A16 window map on the VME-MXI-2 interfaces in your system, the following pages include worksheets, an address map diagram, and an example.

The following steps are used in the example:

1. Identify the RM Mainframe and the MXIbus levels of your system. Determine the amount of A16 space required by each MXIbus device. See Figure 5-19 and Table 5-4.
2. Fill out the RM Mainframe information in Figure 5-21. In this example, the RM Mainframe needs 16 KB of A16 space.
3. Next, analyze the first-level MXIbus links and complete a worksheet for each link. Figure 5-22 is the worksheet for MXIbus #1, which includes MXIbus Device A, MXIbus Device B, VXIbus Mainframe #2, and VMEbus Mainframe #3. MXIbus Device A needs 512 B of A16 space. We fill in the worksheet accordingly. MXIbus Device B and VXIbus Mainframe #2 do not need any A16 space, so we put zeros in the worksheet for these

devices. VMEbus Mainframe #3 needs 4 KB of A16, in addition to the amount of A16 required by MXIbus link #3 connected to it on Level 2.

4. Figure 5-23 is the worksheet for MXIbus #3, which includes VMEbus Mainframes #4 and #5. Mainframe #4 needs 2 KB and Mainframe #5 needs 1 KB of A16 space. We fill in the appropriate spaces on the worksheet.
5. Now we return to Figure 5-22 and fill in the MXIbus #3 information in the space for a second-level MXIbus link connected to VMEbus Mainframe #3. MXIbus #3 needs 2 KB for Mainframe #4 and 1 KB for Mainframe #5. The sum is 3 KB, which rounds up to the next address break of 4 KB. The amount of A16 space required for the window into VMEbus Mainframe #3 is 4 KB plus the 3 KB required by MXIbus #3, which rounds up to the next address break of 8 KB. We enter all of these numbers into the worksheet.
6. We now fill in Figure 5-21 for MXIbus #1. MXIbus #1 requires 512 B for MXIbus Device A and 8 KB for VMEbus Mainframe #3. The sum of these values rounds up to the nearest address break of 16 KB. We record this information on the worksheet.
7. Figure 5-21 is now completed for MXIbus #2. The only device on MXIbus #2 is VMEbus Mainframe #6, which needs 2 KB of A16 space. We enter this value into the worksheet.
8. The total amount of A16 space required by the system is now computed and found to be 34 KB, which rounds up to the next address break of 48 KB. This number does not exceed the maximum of 48 KB, so this configuration of A16 space is acceptable.
9. The next step is to determine the range of addresses, or base address, size, and direction of the A16 window for each VME-MXI-2 in the system. We first assign A16 space to the VMEbus RM Mainframe. From Figure 5-21, we see it needs 16 KB of A16 space, so we assign it the bottom 16 KB of A16 space, addresses 0 through 3FFF hex. See Figure 5-20 for a pictorial representation of this assignment.

10. Each first-level MXIbus link is connected to the RM through a VME-MXI-2. The A16 window for MXIbus link #1 is 16 KB in size. We assign the next lowest available 16 KB portion of A16 space to MXIbus link #1, which is address range 4000 to 7FFF hex. (See Figure 5-20.) The base address of this window is 4000, which we enter into Figure 5-21. The Size field for the window is *i* where the size of the window = $256 * 2^{8-i}$. 16 KB = $256 * 2^{8-2}$, so Size = 2. The direction of the window is in relation to the mainframe; therefore, Direction = *Out*.
11. The other first-level MXIbus link is MXIbus #2, which needs 2 KB of A16 space. The next lowest available 2 KB portion of A16 space is 8000 through 87FF hex. We set the base address of the window to 8000. To determine the Size value, 2 KB = $256 * 2^{8-5}$, so Size = 5. The direction of the window is in relation to the mainframe; therefore, Direction = *Out*. We enter all of these values into the worksheet in Figure 5-21.
12. The VXI-MXI-2 in VXIbus Mainframe #2 will be configured so that all A16 space is mapped outward, because the mainframe does not require any A16 space. To do this, we set Base = 0, Size = 0, and Direction = *Out.*
13. The VME-MXI-2 in VMEbus Mainframe #3 should be assigned the lowest available 8 KB of space assigned to MXIbus #1. Therefore, the base should be 4000 hex, and because 8 KB = $256 * 2^{8-3}$, Size = 3. The direction of the window is in relation to the mainframe; therefore, it is *In*. The VME-MXI-2 connected to MXIbus #3 must be assigned a window within the range of addresses assigned to Mainframe #3. Devices in Mainframe #3 need 4 KB of the 8 KB assigned to the mainframe. The other 4 KB can be assigned to MXIbus #3. Therefore, we assign addresses 4000 to 4FFF hex to devices in Mainframe #3, and addresses 5000 through 5FFF to MXIbus #3. For the VME-MXI-2 connected to MXIbus #3, we set Base = 5000, Size = 4 because 4 KB = $256 * 2^{8-4}$, and the direction toward MXIbus #3, or *Out*.
14. The 4 KB assigned to MXIbus #3 is further divided between VMEbus Mainframes #4 and #5. We assigned the bottom portion, 5000 to 57FF, to VMEbus Mainframe #4, and the next portion, 5800 to 5BFF, to VMEbus Mainframe #5. Therefore, for VMEbus Mainframe #4, we assign Base = 5000, Size = 5 because 2 KB = $256 * 2^{8-5}$, and Direction = *In*. For VMEbus Mainframe #5, Base = 5800, Size = 6 because 1 KB = $256 * 2^{8-6}$, and Direction = *In*.

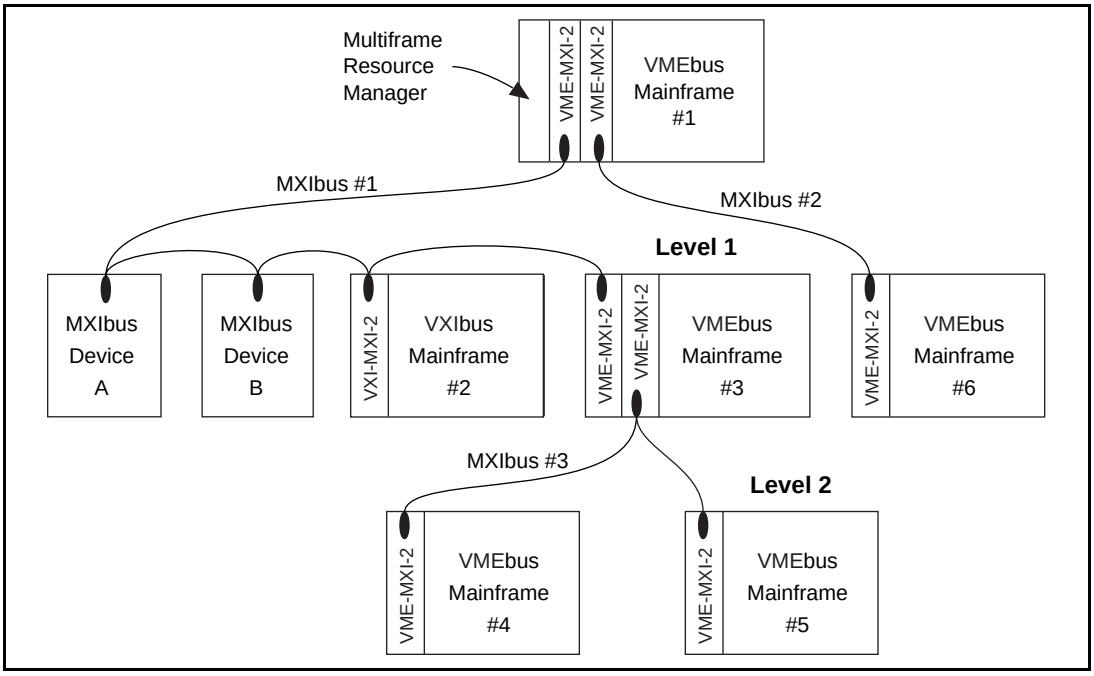


Figure 5-19. Example VMEbus/MXIbus System Diagram

Table 5-4. Example VMEbus/MXIbus System Required A16 Space

Device	Amount of A16 Space Required
VMEbus Mainframe #1	16 KB
MXIbus Device A	512 B
MXIbus Device B	0 B
VXIbus Mainframe #2	0 B
VMEbus Mainframe #3	4 KB
VMEbus Mainframe #4	2 KB
VMEbus Mainframe #5	1 KB
VMEbus Mainframe #6	2 KB

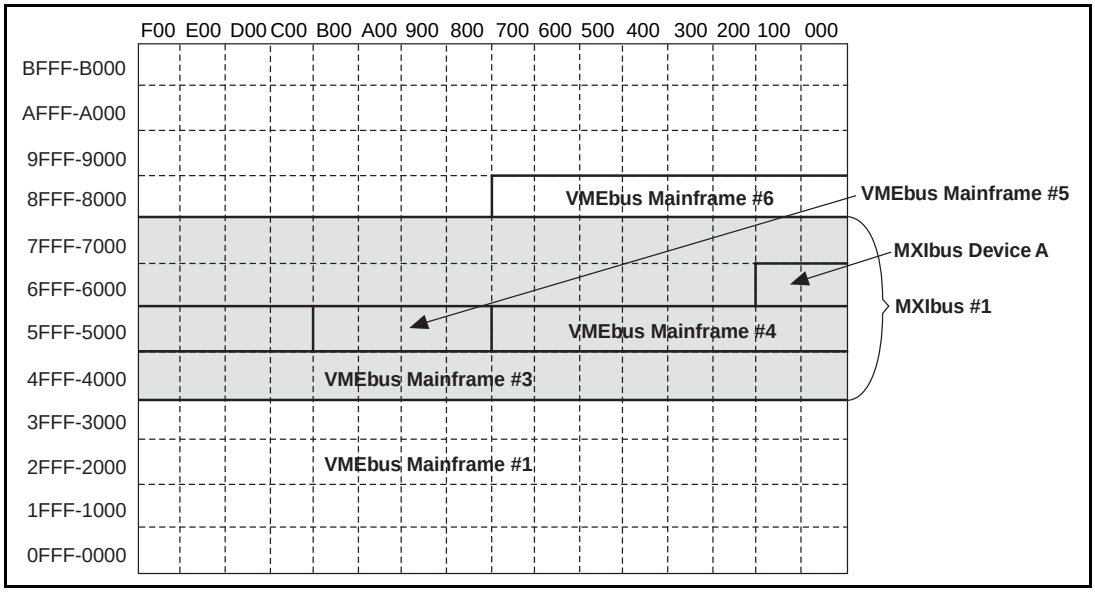


Figure 5-20. Example A16 Space Address Map Diagram

Resource Manager Mainframe: *VMEbus Mainframe #1*	
Amount of A16 space required for this mainframe:	*16 KB*
Round up to next address break:	* *16 KB*
First-Level MXIbus Link: *MXIbus #1*	
Amount of A16 space required for devices connected to this VME-MXI-2:	*8 KB + 512*
Round up to next address break:	* *16 KB*
A16 Window: Base *4000* Size: *2* Direction: *Out*	
First-Level MXIbus Link: *MXIbus #2*	
Amount of A16 space required for devices connected to this VME-MXI-2:	*2 KB*
Round up to next address break:	* *2 KB*
A16 Window: Base *8000* Size: *5* Direction: *Out*	
First-Level MXIbus Link:	
Amount of A16 space required for devices connected to this VME-MXI-2:	
Round up to next address break:	*
A16 Window: Base Size: Direction:	
First-Level MXIbus Link:	
Amount of A16 space required for devices connected to this VME-MXI-2:	
Round up to next address break:	*
A16 Window: Base Size: Direction:	
Total Amount of A16 Space Required by System: (Add numbers after the "*")	*34 KB*
Round up to Next Address Break: (If this number is greater than 48 KB, reorganize devices and try again.)	*48 KB*

Figure 5-21. Worksheet 1: Summary of A16 Address Map Example

MXIbus Link: *MXIbus #1*

Device: *MXIbus Device A*

Amount of A16 space required by this device: *512*

A16 space requirement for each second-level MXIbus link connected to this device:

#1 _____ + #2 _____ = *0*

Round up to next address break: _____ _____

Total amount of A16 space required for this window: = *512*

Round up total amount to the next address size break: * *512*

First-Level VME-MXI-2:

A16 Window: Base: _____ Size: _____ Direction: _____

Second-Level VME-MXI-2 #1: _____

A16 Window: Base: _____ Size: _____ Direction: _____

Second-Level VME-MXI-2 #2: _____

A16 Window: Base: _____ Size: _____ Direction: _____

Device: *MXIbus Device B*

Amount of A16 space required by this device: *0*

A16 space requirement for each second-level MXIbus link connected to this device:

#1 _____ + #2 _____ = *0*

Round up to next address break: _____ _____

Total amount of A16 space required for this window: = *0*

Round up total amount to the next address size break: * *0*

First-Level VME-MXI-2:

A16 Window: Base: _____ Size: _____ Direction: _____

Second-Level VME-MXI-2 #1: _____

A16 Window: Base: _____ Size: _____ Direction: _____

Second-Level VME-MXI-2 #2: _____

A16 Window: Base: _____ Size: _____ Direction: _____

Device: *VXIbus Mainframe #2*

Amount of A16 space required by this device: *0*

A16 space requirement for each second-level MXIbus link connected to this device:

#1 _____ + #2 _____ = *0*

Round up to next address break: _____ _____

Total amount of A16 space required for this window: = *0*

Round up total amount to the next address size break: * *0*

First-Level VME-MXI-2:

A16 Window: Base: *0000* Size: *0* Direction: *Out*

Second-Level VME-MXI-2 #1: _____

A16 Window: Base: _____ Size: _____ Direction: _____

Second-Level VME-MXI-2 #2: _____

A16 Window: Base: _____ Size: _____ Direction: _____

Figure 5-22. Worksheet 2 for MXIbus #1 of A16 Address Map Example (Continues)

MXIbus Link: *MXIbus #1 (Continued)*

Device: *VMEbus Mainframe #3*

Amount of A16 space required by this device: *4 KB*

A16 space requirement for each second-level MXIbus link connected to this device:

#1 *2 KB + 1 KB* + #2 _____ = *3 KB*

Round up to next address break: *4 KB* _____

Total amount of A16 space required for this window: = *7 KB*

Round up total amount to the next address size break: * *8 KB*

First-Level VME-MXI-2:

A16 Window: Base: *4000* Size: *3* Direction: *In*

Second-Level VME-MXI-2 #1: *MXIbus #3*

A16 Window: Base: *5000* Size: *4* Direction: *Out*

Second-Level VME-MXI-2 #2: _____

A16 Window: Base: _____ Size: _____ Direction: _____

Figure 5-22. Worksheet 2 for MXIbus #1 of A16 Address Map Example (Continued)

MXIbus Link: *MXIbus #3*

Device: *VMEbus Mainframe #4*

Amount of A16 space required by this device: *2 KB*

A16 space requirement for each second-level MXIbus link connected to this device:

#1 _____ + #2 _____ = *0*

Round up to next address break: _____ _____

Total amount of A16 space required for this window: = *2 KB*

Round up total amount to the next address size break: * *2 KB*

First-Level VME-MXI-2:

A16 Window: Base: *5000* Size: *5* Direction: *In*

Second-Level VME-MXI-2 #1: _____

A16 Window: Base: _____ Size: _____ Direction: _____

Second-Level VME-MXI-2 #2: _____

A16 Window: Base: _____ Size: _____ Direction: _____

Device: *VMEbus Mainframe #5*

Amount of A16 space required by this device: *1 KB*

A16 space requirement for each second-level MXIbus link connected to this device:

#1 _____ + #2 _____ = *0*

Round up to next address break: _____ _____

Total amount of A16 space required for this window: = *1 KB*

Round up total amount to the next address size break: * *1 KB*

First-Level VME-MXI-2:

A16 Window: Base: *5800* Size: *6* Direction: *In*

Second-Level VME-MXI-2 #1: _____

A16 Window: Base: _____ Size: _____ Direction: _____

Second-Level VME-MXI-2 #2: _____

A16 Window: Base: _____ Size: _____ Direction: _____

Device: _____

Amount of A16 space required by this device: _____

A16 space requirement for each second-level MXIbus link connected to this device:

#1 _____ + #2 _____ = _____

Round up to next address break: _____ _____

Total amount of A16 space required for this window: = _____

Round up total amount to the next address size break: * _____

First-Level VME-MXI-2:

A16 Window: Base: _____ Size: _____ Direction: _____

Second-Level VME-MXI-2 #1: _____

A16 Window: Base: _____ Size: _____ Direction: _____

Second-Level VME-MXI-2 #2: _____

A16 Window: Base: _____ Size: _____ Direction: _____

Figure 5-23. Worksheet 3 for MXIbus #3 of A16 Address Map Example

Worksheets for Planning Your VMEbus/MXIbus A16 Address Map

Use the worksheets on the following pages for planning an A16 address map for your VMEbus/ MXIbus system. Follow the procedures used to fill out the worksheets for the example VMEbus/MXIbus system.

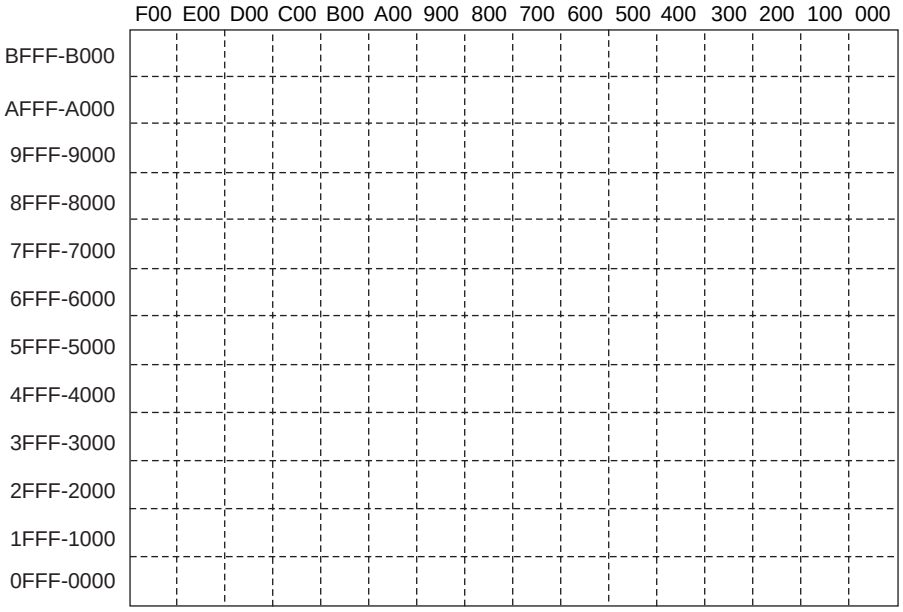


Figure 5-24. A16 Space Address Map Diagram for Your VMEbus/MXIbus System

Resource Manager Mainframe: ____________________

Amount of A16 space required for this mainframe: ________

Round up to next address break: * ________

First-Level MXIbus Link: ____________________

Amount of A16 space required for devices connected to this VME-MXI-2: ________

Round up to next address break: * ________

A16 Window: Base __________ Size: __________ Direction: __________

First-Level MXIbus Link: ____________________

Amount of A16 space required for devices connected to this VME-MXI-2: ________

Round up to next address break: * ________

A16 Window: Base __________ Size: __________ Direction: __________

First-Level MXIbus Link: ____________________

Amount of A16 space required for devices connected to this VME-MXI-2: ________

Round up to next address break: * ________

A16 Window: Base __________ Size: __________ Direction: __________

First-Level MXIbus Link: ____________________

Amount of A16 space required for devices connected to this VME-MXI-2: ________

Round up to next address break: * ________

A16 Window: Base __________ Size: __________ Direction: __________

Total Amount of A16 Space Required by System: ________

(Add numbers after the "*")

Round up to Next Address Break: ________

(If this number is greater than 48 KB, reorganize devices and try again.)

