VME-MXI
User Manual

November 1993 Edition

Part Number 320330-01

National Instruments Corporate Headquarters
6504 Bridge Point Parkway
Austin, TX 78730-5039
(512) 794-0100
Technical support fax: (800) 328-2203
(512) 794-5678

Branch Offices:
Australia (03) 879 9422, Austria (0662) 435986, 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, Italy 02/48301892, Japan (03) 3788-1921, Netherlands 03480-33466, Norway 32-848400,
Spain (91) 640 0085, Sweden 08-730 49 70, Switzerland 056/20 51 51, U.K. 0635 523545

Limited Warranty

The National Instruments MXIbus boards are 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.

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

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.

Contents

About This Manual xi
Organization of This Manual xi
How to Use This Documentation Set xii
Related Documentation xii
Customer Communication xii

Chapter 1
General Information 1-1
Overview 1-4
Front Panel Features 1-5
What Your Kit Should Contain 1-6
Optional Equipment 1-6
Unpacking 1-7

Chapter 2
General Description 2-1
Electrical Characteristics 2-1
VMEbus Modules 2-2
VME-MXI Functional Description 2-5

Chapter 3
Configuration and Installation 3-1
Configuration 3-1
VMEbus System Controller 3-4
MXIbus Logical Address 3-5
VMEbus Request Level 3-6
VMEbus Timeout Value 3-7
VMEbus Timeout Chain Position 3-9
Interlocked Arbitration Mode 3-12
MXIbus System Controller 3-13
MXIbus System Controller Timeout 3-13
MXIbus Fairness Option 3-14
Reset Signal Select 3-15
Installing the VME-MXI Hardware 3-16
MXIbus Termination 3-16
INTX Termination 3-19
Removing the INTX Daughter Card from the VME-MXI 3-20
Installing the INTX Daughter Card onto the VME-MXI 3-20
VME-MXI Installation Instructions 3-21
Connecting the INTX Cable 3-22
Connecting the MXIbus Cable 3-23
System Power Cycling Requirements 3-25

Chapter 4
Register Descriptions 4-1
Register Maps 4-1
Register Sizes 4-1
Register Description Format 4-1
Hard and Soft Reset 4-1
MXIbus Configuration Registers 4-4
ID Register 4-4
Device Type Register 4-6
Status/Control Register 4-7
VMEbus Extender Registers 4-9
Logical Address Window Register 4-9
A16 Window Map Register 4-13
A24 Window Map Register 4-17
A32 Window Map Register 4-21
INTX Interrupt Configuration Register 4-25
INTX Utility Configuration Register 4-26
Subclass Register 4-28
MXIbus Defined Registers 4-29
MXIbus Status/Control Register 4-29
MXIbus Lock Register 4-33
MXIbus IRQ Configuration Register 4-34
Read LA Register 4-36
Interrupt Status/Control Register 4-37
Status/ID Register 4-40
IRQ Acknowledge Registers 4-41

Chapter 5
Programming Considerations 5-1
System Configuration 5-1
Planning a MXIbus System Logical Address Map 5-1
Base/Size Configuration Format 5-3
High/Low Configuration Format 5-5
Steps to Follow When Planning a System Logical Address Map 5-5
Worksheets for Planning Your VMEbus/MXIbus Logical Address Map 5-13
Alternative Worksheets for Planning Your MXIbus Logical Address Map 5-18
Planning a VMEbus/MXIbus System A16 Address Map 5-21
Worksheets for Planning Your VMEbus/MXIbus A16 Address Map 5-29
MXIbus RM Operation 5-35
Configuring the Logical Address Window 5-35
Configuring the Logical Address Window Example 5-36
Configuring the A24 and A32 Addressing Windows 5-38

Chapter 6
Theory of Operation 6-1
VMEbus Address and Address Modifier Transceivers 6-1
VMEbus System Controller Functions 6-1
VMEbus Data Transceivers 6-1

VMEbus Control Signal Transceivers 6-1
VMEbus Requester and Arbiter Circuitry 6-2
SYSFAIL, ACFAIL, and SYSRESET 6-2
Interrupt Circuitry 6-2
Parity Check and Generation 6-4
A32, A24, A16, and LA Windows 6-5
VME-MXI Configuration Registers 6-5
MXIbus Master Mode State Machine 6-5
MXIbus Slave Mode State Machine 6-9
MXIbus Address/Data and Address Modifier Transceivers 6-10
MXIbus System Controller Functions 6-10
MXIbus Control Signals Transceivers 6-11
MXIbus Requester and Arbiter Circuitry 6-11

Appendix A
Specifications A-1

Appendix B
Mnemonics Key B-1

Appendix C
Connector Descriptions C-1
MXIbus Connector C-1
INTX Connector C-3

Appendix D
Customer Communication D-1

Glossary Glossary-1

Index Index-1

Figures

Figure 1-1. VME-MXI Interface Module....1-2
Figure 1-2. VME-MXI Interface Module with INTX Option....1-3

Figure 2-1. VME-MXI Block Diagram....2-5
Figure 2-2. VME-MXI INTX Daughter Card Option Block Diagram....2-8

Figure 3-1. VME-MXI Parts Locater Diagram....3-2
Figure 3-2. VME-MXI with INTX Parts Locator Diagram....3-3
Figure 3-3. VMEbus System Controller Selection....3-4
Figure 3-4. Logical Address Selection....3-6
Figure 3-5. VMEbus Requester Jumper Settings....3-7
Figure 3-6. VMEbus Timeout Value Selection....3-8
Figure 3-7. VMEbus Timeout; One VME-MXI in Chassis....3-9
Figure 3-8. VMEbus Timeout; Multiple VME-MXIs in VMEbus Chassis....3-10
Figure 3-9. VMEbus Timeout; Multiple VME-MXIs in VXIbus Mainframe....3-10
Figure 3-10. No VMEbus Timeout; Multiple VME-MXIs in VMEbus Chassis....3-11
Figure 3-11. No VMEbus Timeout; Multiple VME-MXIs in VXIbus Mainframe....3-11
Figure 3-12. Interlocked Arbitration Mode Selection....3-13
Figure 3-13. MXIbus System Controller Selection....3-13
Figure 3-14. MXIbus System Controller Timeout Value Selection....3-14
Figure 3-15. MXIbus Fair Requester Selection....3-15
Figure 3-16. Reset Signal Selection Settings....3-15
Figure 3-17. MXIbus Daisy-Chain....3-17
Figure 3-18. MXIbus Terminating Networks....3-18
Figure 3-19. VME-MXI INTX Parts Locator Diagram....3-19
Figure 3-20. INTX Terminator Example....3-21
Figure 3-21. MXIbus Single-Ended Cable Configuration....3-24
Figure 3-22. MXIbus Dual-Ended Cable Configuration....3-25

Figure 4-1. VME-MXI Register Map....4-3

Figure 5-1. VMEbus/MXIbus System with MXIbus RM on a PC....5-2
Figure 5-2. VMEbus/MXIbus System with MXIbus RM in a VMEbus Chassis....5-2
Figure 5-3. Base and Size Combinations....5-4
Figure 5-4. Address Range Allocation for Different Size Values....5-4
Figure 5-5. Example VMEbus/MXIbus System....5-8
Figure 5-6. Logical Address Map Diagram for Example VMEbus/MXIbus System....5-9
Figure 5-7. Worksheet 1 for Example VMEbus/MXIbus System....5-10
Figure 5-8. Worksheet 2 for Example VMEbus/MXIbus System....5-11
Figure 5-9. Worksheet 3 for Example VMEbus/MXIbus System....5-12
Figure 5-10. Worksheet 4 for Example VMEbus/MXIbus System....5-12
Figure 5-11. Logical Address Map Example with Alternative Worksheet....5-20
Figure 5-12. A16 Space Allocations for all Size Values....5-22
Figure 5-13. Example VMEbus/MXIbus System....5-24
Figure 5-14. Example A16 Space Address Map....5-25
Figure 5-15. Worksheet 1 for A16 Address Map Example....5-26
Figure 5-16. Worksheet 2 for A16 Map Example....5-27
Figure 5-17. Worksheet 3 for A16 Map Example....5-28

Figure 6-1. Master to Slave VMEbus/MXIbus Transfers 6-6
Figure 6-2. Deadlock Situation 6-8

Figure C-1. MXIbus Connector C-1
Figure C-2. INTX Connector C-3

Tables

Table 2-1. VME-MXI VMEbus Signals 2-1
Table 2-2. MXIbus Transceiver Requirements 2-2
Table 2-3. VME-MXI VMEbus Compliance Levels 2-3

Table 3-1. MXIbus System Power Cycling Requirements 3-26

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

Table 5-1. Base and Size Combinations 5-3
Table 5-2. Example VMEbus/MXIbus System Required Logical Addresses 5-8
Table 5-3. Amount of A16 Space Allocated for all Size Values 5-21
Table 5-4. Example VMEbus/MXIbus System Required A16 Space 5-25
Table 5-5. Logical Address Assignments for Example VMEbus/MXIbus System 5-36

Table 6-1. VME-MXI Addresses for VMEbus Interrupt Levels 6-4
Table 6-2. VMEbus to MXIbus Address Modifier Line Map 6-6
Table 6-3. Transfer Responses for VMEbus Address Modifiers 6-7
Table 6-4. VMEbus/MXIbus Transfer Size Comparison 6-7

Table C-1. MXIbus Connector Signal Assignments C-1
Table C-2. MXIbus Signal Groupings C-2
Table C-3. INTX Connector Signal Assignments C-3
Table C-4. INTX Signal Groupings C-4

About This Manual

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

Organization of This Manual

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

- Chapter 1, *General Information*, describes the VME-MXI features, lists the contents of your VME-MXI kit, and explains how to unpack the VME-MXI kit.
- Chapter 2, *General Description*, contains the physical and electrical specifications for the VME-MXI and describes the characteristics of key components.
- Chapter 3, *Configuration and Installation*, describes the configuration and installation of the VME-MXI hardware.
- Chapter 4, *Register Descriptions*, contains detailed descriptions of the VME-MXI registers, which are used to configure and control the module's operation.
- Chapter 5, *Programming Considerations*, explains important considerations for programming the VME-MXI and configuring a system using VME-MXIs.
- Chapter 6, *Theory of Operation*, contains a functional overview of the VME-MXI board and explains the operation of each functional block making up the VME-MXI.
- Appendix A, *Specifications*, lists the specifications of the VME-MXI.
- Appendix B, *Mnemonics Key*, contains an alphabetical listing of all mnemonics used in this manual.
- Appendix C, *Connector Descriptions*, describes the connector pin assignments for the MXIbus connector.
- Appendix D, *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.

How to Use This Manual

If you will be installing your VME-MXI into a system with a MXIbus Resource Manager, you need to read only Chapters 1 through 3 of this manual. If you have more than two VME-MXIs extending your system, you will find useful system configuration information in Chapter 5. If you are writing your own MXIbus Resource Manager routines, you can find programming information and descriptions of the VME-MXI hardware in Chapters 4 through 6.

Related Documentation

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

- *IEEE Standard for a Versatile Backplane Bus: VMEbus*, ANSI/IEEE Standard 1014-1987
- *Multisystem Extension Interface Bus Specification*, Version 1.2 (part number 340007-01)
- VXI-1, *VXIbus System Specification*, Rev. 1.4, VXIbus Consortium (available from National Instruments Corporation, part number 350083-01)
- VXI-6, *VXIbus Mainframe Extender Specification*, Rev. 1.0, VXIbus Consortium (available from National Instruments Corporation, part number 340258-01)

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 D, *Customer Communication*, at the end of this manual.

Chapter 1
General Information

This chapter describes the VME-MXI features, lists the contents of your VME-MXI kit, and explains how to unpack the VME-MXI kit.

The VME-MXI interface is a single-slot, 6U VMEbus board that interfaces the VMEbus to the MXIbus (Multisystem Extension Interface bus). A VMEbus mainframe equipped with a VME-MXI can be transparently connected to other MXIbus devices such as other VMEbus mainframes, MXIbus instruments, or MXIbus-equipped personal computers. The VME-MXI interface module uses address mapping to transparently translate bus cycles on the VMEbus to the MXIbus and vice versa.

The VME-MXI is available with an Interrupt and Timing Extension (INTX) daughter card option. If you ordered this option, the INTX card is already installed on your VME-MXI. The INTX daughter card plugs into the two daughter card connectors on the VME-MXI. Because this manual describes the VME-MXI with and without this option, you can find information on the INTX card throughout this manual.

Figure 1-1 shows the VME-MXI interface module without the INTX option. Figure 1-2 shows the VME-MXI interface module with the INTX option.