Figure 5-25. Worksheet 1: Summary of Your A16 Address Map

MXIbus Link: *MXIbus #1*

Device: ____________________

Amount of A16 space required by this device: ________

A16 space requirement for each second-level MXIbus link connected to this device:

#1 ________ + #2 ________ = ________

Round up to next address break: ________ ________

Total amount of A16 space required for this window: = ________

Round up total amount to the next address size break: * ________

First-Level VME-MXI-2:

A16 Window: Base: ________ Size: ________ Direction: ________

Second-Level VME-MXI-2 #1: ____________________

A16 Window: Base: ________ Size: ________ Direction: ________

Second-Level VME-MXI-2 #2: ____________________

A16 Window: Base: ________ Size: ________ Direction: ________

Device: ____________________

Amount of A16 space required by this device: ________

A16 space requirement for each second-level MXIbus link connected to this device:

#1 ________ + #2 ________ = ________

Round up to next address break: ________ ________

Total amount of A16 space required for this window: = ________

Round up total amount to the next address size break: * ________

First-Level VME-MXI-2:

A16 Window: Base: ________ Size: ________ Direction: ________

Second-Level VME-MXI-2 #1: ____________________

A16 Window: Base: ________ Size: ________ Direction: ________

Second-Level VME-MXI-2 #2: ____________________

A16 Window: Base: ________ Size: ________ Direction: ________

Device: ____________________

Amount of A16 space required by this device: ________

A16 space requirement for each second-level MXIbus link connected to this device:

#1 ________ + #2 ________ = ________

Round up to next address break: ________ ________

Total amount of A16 space required for this window: = ________

Round up total amount to the next address size break: * ________

First-Level VME-MXI-2:

A16 Window: Base: ________ Size: ________ Direction: ________

Second-Level VME-MXI-2 #1: ____________________

A16 Window: Base: ________ Size: ________ Direction: ________

Second-Level VME-MXI-2 #2: ____________________

A16 Window: Base: ________ Size: ________ Direction: ________

Figure 5-26. Worksheet 2 for MXIbus #1 A16 Address Map

MXIbus Link: *MXIbus #2*

Device: ____________________

Amount of A16 space required by this device: ________

A16 space requirement for each second-level MXIbus link connected to this device:

#1 ________ + #2 ________ = ________

Round up to next address break: ________ ________

Total amount of A16 space required for this window: = ________

Round up total amount to the next address size break: * ________

First-Level VME-MXI-2:

A16 Window: Base: ________ Size: ________ Direction: ________

Second-Level VME-MXI-2 #1: ____________________

A16 Window: Base: ________ Size: ________ Direction: ________

Second-Level VME-MXI-2 #2: ____________________

A16 Window: Base: ________ Size: ________ Direction: ________

Device: ____________________

Amount of A16 space required by this device: ________

A16 space requirement for each second-level MXIbus link connected to this device:

#1 ________ + #2 ________ = ________

Round up to next address break: ________ ________

Total amount of A16 space required for this window: = ________

Round up total amount to the next address size break: * ________

First-Level VME-MXI-2:

A16 Window: Base: ________ Size: ________ Direction: ________

Second-Level VME-MXI-2 #1: ____________________

A16 Window: Base: ________ Size: ________ Direction: ________

Second-Level VME-MXI-2 #2: ____________________

A16 Window: Base: ________ Size: ________ Direction: ________

Device: ____________________

Amount of A16 space required by this device: ________

A16 space requirement for each second-level MXIbus link connected to this device:

#1 ________ + #2 ________ = ________

Round up to next address break: ________ ________

Total amount of A16 space required for this window: = ________

Round up total amount to the next address size break: * ________

First-Level VME-MXI-2:

A16 Window: Base: ________ Size: ________ Direction: ________

Second-Level VME-MXI-2 #1: ____________________

A16 Window: Base: ________ Size: ________ Direction: ________

Second-Level VME-MXI-2 #2: ____________________

A16 Window: Base: ________ Size: ________ Direction: ________

Figure 5-27. Worksheet 3 for MXIbus #2 A16 Address Map

MXIbus Link: *MXIbus #3*						
Device: ____________						
Amount of A16 space required by this device:						________
A16 space requirement for each second-level MXIbus link connected to this device:						
	#1 ________	+	#2 ________		=	________
Round up to next address break:	________		________			
Total amount of A16 space required for this window:					=	________
Round up total amount to the next address size break:					*	________
First-Level VME-MXI-2:						
A16 Window:	Base: ________	Size: ________	Direction: ________			
Second-Level VME-MXI-2 #1:	____________					
A16 Window:	Base: ________	Size: ________	Direction: ________			
Second-Level VME-MXI-2 #2:	____________					
A16 Window:	Base: ________	Size: ________	Direction: ________			
Device: ____________						
Amount of A16 space required by this device:						________
A16 space requirement for each second-level MXIbus link connected to this device:						
	#1 ________	+	#2 ________		=	________
Round up to next address break:	________		________			
Total amount of A16 space required for this window:					=	________
Round up total amount to the next address size break:					*	________
First-Level VME-MXI-2:						
A16 Window:	Base: ________	Size: ________	Direction: ________			
Second-Level VME-MXI-2 #1:	____________					
A16 Window:	Base: ________	Size: ________	Direction: ________			
Second-Level VME-MXI-2 #2:	____________					
A16 Window:	Base: ________	Size: ________	Direction: ________			
Device: ____________						
Amount of A16 space required by this device:						________
A16 space requirement for each second-level MXIbus link connected to this device:						
	#1 ________	+	#2 ________		=	________
Round up to next address break:	________		________			
Total amount of A16 space required for this window:					=	________
Round up total amount to the next address size break:					*	________
First-Level VME-MXI-2:						
A16 Window:	Base: ________	Size: ________	Direction: ________			
Second-Level VME-MXI-2 #1:	____________					
A16 Window:	Base: ________	Size: ________	Direction: ________			
Second-Level VME-MXI-2 #2:	____________					
A16 Window:	Base: ________	Size: ________	Direction: ________			

Figure 5-28. Worksheet 4 for MXIbus #3 A16 Address Map

MXIbus Link: *MXIbus #4*						
Device: ____________						
Amount of A16 space required by this device:						________
A16 space requirement for each second-level MXIbus link connected to this device:						
	#1 ________	+	#2 ________		=	________
Round up to next address break:	________		________			
Total amount of A16 space required for this window:					=	________
Round up total amount to the next address size break:					*	________
First-Level VME-MXI-2:						
A16 Window:	Base: ________	Size: ________	Direction: ________			
Second-Level VME-MXI-2 #1:	____________					
A16 Window:	Base: ________	Size: ________	Direction: ________			
Second-Level VME-MXI-2 #2:	____________					
A16 Window:	Base: ________	Size: ________	Direction: ________			
Device: ____________						
Amount of A16 space required by this device:						________
A16 space requirement for each second-level MXIbus link connected to this device:						
	#1 ________	+	#2 ________		=	________
Round up to next address break:	________		________			
Total amount of A16 space required for this window:					=	________
Round up total amount to the next address size break:					*	________
First-Level VME-MXI-2:						
A16 Window:	Base: ________	Size: ________	Direction: ________			
Second-Level VME-MXI-2 #1:	____________					
A16 Window:	Base: ________	Size: ________	Direction: ________			
Second-Level VME-MXI-2 #2:	____________					
A16 Window:	Base: ________	Size: ________	Direction: ________			
Device: ____________						
Amount of A16 space required by this device:						________
A16 space requirement for each second-level MXIbus link connected to this device:						
	#1 ________	+	#2 ________		=	________
Round up to next address break:	________		________			
Total amount of A16 space required for this window:					=	________
Round up total amount to the next address size break:					*	________
First-Level VME-MXI-2:						
A16 Window:	Base: ________	Size: ________	Direction: ________			
Second-Level VME-MXI-2 #1:	____________					
A16 Window:	Base: ________	Size: ________	Direction: ________			
Second-Level VME-MXI-2 #2:	____________					
A16 Window:	Base: ________	Size: ________	Direction: ________			

Figure 5-29. Worksheet 5 for MXIbus #4 A16 Address Map

Multiframe RM Operation

On power-up, all MXIbus devices are isolated from each other because all address mapping windows are disabled. The multiframe RM performs the following:

- Identifies any VXIbus devices in the system
- Manages system self-tests
- Configures and enables the address map windows for logical addresses, A16, A24, and A32
- Establishes initial Commander/Servant system hierarchy, if any
- Initiates normal system operation

Configuring the Logical Address Window

To identify all devices in the VMEbus/MXIbus system, the RM performs the following steps, starting where the RM is located.

1. If the multiframe RM resides in a PC, it scans all logical addresses from 1 to FE (the RM is at address 0) to find all devices on the MXIbus. For each logical address, it reads the VXIbus ID Register (located at offset 0 within the device's configuration space). If the read is successful (that is, no BERR), a device is present at that logical address. If the read returns a BERR, no device is present at that logical address. The RM records all logical addresses found. For each mainframe extender found, it performs Step 2.

   If the multiframe RM is in a VMEbus mainframe, it performs Step 2 for the mainframe in which the RM is installed.

2. For the current mainframe, the RM does the following:

   A. Scans all logical addresses (0 to FF) in the mainframe to find all static configuration (SC) and dynamic configuration (DC) devices, skipping over logical addresses occupied by previously encountered devices. Finds the Slot 0 device and uses it to move all DC devices in the mainframe to the lowest unused logical addresses. Records all logical addresses found and allocated.

Notice that it is not possible to detect duplicate logical addresses because devices are found by reading the VXIbus ID Register. If two devices share a logical address, they will both respond to an address access without any indication of an error.

B. For each mainframe extender found in the mainframe, starting with the lowest addressed one the RM:

   i. Sets the extender logical address window to map all of the logical address space outward and enables the window.

   ii. Scans all logical addresses (0 to FF) in the window, skipping logical addresses occupied by previously encountered devices.

   iii. For each extender found in Step ii, starting with the lowest addressed one, the RM:

      a. Sets the extender logical address mapping window to map all of the logical address space inward and enables the window.

      b. Repeats Step 2 recursively.

      c. Sets the extender inward logical address mapping window to cover the range up to (but not including) the extender with the next highest logical address that was found in the logical address space.

   iv. Sets the extender outward logical address mapping window to cover the range of the devices connected to that extender.

Configuring the Logical Address Window Example

This example illustrates how the multiframe RM identifies devices in a VMEbus/MXIbus system and configures the logical address windows. The system used is the example VMEbus/MXIbus system shown in Figure 5-5. Table 5-5 shows the logical addresses we assigned to the devices in that system before bringing up the system. MXIbus devices can only be statically configured for the RM to find all devices connected on a MXIbus link. Therefore, each device must have a logical address that was configured before the RM executes.

Table 5-5. Logical Address Assignments for Example VMEbus/MXIbus System

Device	Logical Address Assignments
VMEbus Mainframe #1	
VME-MXI-2 on MXIbus #1	0
VME-MXI-2 on MXIbus #2	1
MXIbus Device A	64
MXIbus Device B	68
VXIbus Mainframe #2	
VXI-MXI-2	40
VMEbus Mainframe #3	
VME-MXI-2 on MXIbus #1	60
VME-MXI-2 on MXIbus #3	61
VMEbus Mainframe #4	
VME-MXI-2	62
VMEbus Mainframe #5	
VME-MXI-2	63
VMEbus Mainframe #6	
VME-MXI-2	2

The RM performs the following steps:

1. Scans logical addresses (0 to FF) and identifies all devices in VMEbus Mainframe #1. Finds the VME-MXI-2 interfaces at logical addresses 0 and 1.
2. Enables the logical address window of the VME-MXI-2 found at logical address 0 for the entire outward mapping range of 0 to FF. Scans all logical addresses, skipping all previously encountered devices, and finds the VME-MXI-2 in VMEbus Mainframe #3, the VXI-MXI-2 in VXIbus Mainframe #2, MXIbus Device A, and MXIbus Device B.
3. Enables the logical address window of the VME-MXI-2 in VMEbus Mainframe #3 for the entire inward mapping range of 0 to FF. Scans all logical addresses, skipping all previously encountered devices, and finds the VME-MXI-2 at logical address 61.
4. Enables the logical address window of the VME-MXI-2 found at logical address 61 for the entire outward mapping range of 0 to FF. Scans all logical addresses, skipping all previously encountered

devices, and finds the VME-MXI-2 in VMEbus Mainframe #4 and the VME-MXI-2 in VMEbus Mainframe #5.

5. Enables the logical address window of the VME-MXI-2 in VMEbus Mainframe #4 for the entire inward mapping range of 0 to FF. Scans all logical addresses, skipping all previously encountered devices. Because VMEbus Mainframe #4 has no VXIbus devices, the logical address window does not need to be programmed.
6. Enables the logical address window of the VME-MXI-2 in VMEbus Mainframe #5 for the entire inward mapping range of 0 to FF. Scans all logical addresses, skipping all previously encountered devices and previously defined address ranges. Because VMEbus Mainframe #5 has no VXIbus devices, the logical address window does not need to be programmed.
7. Sets the logical address window of the VME-MXI-2 found at logical address 61 to cover the ranges of the VME-MXI-2 in VMEbus Mainframe #4 (62) and the VME-MXI-2 in VMEbus Mainframe #5 (63). Enables the logical address window of the VME-MXI-2 at logical address 61 with an outward range of 62 to 63 by writing the value 4762 hex to the Logical Address Window Register (Base/Size format).
8. Sets the logical address window of the VME-MXI-2 found in VMEbus Mainframe #3 at logical address 60 to cover the devices in that mainframe (60 to 61) and the ranges required of its Level 2 devices: the VME-MXI-2 in VMEbus Mainframe #4 (62) and the VME-MXI-2 in VMEbus Mainframe #5 (63). Enables the logical address window of the VME-MXI-2 at logical address 60 with an inward range of 60 to 63 by writing the value 6660 hex to the Logical Address Window Register (Base/Size format).
9. Enables the logical address window of the VXI-MXI-2 in VXIbus Mainframe #2 for the entire inward mapping range of 0 to FF. Scans all logical addresses, skipping all previously encountered devices and defined ranges. Finds the Slot 0 device and uses it to move all DC devices in VXIbus Mainframe #2 to the lowest unused logical addresses. No more VXI-MXI-2 interfaces are found. The RM enables the logical address window for the VXI-MXI-2 in VXIbus Mainframe #2 with an inward range of 40 to 5F hex by writing the value 6340 hex to the Logical Address Window Register (Base/Size format).
10. Sets the logical address window of the VME-MXI-2 found in VMEbus Mainframe #1 at logical address 0 to cover the devices connected to that extender: the VME-MXI-2 in VMEbus Mainframe #3 (60 to 63), the VXI-MXI-2 in VXIbus

Mainframe #2 (40 to 5F), MXIbus Device A (64 to 67), and MXIbus Device B (68). Enables the logical address window of the VME-MXI-2 at logical address 0 with an outward range of 40 to 7F by writing the value 4240 hex to the Logical Address Window Register (Base/Size format).

11. Enables the logical address window of the VME-MXI-2 found at logical address 1 for the entire outward mapping range of 0 to FF. Scans all logical addresses, skipping all previously encountered devices, and finds the VME-MXI-2 in VMEbus Mainframe #6.
12. Enables the logical address window of the VME-MXI-2 in VMEbus Mainframe #6 for the entire inward mapping range of 0 to FF. Scans all logical addresses, skipping all previously encountered devices and defined ranges. No more VME-MXI-2 interfaces are found. Because VMEbus Mainframe #6 has no VXIbus devices, the logical address window does not need to be programmed.
13. Sets the logical address window of the VME-MXI-2 found in VMEbus Mainframe #1 at logical address 1 to cover the devices connected to that extender: the VME-MXI-2 in VMEbus Mainframe #6 (2). Enables the logical address window of the VME-MXI-2 at logical address 1 with an outward range of 2 to 3 by writing the value 4702 hex to the Logical Address Window Register (Base/Size format).

Configuring the A24 and A32 Addressing Windows

After the logical address space is configured for the system, the multiframe RM configures the A16, A24, and A32 address space. The logical address configuration forms a tree topology. Starting at the bottom of the tree and working up, add up the amount of memory needed by each mainframe and the devices on levels below it. That amount is then rounded up to the next power of two if the Base/Size format is used.

Starting at the root mainframe and working down each branch of the tree, assign memory starting with the largest memory window requirements at the top of the memory space, descending in order of window size and address location.

Each mainframe's A16, A24, and A32 address ranges define the address space occupied by the devices in that mainframe and on levels below that mainframe. These address ranges cannot overlap the defined range of any other mainframe unless that mainframe is on a level below the mainframe.

Chapter 6

VXI*plug&play* for the VME-MXI-2

This chapter describes the contents of the VXI*plug&play* disk that came with your VME-MXI-2 kit. The disk contains a VXI*plug&play* soft front panel and a VXI*plug&play* knowledge base file.

VME-MXI-2 VXI*plug&play* Soft Front Panel

The VXI*plug&play* soft front panel that comes with your VME-MXI-2 kit complies with VXI*plug&play* document VPP-7, *Soft Front Panel Specification*. This section describes the options you can configure using the soft front panel.

Use the soft front panel to configure programmable features on the VME-MXI-2. Because this same soft front panel also works with the VXI-MXI-2, you can easily configure a *hybrid* VXI/VME system. The settings that you change using the soft front panel are stored in the user-configurable half of the EEPROM on the VME-MXI-2. As a result, the changes remain intact through power cycles.

Installing the Soft Front Panel

To run the soft front panel, the host computer must be running the Windows operating system and have the VTL/VISA I/O driver language installed. If you are not using Windows and VTL/VISA, you must follow the instructions in Appendix B, *Programmable Configurations*, to access and modify the programmable features because you will not be able to use the soft front panel.

To install the soft front panel on your system, go to the Windows Program Manager's **File** menu and click on the **Run** option. Type the following command at the prompt

```
x: setup
```

where *x* is the letter of the floppy drive into which you inserted the VXI*plug&play* disk.

Before running the soft front panel, you must enable A24 or A32 accesses to the VME-MXI-2 that you want to configure. You can do this either by using a VXIbus Resource Manager or by programming the VXIbus Offset Register (VOR) and setting the A24/A32 ENABLE bit in the VXIbus Control Register (VCR) as described in the *VMEbus A24/A32 Registers* section of Chapter 4, *Register Descriptions*.

If you have more than one mainframe extender, you must also initialize any Extender Window registers on extenders between the host and the VME-MXI-2 so that the host can access both the VXIbus configuration (A16) registers and the A24/A32 space of the VME-MXI-2. Notice that if you are using a VXIbus Resource Manager, the initialization is performed automatically.

Using the Soft Front Panel

After successfully running the soft front panel, you will see the panel as shown in Figure 6-1.

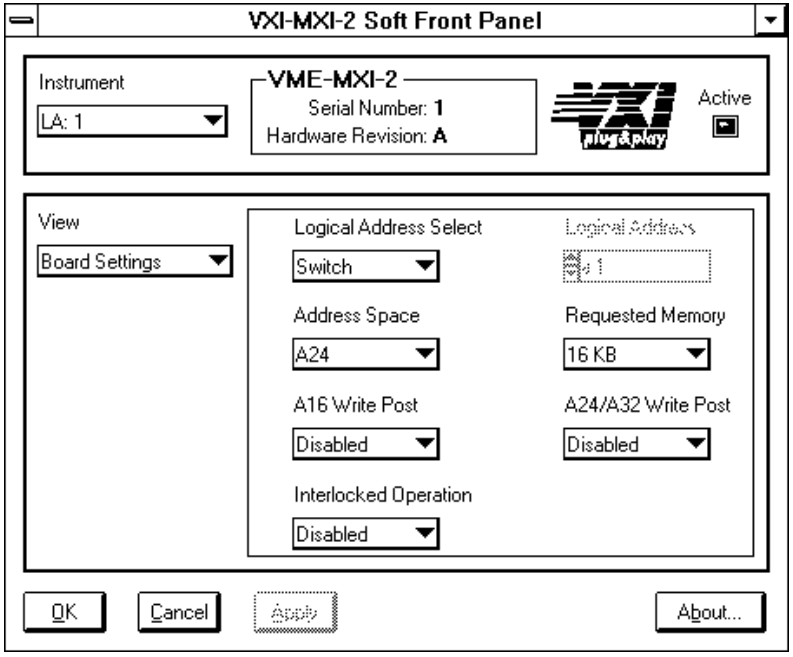


Figure 6-1. VME-MXI-2 VXI*plug&play* Soft Front Panel

If you have more than one VXI/VME-MXI-2 in your system, use the **Instrument** control in the upper-left corner to select which one you want to configure. The soft front panel selects the first VXI/VME-MXI-2 it finds upon execution. The top-center area of the panel indicates whether the currently selected instrument is a VXI-MXI-2 or a VME-MXI-2. Notice that this area of the panel also displays the serial number and hardware revision of the currently selected instrument.

The configurable features on the soft front panel are grouped into three categories: board settings, VMEbus settings, and MXIbus settings. Use the **View** control to select which group to display. These groups are described later in this section.

Click on the **OK** button to exit the soft front panel and save to the instrument's onboard EEPROM any changes you have made. Alternatively, you can use the **Apply** button to save your changes to the EEPROM without exiting the soft front panel, or the **Cancel** button will exit the panel without saving any changes. Changes to an instrument's settings are also saved to its EEPROM when you switch to a different instrument using the **Instrument** select control in the upper-left corner of the soft front panel.

The **About** button brings up an information display that shows the revision of the soft front panel.

Board Settings

The Board settings group contains controls that affect the VME-MXI-2 as a whole. Access this group by setting the **View** control to **Board Settings** as shown in Figure 6-1.

Logical Address Select and Logical Address

You can set or modify the logical address of the VME-MXI-2 either within the VME-MXI-2 soft front panel itself or with the onboard 8-position DIP switch. To select the configuration method you prefer, use the **Logical Address Select** control.

The default selection is the **Switch** option. Notice that the **Logical Address** control is inaccessible, since it would have no effect. In this option you need to change the hardware switch setting on the VME-MXI-2 module if you want to change the logical address.

If you select **Software** for this option, you can then use the **Logical Address** control to select a logical address within the range of 1 to 254. If you use this option, the hardware switch setting has no effect and you must use the VME-MXI-2 soft front panel to change the logical address.

Address Space and Requested Memory

The VME-MXI-2 requires at least 16 KB of address space in A24 space or at least 64 KB in A32 space. Use the **Address Space** control to select whether you want to use A24 space or A32 space. Use the **Requested Memory** control to set the amount of memory space that the VME-MXI-2 will require. You can select up to 8 MB in A24 space and up to 2 GB in A32 space.

These controls are necessary if you change the amount of DRAM installed on the VME-MXI-2. The amount of memory you set with the **Requested Memory** control should match the amount of DRAM installed on the VME-MXI-2. If no DRAM is installed, you should set it to 16 KB. Notice that the smallest valid amount in A32 space is 64 KB.

Caution: ***If you install DRAM into the VME-MXI-2, do not attempt to use the first 4 KB of memory space. This 4 KB space maps to the registers on the VME-MXI-2 and does not access onboard DRAM. Accessing this region will cause your VME-MXI-2 to behave incorrectly.***

If you do not want to lose 4 KB of DRAM you can get around this limitation by setting the **Requested Memory** control to double the amount that is installed on the VME-MXI-2, since the DRAM is aliased throughout the remainder of the requested memory space. The DRAM should then be accessed in the upper half of the requested memory space.

A16 Write Post and A24/A32 Write Post

The VME-MXI-2 can increase performance with its capability to post write cycles from both the MXIbus and the VMEbus. Write cycles should be posted only to devices that cannot return a *BERR* signal, because the *BERR* will not be reported to the originating master. Use the control appropriate for either A16 write posting or A24/A32 write posting. For either control, the options are **Enabled** and **Disabled**. By default, both options are disabled.

The **A16 Write Post** control affects only write cycles that map through the Extender A16 window from the VMEbus to the MXIbus and vice-versa. A16 write cycles in VXI configuration space are never posted regardless of the setting of this control.

The **A24/A32 Write Post** control affects write cycles that map through the Extender A24 window and Extender A32 window from the VMEbus to the MXIbus and vice-versa. This control also affects write cycles to the VME-MXI-2 module via its requested memory space from both the VMEbus and the MXIbus. For more information on the A16, A24, and A32 windows, refer to Chapter 4, *Register Descriptions*.

Interlocked

Interlocked arbitration mode is an optional mode of operation in which at any given moment the system can perform as if it were one large VMEbus mainframe with only one master of the entire system—VMEbus and MXIbus. This mode of operation prevents deadlocks by interlocking all arbitration in the VMEbus/MXIbus system.

The options for this control are **Enabled** and **Disabled**. By default, this option is disabled, which puts the VME-MXI-2 in normal operating mode.

In normal operating mode (non-interlocked), multiple masters can operate simultaneously in the VMEbus/MXIbus system. A deadlock occurs when a MXIbus master requests use of a VMEbus resource in another VMEbus mainframe while a VMEbus master in that mainframe is in the process of requesting a resource across the MXIbus. When this situation occurs, the VMEbus master must give up its bus ownership to resolve the conflict. The *RETRY* signal is used to terminate the transfer on the VMEbus; however, devices in the VMEbus mainframe must be able to detect a *RETRY* caused by a deadlock condition so that they can retry the operation. Any master device that cannot detect the RETRY protocol will interpret the response as a *BERR* signal instead.

The VME-MXI-2 is shipped from the factory configured for normal operating mode. If MXIbus transfers will be occurring both into and out of the mainframe, and the VMEbus modules in your system do not have the capability for handling RETRY conditions, you may want to configure the VME-MXI-2 for interlocked arbitration mode. In this mode, no software provisions for deadlock conditions are required. However, parallel processing in separate VMEbus mainframes is no longer possible, and system performance may be lower than in normal operating mode.

In a VMEbus/MXIbus system, you can configure some VME-MXI-2 modules for normal operating mode and others for interlocked arbitration mode. The VMEbus mainframes configured in interlocked arbitration mode will be interlocked with each other and the mainframes configured for normal operating mode can perform transfers in parallel.

This type of system configuration is recommended if you have one of the following situations:

- A VMEbus mainframe with only slave devices and no masters. Without bus masters, there is no chance for deadlock. You can configure the VME-MXI-2 devices in this mainframe for normal operating mode.
- A VMEbus mainframe with both masters and slaves, but the masters communicate only with the slaves in their mainframe. The masters never attempt transfers across the MXIbus, so there is no chance for deadlock when a MXIbus master attempts a transfer into the VMEbus mainframe. You can configure the VME-MXI-2 devices in this mainframe for normal operating mode.
- A VMEbus mainframe in which all masters that perform cycles across the MXIbus support the VME64 RETRY protocol. You can configure the VME-MXI-2 devices in this mainframe for normal operating mode because all masters that could cause a deadlock will automatically retry the operation.

In Chapter 4, *Register Descriptions*, the INTLCK bit is described in the *VME-MXI-2 Control Register (VMCR)* section. You can use this bit to enable the interlocked mode of arbitration. However, you may prefer to have the VME-MXI-2 automatically enable interlocked mode during its self-configuration, so that you do not need to access the INTLCK bit at each power-on. Interlocked mode is disabled in the default configuration of the VME-MXI-2.

VME Bus Settings

Use the options in this group to control features of the VMEbus interface on the VME-MXI-2. Access these controls by setting the **View** control to **VMEbus** as shown in Figure 6-2.

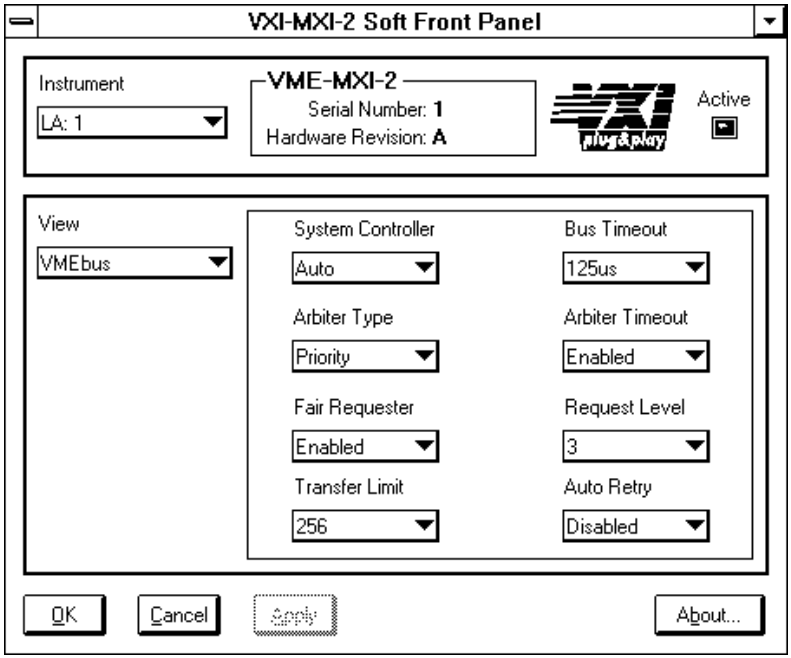


Figure 6-2. VME-MXI-2 VMEbus Settings

System Controller

You can use the **System Controller** control to override the automatic first slot detection circuit on the VME-MXI-2. When the control is set to **Auto** (the default setting), the first slot detection circuit will be active.

Otherwise, choose either the **Yes** or **No** option. You would need to run the VME-MXI-2 soft front panel again if you decide to change the VMEbus System Controller setting at a later time.

Warning: ***Do not install a VME-MXI-2 configured for VMEbus System Controller (the* Yes *option of this control) into another slot without first reconfiguring it. Neglecting to do this could result in damage to the VME-MXI-2, the VMEbus backplane, or both.***

This means that you should use either the* No *option to configure it not to be the System Controller or the* Auto *option for automatic configuration whenever you install the VME-MXI-2 into a slot other than the first slot.

Bus Timeout

The VMEbus Bus Timeout (BTO) is a watchdog timer for transfers on the VMEbus Data Transfer bus. After the specified amount of time has elapsed, the BTO circuitry terminates a VMEbus cycle if no slave has responded. The VME-MXI-2 must provide the VMEbus BTO for proper operation because when a MXIbus cycle is involved, the VMEbus timeout must be disabled and the MXIbus BTO enabled. You should disable the BTO of any other BTO module residing in the mainframe. If this is not possible, set it to its maximum setting to give the MXIbus cycles as much time as possible to complete.

The lowest value in the allowable range is 15 µs and the highest is 256 ms. The default value is 125 µs.

Arbiter Type

You can use the **Arbiter Type** feature to configure the VME-MXI-2 as either a Priority or Round Robin VMEbus arbiter. This control is applicable only if the VME-MXI-2 you are configuring is a VMEbus System Controller device. The default value is **Priority**.

When configured for **Priority** arbitration, the VME-MXI-2 grants the bus to the highest pending bus request level. In **Round Robin** arbitration mode, the VME-MXI-2 grants the bus to the next highest bus request level after the level of the previous bus owner. This effectively gives the same priority to each bus request level. Refer to the VMEbus specification for more information on the different types of arbiters.

Arbiter Timeout

An arbitration timeout feature is available on the VME-MXI-2 when it is acting as the VMEbus arbiter. This feature applies only to a VMEbus System Controller VME-MXI-2. The default value is **Enabled**.

The timer begins when the arbiter circuit on the VME-MXI-2 drives one of the *BGOUT* lines on the backplane. If no device takes over the bus within the timeout limit, the *BGOUT* is removed and the bus is either idle or granted to another requester.

Fair Requester

The VME-MXI-2 is always a Release On Request requester. However, you can configure whether the VME-MXI-2 acts as either a fair or unfair requester on the VMEbus. The default value for this control is **Enabled**, signifying a fair requester. For more information on the different types of requesters, refer to the VMEbus specification.

Request Level

The VME-MXI-2 uses one of the four VMEbus request levels (0 to 3) to request use of the VME Data Transfer Bus (DTB). The VME-MXI-2 requests use of the DTB whenever an external MXIbus device attempts a transfer that maps into the VMEbus mainframe.

The VME-MXI-2 uses VMEbus request level 3 in its factory-default setting. This is suitable for most systems. However, you can change the VME-MXI-2 to use any of the other three request levels (0, 1, or 2) by changing the setting of the **Request Level** control. You may want to change request levels to change the priority of the VME-MXI-2 request signal. For more information, refer to the VMEbus specification.

Transfer Limit

You can use this feature to control how many data transfers the VME-MXI-2 will perform on the VMEbus before releasing it to another master device that is requesting use of the bus.

The available options you can choose from are **16**, **64**, and **256** transfers. If you do not want the VME-MXI-2 to hold the VMEbus long enough to perform 256 transfers (the default value), you can use this control to select a smaller value.

Auto Retry

The VME-MXI-2 has an automatic retry feature for cycles that map from the VMEbus to the MXIbus. You can select **Enabled** or **Disabled** for this control. By default this option is disabled.

Normally, when a cycle maps from the VMEbus to the MXIbus, any retry response received on the MXIbus is passed to the VMEbus. If you enable the **Auto Retry** feature, the VME-MXI-2 automatically retries any MXI cycle that receives a retry response instead of passing a retry response back to the VMEbus. The VME-MXI-2 automatically continues to retry the MXI cycle until it receives either a *DTACK* or *BERR* response, which it then passes to the VMEbus.

Notice that the VME-MXI-2 has a limit on the number of automatic retries it will perform on any one cycle. If the limit is exceeded and the VME-MXI-2 receives another retry, it will pass a retry back to the VMEbus even though **Auto Retry** is enabled.

MXI Bus Settings

Use the options in this group to control features of the MXIbus interface on the VME-MXI-2 module. Access these controls by setting the **View** control to **MXIbus** as shown in Figure 6-3.

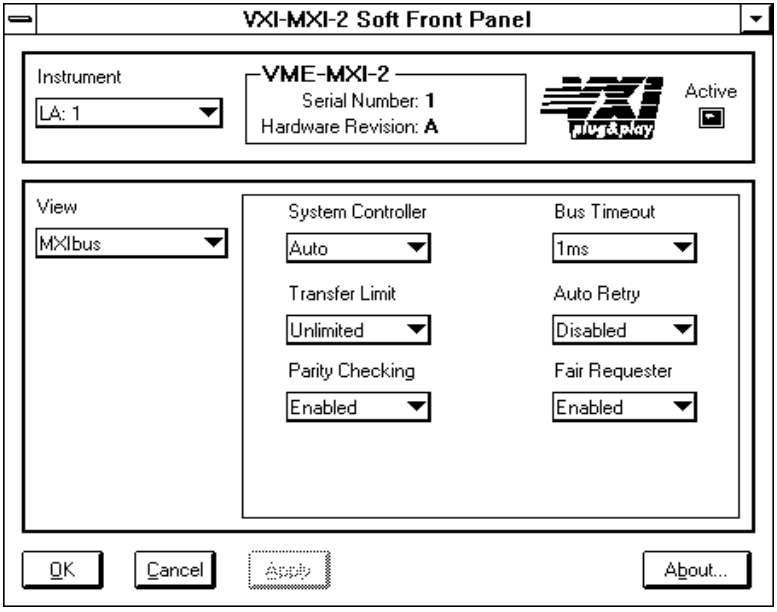


Figure 6-3. VME-MXI-2 MXIbus Settings

System Controller

You can use the **System Controller** control to determine whether the VME-MXI-2 acts as the MXIbus System Controller. When the **Auto** setting (the default setting) is active, the VME-MXI-2 automatically can sense from the MXIbus cable whether it should be the controller.

You can select either **Yes** or **No** to manually determine if the VME-MXI-2 should be the MXIbus System Controller. You must still be certain to cable the MXIbus system appropriately when you make either of these selections.

Bus Timeout

The MXIbus Bus Timeout (BTO) is a watchdog timer for transfers on the MXIbus. The MXIbus BTO unit operates only when the VME-MXI-2 is acting as the MXIbus System Controller. The functionality of this control is similar to that of the **Bus Timeout** control described previously under the *VME Bus Settings* section. The options range from 8 µs to 128 ms, with a default value of 1 ms.

After the specified amount of time has elapsed, the BTO circuitry terminates a MXIbus cycle if no slave has responded. The BTO circuitry is automatically deactivated when the VME-MXI-2 is not acting as the MXIbus System Controller. The BTO is also disabled when the current MXIbus cycle maps to the VMEbus through a VME-MXI-2.

Transfer Limit

You can use this feature to control how many data transfers the VME-MXI-2 will perform on the MXIbus before releasing it to another master device that is requesting use of the bus. The default setting holds the MXIbus for an **Unlimited** period of time.