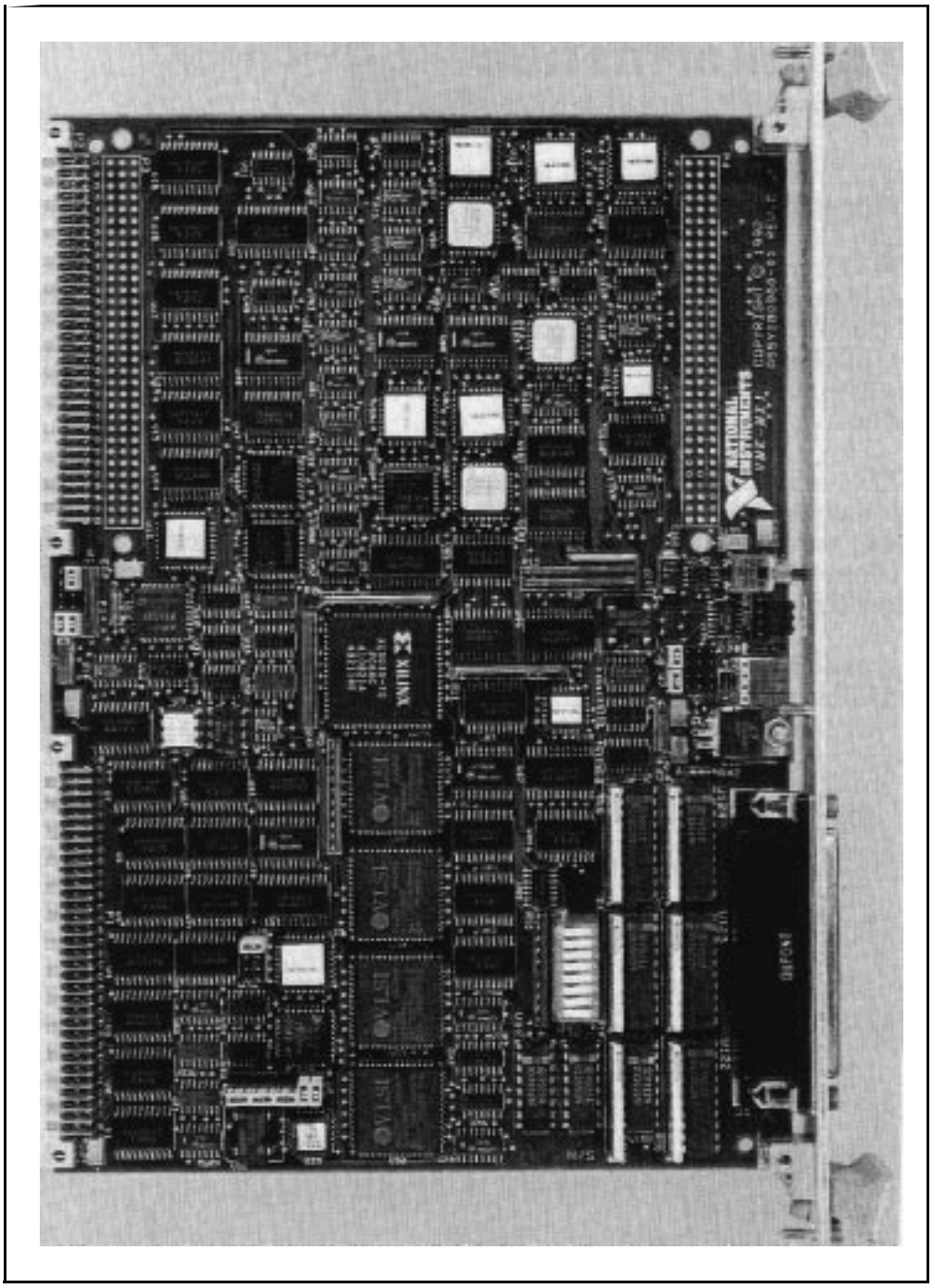


Figure 1-1. VME-MXI Interface Module

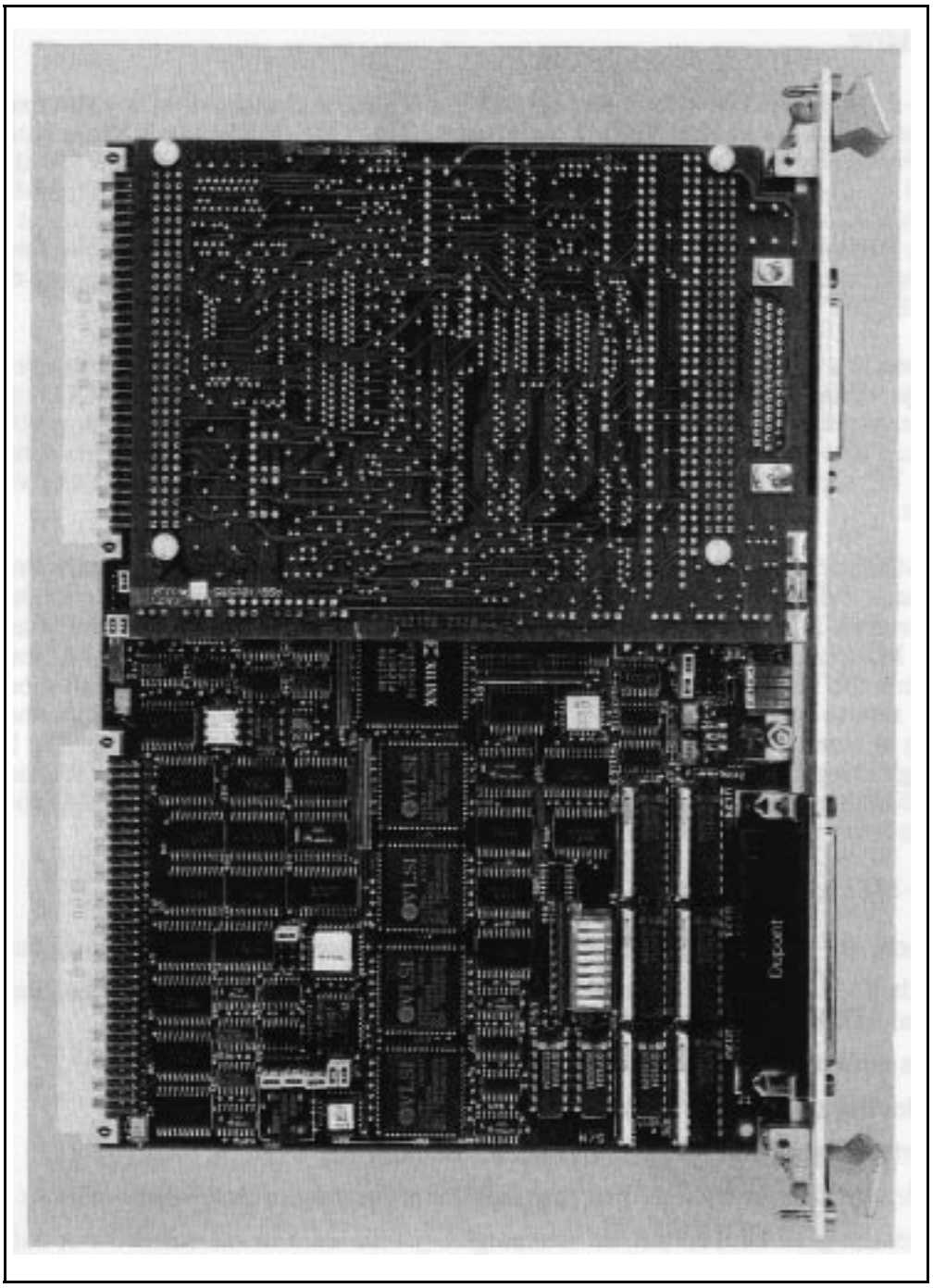


Figure 1-2. VME-MXI Interface Module with INTX Option

Overview

The VME-MXI can be installed in any 6U slot in a VMEbus chassis, or in any slot except Slot 0 of a B-Size, C-Size, or D-Size VXIbus mainframe. The VME-MXI has a VXIbus extended class Register-Based register set making the VME-MXI compatible with the VXIbus specification, Revision 1.4. The VME-MXI converts A32, A24, A16, D32, D16, and D08(EO) VMEbus cycles into MXIbus cycles and vice versa. The VME-MXI has four address windows that map bus cycles into and out of the VMEbus chassis. These four windows represent the three VMEbus address spaces (A32, A24, and A16) plus a dedicated window for mapping the VXIbus/MXIbus configuration space (the upper 16 kilobytes 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 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. This configuration makes the PC appear to be embedded on a VMEbus module 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 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 personal computers (PCs)
- Allows multiple VMEbus mainframes to appear as a single VMEbus system
- Provides integrated block mode for high-performance data transfers
- Supports dynamic configuration of MXIbus devices
- Provides optional interlocked bus operation for prevention of deadlock conditions
- Includes daughter card connector scheme giving additional functionality for optional daughter cards
- Is fully compatible with VMEbus, VXIbus, and MXIbus specifications
- Has no restrictions on Commander/Servant hierarchy or physical location of devices

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

- VMEbus System Controller functions:
  - 16-MHz system clock driver
  - VME bus timeout (BTO)

  - Data transfer bus arbiter (PRI ARBITER)
  - Interrupt acknowledge daisy-chain driver
  - Pushbutton system reset switch
- VMEbus master capabilities:
  - Access to A16, A24, and A32 address space
  - D08(EO), D16, and D32 accesses
  - Release-on-Request bus requester (jumper-selectable arbitration level)
- VMEbus slave accesses:
  - A16, A24, and A32 address space
  - D08(EO), D16, and D32 accesses

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

Front Panel Features

The VME-MXI has the following front panel features:

- Three front panel LEDs
  - *FAILED* LED indicates that the VMEbus SYSFAIL line is asserted.
  - *VME ACCESS* LED indicates when the VME-MXI is accessed from the VMEbus.
  - *MXI ACCESS* LED indicates when the VME-MXI is accessed from the MXIbus.
- MXIbus connector
- System reset pushbutton
- INTX connector (if your VME-MXI includes the INTX daughter card option)

What Your Kit Should Contain

Your VME-MXI kit should contain the following components:

Component	Part Number
Standard VME-MXI Interface Module or Enhanced VME-MXI Interface Module with INTX option	180960-01 180960-02
VME-MXI User Manual	320330-01

Optional Equipment

Equipment	Part Number
Type M1 MXIbus Cables Straight-point connector to straight-point connector: – 1 m – 2 m – 4 m – 8 m – 20 m	 180758-01 180758-02 180758-04 180758-08 180758-20
Type M2 MXIbus Cables Straight-point connector to right-angle daisy-chain connector: – 1 m – 2 m – 4 m – 8 m – 20 m	 180760-01 180760-02 180760-04 180760-08 180760-20
Type M3 MXIbus Cables Right-angle point connector to right-angle daisy-chain connector: – 1 m – 2 m – 4 m – 8 m – 20 m	 180761-01 180761-02 180761-04 180761-08 180761-20
MXIbus Terminating Pac (External)	180780-01

The following optional equipment is also available and may be necessary if your VME-MXI includes the INTX daughter card.

Equipment	Part Number
Type INTX1 Cables Straight-point connector to straight-point connector:	
– 1 m	180980-01
– 2 m	180980-02
– 4 m	180980-04
– 8 m	180980-08
– 20 m	180980-20
Type INTX2 Cables Right-angle point connector to right-angle daisy-chain connector:	
– 1 m	180982-01
– 2 m	180982-02
– 4 m	180982-04
– 8 m	180982-08
– 20 m	180982-20

Unpacking

Follow these steps when unpacking your VME-MXI:

1. Before attempting to configure or install the VME-MXI, inspect the shipping container and its contents for damage. If damage appears to have been caused in shipment, file a claim with the carrier. Retain the packing material for possible inspection and/or for reshipment.

2. Verify that the pieces contained in the package you received match the kit parts list. *Do not* remove the board from its bag at this point.

3. Your VME-MXI module is shipped packaged in an antistatic plastic bag to prevent electrostatic damage to the module. Several components on the module can be damaged by electrostatic discharge. To avoid such damage while handling the module, touch the plastic bag to a metal part of your grounded VMEbus mainframe chassis before removing the module from the bag.

4. As you remove the VME-MXI module from its bag, be sure to handle it only by its edges. Avoid touching any of the IC components or connectors. Inspect the module for loose components or any other sign of damage. Notify National Instruments if the module appears damaged in any way. *Do not* install a damaged module into your VMEbus mainframe.

Chapter 2
General Description

This chapter contains the physical and electrical specifications for the VME-MXI and describes the characteristics of key interface board components.

Electrical Characteristics

All integrated circuit drivers and receivers used on the VME-MXI meet the requirements of the VMEbus specification. Table 2-1 contains a list of the VMEbus signals used by the VME-MXI and the electrical loading presented by the circuitry on the interface board (in terms of device types and their part numbers).

Note: Throughout this manual, an asterisk (*) following a bus signal mnemonic indicates that the signal is active low.

Table 2-1. VME-MXI VMEbus Signals

Bus Signals	Driver Device Part Number	Receiver Device Part Number
D[31–0], A[31–1]	F245	F245
AM[5–0], LWORD	ALS646–1	ALS646–1
DS0*, DS1*	F125	ALS244
AS*, WRITE*	F125	ALS240
SYSCLK	F125	–
BG[3-0]IN*	–	F21, GAL22V10
BG[3-0]OUT*	LS32	–
BBSY*, SYSFAIL*, ACFAIL*	F38	ALS240
BR[3-0]*, DTACK*, BERR*	F38	ALS244
SYSRESET*	F38	ALS244
IACK*	F38	ALS244

(continues)

Table 2-1. VME-MXI VMEbus Signals (continued)

Bus Signals	Driver Device Part Number	Receiver Device Part Number
IACKIN*	–	LS540
IACKOUT*	GAL22V10	–
IRQ[7-1]*	AS760	LS540

All MXIbus transceivers meet the requirements of the MXIbus specification. Table 2-2 lists the components used.

Table 2-2. MXIbus Transceiver Requirements

Transceivers	Component Designation
Data Transceivers	DS3862
Control Transceivers	DS3662

VMEbus Modules

The VME-MXI has the following VMEbus modules:

- VMEbus Requester
- VMEbus Master
- VMEbus Slave
- Interrupter
- IACK Daisy-Chain Driver

When the VME-MXI is configured as a VMEbus System Controller, it also has the following VMEbus modules:

- VMEbus Timer
- Arbiter
- System Clock Driver

The VME-MXI does not support the following VMEbus modules:

- Serial Clock Driver
- Power Monitor

Table 2-3 indicates the VME-MXI VMEbus compliance levels.

Table 2-3. VME-MXI VMEbus Compliance Levels

Compliance Notation	Description
Bus Slave Compliance Levels	
D08(EO)	8-bit data path to configuration registers and MXIbus
D16 & D08(EO)	8-bit or 16-bit data path to configuration registers or MXIbus
D32	32-bit data path to MXIbus
A16	Responds to 16-bit short I/O addresses when specified on the address modifier lines
A24	Responds to 24-bit memory addresses when specified on the address modifier lines
A32	Responds to 32-bit memory addresses when specified on the address modifier lines
ADO	Accommodates address-only cycles
BLT	Responds to block mode transfers
RMW	Can accept Read-Modify-Write cycles
DTB Arbiter Compliance Level	
PRI	Monitors BR3* through BR0* and drives BG3OUT* through BG0OUT* and BCLR*
DTB Requester Compliance Level	
ROR	Release-on-Request

(continues)

Table 2-3. VME-MXI VMEbus Compliance Levels (continued)

Compliance Notation	Description
Bus Master Compliance Levels	
D08(EO)	8-bit data path from MXIbus
D16 & D08(EO)	8-bit or 16-bit data path from MXIbus
D32	32-bit data path from MXIbus
A16	Generates 16-bit short I/O addresses when specified by the MXIbus address modifier lines
A24	Generates 24-bit memory addresses when specified by the MXIbus address modifier lines
A32	Generates 32-bit memory addresses when specified by the MXIbus address modifier lines
BLT	Generates block mode transfers when specified by the MXIbus address modifier lines
RMW	Can generate Read-Modify-Write cycles
Interrupter Compliance Levels	
I(7-1)	Can generate an interrupt request on interrupt lines IRQ7 through IRQ1
D16 & D32	Responds to 16-bit and 32-bit interrupt acknowledge cycles by providing a 16-bit Status/ID byte on D00 through D15
ROAK	Releases its interrupt request line when its Status/ID is read during an interrupt acknowledge cycle
Interrupt Handler Compliance Levels	
IH(7-1)	Can generate interrupt acknowledge cycles in response to interrupt requests on IRQ7 through IRQ1
D16	Generates a 16-bit interrupt acknowledge cycle in response to a VMEbus interrupt request

VME-MXI Functional Description

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

Figure 2-1 is a functional block diagram of the VME-MXI. Refer to Chapter 6, *Theory of Operation* for more details about the major components of the VME-MXI.

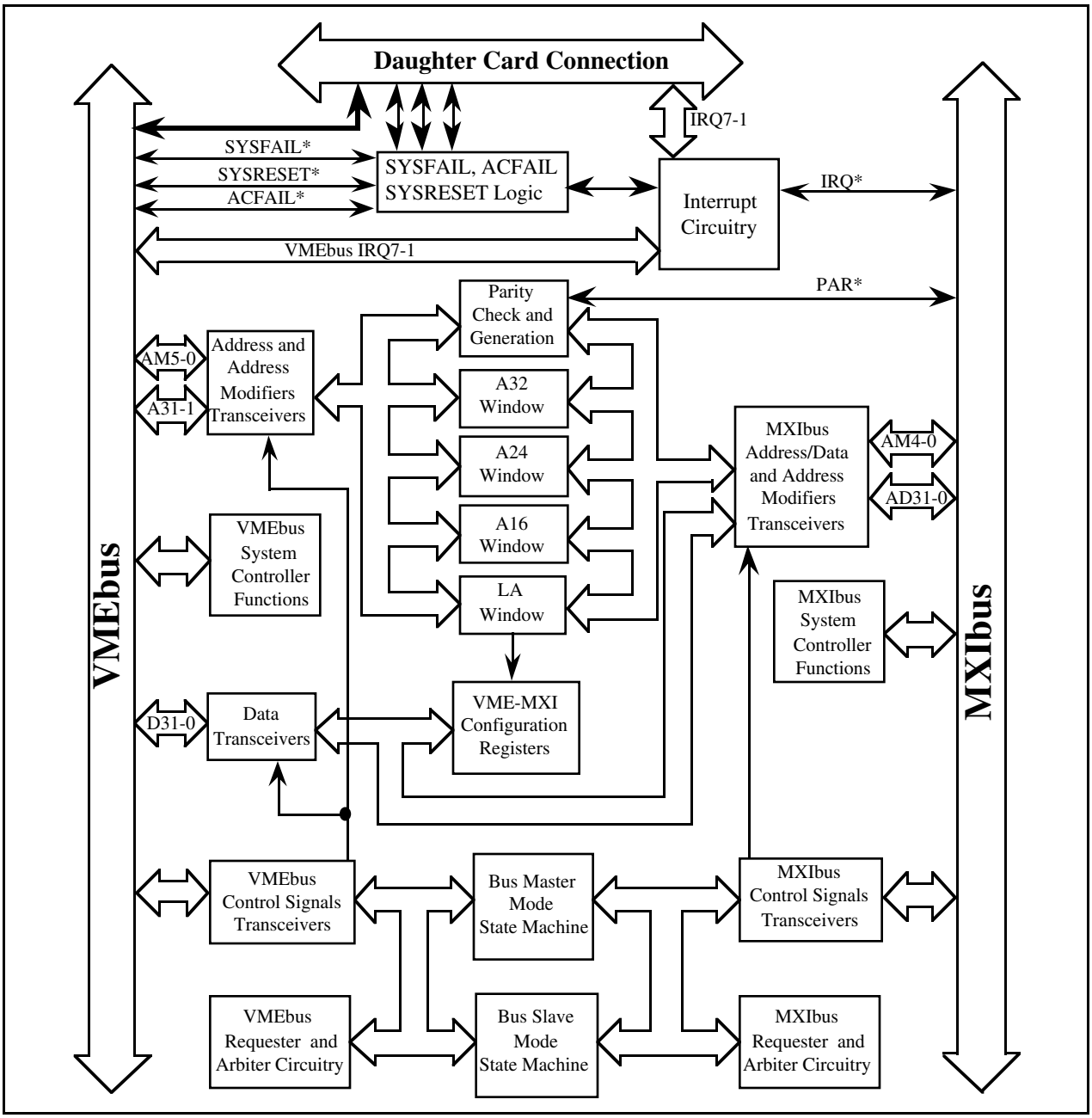


Figure 2-1. VME-MXI Block Diagram

- VMEbus Address and Address Modifiers Transceivers — These transceivers control the direction of the VMEbus address lines and latch the status of the address lines on the falling edge of the VMEbus address strobe.
- VMEbus System Controller Functions — If the VME-MXI is selected as the VMEbus System Controller, this circuitry generates the 16-MHz system clock, provides the VMEbus arbiter and the VMEbus Bus Timer Unit.
- VMEbus Data Transceivers — These transceivers control the direction of the VMEbus data lines and meet VMEbus specifications for timing and signal loading.
- VMEbus Control Signals Transceivers — These transceivers control the direction of the VMEbus control signals and meet VMEbus specifications for timing and signal loading.
- VMEbus Requester and Arbiter Circuitry — This circuitry is used to request the VMEbus and to provide the VMEbus arbiter function if the VME-MXI is the VMEbus System Controller.
- SYSFAIL, ACFAIL, and SYSRESET — Through this circuitry, the VMEbus signals SYSFAIL, ACFAIL and SYSRESET connect to the corresponding signals on the daughter card connections. These three signals can also be individually enabled to generate a VMEbus interrupt. With control bits in onboard registers, SYSFAIL and SYSRESET can also be driven on the VMEbus backplane.
- Interrupt Circuitry — This circuitry generates and receives interrupt requests on the VMEbus, the MXIbus, and on boards plugged into the daughter card connectors. Interrupt requests routed between VMEbus chassis can be transparently serviced by interrupt handlers in VMEbus chassis other than the requester's own chassis.
- Parity Check and Generation — This circuitry checks and generates MXIbus parity.
- A32, A24, A16 and LA Windows — These address windows assign portions of the MXIbus address space to the VMEbus chassis and vice versa.
- VME-MXI Configuration Registers — These registers provide all the configuration information required by the VME-MXI and are accessible from both the VMEbus and the MXIbus.
- MXIbus Master Mode State Machine — This state machine converts VMEbus cycles mapped out of a MXIbus window to the MXIbus into MXIbus cycles.

• MXIbus Slave Mode State Machine	This state machine converts MXIbus cycles mapped through a MXIbus window to the VMEbus mainframe into VMEbus cycles.
• MXIbus Address/Data and Address Modifiers Transceivers	These transceivers and associated circuitry control the direction of the MXIbus address and data lines. When a VMEbus transfer is mapped out to the MXIbus, the VMEbus address/data lines are multiplexed into the MXIbus address/data lines. When a MXIbus transfer is mapped into the VMEbus, the MXIbus address/data lines are demultiplexed into separate VMEbus address and data lines.
• MXIbus System Controller Functions	If the VME-MXI is the MXIbus System Controller, this circuitry provides the MXIbus arbiter, interrupt daisy-chain generation, and the MXIbus System Controller timeout logic.
• MXIbus Control Signals Transceivers	These transceivers control the direction of the MXIbus control signals.
• MXIbus Requester and Arbiter Circuitry	This circuitry is used to request the MXIbus when a VMEbus transfer is mapped into a MXIbus window.
• Daughter Card Connection	The two daughter card connectors can be used to add additional functionality to the VME-MXI in the form of plug-in daughter cards.

The following information applies only to VME-MXI kits that include the INTX daughter card option. Figure 2-2 is a block diagram of the circuitry of the INTX daughter card.

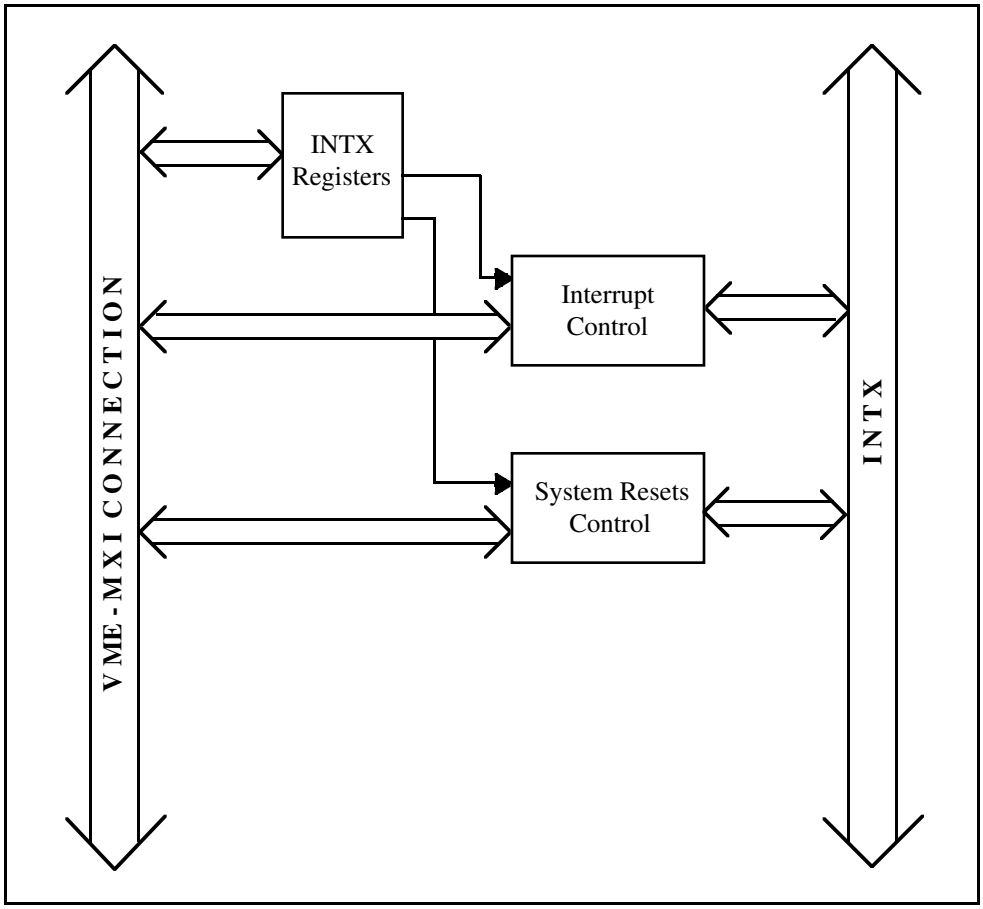


Figure 2-2. VME-MXI INTX Daughter Card Option Block Diagram

- INTX Registers

  The INTX card has two onboard registers that reside in the VME-MXI configuration space: the INTX Interrupt Configuration Register and the INTX Utility Configuration Register. These registers configure the mapping of the VMEbus interrupt lines, and the SYSRESET, SYSFAIL, and ACFAIL lines to and from the INTX connector. The INTX card also drives the Extended Device Type Class field in the VMEbus Status/Control Register when that register is accessed on the VME-MXI.

- Interrupt Control

The interrupt control logic maps the VMEbus interrupt lines to and from the corresponding INTX interrupt lines. In conjunction with the VME-MXI circuitry, the interrupt requests routed between VMEbus mainframes through the INTX connector can be transparently serviced by interrupt handlers in VMEbus mainframes other than the mainframe from which the request was generated. This process takes advantage of transparent MXIbus interrupt acknowledge cycles.

When an interrupt request received from across the INTX is driven on the corresponding VMEbus interrupt line, an interrupt handler in the receiving VMEbus mainframe generates an interrupt acknowledge cycle for that interrupt request. This interrupt acknowledge cycle is transparently converted into a MXIbus interrupt acknowledge cycle for that interrupt request level. Similarly, when a VMEbus interrupt line is driven out of the VMEbus mainframe across the INTX connection, an interrupt handler in another VMEbus mainframe can generate an interrupt acknowledge cycle to handle that interrupt. The VME-MXI in the requesting mainframe recognizes that the MXIbus interrupt acknowledge cycle is for the request it is driving and converts the cycle into a VMEbus interrupt acknowledge cycle that can service the VMEbus interrupt requester.

- System Resets Control

The system resets control circuitry maps the VMEbus signals SYSRESET, SYSFAIL, and ACFAIL to the corresponding signals on the INTX connection.

Chapter 3
Configuration and Installation

This chapter describes the configuration and installation of the VME-MXI.

Configuration

Before installing the VME-MXI in the VMEbus mainframe, configure the VME-MXI to suit the needs for your VMEbus system. The VME-MXI module contains jumpers, switches, and slide switches that you can use to configure the following options:

- VMEbus System Controller (slide switch S5)
- MXIbus Logical Address (switch U94)
- VMEbus Request Level (jumpers W2, W3, W4, W5, and W6)
- VMEbus Timeout Value (jumper W8)
- VMEbus Timeout Chain Position (jumper W1)
- Interlocked Arbitration Mode (slide switch S2)
- MXIbus System Controller (slide switch S4)
- MXIbus System Controller Timeout (jumper W7)
- MXIbus Fairness Option (slide switch S3)
- Reset Signal Select (slide switch S1)

Figure 3-1 shows the locations and factory default settings of the VME-MXI configuration jumpers and switches for a VME-MXI without the INTX option.

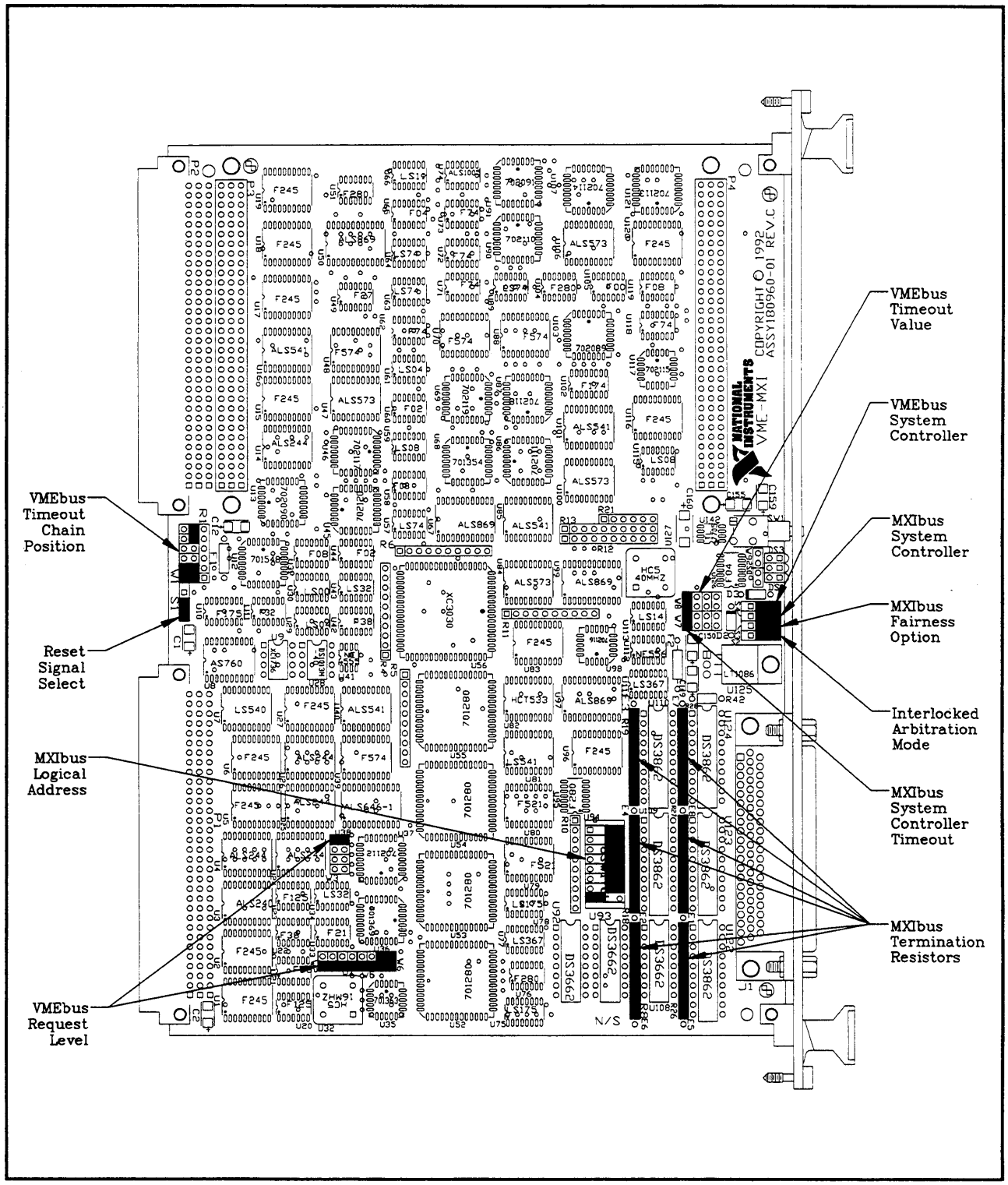


Figure 3-1. VME-MXI Parts Locator Diagram

Figure 3-2 shows the locations and factory default settings of the VME-MXI configuration jumpers and switches for a VME-MXI with the INTX option.