The other options you can choose from are **16**, **64**, and **256** transfers. If you do not want the VME-MXI-2 to hold the MXIbus for an unlimited period of time, you can use this control to select one of these values.

Auto Retry

The VME-MXI-2 has an automatic retry feature for cycles that map from the MXIbus to the VMEbus. This feature works in the same manner as the **Auto Retry** control described previously under the *VME Bus Settings* section. You can select **Enabled** or **Disabled** for this control. By default, this option is disabled.

Normally, when a cycle maps from the MXIbus to the VMEbus, any retry response received on the VMEbus is passed to the MXIbus. If you enable the **Auto Retry** feature, the VME-MXI-2 automatically retries any VME cycle that receives a retry response instead of passing a retry response on to the MXIbus. The VME-MXI-2 automatically continues to retry the VME cycle until it receives either a *DTACK* or *BERR* response, which it then passes to the MXIbus.

Notice that the VME-MXI-2 has a limit on the number of automatic retries it will perform on any one cycle. If the limit is exceeded and the VME-MXI-2 receives another retry, it will pass a retry back to the MXIbus even though **Auto Retry** is enabled.

Parity Checking

You can use the **Parity Checking** control if you want to disable MXIbus parity checking. By default, MXIbus parity checking is set to **Enabled**, and should not be disabled under normal circumstances. MXIbus parity is always generated regardless if checking is enabled or disabled.

Fair Requester

You can use the **Fair Requester** control to configure the VME-MXI-2 as either a fair or unfair requester on the MXIbus. The default setting is **Enabled** (fair requester), which causes the VME-MXI-2 to request the MXIbus only when there are no requests pending from other masters. This prevents other MXIbus masters from being starved of bandwidth. The VME-MXI-2 will request the bus at any time when this setting is disabled (unfair requester).

VME-MXI-2 VXI*plug&play* Knowledge Base File

A VXI*plug&play* Knowledge Base File is included on the VXI*plug&play* disk in this kit. This file conforms to VPP-5, *VXI Component Knowledge Base Specification*. This file contains detailed information about the VME-MXI-2 such as address space requirements and power consumption. The knowledge base file is intended to be used with software tools that aid in system design, integration, and verification.

The knowledge base file is directly accessible on the disk as a text file with a `.kb` extension. It is also installed on your system when the Setup program is executed as described earlier in this chapter. The knowledge base file is directly accessible for users who are not running Windows and cannot use the Setup program to install the file.

Appendix

A

Specifications

This appendix lists various module specifications of the VME-MXI-2, such as physical dimensions and power requirements.

MXIbus Capability Descriptions

- Master-mode A32, A24 and A16 addressing
- Master-mode block transfers and synchronous block transfers
- Slave-mode A32, A24, and A16 addressing
- Slave-mode block transfers and synchronous block transfers
- Master-mode D32, D16, and D08 data sizes
- Slave-mode D32, D16, and D08 data sizes
- Optional MXIbus System Controller
- Can be a fair MXIbus requester
- Can lock the MXIbus for indivisible transfers
- Can terminate the MXIbus
- MXIbus master retry support
- MXIbus slave retry support
- Interrupt handler for levels 7 to 1
- Interrupt requester for levels 7 to 1
- MXIbus D32, D16, D08(O) interrupt handler
- MXIbus D32, D16, D08(O) interrupter
- Release on Acknowledge or Register Access interrupter
- MXIbus bus timer (programmable limit)
- Automatic MXIbus System Controller detection
- Automatic MXIbus termination detection

VMEbus Capability Codes

Capability Code	Description
A32, A24, A16 (master)	VMEbus master A32, A24, and A16 addressing
A32, A24, A16 (slave)	VMEbus slave A32, A24, and A16 addressing
D32, D16, D08(EO) (master)	VMEbus master D32, D16, and D08 data sizes
D32, D16, D08(EO) (slave)	VMEbus slave D32, D16, and D08 data sizes
BLT, MBLT (master)	VMEbus master block and D64 transfers
BLT, MBLT (slave)	VMEbus slave block and D64 transfers
RMW (master)	VMEbus master read/modify/write transfers
RMW (slave)	VMEbus slave read/modify/write transfers
RETRY (master)	VMEbus master retry support
RETRY (slave)	VMEbus slave retry support
FSD	First slot detector
SCON	VMEbus System Controller
PRI, RRS	Prioritized or Round Robin Select arbiter
ROR, FAIR	Release on Request and FAIR bus requester
IH(7–1)	Interrupt handler for levels 7–1
I(7–1)	Interrupt requester for levels 7–1
D32, D16, D08(O) (Interrupt Handler)	VMEbus D32, D16, D08(O) interrupt handler
D32, D16, D08(O) (Interrupter)	VMEbus D32, D16, D08(O) interrupter
ROAK, RORA	Release on Acknowledge or Register Access interrupter
BTO(*x*)	VMEbus bus timer (programmable limit)

Requirements

Characteristic	Specification
A16 Space	64 B
A24 or A32 Space	16 KB minimum (programmable)

Environmental

Characteristic	Specification
Temperature	0° to 55° C operating; -40° to 85° C storage
Relative Humidity	0% to 95% noncondensing, operating; 0% to 95% noncondensing, storage
EMI	FCC Class A Verified

Physical

Characteristic	Specification
Board Dimensions	VMEbus double-height board 233.36 by 160 mm (9.187 by 6.2999 in.)
Connectors	Single fully implemented MXI-2 bus connector
Slot Requirements	Single VMEbus double-height slot
Compatibility	Fully compatible with VMEbus specification
MTBF	97,000 hours at 25° C
Weight	0.33 Kg (0.73 lb) typical (no DRAM installed)

Electrical

Source	DC Current Ratings	
	Typical	Maximum
+5 VDC	2.2 A	3.2 A

Performance

VME Transfer Rate	
Peak	33 MB/s
Sustained	23 MB/s

Appendix

B

Programmable Configurations

This appendix describes some features of the VME-MXI-2 that are configured by programming an onboard EEPROM through software rather than by onboard switches or jumpers.

Configuring the EEPROM

The EEPROM settings are loaded into the VME-MXI-2 registers after each power-up or hard reset. The VME-MXI-2 must be reset either with a power cycle or by asserting the VMEbus SYSRESET* signal after the EEPROM is written for the changes to take effect. The EEPROM retains its settings even when power is removed from the VME-MXI-2. Once you program the settings into the EEPROM, they need not be programmed again unless you want to make further changes to the settings.

The EEPROM is accessible in the VME-MXI-2 A24 or A32 memory region defined by the VIDR, VDTR, VCR, and VOR registers. It is required that when the Resource Manager executes, it allocates A24 or A32 space to the VME-MXI-2 before the EEPROM can be accessed. If you are not using a multiframe VXIbus Resource Manager, you must allocate A24 or A32 space to the VME-MXI-2 by writing a base address to the VOR and then setting the A24/A32 ENABLE bit in the VCR. The space (either A24 or A32) and amount of address space that the VME-MXI-2 requires can be determined by reading the VIDR and VDTR. Following this allocation, the IOCONFIG bit in the VME-MXI-2 Control Register 2 (VMCR2) must be written with a 1 before the EEPROM is accessible. The IOCONFIG bit should be written with a 0 after accesses to the EEPROM are complete to prevent unintentional accesses to the EEPROM.

The EEPROM must be written with 8-bit accesses. Also, after each write access to the EEPROM, the location written should be

continuously read back until the value written is returned before attempting any further write accesses.

After all changes have been written to the EEPROM, the 32-bit value stored at offset 2FFC hex from the VME-MXI-2 A24 or A32 base address should be incremented. This 32-bit value stores the number of times the EEPROM has been written, since there is a limit of 10,000 writes before writes to the part become unreliable. The 32-bit value can be read with a 32-bit access but must be written with 8-bit accesses followed by reads as described in the previous paragraph.

The following pseudo code is an example of a VME-MXI-2 EEPROM programming sequence. Assume that the VME-MXI-2 has been allocated 200000 hex for its base A24 address. All numbers in the example are in hexadecimal. The four lines of code labeled with the comment "`/* set options here */`" should be repeated for each configuration setting that is being written to the EEPROM. *xxxxxx* represents the address of the EEPROM location being changed, and *xx* represents the value to change it to.

```
LONG writecount, temp;                  /* program variables */
a24_byte_write(address, data);          /* function prototype */
a24_byte_read(address, data);           /* function prototype */
a24_long_read(address, data);           /* function prototype */

a24_byte_write(20075B, 81);             /* set IOCONFIG in VMCR2 */
a24_long_read(202FFC, writecount);      /* get EEPROM write count */
if (writecount < 2710) {                /* check limit (10,000 dec) */
   a24_byte_write(xxxxxx, xx);          /* set options here */
   do {
   a24_byte_read(xxxxxx, temp);
   } while (temp != xx);
   writecount = writecount + 1;         /* increment write count */
   a24_byte_write(202FFC, (writecount & FF000000) >> 24);
   do {
   a24_byte_read(202FFC, temp);
   } while (temp != (writecount & FF000000) >> 24);
   a24_byte_write(202FFD, (writecount & 00FF0000) >> 16);
   do {
   a24_byte_read(202FFD, temp);
   } while (temp != (writecount & 00FF0000) >> 16);
   a24_byte_write(202FFE, (writecount & 0000FF00) >> 8);
   do {
```

```
   a24_byte_read(202FFE, temp);
   } while (temp != (writecount & 0000FF00) >> 8);
   a24_byte_write(202FFF, writecount & 000000FF);
   do {
   a24_byte_read(202FFF, temp);
   } while (temp != writecount & 000000FF);
}
else {                                  /* write limit reached */
   print("Write limit reached - can't write.");
}
a24_byte_write(20075B, 01);             /* clear IOCONFIG in VMCR2 */
```

The following sections describe the features that you can configure by writing to the EEPROM on the VME-MXI-2.

VME-MXI-2 Requested Memory Space

The VME-MXI-2 requires at least 16 KB of either A24 or A32 space. You might want to change the amount of space requested or whether the VME-MXI-2 is an A24 or A32 device. This is especially important when changing the amount of DRAM installed on the VME-MXI-2. The amount of space requested by the VME-MXI-2 should match the amount of DRAM installed. Set it to 16 KB when no DRAM is installed.

Caution: ***If you install DRAM into the VME-MXI-2, do not attempt to use the first 4 KB of memory space. This 4 KB space maps to the registers on the VME-MXI-2 and does not access onboard DRAM. Accessing this region will cause your VME-MXI-2 to behave incorrectly.***

If you do not want to lose 4 KB of DRAM you can get around this limitation by setting the requested memory to double the amount that is installed on the VME-MXI-2, because the DRAM is aliased throughout the remainder of the requested memory space. The DRAM should then be accessed in the upper half of the requested memory space.

To change whether the VME-MXI-2 is an A24 or A32 device, write the EEPROM byte at offset 2016 hex from the VME-MXI-2 base address. Write a 4F hex for A24 or a 5F hex for A32.

To change the amount of space that the VME-MXI-2 requests, write the EEPROM byte at offset 201E hex from the VME-MXI-2 base address. The following table gives the value that should be written for the corresponding size. Notice that the value you should write for any given size differs depending on whether you are requesting A24 or A32 space.

Size	A24 Value (Hex)	A32 Value (Hex)
16 KB	9F	N/A
32 KB	8F	N/A
64 KB	7F	FF
128 KB	6F	EF
256 KB	5F	DF
512 KB	4F	CF
1 MB	3F	BF
2 MB	2F	AF
4 MB	1F	9F
8 MB	0F	8F
16 MB	N/A	7F
32 MB	N/A	6F
64 MB	N/A	5F
128 MB	N/A	4F
256 MB	N/A	3F
512 MB	N/A	2F
1 GB	N/A	1F
2 GB	N/A	0F

VMEbus Timer Limit

The VMEbus Bus Timeout (BTO) is a watchdog timer for transfers on the VMEbus Data Transfer bus. After the specified amount of time has elapsed, the BTO circuitry terminates a VMEbus cycle if no slave has responded. The VME-MXI-2 must provide the VMEbus BTO for proper operation because when a MXIbus cycle is involved, the VMEbus timeout must be disabled and the MXIbus BTO enabled. You should disable the BTO of any other BTO module residing in the mainframe. If this is not possible, set it to its maximum setting to give the MXIbus cycles as much time as possible to complete.

The lowest value in the allowable range is 15 µs and the highest is 256 ms. The default value is 125 µs.

To change the VMEbus timeout limit of the VME-MXI-2, write the EEPROM byte at offset 206F hex from the VME-MXI-2 base address. The following table gives the value that should be written for the corresponding time limit.

Time Limit	Value (Hex)
Timer Disabled	C0
15 µs	C1
30 µs	C2
60 µs	C3
125 µs	C4 (default)
250 µs	C5
500 µs	C6
1 ms	C7
2 ms	C8
4 ms	C9
8 ms	CA
16 ms	CB
32 ms	CC
64 ms	CD
128 ms	CE
256 ms	CF

VMEbus Arbiter

Arbiter Type

You can configure the VME-MXI-2 as either a Priority or Round Robin VMEbus arbiter. This setting is applicable only if the VME-MXI-2 you are configuring is the first slot device. The default is Priority.

When configured for Priority arbitration, the VME-MXI-2 grants the bus to the highest bus request level pending. In Round Robin arbitration mode, the VME-MXI-2 grants the bus to the next highest bus request level after the level of the previous bus owner. This effectively gives the same priority to each bus request level. Refer to the VMEbus specification for more information on the different types of arbiters.

Arbiter Timeout

An arbitration timeout feature is available on the VME-MXI-2 when it is acting as the VMEbus arbiter. This feature applies only to a VME-MXI-2 in the first slot. This feature is enabled by default.

The timer begins when the arbiter circuit on the VME-MXI-2 drives one of the *BGOUT* lines on the backplane. If no device takes over the bus within the timeout limit, the *BGOUT* is removed and the bus is either idle or granted to another requester.

To change the VMEbus arbiter type of the VME-MXI-2, write the EEPROM byte at offset 20B4 hex from the VME-MXI-2 base address. The following table gives the value that should be written for the corresponding arbiter type. The values shown in the table are hexadecimal.

Timeout Status	Priority Arbiter	Round Robin Arbiter
Arbiter Timeout Enabled	00 (default)	80
Arbiter Timeout Disabled	40	C0

VMEbus Requester

Request Level

The VME-MXI-2 uses one of the four VMEbus request levels (0 to 3) to request use of the VME Data Transfer Bus (DTB). The VME-MXI-2 requests use of the DTB whenever an external MXIbus device attempts a transfer that maps into the VMEbus mainframe.

The VME-MXI-2 uses VMEbus request level 3 in its factory-default setting. This is suitable for most VMEbus systems. However, you can change the VME-MXI-2 to use any of the other three request levels (0, 1, or 2) by writing to the EEPROM. You may want to change request levels to change the priority of the VME-MXI-2 request signal.

Fair Request

The VME-MXI-2 is always a Release On Request requester. However, you can configure whether the VME-MXI-2 acts as either a fair or unfair requester on the VMEbus. By default, the VME-MXI-2 is a fair requester. For more information on the different types of requesters, refer to the VMEbus specification.

To change the VMEbus requester type of the VME-MXI-2, write the EEPROM byte at offset 20B5 hex from the VME-MXI-2 base address. The following table gives the value that should be written for the corresponding requester type. The values shown in the table are hexadecimal.

Bus Request Level	Fair Requester	Unfair Requester
Bus Request Level 3	07 (default)	17
Bus Request Level 2	06	16
Bus Request Level 1	05	15
Bus Request Level 0	04	14

MXIbus Timer Limit

The MXIbus Bus Timeout (BTO) is a watchdog timer for transfers on the MXIbus. The MXIbus BTO unit operates only when the VME-MXI-2 is acting as the MXIbus System Controller. The functionality is similar to that of the VMEbus timer limit described previously. The options range from 8 µs to 128 ms, with a default value of 1 ms.

After the specified amount of time has elapsed, the BTO circuitry terminates a MXIbus cycle if no slave has responded. The BTO circuitry is automatically deactivated when the VME-MXI-2 is not acting as the MXIbus System Controller. The BTO is also disabled when the current MXIbus cycle maps to the VMEbus through a VME-MXI-2.

To change the MXIbus timeout limit of the VME-MXI-2, write the EEPROM byte at offset 2067 hex from the VME-MXI-2 base address. The following table gives the value that should be written for the corresponding time limit.

Time Limit	Value (hex)
Timer Disabled	00
8 µs	01
15 µs	02
30 µs	03
60 µs	04
125 µs	05
250 µs	06
500 µs	07
1 ms	08 (default)
2 ms	09
4 ms	0A
8 ms	0B
16 ms	0C
32 ms	0D
64 ms	0E
128 ms	0F

MXIbus Fair Requester and MXIbus Parity Checking

You can configure whether the VME-MXI-2 acts as either a fair or unfair requester on the MXIbus. The default is a fair requester, which causes the VME-MXI-2 to request the MXIbus only when there are no requests pending from other masters. This prevents other masters from being starved of bandwidth. The VME-MXI-2 will request the bus at any time when configured for unfair operation.

MXIbus parity checking can also be disabled in the same EEPROM location as the MXIbus fair requester setting. By default, MXIbus parity checking is enabled and should not be disabled under normal circumstances. MXIbus parity is always generated regardless if checking is enabled or disabled.

To change the MXIbus requester type or the MXIbus parity checking setting of the VME-MXI-2, write the EEPROM byte at offset 2065 hex from the VME-MXI-2 base address. The following table gives the value that should be written for the corresponding requester type and parity checking combination.

Parity Checking Status	Fair Requester	Unfair Requester
Parity Checking Enabled	E5 (default)	C5
Parity Checking Disabled	E1	C1

Interlocked Arbitration Mode

Interlocked arbitration mode is an optional mode of operation in which at any given moment the system can perform as if it were one large VMEbus mainframe with only one master of the entire system—VMEbus and MXIbus. This mode of operation prevents deadlocks by interlocking all arbitration in the VMEbus/MXIbus system.

To change the arbitration mode of the VME-MXI-2, the EEPROM bytes at offsets 2035 and 2037 must be written. Write a 0 to each location for normal arbitration mode, or write a 1 to each location for interlocked arbitration mode.

For more information on interlocked mode, refer to Chapter 6, *VXIplug&play for the VME-MXI-2*.

Appendix

C

VME-MXI-2 Front Panel Configuration

This appendix describes the front panel and connectors on the VME-MXI-2 interface module. This material contains the information relevant to VXI*plug&play* Specification VPP-8, *VXI Module/ Mainframe to Receiver Interconnection*.

The VME-MXI-2 module is National Instruments part number 183105*x*-01, where *x* is the hardware revision letter.

Front Panel

Figure C-1 shows the front panel layout of the VME-MXI-2. The drawing shows dimensions relevant to key elements on the front panel. Dimensions are in mm (inches). The VME-MXI-2 front panel thickness is 2.49 mm (.098 in.).