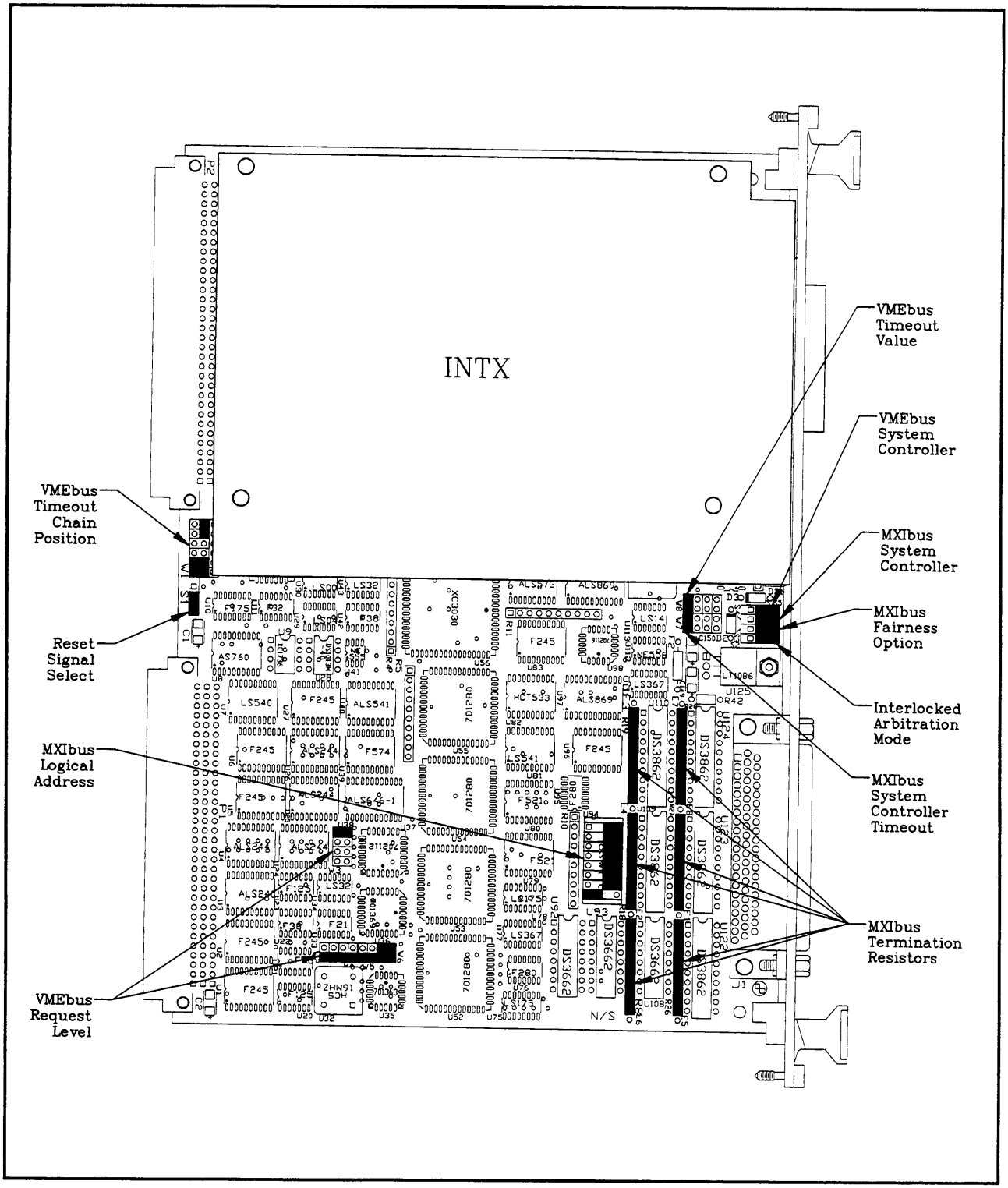


Figure 3-2. VME-MXI with INTX Parts Locator Diagram

VMEbus System Controller

The VME-MXI is shipped from the factory configured to be installed in Slot 1 of the VMEbus chassis as the VMEbus System Controller. If another device is already in Slot 1, you must decide which device will be the System Controller and reconfigure the other device as non-System Controller.

Warning: Do not install a device configured for Slot 1 into another slot without first reconfiguring it for Non-Slot 1 use. Doing so could result in damage to the Non-Slot 1 device, the VMEbus backplane, or both.

When the VME-MXI is the VMEbus System Controller, it has the VMEbus Data Transfer Bus Arbiter capability (PRI ARBITER) and it drives the 16-MHz VMEbus system clock. The VMEbus Data Transfer Bus Arbiter circuitry accepts bus requests on all four VMEbus request levels, prioritizes the requests, and grants the bus to the highest priority requester. The VMEbus system clock is driven by an onboard 16-MHz oscillator with a 50% ±5% duty cycle.

If you want to install the VME-MXI into a VXIbus mainframe, you can install it in any slot except Slot 0. The VME-MXI has VXIbus configuration registers, which makes it compatible with the VXIbus specification. However, the VME-MXI does not have the CLK10 and MODID circuitry required by a Slot 0 VXIbus device. Therefore, configure the VME-MXI as a non-System Controller and install it in any other slot.

If you want to use the optional interlocked bus mode, you must install the VME-MXI as the System Controller. Refer to the *Interlocked Arbitration Mode* section later in this chapter for a description of the optional interlocked bus mode. Figure 3-3(a) shows the default configuration setting for the VME-MXI installed as the VMEbus System Controller. To configure the VME-MXI as a non-System Controller, change slide switch S5 as depicted in Figure 3-3(b).

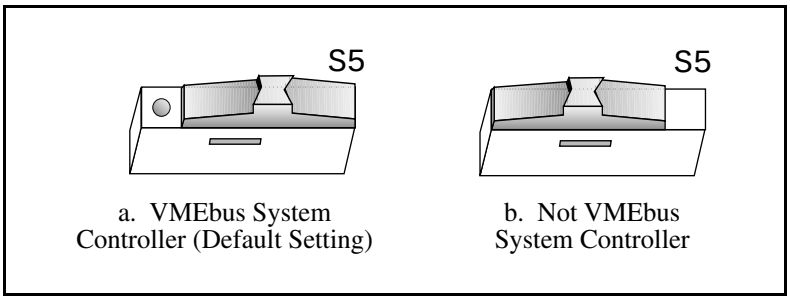


Figure 3-3. VMEbus System Controller Selection

MXIbus Logical Address

Each device in a MXIbus system is assigned a unique number between 0 and 254. This 8-bit number, called the *logical address*, defines the base address for the configuration registers located on the device. With unique logical addresses, each MXIbus device in the system is assigned 64 bytes of configuration space in the upper 16 kilobytes of A16 space.

The range of VMEbus A16 addresses occupied by the VME-MXI is defined by the logical address. The logical address corresponds to bits 13 through 6 of the A16 address when bits 15 and 14 are both one. The lower bound of addresses in this range is determined using the following equation:

(Logical Address) * 40 hex + C000 hex

The upper bound of addresses in this range is determined using the following equation:

(Logical Address) * 40 hex + C03F hex

For example, if the VME-MXI logical address is one, it will occupy A16 addresses C040 through C07F hex. If the VME-MXI logical address is 54 hex, the range of A16 addresses occupied is D500 through D53F hex.

If the VME-MXI is installed in a VXIbus mainframe, the VXIbus devices in that frame will also have logical addresses. Some VXIbus devices have dynamically configurable logical addresses. These devices have an initial logical address of hex FF, which indicates that they can be dynamically configured. While the VME-MXI does support dynamic configuration of VXIbus devices within its mainframe, it cannot itself be dynamically configured. Therefore, do not set the logical address for the VME-MXI to hex FF.

The MXIbus RM has Logical Address 0 by definition. The VME-MXI does not have MXIbus RM capability, so do not set the logical address for the VME-MXI to 0. If you are configuring a multiple chassis VMEbus/MXIbus system, refer to Chapter 5, *Programming Considerations*, for instructions on planning a VMEbus/MXIbus system logical address map. If you are connecting only a PC with a MXIbus interface to the VME-MXI, leave the logical address at the default setting of 1. Using this setting, you can install devices with all other possible logical addresses in the VMEbus chassis.

An 8-bit DIP switch at position U94 selects the logical address for the VME-MXI. Refer to Figure 3-1 to find the location of this DIP switch. The ON position on the DIP switch corresponds to a logic value of 0, and the OFF position corresponds to a logic value of 1. This switch is set at the factory to a default logical address of 1. Verify that the logical address assigned to the VME-MXI is not used by any other statically configured VMEbus device in your system. Remember that logical addresses hex 0 and FF are not allowed for the VME-MXI. Figure 3-4 shows switch settings for logical address hex 1 and 54.

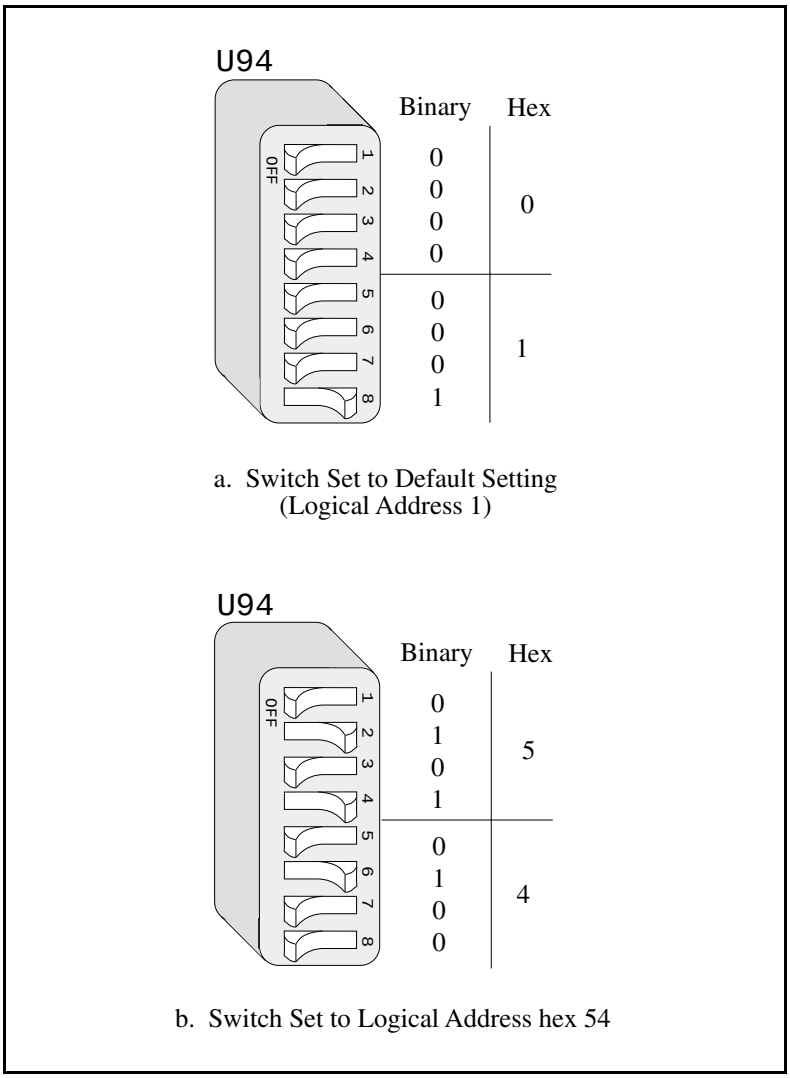


Figure 3-4. Logical Address Selection

VMEbus Request Level

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

The VME-MXI is shipped from the factory configured to use VMEbus request level 3. Request level 3 is the highest priority request level and request level 0 is the lowest. You can change the VME-MXI to use any of the other three request levels by changing the jumper configuration of the pin arrays at locations W2, W3 , W4, W5, and W6. You may want to change request levels to change the priority of the VME-MXI request signal. For more information, refer to the VMEbus specification.

To change the VMEbus request level of the VME-MXI, rearrange the jumpers on the pin arrays as shown in Figure 3-5.

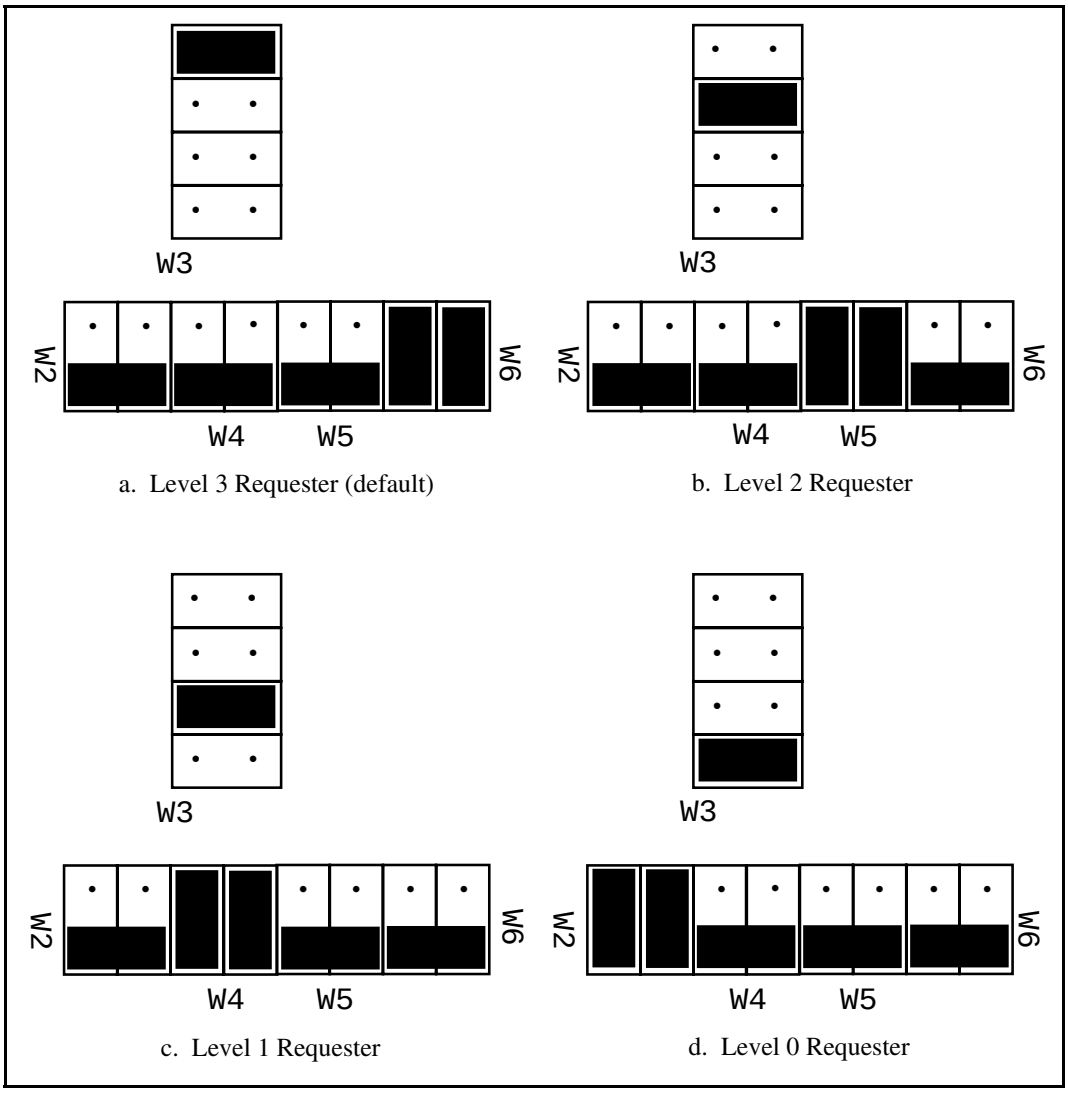


Figure 3-5. VMEbus Requester Jumper Settings

VMEbus Timeout Value

A VME-MXI installed in a VMEbus chassis must contain the only VMEbus Bus Timeout Unit (BTO) circuitry for the VMEbus chassis. If you have multiple VME-MXI interfaces in a chassis, the BTO must be enabled on one of them. If you are installing multiple VME-MXIs in a VXIbus mainframe, place them in adjacent slots, with the BTO enabled on the VME-MXI that is closest to the VMEbus System Controller.

The BTO monitors the current bus cycle and asserts the bus error (BERR) signal if a data transfer acknowledge (DTACK) or BERR is not received from the selected slave within the given amount of time after data strobe (DS1 or DS0) becomes active. Whenever a VMEbus transfer that maps out to the MXIbus occurs, the VMEbus timeout on the VME-MXI is disabled and the MXIbus System Controller BTO monitors the transfer. With this configuration, VMEbus transfers can have short bus timeout values and MXIbus transfers can have much longer timeout values.

You can either disable the VMEbus timeout value or set it to 100, 200, or 400 µsec by rearranging the jumper selection at location W8, as shown in Figure 3-6. If the VME-MXI is not the VMEbus System Controller, remember to enable the BTO on the VME-MXI and to disable the VMEbus BTO on the System Controller device.

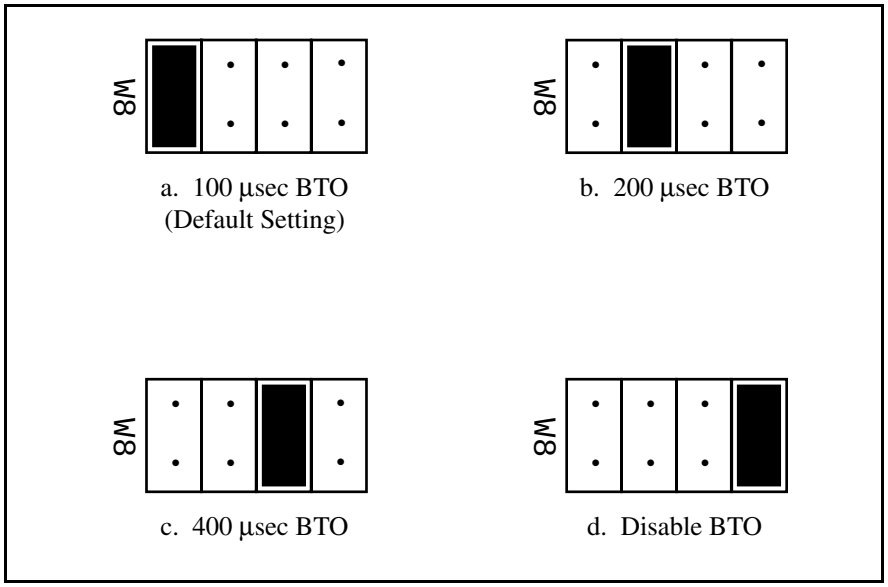


Figure 3-6. VMEbus Timeout Value Selection

VMEbus Timeout Chain Position

The jumper block at location W1 indicates the location of the VME-MXI interface in relation to other VME-MXIs installed in the chassis. If your system contains only one VME-MXI, set the W1 jumper block to the configuration shown in Figure 3-7.

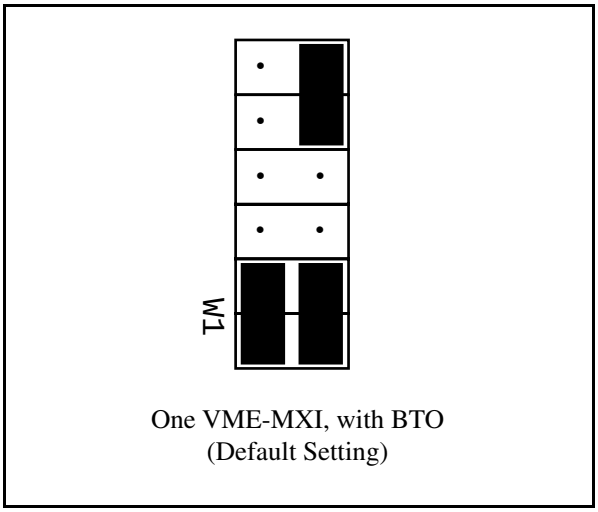
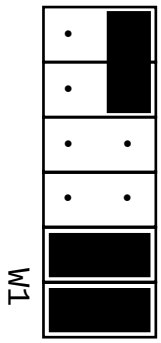


Figure 3-7. VMEbus Timeout; One VME-MXI in Chassis

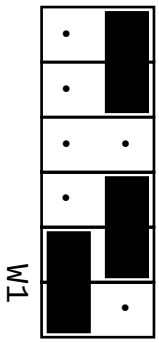
When you have multiple VME-MXI modules installed in a VMEbus chassis, one of the user-defined pins on the P2 connector is used to send a signal to the VME-MXI with the VMEbus BTO to indicate that an outward MXIbus transfer is in progress. When multiple VME-MXI modules are installed in a VXIbus mainframe, they must be installed in adjacent slots so that the VXIbus local bus can be used to send a signal to the VME-MXI with the VMEbus BTO to indicate that an outward MXIbus transfer is in progress.

The following figures show how to configure jumper block W1 to select how the VXIbus local bus or the VMEbus user-defined pins are used to disable the VMEbus timeout during outward MXIbus transfers.

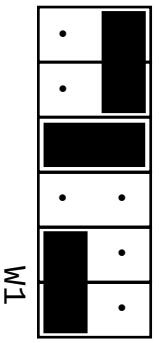
If your system contains more than one VME-MXI, select which card will supply the VMEbus timeout, and set the W1 jumper block according to the VME-MXI's position in relation to the adjacent VME-MXIs. Figure 3-8 shows three possible settings for a VMEbus chassis. Figure 3-9 shows three possible settings for a VXIbus mainframe.



a. VME-MXI with BTO, Multiple VME-MXIs in Chassis, User-Defined Pin C30 (Suggested Configuration)

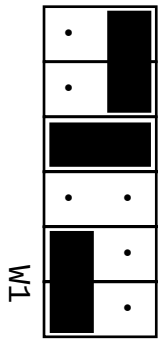


b. VME-MXI with BTO, Multiple VME-MXIs in Chassis, User-Defined Pin A05

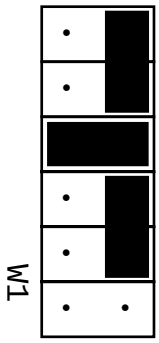


c. VME-MXI with BTO, Multiple VME-MXIs in Chassis, User-Defined Pin C05

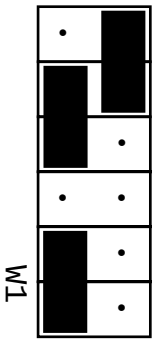
Figure 3-8. VMEbus Timeout; Multiple VME-MXIs in VMEbus Chassis



a. Non-Slot 0 VME-MXI with BTO, the VME-MXI Closest to Slot 0, Multiple VME-MXIs in VXIbus Mainframe



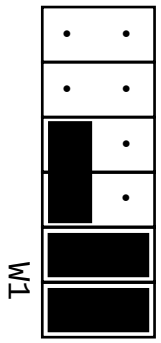
b. Non-Slot 0 VME-MXI with BTO, VME-MXI Located Between Two VME-MXIs, Multiple VME-MXIs in VXIbus Mainframe



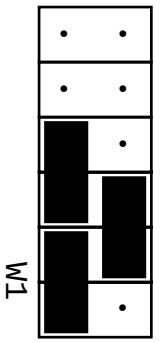
c. Non-Slot 0 VME-MXI with BTO, the VME-MXI Furthermost from Slot 0, Multiple VME-MXIs in VXIbus Mainframe

Figure 3-9. VMEbus Timeout; Multiple VME-MXIs in VXIbus Mainframe

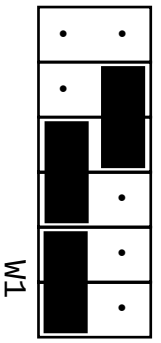
For those VME-MXIs that do not supply the VMEbus timeout, set the W1 jumper block to reflect each VME-MXI's position in relation to the other VME-MXIs in the chassis. Figure 3-10 shows jumper settings for installation in a VMEbus chassis. Figure 3-11 shows possible jumper settings for installation in a VXIbus mainframe.



a. VME-MXI without BTO, Multiple VME-MXIs in Chassis, User-Defined Pin C30 (Suggested Configuration)

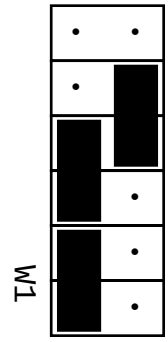


b. VME-MXI without BTO, Multiple VME-MXIs in Chassis, User-Defined Pin A05

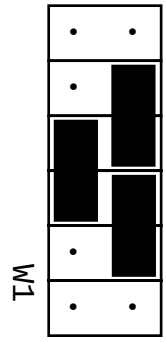


c. VME-MXI without BTO, Multiple VME-MXIs in Chassis, User-Defined Pin C05

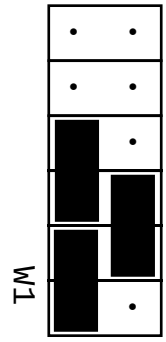
Figure 3-10. No VMEbus Timeout; Multiple VME-MXIs in VMEbus Chassis



a. Non-Slot 0 VME-MXI without BTO, the VME-MXI Closest to Slot 0, Multiple VME-MXIs in VXIbus Mainframe



b. Non-Slot 0 VME-MXI without BTO, VME-MXI Located Between Two VME-MXIs, Multiple VME-VXIs in VXIbus Mainframe



c. Non-Slot 0 VME-MXI without BTO, the VME-MXI Furthermost from Slot 0, Multiple VME-MXIs in VXIbus Mainframe

Figure 3-11. No VMEbus Timeout; Multiple VME-MXIs in VXIbus Mainframe

Interlocked Arbitration Mode

Interlocked arbitration mode is an optional mode of operation in which the system performs as one large VMEbus chassis with only one master of the entire system (VMEbus and MXIbus) at any given moment. This mode of operation prevents deadlocks by interlocking all arbitration in the VMEbus/MXIbus system. Refer to Chapter 6 for a thorough discussion of interlocked arbitration mode.

In the normal operating mode, there can be multiple masters operating simultaneously in the VMEbus/MXIbus system. A deadlock occurs when a MXIbus master requests use of a VMEbus resource in another VMEbus chassis while a VMEbus master in that chassis is in the process of requesting a resource across the MXIbus. To resolve the conflict, the VME-MXI will assert BERR on the VMEbus to terminate the transfer and cause the VMEbus master to give up its bus ownership. Devices in the VMEbus chassis must be able to detect a BERR caused by a deadlock condition so that they can retry the operation.

The VME-MXI is shipped from the factory configured for normal operating mode. If MXIbus transfers will be occurring both into and out of the chassis, and the VMEbus modules in your system do not have the capability for handling BERR exceptions caused by deadlock conditions, you may want to configure the VME-MXI for interlocked arbitration mode. In this mode, no changes will need to be made to software. However, parallel processing in separate VMEbus chassis is no longer possible, and system performance may be lower than in normal operating mode.

VMEbus requesters are awarded the bus when they receive an active signal on the daisy-chained bus grant line. Requesters closest to the VMEbus System Controller have higher priority, therefore, than devices installed in slots farther from the VMEbus System Controller. In addition, four bus request levels further prioritize modules. For proper operation in interlocked arbitration mode, only one chassis can have a requester at a higher priority than the VME-MXIs in that chassis. In all the other chassis, the VME-MXIs must be the highest priority requesters. In other words, the VME-MXIs should be installed in Slot 1 and the adjacent slots with the option that one chassis in the MXIbus link (for example, a multiframe RM) has a VMEbus requester that is the VMEbus System Controller. Devices in the chassis can be configured to operate with the same or different VMEbus requester levels.

Note: Interlocked arbitration mode has a potential for long access times. Therefore, you should configure bus timeouts for adequate times.

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

This type of system configuration is recommended if you have a VMEbus chassis where the masters communicate only with the slaves in their chassis. 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 chassis. You can configure the VME-MXIs in this chassis for normal operating mode.

Select interlocked arbitration mode by changing the default setting of the slide switch at location S2, as shown in Figure 3-12.

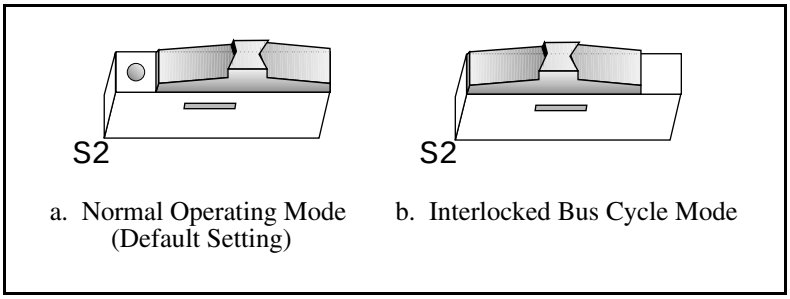


Figure 3-12. Interlocked Arbitration Mode Selection

MXIbus System Controller

The slide switch at location S4 selects whether the VME-MXI interface module is the MXIbus System Controller. The MXIbus System Controller is the first device in the MXIbus daisy-chain. The System Controller supplies the arbitration circuitry for MXIbus arbitration, the MXIbus interrupt acknowledge daisy-chain driver, and the MXIbus bus timeout unit. The VME-MXI is shipped from the factory configured for non-MXIbus System Controller operation. If the VME-MXI is the first device in the MXIbus link, change the setting of the S4 slide switch as shown in Figure 3-13 to configure the VME-MXI as the MXIbus System Controller.

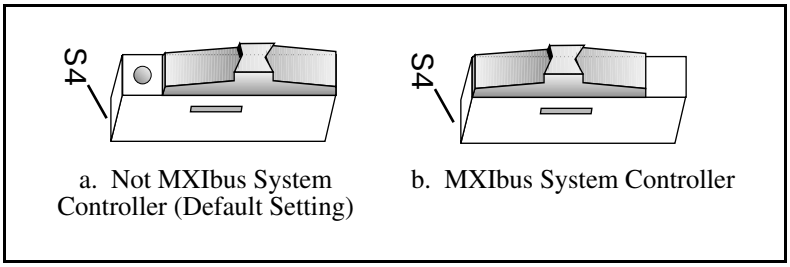


Figure 3-13. MXIbus System Controller Selection

MXIbus System Controller Timeout

The MXIbus System Controller is also responsible for the MXIbus system timeout. The timeout period begins when a MXIbus data strobe (DS) is received. The period stops at the detection of a MXIbus DTACK or BERR. If a timeout occurs, the MXIbus System Controller sends a MXIbus BERR to clear the MXIbus system. On power up, this timeout is between 100 and 400 µsec as configured by the jumper array at location W7. You can extend the timeout to a value between 10 and 40 msec by setting the LNGMXSCTO bit in the MXIbus Control Register to one. A long MXIbus System Controller timeout is desirable in MXIbus systems with many devices or in situations where one or more MXIbus devices use a large amount of MXIbus bandwidth.

Figure 3-14 shows how to position the jumper array W7 to set the MXIbus System Controller timeout value. When the LNGMXSCTO bit in the MXIbus Control Register is zero, the selected timeout value is in µsec. When the LNGMXSCTO bit is one, the selected timeout value is in msec. When the VME-MXI is not configured to be the MXIbus System Controller, the setting of this jumper array has no effect.

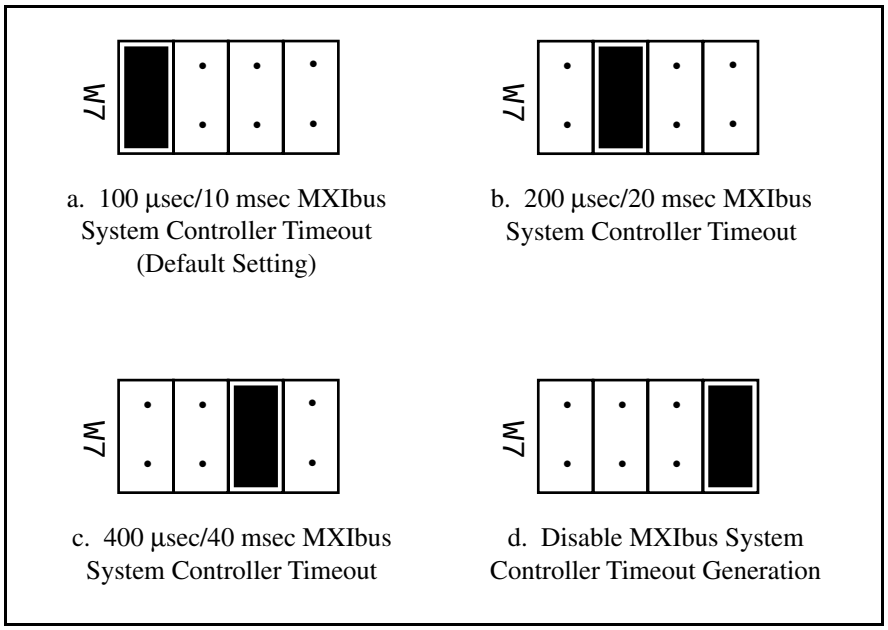


Figure 3-14. MXIbus System Controller Timeout Value Selection

MXIbus Fairness Option

The MXIbus fairness feature ensures that all requesting devices will be granted use of the MXIbus. This feature prevents a high priority MXIbus device from consuming all of the MXIbus bandwidth. If MXIbus fairness is enabled, a MXIbus master will not request the bus until it detects that no other devices are requesting the bus. MXIbus fairness ensures that all MXIbus masters have an equal opportunity to use the MXIbus.