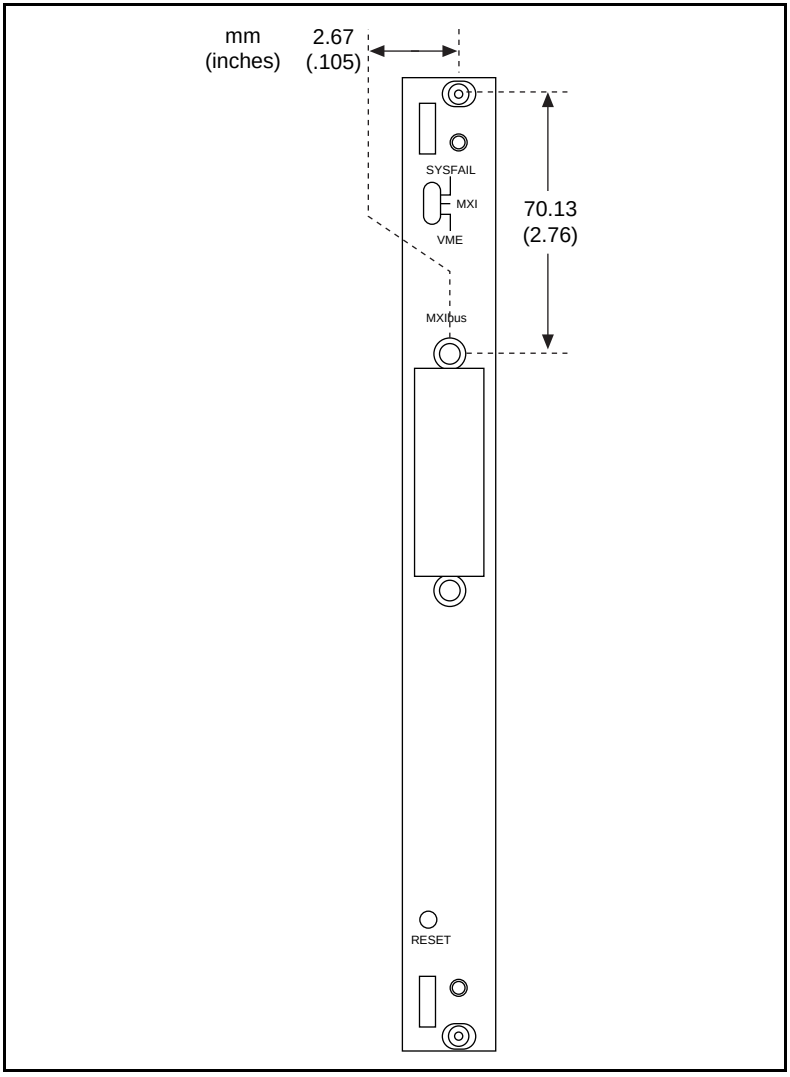


Figure C-1. VME-MXI-2 Front Panel Layout

MXI-2 Connector

The MXI-2 connector is a 144-pin female connector manufactured by Meritec (Meritec part number 182800A-01). The mating cable assembly is National Instruments part number 182801A-*xxx*, where *xxx* is the length in meters.

Figure C-2 shows the MXI-2 connector on the VME-MXI-2. The drawing shows the pinout assignments for each pin, which are described in Table C-1.

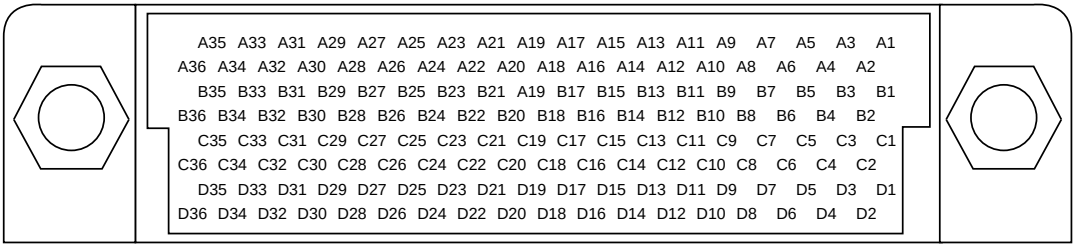


Figure C-2. MXI-2 Connector

Table C-1 lists the signal assignments for the MXI-2 connector.

Table C-1. MXI-2 Connector Signal Assignments

Pin	Signal Name	Pin	Signal Name	Pin	Signal Name	Pin	Signal Name
A1	AD(31)*	B1	AD(14)*	C1	AM(4)*	D1	BUSY*
A2	GND	B2	GND	C2	GND	D2	GND
A3	AD(30)*	B3	AD(13)*	C3	AM(3)*	D3	IRQ(1)*
A4	GND	B4	GND	C4	GND	D4	GND
A5	AD(29)*	B5	AD(12)*	C5	AM(2)*	D5	IRQ(2)*
A6	GND	B6	GND	C6	GND	D6	GND
A7	AD(28)*	B7	AD(11)*	C7	AM(1)*	D7	IRQ(3)*

(continues)

Table C-1. MXI-2 Connector Signal Assignments (Continued)

Pin	Signal Name	Pin	Signal Name	Pin	Signal Name	Pin	Signal Name
A8	GND	B8	GND	C8	GND	D8	GND
A9	AD(27)*	B9	AD(10)*	C9	AM(0)*	D9	IRQ(4)*
A10	GND	B10	GND	C10	GND	D10	GND
A11	AD(26)*	B11	AD(9)*	C11	WR*	D11	IRQ(5)*
A12	GND	B12	GND	C12	GND	D12	GND
A13	AD(25)*	B13	AD(8)*	C13	SIZE*	D13	IRQ(6)*
A14	GND	B14	GND	C14	GND	D14	GND
A15	AD(24)*	B15	AD(7)*	C15	DISBTO*	D15	IRQ(7)*
A16	GND	B16	GND	C16	GND	D16	GND
A17	AD(23)*	B17	AD(6)*	C17	ACFAIL*	D17	TRG(0)+
A18	GND	B18	GND	C18	GND	D18	TRG(0)-
A19	AD(22)*	B19	AD(5)*	C19	SYSRESET*	D19	TRG(1)+
A20	GND	B20	GND	C20	GND	D20	TRG(1)-
A21	AD(21)*	B21	AD(4)*	C21	SYSFAIL*	D21	TRG(2)+
A22	GND	B22	GND	C22	GND	D22	TRG(2)-
A23	AD(20)*	B23	AD(3)*	C23	BERR*	D23	TRG(3)+
A24	GND	B24	GND	C24	GND	D24	TRG(3)-
A25	AD(19)*	B25	AD(2)*	C25	DTACK*	D25	TRG(4)+
A26	GND	B26	GND	C26	GND	D26	TRG(4)-
A27	AD(18)*	B27	AD(1)*	C27	DS*	D27	TRG(5)+
A28	GND	B28	GND	C28	GND	D28	TRG(5)-
A29	AD(17)*	B29	AD(0)*	C29	AS*	D29	TRG(6)+
A30	GND	B30	GND	C30	GND	D30	TRG(6)-
A31	AD(16)*	B31	CONVERT*	C31	BREQ*	D31	TRG(7)+
A32	GND	B32	GND	C32	GND	D32	TRG(7)-
A33	AD(15)*	B33	PAR*	C33	GIN*	D33	CLK10+
A34	GND	B34	GND	C34	GND	D34	CLK10-
A35	5 V	B35	TERMPOWER	C35	GOUT*	D35	MXISC*
A36	5 V	B36	TERMPOWER	C36	GND	D36	ENDDEV

The characteristic impedance of the MXIbus signals is 120 Ω. Table C-2 lists additional characteristics of the MXIbus signals.

Table C-2. MXIbus Signal Characteristics

Signal Category	Voltage Range	Max Current	Frequency Range
Each single-ended signal	0 to 3.4 V	60 mA	DC to 10 Mhz
Each differential signal (D17–D34)	0 to 5 V	80 mA	DC to 10 Mhz
Each 5 V (A35, A36)	5 V	1.75 A fused	DC
Each TERMPOWER (B35, B36)	3.4 V	1.75 A fused	DC

Differences and Incompatibilities between the VME-MXI and the VME-MXI-2

Appendix
D

This appendix describes the differences and incompatibilities between the first-generation MXIbus-to-VMEbus interface, the VME-MXI, and the VME-MXI-2. This information may be helpful for users of the VME-MXI who are moving to the VME-MXI-2.

MXIbus Connector

The VME-MXI-2 interfaces the VMEbus to the National Instruments next-generation MXIbus (MXI-2), while the VME-MXI used the first-generation MXIbus. MXI-2 uses new higher-density connectors and cables, which means that the VME-MXI-2 cannot be readily connected to any first-generation MXIbus device such as the VME-MXI.

A major benefit of MXI-2 is that it combines the MXIbus functionality with all the functionality of the INTX enhancement for the VME-MXI onto a single connector. The INTX enhancement extends the VMEbus utility signals (ACFAIL*, SYSRESET*, and SYSFAIL*) and all seven VMEbus interrupts.

As the following table shows, you need only one cable with the VME-MXI-2 module, whereas the Enhanced VME-MXI requires an additional cable for INTX functionality.

First-Generation MXIbus	MXI-2
Enhanced VME-MXI	VME-MXI-2
MXIbus Cable	MXI-2 Cable
INTX Cable	

In addition to the INTX functionality, MXI-2 incorporates new data transfer protocols that achieve higher performance than is possible on the first-generation MXIbus.

Configuration Switches and Jumpers

Some of the configurable features of the VME-MXI are software programmable settings on the VME-MXI-2; some others are now implemented by automatic configuration circuits on the VME-MXI-2 instead of by onboard switches or jumpers.

One configuration switch on the VME-MXI selected whether the front-panel pushbutton asserted the VMEbus SYSRESET* or ACFAIL* signal. This is not implemented on the VME-MXI-2. The VME-MXI-2 will always assert SYSRESET* when the front-panel pushbutton is pressed.

The following table lists the configurable features that are automatic on the VME-MXI-2 and require no attention.

Configurable Feature	VME-MXI-2 Implementation
MXIbus System Controller	Automatic
MXIbus Termination	Switches available to override automatic detection
VMEbus System Controller	Automatic

The following table lists the configurable features that are programmable on the VME-MXI-2, either through a writable register or a writable location in the onboard EEPROM. Keep in mind that configurations you write to a register will be lost during any hard reset or power cycle, while configurations you write to the EEPROM will remain even through resets and power cycles.

Configurable Feature	VME-MXI-2 Implementation
Interlocked Arbitration Mode	VME-MXI-2 Status/Control Register (VMSR/VMCR) or EEPROM
VMEbus Timeout Length	EEPROM
VMEbus Request Level	EEPROM
MXIbus Timeout Length	Shared MXIbus Status/Control Register (SMSR/SMCR) or EEPROM
MXIbus Fair Requester	Shared MXIbus Status/Control Register (SMSR/SMCR) or EEPROM
MXIbus Parity Checking	Shared MXIbus Status/Control Register (SMSR/SMCR) or EEPROM

VXIbus Model Code

The VXIbus Device Type Register (VDTR) on the VME-MXI-2 returns a different model code than the VME-MXI because it includes new capabilities and is not an identical replacement for the VME-MXI.

Required Memory Space

The VME-MXI-2 register set is too large to fit in its 64-byte VXIbus configuration area. In addition, you can install onboard DRAM on the VME-MXI-2. For both of these reasons the VME-MXI-2 requests at least 16 KB of either A24 or A32 space, whereas the VME-MXI was an A16-only device. As a result, the VME-MXI-2 has a VXIbus Offset Register (VOR), a Required Memory field in the VXIbus Device Type Register (VDTR), and an A24/A32 ENABLE bit in the VXIbus Status/Control Register (VSR/VCR).

Sysfail Inhibit

The VME-MXI-2 provides a Sysfail Inhibit bit in the VXIbus Status/Control Register (VSR/VCR) to prevent it from asserting the SYSFAIL* signal as defined by the VXIbus specification. The first-generation VME-MXI did not.

VME-MXI-2 Status/Control Register (VMSR/VMCR)

The RMWMODE bit is no longer implemented. The VME-MXI-2 uses the MXI-2 bus CONVERT* signal to determine how to pass a MXIbus access to the VMEbus. When CONVERT* is not asserted, MXIbus block and RMW accesses are passed to the VMEbus unmodified. When CONVERT* is asserted, the VME-MXI-2 converts MXIbus block and indivisible accesses into single VMEbus accesses.

The Long MXIbus System Controller Timeout bit (LNGMXSCTO) is no longer implemented. The MXIbus timer of the VME-MXI-2 is programmable in the EEPROM and covers an even broader range of times than the MXIbus timer of the VME-MXI.

The MXSRSTINT, MXACFAILINT, and MXSYSFINT bits are no longer implemented in the VME-MXI-2 Status Register (VMSR). Likewise, the MXSRSTEN and MXACFAILEN bits in the VME-MXI-2 Control Register (VMCR) are no longer implemented. For more information about these bits, refer to the following section, *Local Interrupt Conditions*.

Local Interrupt Conditions

The first-generation MXIbus has a single interrupt line. MXI-2 has seven interrupt lines, which correspond to the VMEbus interrupt lines. The VME-MXI has some interrupt conditions that would assert the single MXIbus interrupt directly. Since MXI-2 does not have this single MXIbus interrupt, the register bits that would enable these conditions are not implemented on the VME-MXI-2. Specifically, the MXSRSTINT, MXSRSTEN, MXACFAILINT, MXACFAILEN, and MXSYSFINT bits in the VME-MXI-2 Status/Control Register (VMSR/VMCR) are not implemented. Also, the entire MXIbus IRQ Configuration Register (offset 24 hex on the VME-MXI) is not implemented.

As an alternative, all these interrupt conditions can be routed to one of the VMEbus interrupt lines, which then can be routed to the corresponding MXI-2 interrupt line. Also, the utility signals SYSRESET*, ACFAIL*, and SYSFAIL* can be routed to MXI-2 to be detected at the destination as a utility signal rather than generating an interrupt and sending the interrupt to the destination. In fact, this is the only solution available for the SYSRESET* signal on the VME-MXI-2. The VME-MXI-2 cannot generate an interrupt from SYSRESET*.

Refer to the register descriptions for the VXIbus Interrupt Configuration Register (VICR), VXIbus Utility Configuration Register (VUCR), and VMEbus Interrupt Status/Control Register (VISTR/VICTR) in Chapter 4 for more information on these alternatives to the local interrupt conditions on the single MXIbus interrupt line. Notice that these same registers and solutions work on an Enhanced VME-MXI when the destination has an INTX connection.

Hard Reset

The VMEbus Status/ID Register (VSIDR) on the VME-MXI-2 is cleared on a hard reset. This register was unaffected by a hard reset on the VME-MXI.

The INTLK bit in the VME-MXI-2 Status Register (VMSR) is set to the value stored in the EEPROM on a hard reset. By default, the value is 0. The INTLK bit was unaffected by a hard reset on the VME-MXI since it was an onboard switch.

Soft Reset

The following register bits, which are cleared by a soft reset on the VME-MXI, are unaffected by a soft reset on the VME-MXI-2.

- CMODE in the VME-MXI-2 Status/Control Register (VMSR/VMCR)
- DIRQ[7:1] in the VMEbus Interrupt Control Register (VICTR)

Appendix E

Configuring a Two-Frame System

This appendix describes how to configure a system containing two mainframes linked by VME-MXI-2 mainframe extenders.

Configuring Two VME-MXI-2 Modules for a Two-Frame System

The factory configuration of the VME-MXI-2 is suitable for the most common system configurations. However, if you are setting up a VME system using VME-MXI-2 modules to extend from one mainframe to another, you need to reconfigure the VME-MXI-2 interfaces. You can find more information about configuring a multiframe system in Chapter 3, *VME-MXI-2 Configuration and Installation*, which describes the switch settings, and Chapter 5, *System Configuration*, which describes the partitions of system resources, including logical addresses. This appendix is a quick reference for systems such as the one in Figure E-1, which consists of two VME mainframes connected by a single MXIbus link.

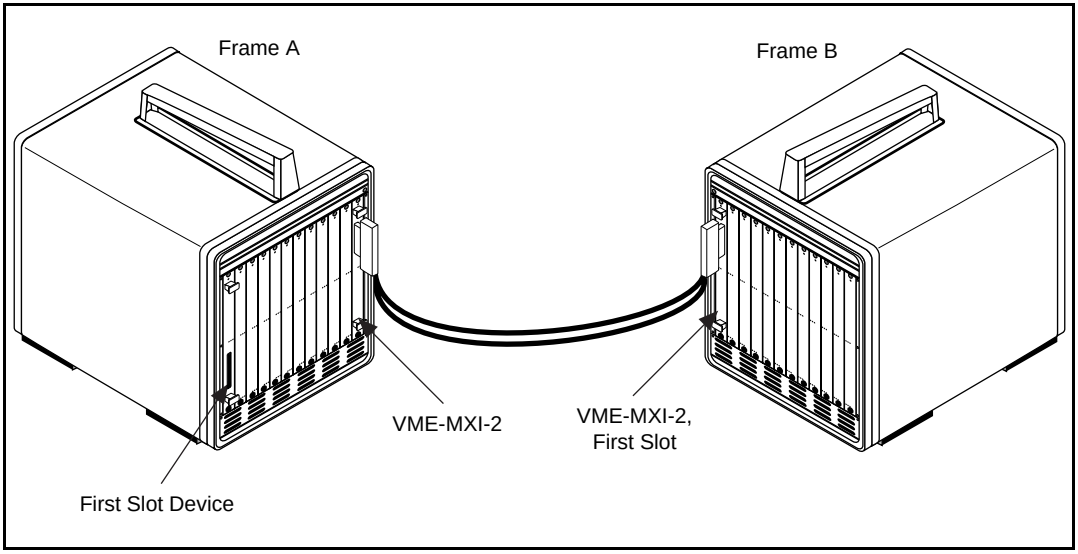


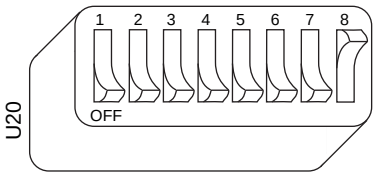
Figure E-1. A Two-Frame VME System

In the example shown in Figure E-1, Frame A contains a VME-MXI-2 installed in a slot other than the first slot. It is logical address 1. Frame B contains a VME-MXI-2 installed in the first slot. It is logical address 80 hex.

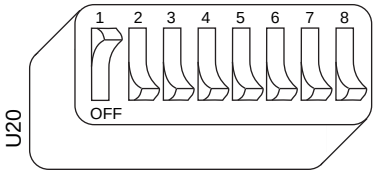
VME-MXI-2 Logical Address

Frame A contains logical addresses in the range of 0 to 7F hex. The VME-MXI-2 is logical address 1 (A16 base address C040 hex), which is the default logical address. Figure E-2a shows the switch setting for logical address 1. Ensure that no other devices in that frame have logical address 1. In addition, no devices in Frame A should have logical addresses of 80 hex or above.

Frame B contains logical addresses from 80 hex to FE hex. The VME-MXI-2 in Frame B is logical address 80 hex (A16 base address E000 hex), as shown in Figure E-2b. Make sure that no other devices in Frame B have logical addresses of 80 hex or below.



a. Switch Set to A16 Base Address C040 Hex (Default)



b. Switch Set to A16 Base Address E000 Hex

Figure E-2. A16 Base Address Selection

VMEbus First Slot

The VME-MXI-2 automatically detects if it is installed in the first slot. Because of the automatic detection feature, you can install the VME-MXI-2 in any slot of a VMEbus mainframe. In the two-frame system described in this appendix, the VME-MXI-2 is installed in the first slot in Frame B, but in a different slot in Frame A. You could also install both in the first slot of their respective mainframes, or both in slots other than the first slot.

MXIbus System Controller

The default setting of the VME-MXI-2 is to automatically detect from the MXIbus cable if it is the MXIbus System Controller. With automatic detection, you can connect the cable in either direction. Notice that one end of the cable is labeled to designate it as the end to attach to the MXIbus System Controller. The VME-MXI-2 you connect to the labeled end of the cable will take on the responsibilities of the MXIbus System Controller.

VMEbus BTO Unit

In each mainframe, the VME-MXI-2 must be the sole bus timer on the VMEbus regardless of its slot location within the mainframe. Be sure to disable the bus timers on all other modules in the mainframes for proper operation.

Address Mapping

If you are not using a VXIbus multiframe Resource Manager, you will have to configure the four address mapping registers on the two VME-MXI-2 modules. For simplicity sake, you can map the lower half of each address space (Logical Address, A16, A24, and A32) to Frame A and the upper half to Frame B. To accomplish this, you should write the value 6100 hex into each of the Extender window registers (VWR0, VWR1, VWR2, and VWR3) on the VME-MXI-2 in Frame A and the value 6180 hex into each of the Extender window registers on the VME-MXI-2 in Frame B. You should configure all addressable resources in Frame A to be located in the lower half of their respective address spaces. Likewise, all addressable resources in Frame B should be located in the upper half of their respective address space.

In addition to address space mapping, you might want to map VMEbus utility signals (SYSRESET*, SYSFAIL*, and ACFAIL*) and interrupt requests between frames. For information on how to do this, refer to Chapter 4, *Register Descriptions*, for information about the VXIbus Interrupt Configuration Register (VICR) and VXIbus Utility Configuration Register (VUCR).

Appendix F

DMA Programming Examples

This appendix contains two example programs for using the DMA controllers on the VME-MXI-2. If you are using a version of the National Instruments NI-VXI software that has remote DMA controller functionality, this information is not necessary because you can make use of the VME-MXI-2 module's DMA controllers from the NI-VXI high-level function calls.

Overview of Programming Examples

The DMA controllers each have registers that define the source and destination of the data. Data is always transferred from the source to the destination during a DMA operation. The DMA controllers can transfer data between devices located on different buses—VMEbus, MXIbus, and DRAM onboard the VME-MXI-2—as well as between devices located on the same bus. The only limitation regarding location of the devices is that MXIbus synchronous burst transfers cannot be used for either the source or destination when both devices are located on the MXIbus. The source and destination can each use differing data widths, address spaces, and transfer types (single or block) during a DMA operation. Detailed descriptions of each DMA register can be found in the *VMEbus A24/A32 Registers* section of Chapter 4, *Register Descriptions*.

The only difference between the two examples in this appendix is that Example 1 does not make use of the DMA interrupt; Example 2 does show this functionality. The examples use pseudo code that mostly resembles the C language. Constant numbers in the examples preceded by *0x* are in hexadecimal notation. Both examples contain the two functions `write` and `read`. These are meant to represent the method for performing VMEbus or MXIbus data transfers to (write) and from (read) devices.

Parameter Descriptions

The parameters for both functions are ADDRESS_SPACE, ADDRESS, TRANSFER_SIZE, and DATA.

- ADDRESS_SPACE represents the VMEbus address space in which the write or read will take place. The examples assume the VME-MXI-2 is located in A24 space.
- ADDRESS represents the address in the memory space to which to perform the write or read. In the examples, A24BASE represents the base A24 address of the VME-MXI-2. Any register name in the examples represents the offset of that register defined in Chapter 4, *Register Descriptions*.
- The TRANSFER_SIZE parameter can be one of BYTE, WORD, or LONGWORD representing 8-bit, 16-bit, and 32-bit data transfers, respectively.
- The DATA parameter is a constant for writes that represents the data to be written and is a program variable for reads that store the data read.

Example 1: DMA Operation without Interrupt

This example simply programs DMA controller 1 to move data from the source on the VMEbus to the destination on the MXIbus. The source is located in A24 space beginning at address location 200000 hex. VMEbus 32-bit block cycles are used to read data from the source. The destination is located in A32 space beginning at address location 40000000 hex. MXIbus 32-bit synchronous burst cycles are used to write data to the destination.

```
/*********************************************************
 *                                                       *
 * Initialization: This section needs to be executed only  *
 * once prior to any DMA activity and does not have to be  *
 * repeated for each DMA operation.                      *
 *                                                       *
 *********************************************************/
```

```
/* The following write causes any block transfer to the MXIbus
from either DMA controller to be a synchronous burst transfer by
setting both DMAxMBS bits in the SMCR. You can modify this write
so that both DMA controllers perform normal MXIbus block
transfers, or you can have one DMA controller perform normal
MXIbus block transfers and the other perform synchronous burst
transfers. Remember that MXIbus synchronous burst transfers cannot
be used when both the source and destination are located on the
MXIbus. */

   write(A24, A24BASE + SMCR, BYTE, 0x38);

/* The following write is required to initialize the CHCRx for the
DMA controller that will be used. If you will be using both DMA
controllers, perform this write to both CHCR1 and CHCR2. */

   write(A24, A24BASE + CHCR1, LONGWORD, 0x00004000);

/*********************************************************
 *                                                       *
 * Operation Setup:  This section sets up one of the DMA   *
 * controllers to perform a data transfer from the VMEbus  *
 * to the MXIbus and starts the operation. Repeat this     *
 * process for each DMA operation. You can also perform    *
 * these steps to the other DMA controller to start        *
 * another operation without waiting for the first one to  *
 * complete.                                               *
 *                                                       *
 *********************************************************/


/* The following write sets up the DMA Source Configuration
Register. It indicates that the source is located on the VMEbus
and that 32-bit block transfers with the address modifier code
0x3B will be used to access it. Table F-1, at the end of this
appendix, describes the address modifier codes that can be written
to this register. Remember that if the source is DRAM onboard the
VME-MXI-2, the address modifier code should be written with 0.
This step can be skipped if SCR1 was already written with the same
value from a previous DMA operation. This is useful if you will be
performing several DMA operations where the source device remains
constant. */

   write(A24, A24BASE + SCR1, LONGWORD, 0x00E047BB);
```

```
/* The following write sets up the base address at which the data
will be acquired from the source. Remember that if the source is
DRAM onboard the VME-MXI-2, the offset within the module's space
should be written to this register, not the VMEbus address of the
source. To compute this value from the source's VMEbus address,
just subtract the VME-MXI-2 module's A24 or A32 base address. */

   write(A24, A24BASE + SAR1, LONGWORD, 0x00200000);

/* The following write sets up the DMA Destination Configuration
Register. It indicates that the destination is located on the
MXIbus and that 32-bit block transfers with the address modifier
code 0x0B will be used to access it. Synchronous burst transfers
will actually be used because the DMA1MBS bit in the SMCR was set
in the Initialization section of this example. Table F-1, at the
end of this appendix, describes the address modifier codes that
can be written to this register. Remember that if the destination
is DRAM onboard the VME-MXI-2, the address modifier code should be
written with 0. This step can be skipped if DCR1 was already
written with the same value from a previous DMA operation. This is
useful if you will be performing several DMA operations where the
destination device remains constant. */

   write(A24, A24BASE + DCR1, LONGWORD, 0x00E047CB);

/* The following write sets up the base address at which the data
will be written to the destination. Remember that if the
destination is DRAM onboard the VME-MXI-2, the offset within the
module's space should be written to this register, not the VMEbus
address of the destination. To compute this value from the
destination's VMEbus address, just subtract the VME-MXI-2 module's
A24 or A32 base address. */

   write(A24, A24BASE + DAR1, LONGWORD, 0x40000000);

/* The following write sets up the transfer count for the DMA
operation. Remember that TCRx is written with the number of bytes
to be transferred regardless of the data width being used for the
source or destination. In this example, 4 KB will be transferred.
Also remember the limits imposed on the transfer count when
performing MXIbus synchronous burst operations described in the
TCRx register description in Chapter 4, Register Descriptions. */

   write(A24, A24BASE + TCR1, LONGWORD, 0x00001000);

/* The following write sets the START bit in CHOR1. This causes
the DMA controller to actually begin the operation. */

   write(A24, A24BASE + CHOR1, LONGWORD, 0x00000001);
```

```
/*************************************************************
 *                                                           *
 * Operation Termination:  This section waits for the DMA    *
 * operation to complete. It is important that the           *
 * operation complete before either using the data that      *
 * is being sent to the destination or reprogramming any     *
 * of the DMA registers for another operation.               *
 *                                                           *
 *************************************************************/
/* The following do-while loop waits for the DMA operation to
complete by polling for the DONE bit in CHSR1 to be 1. After
leaving this loop, the DMA operation has completed either
successfully or due to an error. */
   do {
      read(A24, A24BASE + CHSR1, LONGWORD, value);
   } while((value & 0x02000000) == 0);
/* The following if statement checks if any errors occurred during
the DMA operation by checking the state of the ERROR bit that was
read from CHSR1 when the DONE bit became 1. If the expression is
false, the DMA operation completed successfully and the data at
the destination can now be used. If the expression is true, the
SERR[1:0] and DERR[1:0] bits of CHSR1 should be checked to
determine what type of error occurred. */
   if (value & 0x00008000) {
      /* The DMA operation encountered an error. */
   }
```

Example 2: DMA Operation With Interrupt

This example is similar to Example 1 in that it programs DMA controller 1 to perform the same data transfer from the source on the VMEbus to the destination on the MXIbus. The source is located in A24 space beginning at address location 200000 hex. VMEbus 32-bit block cycles are used to read data from the source. The destination is located in A32 space beginning at address location 40000000 hex. MXIbus 32-bit synchronous burst cycles are used to write data to the destination.

This example adds code to make use of the DMA interrupt functionality on the VME-MXI-2. Using the interrupt to determine when a DMA operation is complete can improve performance over the polling method described in Example 1 because the read cycles used to poll CHSRx will be using bandwidth on whichever bus (VMEbus or MXIbus) the host is located. The bandwidth the host is using to poll CHSRx will not be available to the VME-MXI-2 module's DMA controller. Using the DMA interrupt alleviates this problem since the host is not required to poll. Because the DMA interrupt is common between the two DMA controllers, you must be especially careful to ensure that no interrupts are lost when both DMA controllers are running. This example demonstrates how this can be achieved even though only one DMA controller is being used in the example.

```
/*************************************************************
 *                                                           *
 * Initialization:  This section needs to be executed only   *
 * once prior to any DMA activity and does not have to be    *
 * repeated for each DMA operation.                          *
 *                                                           *
 *************************************************************/

/* The following write causes any block transfer to the MXIbus
from either DMA controller to be a synchronous burst transfer by
setting both DMAxMBS bits in the SMCR. You can modify this write
so that both DMA controllers perform normal MXIbus block
transfers, or you can have one DMA controller perform normal
MXIbus block transfers and the other perform synchronous burst
transfers. Remember that MXIbus synchronous burst transfers cannot
be used when both the source and destination are located on the
MXIbus. */

   write(A24, A24BASE + SMCR, BYTE, 0x38);

/* The following write is required to initialize the CHCRx for the
DMA controller that will be used. Notice that the DONE interrupt
condition is being enabled here but the overall DMA interrupt for
this controller is not being enabled yet. This is because the DMA
controller is already in the DONE state on power up. The DMA
interrupt for the controller being used will be enabled after
starting the operation because the DONE condition will then be
clear until the operation is complete. If you will be using both
DMA controllers you should perform this write to both CHCR1 and
CHCR2. */

   write(A24, A24BASE + CHCR1, LONGWORD, 0x02004000);
```

```
/* The following write is required to initialize the DMAICR. In
this example, the DMA interrupt is being routed to VMEbus IRQ5*.
You can route the DMA interrupt to any VMEbus interrupt level in
the DMAICR. You can put the DMA interrupt on the same level as
other interrupt conditions on the VME-MXI-2 as well as interrupt
conditions on other devices. This write is also programming the
DMA interrupt condition to use a 16-bit Status ID when being
acknowledged. You can change this write if you prefer an 8-bit
Status ID. If you select an 8-bit Status ID you should also decide
if you want the contents of the DMAISIDR or the VME-MXI-2 module's
logical address returned during the interrupt acknowledge cycle.
*/

   write(A24, A24BASE + DMAICR, WORD, 0x2805);

/* The following write is required to initialize the DMAIER. This
is simply enabling the DMA interrupt condition to be routed to the
VMEbus. */

   write(A24, A24BASE + DMAIER, BYTE, 0x09);

/* The following write sets up the DMAISIDR. This is just a Status
ID code that the VME-MXI-2 will return during an interrupt
acknowledge cycle for the DMA interrupt condition. You should
program a code that you can use in your interrupt service routine
to uniquely identify the VME-MXI-2 module's DMA interrupt
condition. Remember that you can change only the 5 most
significant bits of the Status ID using the DMAISIDR. For a 16-bit
Status ID, the VME-MXI-2 always uses 011 binary for bits 10
through 8 and its logical address for bits 7 through 0. For an
8-bit Status ID, the VME-MXI-2 can return either the contents of
the DMAISIDR (the value you write here with bits 2 through 0
forced to 011 binary) or the logical address of the VME-MXI-2
module. */

   write(A24, A24BASE + DMAISIDR, WORD, 0x0013);
```

```
/*********************************************************
 *                                                       *
 * Operation setup:  This section sets up one of the DMA   *
 * controllers to perform a data transfer from the VMEbus  *
 * to the MXIbus and starts the operation. This process    *
 * should be repeated for each DMA operation. You can also *
 * perform these steps to the other DMA controller to      *
 * start another operation without waiting for the first   *
 * one to complete.                                        *
 *                                                       *
 *********************************************************/

/* The following write sets up the DMA Source Configuration
Register. It indicates that the source is located on the VMEbus
and that 32-bit block transfers with the address modifier code
0x3B will be used to access it. Table F-1, at the end of this
appendix, describes the address modifier codes that can be written
to this register. Remember that if the source is DRAM onboard the
VME-MXI-2, the address modifier code should be written with 0.
This step can be skipped if SCR1 was already written with the same
value from a previous DMA operation. This is useful if you will be
performing several DMA operations where the source device remains
constant. */

   write(A24, A24BASE + SCR1, LONGWORD, 0x00E047BB);

/* The following write sets up the base address at which the data
will be acquired from the source. Remember that if the source is
DRAM onboard the VME-MXI-2, the offset within the module's space
should be written to this register, not the VMEbus address of the
source. To compute this value from the source's VMEbus address,
just subtract the VME-MXI-2 module's A24 or A32 base address. */

   write(A24, A24BASE + SAR1, LONGWORD, 0x00200000);

/* The following write sets up the DMA Destination Configuration
Register. It indicates that the destination is located on the
MXIbus and that 32-bit block transfers with the address modifier
code 0x0B will be used to access it. Synchronous burst transfers
will actually be used since the DMA1MBS bit in the SMCR was set in
the Initialization section of this example. Table F-1, at the end
of this appendix, describes the address modifier codes that can be
written to this register. Remember that if the destination is DRAM
onboard the VME-MXI-2, the address modifier code should be written
```

```
with 0. This step can be skipped if DCR1 was already written with
the same value from a previous DMA operation. This is useful if
you will be performing several DMA operations where the
destination device remains constant. */

   write(A24, A24BASE + DCR1, LONGWORD, 0x00E047CB);

/* The following write sets up the base address at which the data
will be written to the destination. Remember that if the
destination is DRAM onboard the VME-MXI-2, the offset within the
module's space should be written to this register, not the VMEbus
address of the destination. To compute this value from the
destination's VMEbus address, just subtract the VME-MXI-2 module's
A24 or A32 base address. */

   write(A24, A24BASE + DAR1, LONGWORD, 0x40000000);

/* The following write sets up the transfer count for the DMA
operation. Remember that the TCRx is written with the number of
bytes to be transferred, regardless of the data width being used
for the source or destination. In this example 4 KB will be
transferred. Also remember the limits imposed on the transfer
count when performing MXIbus synchronous burst operations
described in the TCRx register description in Chapter 4, Register
Descriptions. */

   write(A24, A24BASE + TCR1, LONGWORD, 0x00001000);

/* The following write sets the START bit in CHOR1. This causes
the DMA controller to actually begin the operation. */

   write(A24, A24BASE + CHOR1, LONGWORD, 0x00000001);

/* The following write enables the DMA interrupt condition from
DMA controller 1. Since this occurs after the Start bit is set in
CHOR1, the DONE bit will be clear and the interrupt will not
assert until the DMA operation completes. */

   write(A24, A24BASE + CHCR1, LONGWORD, 0x80004000);
/*********************************************************
 *                                                       *
 * Interrupt service routine:  This section demonstrates  *
 * how an interrupt service routine should handle the DMA  *
 * interrupt condition.                                  *
 *                                                       *
 *********************************************************/
```

```
/* The following read generates a 16-bit interrupt acknowledge
cycle for level 5 and stores the Status ID returned in the value
variable. */

   read(IACK, LEVEL5, WORD, value);

/* The following if statement checks if the Status ID returned
from the interrupt acknowledge cycle matches the code for the
VME-MXI-2 module's DMA interrupt condition (assuming the logical
address of the VME-MXI-2 module is 1). The upper bits of the
Status ID code were written to the DMAISIDR in the Initialization
section of this example. If the expression is false, some other
condition asserted the interrupt. */

   if (value == 0x1301) {

   /* At this point it is known that the VME-MXI-2 module's DMA
   interrupt condition is the highest priority interrupter because
   of the Status ID from the interrupt acknowledge cycle. The
   following two sections of code are identical. The first section
   applies if DMA controller 1 interrupted, and the second section
   applies if DMA controller 2 interrupted. */

      /* DMA controller 1 section */

      read(A24, A24BASE + CHSR1, LONGWORD, value);

      /* The following if statement checks if DMA controller 1 is
      currently interrupting. */

      if (value & 0x80000000) {

         /* At this point it is known that DMA controller 1 is
         the interrupter. The DONE and ERROR bits of CHSR1 should
         be checked for a successful operation. This could be
         handled either here or in the main program after the
         interrupt service routine has exited. If handled here,
         the value variable already contains a copy of CHSR1. The
         following three writes re-arm the DMA interrupt
         condition. This must be done because it is possible that
         the other DMA controller is also interrupting. Notice
         that the overall DMA interrupt in CHCR1 (for DMA
         controller 1) is left disabled when the interrupt
         service routine exits. It will be re-enabled the next
         time DMA controller 1 is started (as shown in the last
         write of the operation setup section earlier in this
         example). */
```

```
            write(A24, A24BASE + DMAIER, BYTE, 0x08);
            write(A24, A24BASE + CHCR1, LONGWORD, 0x40004000);
            write(A24, A24BASE + DMAIER, BYTE, 0x09);
            return_from_interrupt();
        }
        /* DMA controller 2 section */
        read(A24, A24BASE + CHSR2, LONGWORD, value);
        /* The following if statement checks if DMA controller 2 is
        currently interrupting. */
        if (value & 0x80000000) {
            /* At this point it is known that DMA controller 2 is
            the interrupter. The DONE and ERROR bits of CHSR2 should
            be checked for a successful operation. This could be
            handled either here or in the main program after the
            interrupt service routine has exited. If handled here,
            the value variable already contains a copy of CHSR2. The
            following three writes re-arm the DMA interrupt
            condition. This must be done because it is possible that
            the other DMA controller is also interrupting. Notice
            that the overall DMA interrupt in CHCR2 (for DMA
            controller 2) is left disabled when the interrupt
            service routine exits. It will be re-enabled the next
            time DMA controller 2 is started (as shown in the last
            write of the operation setup section earlier in this
            example). */
            write(A24, A24BASE + DMAIER, BYTE, 0x08);
            write(A24, A24BASE + CHCR2, LONGWORD, 0x40004000);
            write(A24, A24BASE + DMAIER, BYTE, 0x09);
            return_from_interrupt();
        }
/* The interrupt service routine should never reach this point. If
it did, it would indicate that the Status ID of the VME-MXI-2
module's DMA interrupt condition was returned during the interrupt
acknowledge cycle yet neither DMA controller indicated it was
interrupting. */
    print("Error message.");
    return_from_interrupt();
}
```