The VME-MXI factory default setting has the MXIbus fairness feature disabled. Keep this option disabled if a device in your chassis needs a large portion of the MXIbus bandwidth without interruptions from lower priority requesters. In an unfair system, the order in which you connect the MXIbus devices in the daisy-chain determines the priority of each device's MXIbus request. MXIbus requesters closer to the MXIbus System Controller have higher priority than those further down the MXIbus chain. The MXIbus fairness feature is enabled or disabled by the slide switch at location S3. If you want your VME-MXI to be a fair requester, change the slide switch from the default setting as shown in Figure 3-15.

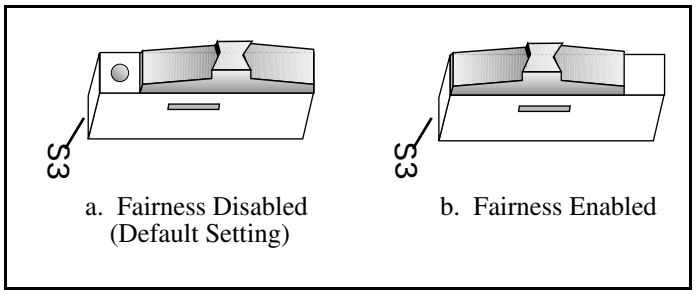


Figure 3-15. MXIbus Fair Requester Selection

Reset Signal Select

The VME-MXI generates a 200 msec active low pulse both on power-up and when you press the pushbutton system reset switch on the front panel. Using slide switch S1, you can route the pulse to either VMEbus signal ACFAIL* or SYSRESET*. See Figure 3-16.

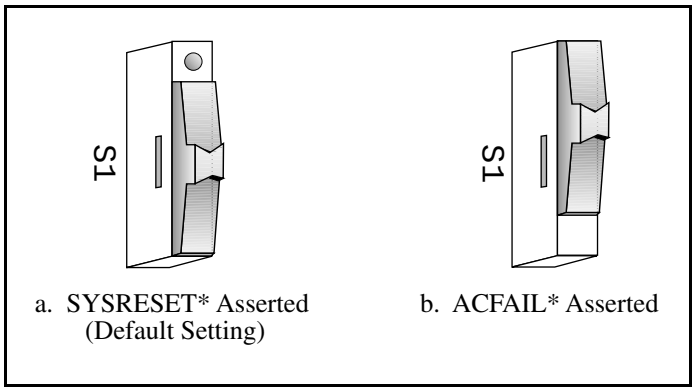


Figure 3-16. Reset Signal Selection Settings

Installing the VME-MXI Hardware

The VME-MXI is a VMEbus extender; it has no onboard intelligence or memory. For the VME-MXI to perform as a MXIbus master, a device with VMEbus master capability must be installed in the VMEbus chassis. For the VME-MXI to perform as a MXIbus slave in A16, A24, or A32 space, a slave VMEbus device with resources in those address spaces must be installed in the VMEbus chassis.

Note: Pay special attention to the configuration of the A16 resources used by the VMEbus cards in your system. The upper 16 kilobytes of A16 space have been reserved for configuration registers on MXIbus devices.

If possible, configure the A16 resources for your VMEbus boards in the lower 48 kilobytes (0000 through BFFF hex) of A16 space, to avoid interfering with MXIbus configuration space. The logical address window mapping window is then used for mapping configuration space for MXIbus devices, and the A16 window mapping window is used for mapping configuration space for VMEbus devices.

If you do configure some of the A16 resources on VMEbus devices in the upper 16 kilobytes (C000 through FFFF hex) of A16 space, you must be careful not to configure any static MXIbus logical addresses in the portions of A16 space occupied by the VMEbus devices.

Warning: The VME-MXI is shipped from the factory configured as the VMEbus System Controller to be installed into Slot 1 of your VMEbus chassis. Installing your VME-MXI into any slot other than Slot 1 without changing its default configuration can damage the VME-MXI, the VMEbus backplane, or both.

If a device is already installed in Slot 1, either reconfigure that device and install it in another slot, or reconfigure your VME-MXI for non-System Controller use. Remember also that the VME-MXI must have the VMEbus BTO. If another device is installed in Slot 1, verify that it does not have a VMEbus BTO enabled. If you will be installing the VME-MXI into a VXIbus chassis, you must configure the VME-MXI for non-System Controller use and install it in any slot except Slot 0.

MXIbus Termination

The MXIbus is a matched impedance bus and requires termination networks at the first and last device in the MXIbus daisy-chain. These terminations minimize reflections caused by impedance discontinuities at the ends of the cables. These termination networks are located at the end device's MXIbus connectors and can be either external self-contained modules or internal plug-in resistor packages. The VME-MXI comes with terminating resistors installed. If you prefer, you can replace them with external resistor packages for easy system reconfiguration. Figure 3-17 shows an example of a daisy-chained MXIbus system, including terminators.

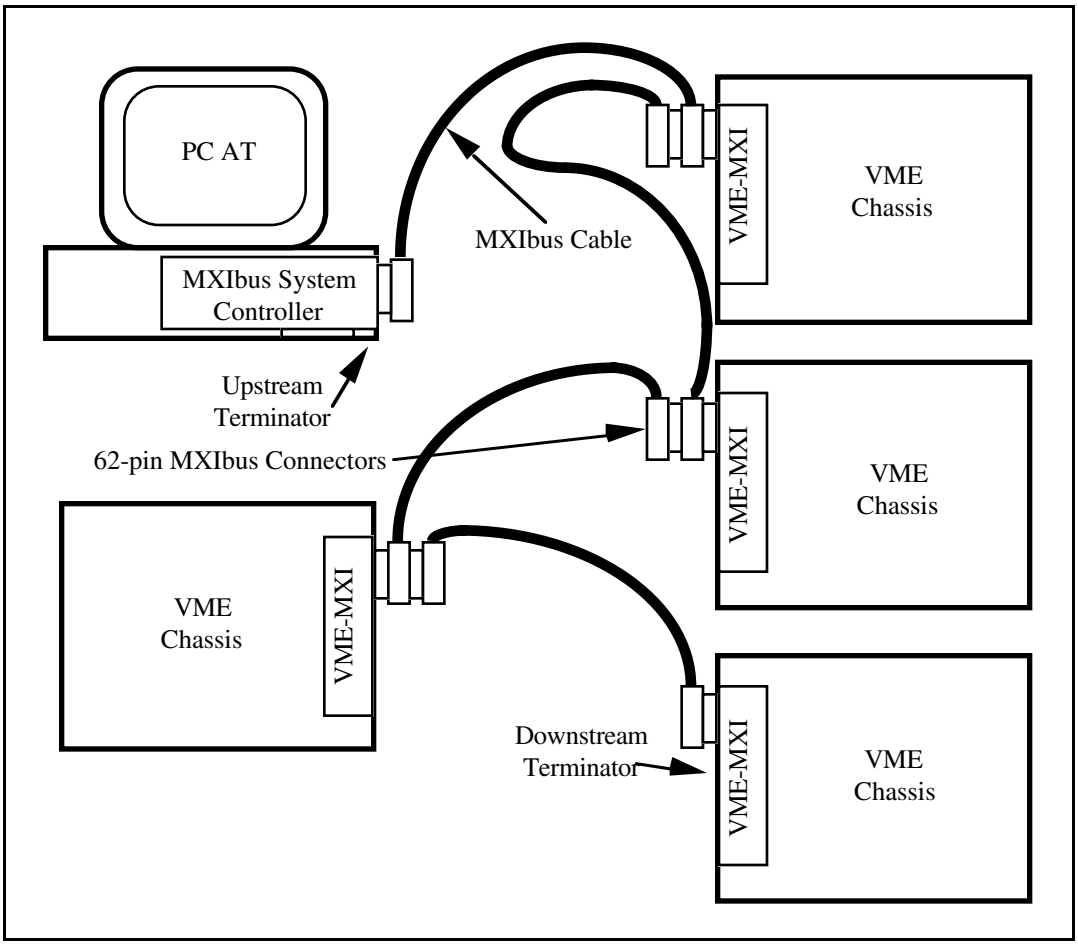


Figure 3-17. MXIbus Daisy-Chain

The VME-MXI uses the TERMPWR connection on the MXIbus connector as the power source for external MXIbus termination networks. TERMPWR is protected by a fuse that limits the maximum current that can be drawn to 2A. The fuse is soldered on the module and is not user-replaceable.

Note: TERMPWR is not intended to provide power to any other device.

The VME-MXI is shipped from the factory with terminating SIP resistor networks installed. If the VME-MXI will be the first or last device in the MXIbus daisy-chain and you will not be using external terminating networks, leave these internal terminators in place. If the VME-MXI is not going to be an end device, or if you will be using external terminators, remove the terminating resistor networks from their sockets and store them in a safe place in case the MXIbus system changes.

Figure 3-18 shows the position of the six MXIbus terminating networks. All six MXIbus networks must be either installed or removed from their sockets.

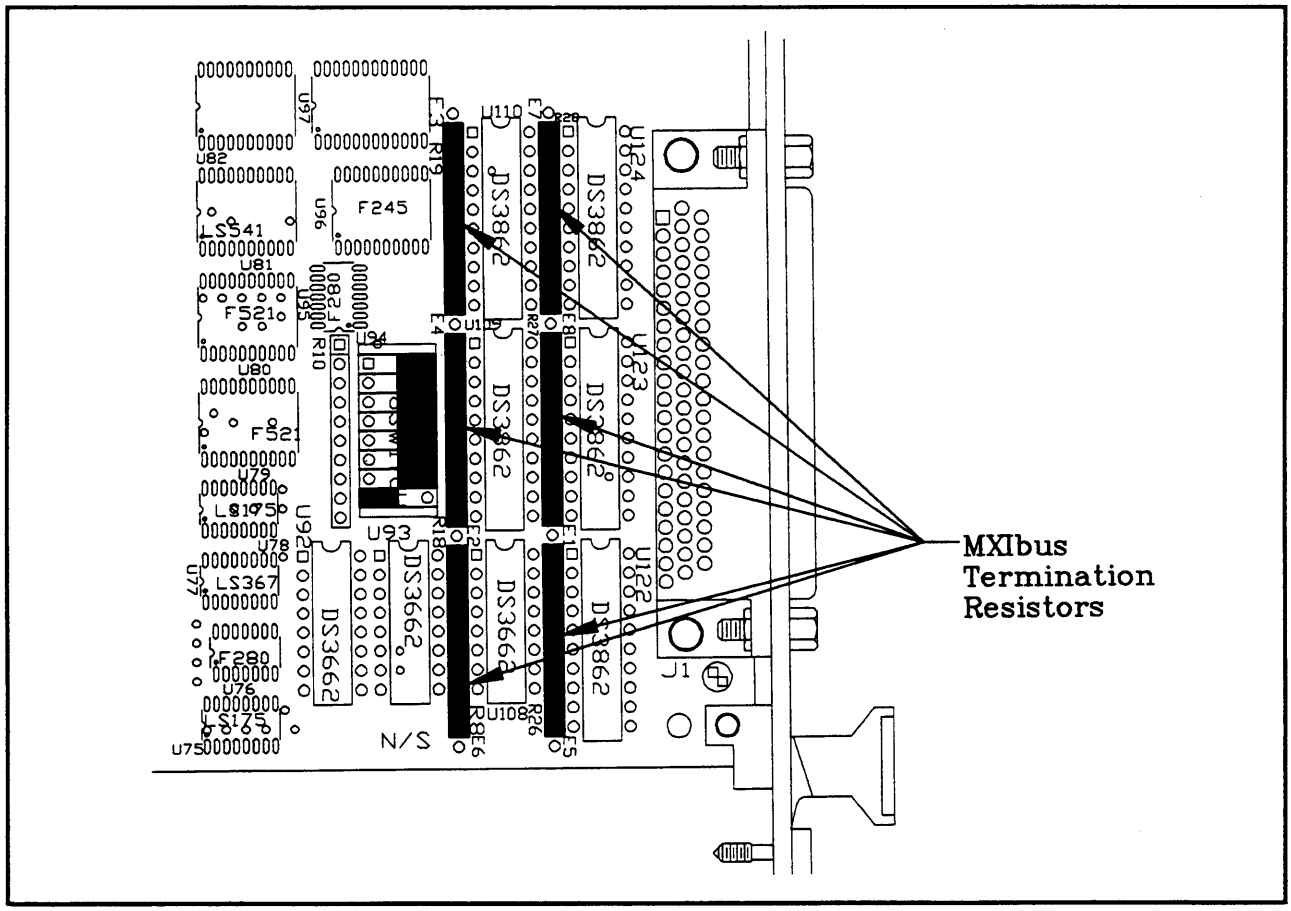


Figure 3-18. MXIbus Terminating Networks

INTX Termination

If your VME-MXI includes the INTX daughter card option, you must follow much the same procedure for termination as for the MXIbus terminators. The INTX bus requires termination networks at the first and last devices in the INTX chain. These terminations minimize reflections caused by impedance discontinuities at the ends of the cables and bias the signal lines to their unasserted state when they are not driven. The INTX daughter card comes with terminating resistors installed. Figure 3-19 is a parts locator diagram of the INTX daughter card, showing the location of the INTX terminators.

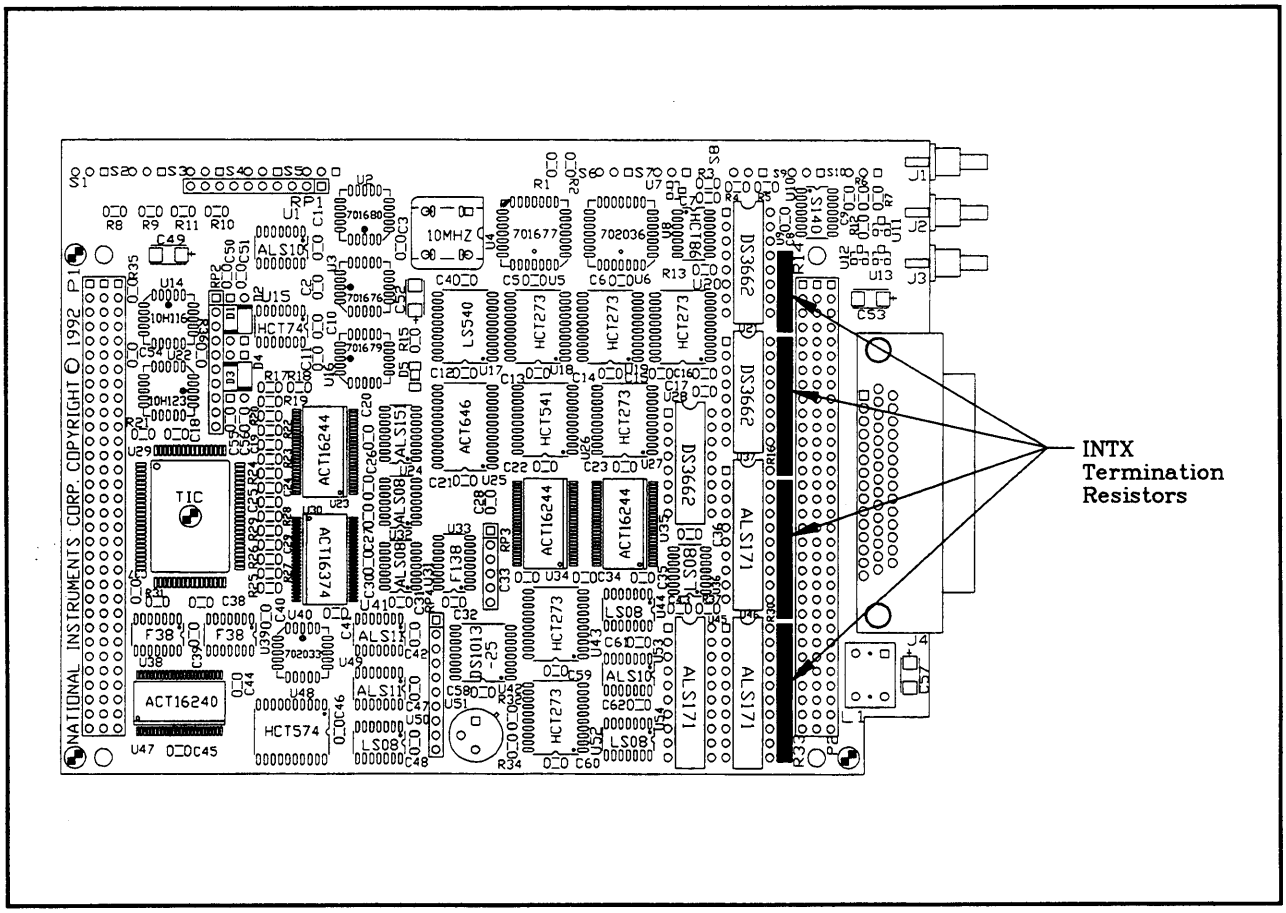


Figure 3-19. VME-MXI INTX Parts Locator Diagram

If the daughter card will be the first or last device in the INTX chain (irrespective of the VME-MXI's position in the MXIbus chain), you should leave these terminators in place. If the daughter card is not going to be an end device, remove all the terminating resistor networks from their sockets. Follow the instructions in the following sections to remove and reinstall the daughter card from the VME-MXI.