Table F-1. Address Modifier Codes

Code (Hex)	Description
3F	A24 supervisory block transfer
3E	A24 supervisory program access
3D	A24 supervisory data access
3C	A24 supervisory 64-bit block transfer
3B	A24 nonprivileged block transfer
3A	A24 nonprivileged program access
39	A24 nonprivileged data access
38	A24 nonprivileged 64-bit block transfer
37	Reserved
36	Reserved
35	Reserved
34	Reserved
33	Reserved
32	Reserved
31	Reserved
30	Reserved
2F	Reserved
2E	Reserved
2D	A16 supervisory access
2C	Reserved
2B	Reserved
2A	Reserved
29	A16 nonprivileged access
28	Reserved
27	Reserved
26	Reserved

(continues)

Table F-1. Address Modifier Codes (Continued)

Code (Hex)	Description
25	Reserved
24	Reserved
23	Reserved
22	Reserved
21	Reserved
20	Reserved
1F	User-defined
1E	User-defined
1D	User-defined
1C	User-defined
1B	User-defined
1A	User-defined
19	User-defined
18	User-defined
17	User-defined
16	User-defined
15	User-defined
14	User-defined
13	User-defined
12	User-defined
11	User-defined
10	User-defined
0F	A32 supervisory block transfer
0E	A32 supervisory program access
0D	A32 supervisory data access
0C	A32 supervisory 64-bit block transfer
0B	A32 nonprivileged block transfer

(continues)

Table F-1. Address Modifier Codes (Continued)

Code (Hex)	Description
0A	A32 nonprivileged program access
09	A32 nonprivileged data access
08	A32 nonprivileged 64-bit block transfer
07	Reserved
06	Reserved
05	Reserved
04	Reserved
03	Reserved
02	Reserved
01	Reserved
00	Reserved

Appendix

G

Mnemonics Key

This appendix contains an alphabetical listing of mnemonics used in this manual to describe signals and terminology specific to MXIbus, VMEbus, VXIbus, and register bits. Refer also to the *Glossary*.

The mnemonic types are abbreviated as follows:

Abbreviation	Meaning
B	Bit
MBS	MXIbus Signal
MXI	MXIbus Terminology
R	Register
VBS	VMEbus Signal
VME	VMEbus Terminology
VXI	VXIbus Terminology

Mnemonic	Type	Definition
A		
A16BASE[7:0]	B	Extender A16 Window Base
A16DIR	B	Extender A16 Window Direction
A16EN	B	Extender A16 Window Enable
A16SIZE[2:0]	B	Extender A16 Window Size
A24/A32 ACTIVE	B	A24/A32 Active
A24/A32 ENABLE	B	A24/A32 Enable
A24BASE[7:0]	B	Extender A24 Window Base
A24DIR	B	Extender A24 Window Direction
A24EN	B	Extender A24 Window Enable
A24SIZE[2:0]	B	Extender A24 Window Size
A32BASE[7:0]	B	Extender A32 Window Base
A32DIR	B	Extender A32 Window Direction
A32EN	B	Extender A32 Window Enable
A32SIZE[2:0]	B	Extender A32 Window Size
ABORT	B	Abort DMA Operation
ACCDIR	B	Access Direction
ACFAIL	B	ACFAIL* Status
ACFAIL*	VBS	VME ACFAIL Signal
ACFIN	B	ACFAIL* In
ACFOUT	B	ACFAIL* Out
ADSPC[1:0]	B	Address Space
AFIE	B	ACFAIL* Interrupt Enable
AFINT	B	VMEbus ACFAIL* Interrupt Status
AM[5:0]	B	Address Modifiers
ASCEND	B	Ascending Addresses
B		
BERR*	VBS/MBS	Bus Error
BKOFF	B	Back Off Status
BKOFFIE	B	Back Off Interrupt Enable

Mnemonic	Type	Definition
BLOCKEN	B	Block Mode DMA

C

CHCR1	R	DMA Channel 1 Control
CHCR2	R	DMA Channel 2 Control
CHOR1	R	DMA Channel 1 Operation
CHOR2	R	DMA Channel 2 Operation
CHSR1	R	DMA Channel 1 Status
CHSR2	R	DMA Channel 2 Status
CLR DMAIE	B	Clear DMA Interrupt Enable
CLRDONE	B	Clear DONE
CLR DONEIE	B	Clear DONE Interrupt Enable
CMODE	B	Comparison Mode/Comparison Mode Status
CONVERT*	MBS	Convert

D

DA[31:0]	B	Destination Address
DAR1	R	DMA Channel 1 Destination Address
DAR2	R	DMA Channel 2 Destination Address
DCR1	R	DMA Channel 1 Destination Configuration
DCR2	R	DMA Channel 2 Destination Configuration
DERR[1:0]	B	Destination Error Status
DEVCLASS[1:0]	B	Device Class
DIRQ[7:1]	B	Drive VMEbus Interrupt Request [7:1]
DMAICR	R	DMA Interrupt Configuration
DMAIEN	B	DMA Interrupt Enable
DMAIER	R	DMA Interrupt Enable
DMAISIDR	R	DMA Interrupt Status/ID
DMAMB S/N*	B	DMA MXIbus Block Synchronous/Normal*
DMASID[7:3]	B	DMA Status/ID 7 through 3
DMA1MBS	B	DMA Controller 1 MXIbus Block Select
DMA2MBS	B	DMA Controller 2 MXIbus Block Select

Mnemonic	Type	Definition
DONE	B	DMA Done
DSYSFAIL	B	Drive SYSFAIL*/ Drive SYSFAIL* Status
DSYSRST	B	Drive SYSRESET*

E

ECR[7:0]	B	Empty Count Register
EDTYPE[3:0]	B	Extended Device Type Class
ENABLE	B	Enable Interrupt
ERROR	B	DMA Error

F

FAIR	B	MXIbus Fair Requester
FCR[7:0]	B	Full Count Register
FCR1	R	DMA Channel 1 FIFO Count
FCR2	R	DMA Channel 2 FIFO Count
FRESET	B	DMA FIFO Reset

I

I1[15:0]	B	Level 1 Interrupter Status ID
I2[31:0]	B	Level 2 Interrupter Status ID
I3[15:0]	B	Level 3 Interrupter Status ID
I4[31:0]	B	Level 4 Interrupter Status ID
I5[15:0]	B	Level 5 Interrupter Status ID
I6[31:0]	B	Level 6 Interrupter Status ID
I7[15:0]	B	Level 7 Interrupter Status ID
IACK*	VME	VMEbus Interrupt Acknowledge
ILVL[2:0]	B	DMA Interrupt Level
INT	B	DMA Interrupt
INTDIR[7:1]	B	Interrupt Direction
INTEN[7:1]	B	Interrupt Enable
INTLCK	B	Interlocked Mode/Interlocked Status

Mnemonic	Type	Definition
IOCONFIG	B	I/O Configuration Space Enable
IRQ*	MBS	MXIbus Interrupt Request
IRQ[7:1]	B	VMEbus Interrupt Request [7:1] Status
ISTAT	B	DMA Interrupt Status

L

LA[7:0]	B	Logical Address Status
LABASE[7:0]	B	Extender Logical Address Window Base
LADIR	B	Extender Logical Address Window Direction
LAEN	B	Extender Logical Address Window Enable
LASIZE[2:0]	B	Extender Logical Address Window Size
LINT[3:1]	B	Local Interrupt Level
LOCKED	B	VMEbus or MXIbus Locked

M

MANID[11:0]	B	Manufacturer ID
MBERR	B	MXIbus Bus Error Status
MBTO[3:0]	B	MXIbus Timeout Value
MODEL[11:0]	B	Model Code
MODID*	B	MODID Line Status
MXISC	B	MXIbus System Controller Status
MXSCTO	B	MXIbus System Controller Timeout Status

O

OFFSET[15:0]	B	VMEbus Offset

P

PAREN	B	MXIbus Parity Enable
PARERR	B	Parity Error Status
PASSED	B	Passed
PORT[1:0]	B	Port

Mnemonic	Type	Definition
POSTERR	B	Write Post Error Status
PRI ARBITER	VME	VMEbus Prioritized Arbiter

R

READY	B	Ready
REQMEM[3:0]	B	Required Memory
RESET	B	Soft Reset
RETRY*	VBS	Retry signal
ROAK	VME	Release on Interrupt Acknowledge
RR ARBITER	VME	VMEbus Round Robin Arbiter

S

S[15:0]	B	Status ID
SA[31:0]	B	Source Address
SABORT	B	DMA Software Abort
SAR1	R	DMA Channel 1 Source Address
SAR2	R	DMA Channel 2 Source Address
SC[15:0]	B	Subclass
SCFG	B	Self Configuration Status
SCR1	R	DMA Channel 1 Source Configuration
SCR2	R	DMA Channel 2 Source Configuration
SERR[1:0]	B	Source Error Status
SET DMAIE	B	Set DMA Interrupt Enable
SET DONEIE	B	Set DONE Interrupt Enable
SFIE	B	SYSFAIL* Interrupt Enable
SFIN	B	SYSFAIL* In
SFINH	B	Sysfail Inhibit
SFINT	B	VMEbus SYSFAIL* Interrupt Status
SFOUT	B	SYSFAIL* Out
SID8	B	8-bit Status/ID
SIDLA	B	Logical Address Status/ID
SMCR	R	Shared MXIbus Control Register

Mnemonic	Type	Definition
SMSR	R	Shared MXIbus Status Register
SRIN	B	SYSRESET* In
SROUT	B	SYSRESET* Out
START	B	Start DMA Operation
Status/ID	VME	VMEbus Interrupt Status/Identification Data
STOP	B	Stop DMA Operation
STOPS	B	DMA Stopped Status
SYSFAIL	B	SYSFAIL* Status
SYSFAIL*	VME	System Failure
SYSRESET*	VME	System Reset

T

TC[31:0]	B	Transfer Count
TCR1	R	DMA Channel 1 Transfer Count
TCR2	R	DMA Channel 2 Transfer Count
TSIZE[1:0]	B	Transfer Size

U

UTIL*	B	Utility Signal Support

V

VCR	R	VXIbus Control Register
VDTR	R	VXIbus Device Type Register
VERSION[3:0]	B	Version Number
VIAR1	R	VMEbus Interrupt Acknowledge Register 1
VIAR2	R	VMEbus Interrupt Acknowledge Register 2
VIAR3	R	VMEbus Interrupt Acknowledge Register 3
VIAR4	R	VMEbus Interrupt Acknowledge Register 4
VIAR5	R	VMEbus Interrupt Acknowledge Register 5
VIAR6	R	VMEbus Interrupt Acknowledge Register 6
VIAR7	R	VMEbus Interrupt Acknowledge Register 7

Mnemonic	Type	Definition
VICR	R	VXIbus Interrupt Configuration Register
VICTR	R	VMEbus Interrupt Control Register
VIDR	R	VXIbus ID Register
VISTR	R	VMEbus Interrupt Status Register
VLAR	R	VME-MXI-2 Logical Address Register
VLR	R	VMEbus Lock Register
VMCR	R	VME-MXI-2 Control Register
VMCR2	R	VME-MXI-2 Control Register 2
VMSR	R	VME-MXI-2 Status Register
VMSR2	R	VME-MXI-2 Status Register 2
VOR	R	VXIbus Offset Register
VSCR	R	VXIbus Subclass Register
VSIDR	R	VMEbus Status ID Register
VSR	R	VXIbus Status Register
VUCR	R	VXIbus Utility Configuration Register
VWR0	R	Extender Logical Address Window Register
VWR1	R	Extender A16 Window Register
VWR2	R	Extender A24 Window Register
VWR3	R	Extender A32 Window Register

X

XFERR	B	Transfer Error

Appendix

H

Customer Communication

For your convenience, this appendix contains forms to help you gather the information necessary to help us solve technical problems you might have as well as a form you can use to comment on the product documentation. Filling out a copy of the *Technical Support Form* before contacting National Instruments helps us help you better and faster.

National Instruments provides comprehensive technical assistance around the world. In the U.S. and Canada, applications engineers are available Monday through Friday from 8:00 a.m. to 6:00 p.m. (central time). In other countries, contact the nearest branch office. You may fax questions to us at any time.

Electronic Services

Bulletin Board Support

National Instruments has BBS and FTP sites dedicated for 24-hour support with a collection of files and documents to answer most common customer questions. From these sites, you can also download the latest instrument drivers, updates, and example programs. For recorded instructions on how to use the bulletin board and FTP services and for BBS automated information, call (512) 795-6990. You can access these services at:

United States: (512) 794-5422 or (800) 327-3077
Up to 14,400 baud, 8 data bits, 1 stop bit, no parity

United Kingdom: 01635 551422
Up to 9,600 baud, 8 data bits, 1 stop bit, no parity

France: 1 48 65 15 59
Up to 9,600 baud, 8 data bits, 1 stop bit, no parity

FTP Support

To access our FTP site, log on to our Internet host, `ftp.natinst.com`, as anonymous and use your Internet address, such as `joesmith@anywhere.com`, as your password. The support files and documents are located in the `/support` directories.

FaxBack Support

FaxBack is a 24-hour information retrieval system containing a library of documents on a wide range of technical information. You can access FaxBack from a touch-tone telephone at the following numbers:

(512) 418-1111 or (800) 329-7177

E-Mail Support (currently U.S. only)

You can submit technical support questions to the appropriate applications engineering team through e-mail at the Internet addresses listed below. Remember to include your name, address, and phone number so we can contact you with solutions and suggestions.

GPIB:	`gpib.support@natinst.com`
DAQ:	`daq.support@natinst.com`
VXI:	`vxi.support@natinst.com`
LabVIEW:	`lv.support@natinst.com`
LabWindows:	`lw.support@natinst.com`
HiQ:	`hiq.support@natinst.com`
VISA:	`visa.support@natinst.com`

Fax and Telephone Support

National Instruments has branch offices all over the world. Use the list below to find the technical support number for your country. If there is no National Instruments office in your country, contact the source from which you purchased your software to obtain support.

	 Telephone	 Fax
Australia	03 9 879 9422	03 9 879 9179
Austria	0662 45 79 90 0	0662 45 79 90 19
Belgium	02 757 00 20	02 757 03 11
Canada (Ontario)	519 622 9310	519 622 9311
Canada (Quebec)	514 694 8521	514 694 4399
Denmark	45 76 26 00	45 76 71 11
Finland	90 527 2321	90 502 2930
France	1 48 14 24 24	1 48 14 24 14
Germany	089 741 31 30	089 714 60 35
Hong Kong	2645 3186	2686 8505
Italy	02 48301892	02 48301915
Japan	03 5472 2970	03 5472 2977
Korea	02 596 7456	02 596 7455
Mexico	95 800 010 0793	5 520 3282
Netherlands	0348 433466	0348 430673
Norway	32 84 84 00	32 84 86 00
Singapore	2265886	2265887
Spain	91 640 0085	91 640 0533
Sweden	08 730 49 70	08 730 43 70
Switzerland	056 200 51 51	056 200 51 55
Taiwan	02 377 1200	02 737 4644
U.K.	01635 523545	01635 523154

Technical Support Form

Photocopy this form and update it each time you make changes to your software or hardware, and use the completed copy of this form as a reference for your current configuration. Completing this form accurately before contacting National Instruments for technical support helps our applications engineers answer your questions more efficiently.

If you are using any National Instruments hardware or software products related to this problem, include the configuration forms from their user manuals. Include additional pages if necessary.

Name _____

Company _____

Address _____

_____

Fax (___)_____ Phone (___)_____

Computer brand _____ Model _____ Processor _____

Operating system (include version number) _____

Clock Speed _____MHz RAM _____MB Display adapter _____

Mouse _____yes _____no Other adapters installed _____

Hard disk capacity _____MB Brand _____

Instruments used _____

National Instruments hardware product model _____ Revision _____

Configuration _____

National Instruments software product _____ Version _____

Configuration _____

The problem is _____

_____

_____

_____

List any error messages _____

_____

_____

_____

The following steps will reproduce the problem _____

_____

_____

Hardware and Software Configuration Form

Record the settings and revisions of your hardware and software on the line to the right of each item. Complete a new copy of this form each time you revise your software or hardware configuration, and use this form as a reference for your current configuration. Completing this form accurately before contacting National Instruments for technical support helps our applications engineers answer your questions more efficiently.

National Instruments Products

VME-MXI-2 Module Part Number _____

Serial Number _____

Revision Number _____

Slot Location _____

Hardware Settings (Chapter 3)

VME-MXI-2 Logical Address Switch Setting (U20) _____

VME-MXI-2 Intermodule Signaling (W2) _____

MXIbus Termination (U21 switches 3 and 4) _____

EEPROM Operation (U21 switches 1 and 2) _____

Onboard DRAM SIMM Size (S2) _____

DRAM SIMMs Installed _____

VXI*plug&play* Soft Front Panel Settings (Chapter 6)

Logical Address Select _____

Logical Address _____

Address Space _____

Requested Memory _____

A16 Write Post _____

A24/A32 Write Post _____

Interlocked Operation _____

VMEbus System Controller _____

VMEbus Bus Timeout _____

VMEbus Arbiter Type _____

VMEbus Arbiter Timeout _____

VMEbus Fair Requester _____

VMEbus Request Level _____

VMEbus Transfer Limit _____

VMEbus Auto Retry _____

MXIbus System Controller _____

MXIbus Bus Timeout _____

MXIbus Transfer Limit _____

MXIbus Auto Retry _____

MXIbus Parity Checking _____

MXIbus Fair Requester _____

Programmable Configurations (Appendix B)

Requested Memory Space _____

VMEbus Timer Limit _____

VMEbus Arbiter Type _____

VMEbus Arbiter Timeout _____

Requested Memory Space _____

VMEbus Request Level _____

VMEbus Fair Requester _____

MXIbus Timer Limit _____

MXIbus Fair Requester _____

MXIbus Parity Checking _____

Interlocked Arbitration Mode _____

Other Products

Computer Make and Model _____

Mainframe Make and Model _____

Microprocessor _____

Clock Frequency _____

Type of Video Board Installed _____

Operating System _____

Operating System Version _____

Operating System Mode _____

Other MXIbus Devices in System _____

_____

Other VMEbus Devices in System _____

_____

Base I/O Address of Other Boards _____

DMA Channels of Other Boards _____

Interrupt Level of Other Boards _____

VXIbus Resource Manager used (if any)
(Make, Model, Version, Software Version) _____

Documentation Comment Form

National Instruments encourages you to comment on the documentation supplied with our products. This information helps us provide quality products to meet your needs.

Title: VME-MXI-2 User Manual

Edition Date: January 1996

Part Number: 321071A-01

Please comment on the completeness, clarity, and organization of the manual.

If you find errors in the manual, please record the page numbers and describe the errors.

Thank you for your help.

Name _____

Title _____

Company _____

Address _____

Phone (_____) _____

Mail to: Technical Publications
National Instruments Corporation
6504 Bridge Point Parkway
Austin, TX 78730-5039

Fax to: Technical Publications
National Instruments Corporation
(512) 794-5678

Glossary

Prefix	Meaning	Value
n-	nano-	10^{-9}
µ-	micro-	10^{-6}
m-	milli-	10^{-3}
K-	kilo-	10^{3}
M	mega-	10^{6}
G-	giga-	10^{9}

Symbols

°	degrees
Ω	ohms
%	percent
±	plus or minus

A

A	amperes
A16 space	The VME 64 KB short address space. In VXI, the upper 16 KB of A16 space is allocated for use by VXI devices configuration registers. This 16 KB region is referred to as VXI configuration space.

A24/A32 Decoder	The logic circuit on the VME-MXI-2 that is responsible for detecting data transfers to the module's registers or DRAM in A24 or A32 address space.
A24 space	The VME 16 MB *standard* address space.
A32 space	The VME 4 GB *extended* address space.
ACFAIL	A VMEbus backplane signal that is asserted when a power failure has occurred (either AC line source or power supply malfunction), or if it is necessary to disable the power supply (such as for a high temperature condition).
address	Character code that identifies a specific location (or series of locations) in memory.
address modifier	One of six signals in the VMEbus specification used by VMEbus masters to indicate the address space in which a data transfer is to take place.
address space	A set of 2^n memory locations differentiated from other such sets in VMEbus systems by six addressing lines known as address modifiers. *n* is the number of address lines required to uniquely specify a byte location in a given space. Valid numbers for *n* are 16, 24, and 32. In VME, because there are six address modifiers, there are 64 possible address spaces.
address window	A portion of address space that can be accessed from the application program.
ANSI	American National Standards Institute
arbiter	Circuitry providing the bus arbitration mechanism for a system.
arbitration	A process in which a potential bus master gains control over a particular bus.
asynchronous	Not synchronized; not controlled by time signals.

B

B	bytes
backplane	An assembly, typically a printed circuit board, with 96-pin connectors and signal paths that bus the connector pins. A C-size VMEbus system will have two sets of bused connectors called J1 and J2.
backoff condition	A method used to resolve a deadlock situation by acknowledging one of the bus masters with either a RETRY or BERR, allowing the data transfer from the other master to complete.
base address	A specified address that is combined with a *relative* address to determine the *absolute* address of a data location. All VME address windows have an associated base address for their assigned VME address spaces.
BERR*	Bus Error signal. This signal is asserted by either a slave device or the BTO unit when an incorrect transfer is made on the Data Transfer Bus (DTB).
binary	A numbering system with a base of 2.
bit	Binary digit. The smallest possible unit of data: a two-state, yes/no, 0/1 alternative. The building block of binary coding and numbering systems. Eight bits make up a *byte*.
block data rate	Transfer rate when using MXIbus block-mode transfers.
block-mode transfer	An uninterrupted transfer of data elements in which the master sources only the first address at the beginning of the cycle. The slave is then responsible for incrementing the address on subsequent transfers so that the next element is transferred to or from the proper storage location. In VME, the data transfer may have no more than 256 elements; MXI does not have this restriction.

BTO unit	Bus Timeout Unit; a functional module that times the duration of each data transfer and terminates the cycle if the duration is excessive. Without the termination capability of this module, a bus master attempt to access a nonexistent slave could result in an indefinitely long wait for a slave response.
bus master	A device that is capable of requesting the Data Transfer Bus (DTB) for the purpose of accessing a slave device.
byte	A grouping of adjacent binary digits operated on as a single unit. Most commonly consists of eight bits.

C

C	Celsius
clearing	Replacing the information in a register, storage location, or storage unit with zeros or blanks.
CLK10	A 10 MHz, ± 100 ppm, individually buffered (to each module slot), differential ECL system clock that is sourced from Slot 0 of a VXIbus mainframe and distributed to Slots 1 through 12 on P2. It is distributed to each slot as a single-source, single-destination signal with a matched delay of under 8 ns.
CMOS	Complementary Metal Oxide Semiconductor; a process used in making chips.
Commander	A message-based device which is also a bus master and can control one or more Servants.
configuration registers	A set of registers through which the system can identify a module device type, model, manufacturer, address space, and memory requirements. In order to support automatic system and memory configuration, the VXIbus specification requires that all VXIbus devices have a set of such registers.
configuration space	The upper 16 KB of A16 space in which the configuration registers for VXIbus devices exist.
controller	An intelligent device (usually involving a CPU) that is capable of controlling other devices.

D

daisy-chain	A method of propagating signals along a bus, in which the devices are prioritized on the basis of their position on the bus.
Data Transfer Bus	DTB; one of four buses on the VMEbus backplane. The DTB is used by a bus master to transfer binary data between itself and a slave device.
deadlock	Unresolved situation in which two devices are vying for the use of a resource.
DIP	Dual Inline Package
DMA	Direct Memory Access; a method by which data is transferred between devices and internal memory without intervention of the central processing unit.
DRAM	Dynamic RAM (Random Access Memory)
DTACK*	Data Acknowledge signal
DTB	See *Data Transfer Bus*.
dynamic configuration	A method of automatically assigning logical addresses to VXIbus devices at system startup or other configuration times.
dynamically configured device	A device that has its logical address assigned by the Resource Manager. A VXI device initially responds at Logical Address 255 when its MODID line is asserted. A MXIbus device responds at Logical Address 255 during a priority select cycle. The Resource Manager subsequently assigns it a new logical address, which the device responds to until powered down.

E

ECL	Emitter-Coupled Logic
EEPROM	Electronically Erasable Programmable Read Only Memory
embedded controller	An intelligent CPU (controller) interface plugged directly into the VME backplane, giving it direct access to the VMEbus. It must have all of its required VME interface capabilities built in.

EMC	Electromechanical Compliance
EMI	Electromagnetic Interference
external controller	In this configuration, a plug-in interface board in a computer is connected to the VME mainframe via one or more VMEbus extended controllers. The computer then exerts overall control over VMEbus system operations.

F

fair requester	A MXIbus master that will not arbitrate for the MXIbus after releasing it until it detects the bus request signal inactive. This ensures that all requesting devices will be granted use of the bus.

H

hard reset	Occurs when the mainframe is powered on and when the VMEbus SYSRESET signal is active. A hard reset restores all the registers on the VME-MXI-2 to their initial values.
hex	Hexadecimal; the numbering system with base 16, using the digits 0 to 9 and letters A to F.
Hz	hertz; cycles per second.

I

IACK	Interrupt Acknowledge; a special data transfer generated by the interrupt handler in response to an interrupt.
IC	Integrated Circuit
IEEE	Institute of Electrical and Electronics Engineers
IEEE-1014	The VME specification.
in.	inches
I/O	input/output; the techniques, media, and devices used to achieve communication between machines and users.

interlocked arbitration mode	Contrasted with *normal operating mode*; an optional mode of operation in which the system performs as one large VMEbus mainframe with only one master of the entire system (VMEbus and MXIbus) at any given moment. In this mode there is no chance for a deadlock situation.
interrupt	A means for a device to request service from another device.
interrupt handler	A VMEbus functional module that detects interrupt requests generated by interrupters and responds to those requests by requesting status and identify information.
interrupter	A device capable of asserting interrupts and responding to an interrupt acknowledge cycle.
interrupt level	The relative priority at which a device can interrupt.
INTX	Interrupt and Timing Extension; a daughter card option that plugs into the two daughter card connectors on the first-generation VME-MXI. It extends the seven VMEbus interrupt lines and the VMEbus utility signals SYSRESET*, SYSFAIL*, and ACFAIL*. This functionality is built into the VME-MXI-2, so this daughter card is not required.
inward cycle	A data transfer cycle that maps from the MXIbus to the VMEbus.
IRQ*	Interrupt signal

K

KB	Kilobytes of memory

L

LED	Light Emitting Diode
logical address	An 8-bit number that uniquely identifies each VXIbus device in a system. It defines the A16 register base address of a device, and indicates Commander and Servant relationships.
LSB	Least Significant Bit (bit 0)

M

MB	Megabytes of memory
m	meters
mainframe extender	A device such as the VME-MXI-2 that interfaces a VMEbus mainframe to an interconnect bus. It routes bus transactions from the VMEbus to the interconnect bus or vice versa. A mainframe extender has a set of registers that defines the routing mechanisms for data transfers, interrupts, and utility bus signals, and has optional VMEbus first slot capability.
mapping	Establishing a range of address space for a one-to-one correspondence between each address in the window and an access in VMEbus memory.
master	A functional part of a MXI/VXI/VMEbus device that initiates data transfers on the backplane. A transfer can be either a read or a write.
master-mode operation	A device is in master mode if it is performing a bus cycle which it initiated.
memory device	A VMEbus device that not only has configuration registers, but also has memory that is accessible through addresses on the VME data transfer bus.
message-based device	An intelligent device that implements the defined VXIbus registers and communication protocols. These devices are able to use Word Serial Protocol to communicate with one another through communication registers.
MBLT	Multiplexed Block Transfer; 8-byte block transfers in which both the Address bus and the Data bus are used to transfer data.
MITE	A National Instruments custom ASIC, a sophisticated dual-channel DMA controller that incorporates the Synchronous MXI and VME64 protocols to achieve high-performance block transfer rates.
MODID	Module Identification lines
MSB	Most Significant Bit (such as bit 15 in a 16-bit register)

MTBF	Mean Time Between Failure
multiframe	A system consisting of more than one mainframe connected together to act as one; it can have multiple first slot devices.
MXI-2	The second generation of the National Instruments MXIbus product line. MXI-2 expands the number of signals on a standard MXIbus cable by including VXI triggers, all VME interrupts, VXI CLK10, SYSFAIL*, SYSRESET*, and ACFAIL*.
MXIbus	Multisystem eXtension Interface Bus; a high-performance communication link that interconnects devices using round, flexible cables.
MXIbus System Controller	A functional module that has arbiter, daisy-chain driver, and MXIbus cycle timeout responsibility. Always the first device in the MXIbus daisy-chain.

N

nonprivileged access	One of the defined types of VMEbus data transfers; indicated by certain address modifier codes. Each of the defined VMEbus address spaces has a defined nonprivileged access mode.
Non-Slot 0 device	A device configured for installation in any slot in a VXIbus mainframe other than Slot 0. Installing such a device into Slot 0 can damage the device, the VXIbus backplane, or both.
normal operating mode	Contrasted with *interlocked arbitration mode*; in this mode there can be masters operating simultaneously in the VMEbus/MXIbus system. Vulnerable to deadlock situations.

O

onboard RAM	The optional RAM installed into the SIMM slots of the VME-MXI-2 module.
outward cycle	A data transfer cycle that maps from the VMEbus to the MXIbus.

P

P1	The minimum connector required for a VMEbus system. It includes 24 address lines, 16 data lines, and all control, arbitration, and interrupt signals.
P2	A second VMEbus connector providing 32 bits of address and data. RETRY support is also located on P2. In VXI, the P2 connector adds trigger, MODID, and CLK10 signals.
P3	A third connector defined by the VXIbus specification that adds a 100 MHz CLK and additional triggering capabilities. The VME-MXI-2 does not have support for P3.
parity	Ensures that there is always either an even number or an odd number of asserted bits in a byte, character, or word, according to the logic of the system. If a bit should be lost in data transmission, its loss can be detected by checking the parity.
PASSED state	The state a VMEbus device enters after its self-tests are complete and the device is ready for normal operation.
PRI	Priority
privileged access	See *supervisory access*.
propagation	The transmission of signal through a computer system.

R

read	To get information from any input device or file storage media.
register-based device	A Servant-only device that supports VXIbus configuration registers. Register-based devices are typically controlled by message-based devices via device-dependent register reads and writes.
retry	An acknowledge by a destination that signifies that the cycle did not complete and should be repeated.

Resource Manager	A message-based Commander located at Logical Address 0, which provides configuration management services such as address map configuration, Commander and Servant mappings, and self-test and diagnostic management.
response	A signal or interrupt generated by a device to notify another device of an asynchronous event. Responses contain the information in the Response register of a sender.
RM	See *Resource Manager*.
RMW	Read-Modify-Write cycle; a bus cycle in which data from a single location is read, modified, and then written back.
ROAK	Release On Acknowledge; a type of VME interrupter which always deasserts its interrupt line in response to an IACK cycle on the VMEbus.
ROR	Release On Request; a type of VMEbus arbitration where the current VMEbus master relinquishes control of the bus only when another bus master requests the VMEbus.

S

s	seconds
Semi-Synchronous Protocol	A one-line, open-collector, multiple-device handshake trigger protocol.
Servant	A device controlled by a Commander; there are message-based and register-based Servants.
setting	To place a binary cell into the 1 state (non-zero).
Shared Memory Protocol	A communication protocol that uses a block of memory that is accessible to both a client and a server. The memory block operates as a message buffer for communications.
signal	Any communication between message-based devices consisting of a write to a Signal register. Sending a signal requires that the sending device have VMEbus master capability.
SIMM	Single In-line Memory Module

slave	A functional part of a MXI/VXI/VMEbus device that detects data transfer cycles initiated by a VMEbus master and responds to the transfers when the address specifies one of the device's registers.
slave-mode operation	A device is in slave mode it if is responding to a bus cycle.
Slot 0 device	A device configured for installation in Slot 0 of a VXIbus mainframe. This device is unique in the VXIbus system in that it performs the VMEbus System Controller functions, including clock sourcing and arbitration for data transfers across the backplane. Installing such a device into any other slot can damage the device, the VXIbus backplane, or both.
SMB	Sub-miniature BNC; a miniature connector for coaxial cable connections.
soft reset	Occurs when the RESET bit in the VXIbus Control Register of the VME-MXI-2 is set. A soft reset clears signals that are asserted by bits in the configuration registers but does not clear configuration information stored in the configuration registers.
Start/Stop Protocol	A one-line, multiple-device protocol, which can be sourced only by the VXI Slot 0 device and sensed by any other device on the VXI backplane.
statically configured device	A device whose logical address cannot be set through software; that is, it is not dynamically configurable.
Status/ID	A value returned during an IACK cycle. In VME, usually an 8-bit value which is either a status/data value or a vector/ID value used by the processor to determine the source. In VXI, a 16-bit value used as a data; the lower 8 bits form the VXI logical address of the interrupting device and the upper 8 bits specify the reason for interrupting.
supervisory access	One of the defined types of VMEbus data transfers; indicated by certain address modifier codes.

synchronous communications — A communications system that follows the command/response cycle model. In this model, a device issues a command to another device; the second device executes the command and then returns a response. Synchronous commands are executed in the order they are received.

Synchronous MXI Block Protocol — A block data transfer protocol defined by the MXI-2 specification for high-performance data transfers.

Synchronous Protocol — The most basic trigger protocol, simply a pulse of a minimum duration on any one of the trigger lines.

SYSFAIL — A VMEbus signal that is used by a device to indicate an internal failure. A failed device asserts this line. In VXI, a device that fails also clears its PASSed bit in its Status register.

SYSRESET — A VMEbus signal that is used by a device to indicate a system reset or power-up condition.

System Controller — See *MXIbus System Controller*, *VMEbus System Controller*.

system hierarchy — The tree structure of the Commander/Servant relationships of all devices in the system at a given time. In the VXIbus structure, each Servant has a Commander. A Commander can in turn be a Servant to another Commander.

system RAM — RAM installed on your personal computer and used by the operating system, as contrasted with onboard RAM, which is installed on the VME-MXI-2.

T

terminators — Also called *terminating networks*; devices located at the ends of a MXIbus daisy-chain that are used to minimize reflections and bias signals to their unasserted states.

TERMPWR — Termination Power; 3.4 VDC for the MXIbus.

trigger — Either TTL or ECL lines used for intermodule timing.

TTL — Transistor-Transistor Logic

V

V	volts
VDC	volts direct current
VME	Versa Module Eurocard or IEEE 1014; the IEEE Standard for a Versatile Backplane Bus.
VME64	ANSI/VITA 1-1994; defines additional VMEbus protocols such as MBLT and RETRY.
VMEbus System Controller	A device configured for installation in Slot 0 of a VXIbus mainframe or the first slot of a VMEbus chassis. This device is unique in the VMEbus system in that it performs the VMEbus System Controller functions, including clock sourcing and arbitration for data transfers across the backplane. Installing such a device into any other slot can damage the device, the VXIbus/VMEbus backplane, or both.
VXIbus	VMEbus Extensions for Instrumentation
VXIbus System Controller	A functional module with circuitry that generates the 16 MHz system clock, provides the VMEbus arbiter and the VMEbus Bus Timer Unit, and drives the VXIbus CLK10 signal.

W

write	Copying data to a storage device.
Word Serial Protocol	The simplest required communication protocol supported by message-based devices in a VXIbus system. It utilizes the A16 communication registers to transfer data using a simple polling handshake method.

Index

Numbers

A

B

C

D

E

F

H

I

J

L

M

O

P

R

S

T

U

V

W

X