Removing the INTX Daughter Card from the VME-MXI

Should you find it necessary to remove the INTX card, follow these steps.

1. Remove the two INTX connector jack sockets on the sides of the INTX connector, and the two MXIbus jack sockets on the sides of the MXIbus connector on the front panel of the VME-MXI.

2. Remove the two screws on the bottom of the VME-MXI board located near the front panel.

3. Remove the front panel of the VME-MXI.

4. Carefully remove the four screws on the top of the daughter card. These screws are made of nylon and can be damaged if not handled carefully.

5. Carefully lift the daughter card out of the daughter card connectors on the VME-MXI.

6. Remove the INTX terminators and store them in a safe place in case your system configuration changes.

Installing the INTX Daughter Card onto the VME-MXI

Follow these steps when you are ready to reinstall the INTX card onto the VME-MXI,

Warning: Improper installation of the INTX daughter card option onto the VME-MXI can result in damage to the INTX daughter card or the VME-MXI. Verify that the connections line up correctly before installing the VME-MXI into the VMEbus mainframe.

1. Carefully line up the daughter connection pins to the daughter card sockets on the VME-MXI and firmly press the connectors together.

2. Replace the four screws on the top of the daughter card. Take care not to overtighten these nylon screws to avoid damaging them.

3. Replace the front panel of the VME-MXI.

4. Replace the two screws on the bottom of the VME-MXI board located near the front panel.

5. Replace the two INTX connector jack sockets on the sides of the INTX connector, and the two MXIbus jack sockets on the sides of the MXIbus connector on the front panel of the VME-MXI.

Figure 3-20 shows an example of a daisy-chained MXIbus and INTX system, including terminators.

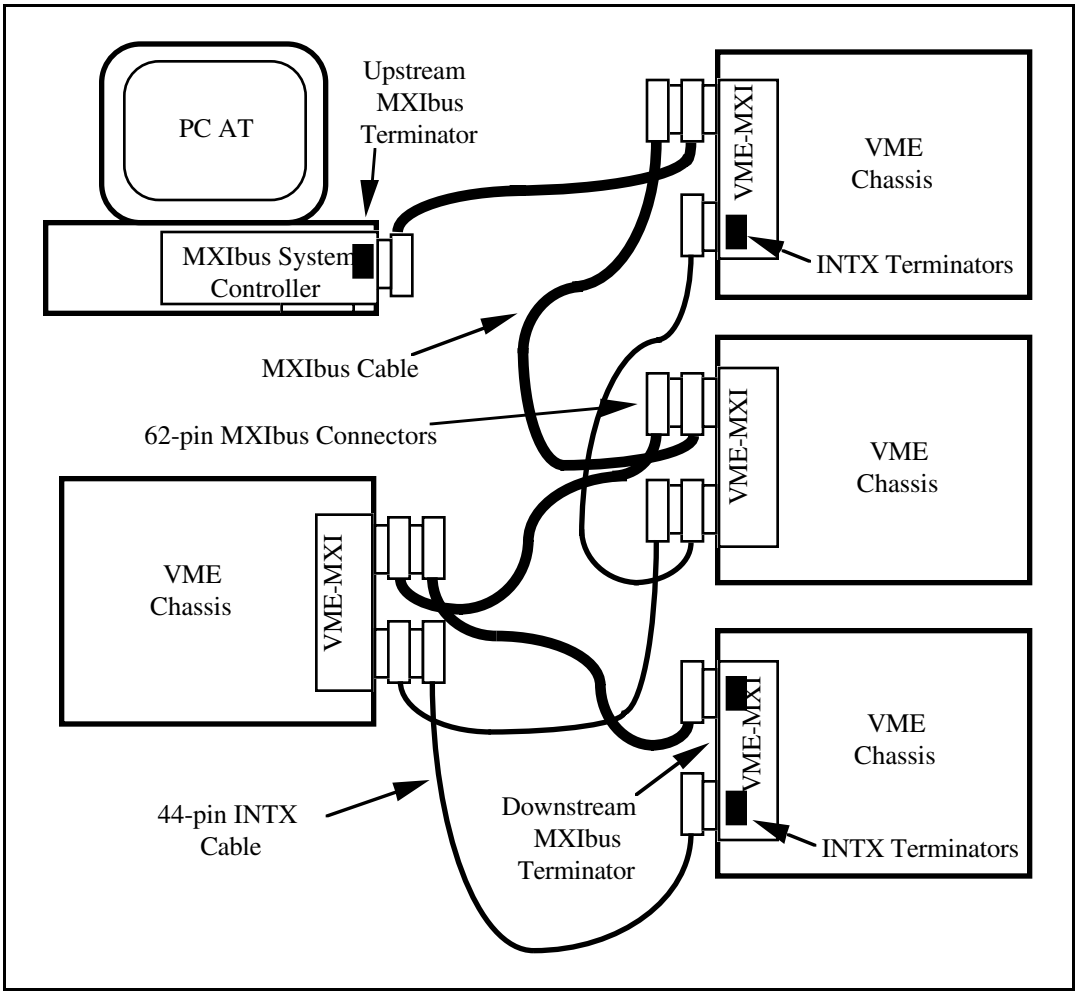


Figure 3-20. INTX Terminator Example

VME-MXI Installation Instructions

Verify the following configuration considerations before installing the VME-MXI:

- If you are installing the VME-MXI in a VMEbus chassis slot other than Slot 1, or into a VXIbus mainframe, verify that you have changed the settings of the VMEbus System Controller slide switch (S5), and the VMEbus relative position jumper (W1).
- If you are installing multiple VME-MXIs into a VXIbus mainframe, install them in adjacent slots with proper relative position jumper settings (W1). The VMEbus BTO must be enabled on one of them.

- If you are using interlocked arbitration mode, all the VME-MXIs must be the highest priority VMEbus requesters in that chassis. However, one, and only one, chassis in the MXIbus link can have a higher priority VMEbus requester than its VME-MXIs.
- The first and last MXIbus devices in the MXIbus link must be terminated.
- No two devices in your MXIbus system can have the same logical address.

After you verify the termination networks, switches, and jumpers, record all settings in the Hardware Configuration Form in Appendix D, *Customer Communication*. You are now ready to install the VME-MXI. Following are general instructions for installing your VME-MXI in your VMEbus chassis. Consult the user manual or technical reference manual of your VMEbus chassis for specific instructions and warnings.

1. Turn off power to the chassis.
2. Remove or open any doors or covers blocking access to the chassis slots.
3. If the VME-MXI will be installed in a C-size or D-size chassis VXIbus mainframe, install a support designed for installing B-size VMEbus cards in larger mainframes.
4. Insert the VME-MXI into the slot of the chassis by aligning the top and bottom of the card with the card guides inside the chassis. Slowly push the VME-MXI straight into the slot until it seats in the backplane receptacles. The front panel of the VME-MXI should be even with the front panel of the chassis.
5. Tighten the retaining screws on the top and bottom edges of the front panel.
6. Check installation.
7. Connect MXIbus cable.
8. Replace or close any doors or covers to the chassis, and restore power to the chassis.

Connecting the INTX Cable

For VME-MXIs with the INTX daughter card option, you must use special INTX cables for routing the additional VMEbus signals to other frames. The INTX cable provides either a straight point to point link (National Instruments part number 180980-*XX* where *XX* is the length in meters) or a single connector to dual connector link (National Instruments part number 180982-*XX*, where *XX* is the length in meters), which gives you the ability to connect more than two devices together.

Notice that while the MXIbus is a prioritized daisy-chain, INTX signals are bused to every device and no priority exists. Like the MXIbus, however, INTX cables must be connected in a daisy-chain fashion to prevent impedance discontinuities from stubs that are created in a star-type configuration.

Secure the INTX cable(s) on the back of the INTX connector using the captive screw elements to ensure that the cable(s) will not accidentally become disconnected.

Connecting the MXIbus Cable

MXIbus devices are daisy-chained together with MXIbus cables. Dual-ended cables are polarized and require proper connection to function properly. The VME-MXI uses a shielded 62-pin high-density D-subminiature device connector specified in the MXIbus specification. When properly configured, MXIbus cables will dress down and away from the VMEbus chassis. Ensure that the proper cable ends are connected to the intended devices. See Figure 3-17.

If your VME-MXI is the first or last device in the MXIbus and you choose to use an external termination network, install it on the VME-MXI connector before attaching the MXIbus cable. Be sure to press the terminator firmly in place and use the captive screw elements to secure the terminator in place.

If your cable has a single connector on each end of the cable (National Instruments part number 180758-*XX*, where *XX* is the length in meters), it is suitable for connecting two MXIbus devices together. This cable is nonpolarized and can be installed with either end connected to either device. Connect one end of the cable to the MXIbus System Controller. Connect the other end of the cable to the second device. Figure 3-21 shows an AT-MXI serving as the MXIbus System Controller connected to a VME-MXI.

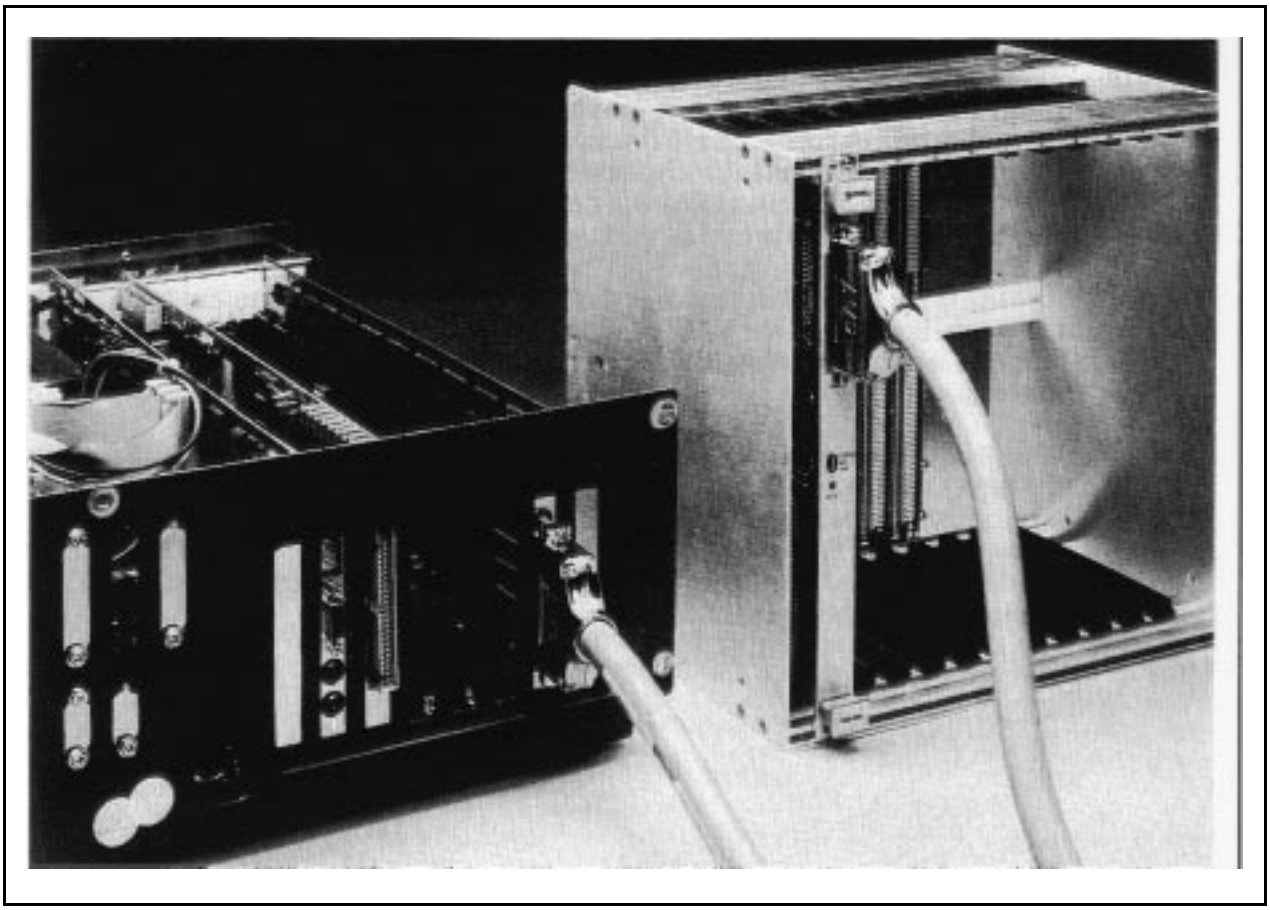


Figure 3-21. MXIbus Single-Ended Cable Configuration

If your MXIbus cable has a single connector on one end and a dual-ended connector on the other end (National Instruments part number 180760-*XX* or 180761-*XX*, where *XX* is the length in meters), you can create a MXIbus system that consists of more than two devices. A MXIbus system is defined as the set of devices physically connected by individual MXIbus cable links. These devices form a daisy-chain in which the relative priority of a device within that chain is determined by its proximity to the first device in the MXIbus system, the MXIbus System Controller. Devices closer to the MXIbus System Controller have a higher priority than others in the daisy-chain. Refer to Figure 3-17 for an example of a MXIbus system.

Begin establishing the system by connecting the end of the cable with the single connector to the MXIbus System Controller and the end of the cable with the dual-ended connectors to the next device in the MXIbus link. If your system contains more than two devices, connect the single connector of the next cable to the back of the dual-ended connector that you connected to the second MXIbus device. Connect the dual-connector end to the next device. Continue in this manner until you have all devices in your system connected.

Note: A MXIbus system may contain no more than eight daisy-chained devices and must have a total cable distance not exceeding 20 meters.

Secure the MXIbus cable(s) on the back of the MXIbus connector (or terminating network) using the captive screw elements to ensure that the cable(s) will not accidentally become disconnected.

Figure 3-22 shows an AT-MXI serving as the MXIbus System Controller connected with the single connector end of the cable, and a VME-MXI connected with the dual-ended connector.

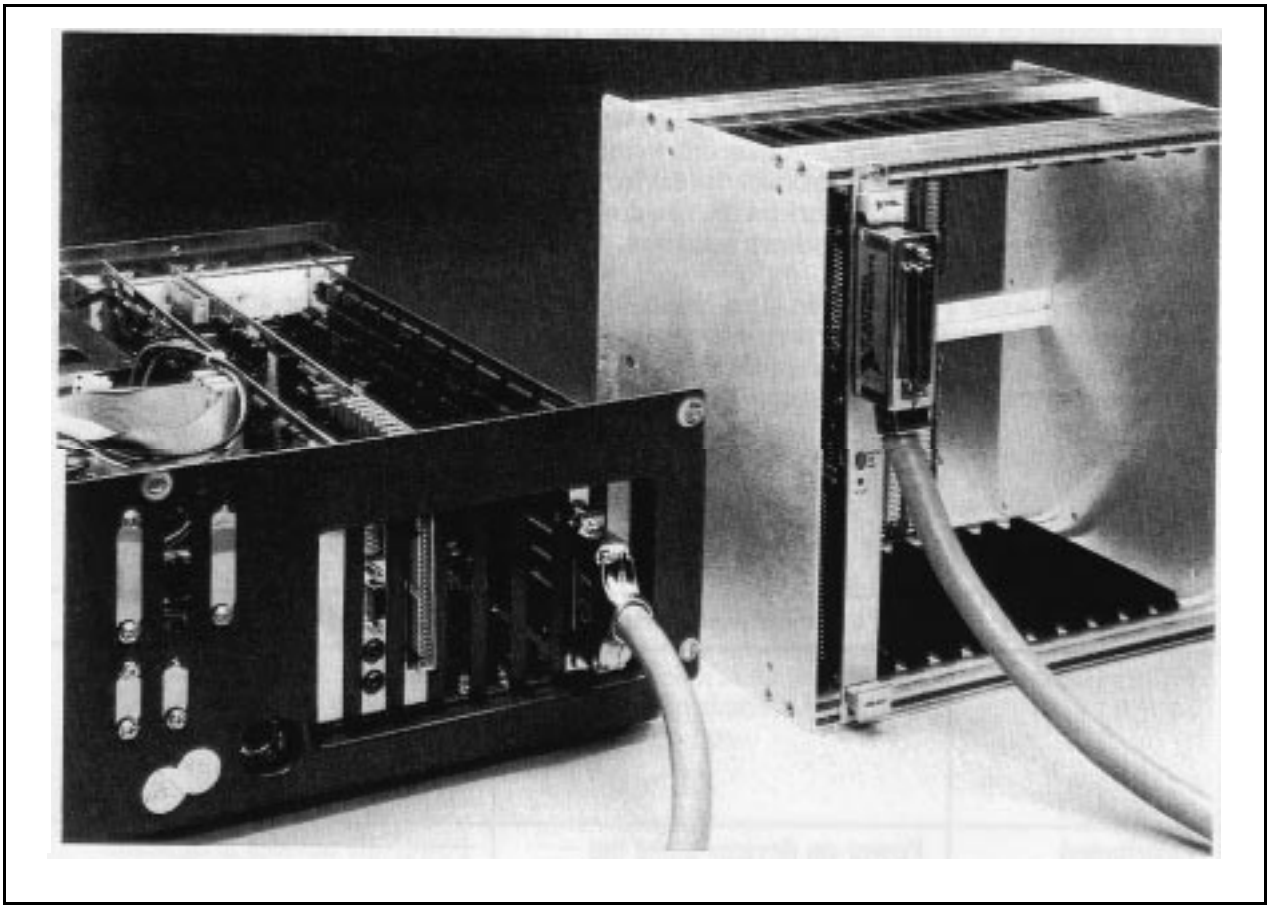


Figure 3-22. MXIbus Dual-Ended Cable Configuration

In a properly configured MXIbus system, the first and last devices in the daisy-chain have only one cable connected to their device connector. MXIbus devices that are neither the first or last device in the daisy-chain have two (and only two) MXIbus cables attached to their device connector.

System Power Cycling Requirements

A distributed architecture such as MXIbus does not have a common power bus or reset signal to ensure that *all* devices within the extended system are initialized at the same time. Therefore, powering-on a device after other devices in the system have already received power and become functional may cause unintended bus activity due to the power-up state of that device. This bus activity could result in a powered, functional MXIbus device making an attempt to respond. If this response initiates bus arbitration, for example, the arbitration mechanism could become hung while waiting for a nonexistent master to assume control of the bus.

To guard against this type of bus activity, you should keep in mind that two types of systems exist with respect to power cycling. The first of these is one in which power is supplied to all devices at roughly the same time as is the case for a system with a master power switch. In such a system, proper operation is guaranteed if the last device to reach 5 volts does so within roughly half of a second of the first device to reach 5 volts. The second type of system is one in which power is applied to each MXIbus device separately. In this type of system, you must power-on devices starting with the device at the *end* of the MXIbus link opposite the MXIbus System Controller and progress towards the MXIbus System Controller. The MXIbus System Controller should be the *last* device on the MXIbus link to receive power. Conversely, when removing system power, you should power-down the device farthest from the MXIbus System Controller *last* so that the termination resistors on the end device ensure that the MXIbus lines remain unasserted throughout the power-down sequence.

When shutting down a system, you must ensure that all devices that could be adversely affected by unintended bus cycles or interrupts have their windows and interrupt mapping disabled. The VME-MXI itself will not be affected during power-down; however, there may be devices in the same frame as the VME-MXI that could be affected.

Table 3-1 summarizes MXIbus system power cycling requirements.

Table 3-1. MXIbus System Power Cycling Requirements

System Type	Power-On Requirements	Power-Off Requirements
Master Power Switch	All devices must receive power within 0.5 seconds of each other.	Disable all A24 and A32 inward mapping to devices that could be adversely affected by an unintended access. Disable all mapping of interrupts.
Distributed	Power-on devices along the MXIbus link beginning with the non-MXIbus System Controller end of the MXIbus link. Make sure the MXIbus System Controller is the last to receive power.	Power-off devices in opposite order of power-on sequence (power off MXIbus System Controller first). Disable all A24 and A32 inward mapping to devices that could be adversely affected by an unintended access. Disable all mapping of interrupts.

Keep in mind that a system can contain only one device acting as the VMEbus Resource Manager (RM). It is important that the RM be run only after all other devices in the system have been powered on. Because many RMs execute automatically upon power-up, you must be sure when working with a distributed system to power-on the device containing the RM last. This implies that any VME-MXIs in that frame must be the MXIbus System Controllers for their respective MXIbus systems to keep the preceding power-on procedures for individual MXIbus systems.

Chapter 4
Register Descriptions

This chapter contains detailed information on the use of the VME-MXI registers, which are used to configure and control the module's operation. All of these configuration registers are accessible from the VMEbus (in the VMEbus configuration space) and from the MXIbus. If you are not writing your own multiframe Resource Manager routines, you can skip over this chapter.

Register Maps

The register map for the VME-MXI configuration registers is shown in Table 4-1 and Figure 4-1. The table gives the register name, the register address, the size of the register in bits, and the type of the register (read only, write only, or read/write). The base address for the VME-MXI configuration space in A16 space is equal to the VMEbus logical address assigned to the VME-MXI shifted left six times and ORed with hex C000.

Register Sizes

The VMEbus supports three different transfer sizes for read/write operations: 8-bit, 16-bit, or 32-bit. Table 4-1 shows the size of the registers on the VME-MXI. All 16-bit registers can be accessed using 8-bit read/write operations.

Register Description Format

Each register bit map shows a diagram of the register with the most significant bit (bit 15 for a 16-bit register, bit 7 for an 8-bit register) shown on the left, and the least significant bit (bit 0) shown on the right. A square is used to represent each bit. Each bit is labeled with a name inside its square. An asterisk (*) after the bit name indicates that the signal is active low. An asterisk is equivalent to an overbar.

Hard and Soft Reset

Each register description indicates whether the bits are cleared by a hard and/or soft reset. A hard reset occurs when the mainframe is powered on and when the VMEbus SYSRESET signal is active. A hard reset clears all the registers on the VME-MXI. A soft reset occurs when the RESET bit in the VMEbus Control Register 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.

Table 4-1. VME-MXI Register Map

Register Name	Offset from Base Address (Hex)	Type	Size
ID Register	0	Read Only	16-bit
Device Type Register	2	Read Only	16-bit
Status/Control Register	4	Read/Write	16-bit
Logical Address Window Register	A	Read/Write	16-bit
A16 Window Map Register	C	Read/Write	16-bit
A24 Window Map Register	E	Read/Write	16-bit
A32 Window Map Register	10	Read/Write	16-bit
INTX Interrupt Configuration *	12	Read/Write	16-bit
INTX Utility Configuration *	18	Read/Write	16-bit
Subclass Register	1E	Read Only	16-bit
MXIbus Status/Control Register	20	Read/Write	16-bit
MXIbus Lock Register	22	Read/Write	16-bit
MXIbus IRQ Configuration Register	24	Read/Write	16-bit
Read LA Register	26	Read Only	16-bit
Interrupt Status/Control Register	2A	Read/Write	16-bit
Status/ID Register	2C	Read/Write	16-bit
Interrupt Acknowledge for IRQ1	32	Read Only	16-bit
Interrupt Acknowledge for IRQ2	34	Read Only	16-bit
Interrupt Acknowledge for IRQ3	36	Read Only	16-bit
Interrupt Acknowledge for IRQ4	38	Read Only	16-bit
Interrupt Acknowledge for IRQ5	3A	Read Only	16-bit
Interrupt Acknowledge for IRQ6	3C	Read Only	16-bit
Interrupt Acknowledge for IRQ7	3E	Read Only	16-bit

* The INTX registers at offsets 12 and 18 are only available on VME-MXIs with the INTX daughter card option. On VME-MXIs without the INTX option, the entire range between offsets 12 and 1C (inclusive) is VME-MXI Reserved Space.

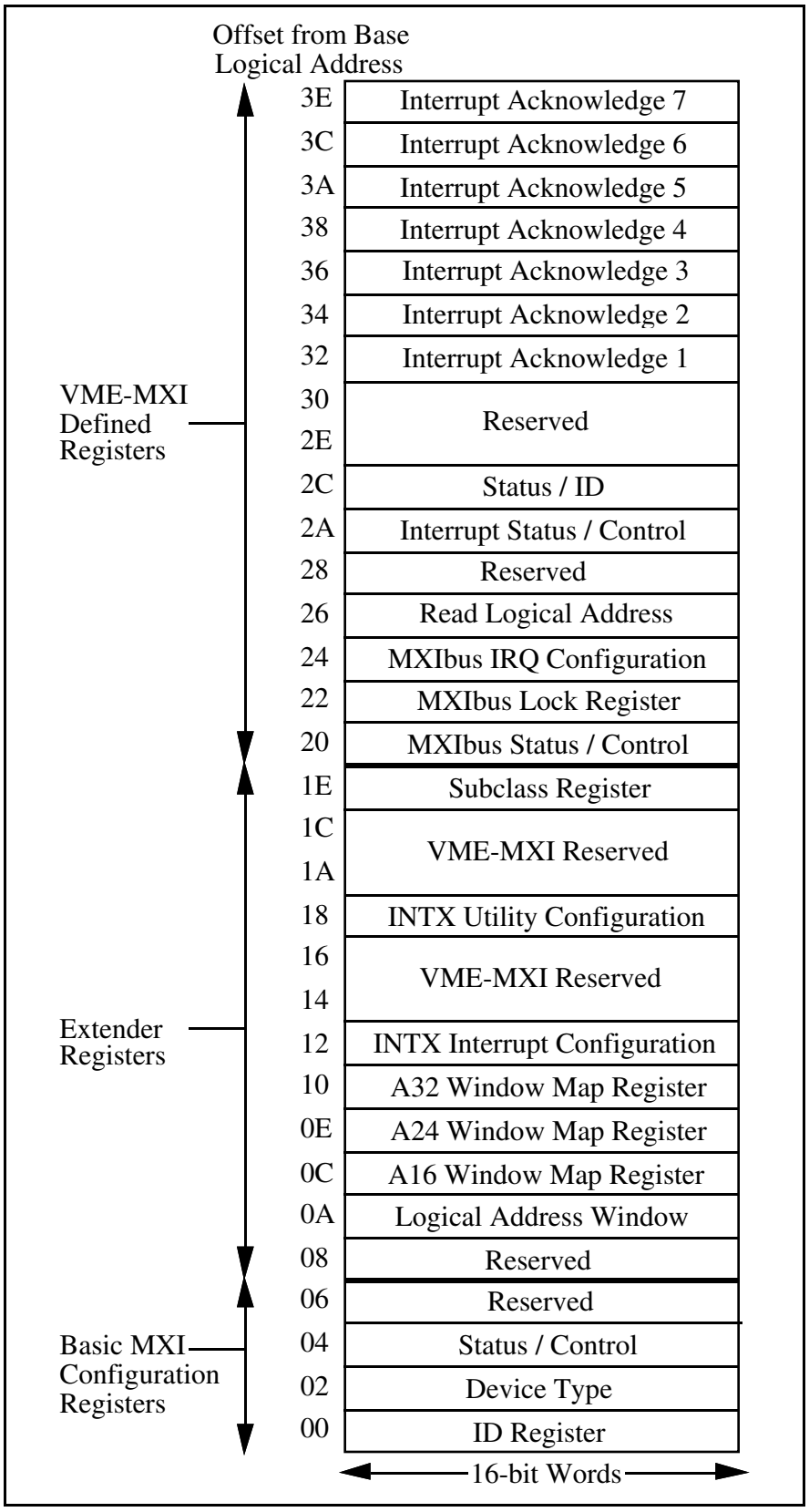


Figure 4-1. VME-MXI Register Map

MXIbus Configuration Registers

These registers are required for all MXIbus devices for identification of the device. These registers are also defined by the VXIbus specification for all VXIbus devices; therefore, the VME-MXI is compatible with the VXIbus specification. The VME-MXI can be installed in either a VMEbus chassis or a VXIbus mainframe.

ID Register

VMEbus Address: Base Address + 0 (hex)

Attributes: Read Only

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	R
0	1	1	1	1	1	1	1	1	1	1	1	0	1	1	0	
DEVCLASS		ADDR		MANID												

This register provides information about this device and its configuration. The bits in this register are configured in hardware as shown above. Hard and soft resets have no effect on this register.

Bit	Mnemonic	Description
15-14r	DEVCLASS	Device Class Bits These bits indicate the device class of the VXIbus/MXIbus device as follows: 00 = Memory 01 = Extended 10 = Message-Based 11 = Register-Based The VME-MXI is an extended device defined by National Instruments; therefore, these bits are configured in hardware as binary 01.
13-12r	ADDR	Address Space Bits These bits indicate the address spaces in which the VXIbus/MXIbus device has operational registers as follows: 00 = A16/A24 01 = A16/A32 10 = Reserved 11 = A16 Only The VME-MXI has operational registers in A16 only; therefore, these bits are configured in hardware as binary 11.

11-0r	MANID	Manufacturer ID Bits This number uniquely identifies the manufacturer of the VXIbus/MXIbus device. These bits are configured in hardware as hex FF6, the VXIbus manufacturer ID number assigned to National Instruments.

Device Type Register

VMEbus Address: Base Address + 2 (hex)

Attributes: Read Only

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

This register indicates how much VMEbus memory is required by this VMEbus device, and identifies this device with a manufacturer's unique model code. The bits in this register are set in hardware to the values shown above. Hard and soft resets have no effect on this register.

Bit	Mnemonic	Description
15-0r	MODEL	Model Code Bits These bits contain a unique number assigned to this device by the manufacturer to identify this device. The model code for the VME-MXI is hex FF1.

Status/Control Register

VMEbus Address: Base Address + 4 (hex)

Attributes: Read/Write

15	14	13	12	11	10	9	8	
1	1	EDTYPE				1	ACCDIR	R
0	0	0	0	0	0	0	0	W

7	6	5	4	3	2	1	0	
VERSION				RDY	PASS	0	RESET	R
0	0	0	0	0	0	0	RESET	W

This register provides status information about this VXIbus/MXIbus device and provides a bit to force the VME-MXI into a Soft Reset state. The RESET bit is cleared on a hard reset. Hard and soft resets have no effect on the other bits on this register.

Bit	Mnemonic	Description
15-14r/w, 13-10w, 9r/w, 8-1w	1	Reserved Bits These bits are reserved and read back as ones. Write a zero when writing to these bits.
13-10r	EDTYPE	Extended Device Type Class Bits These bits are driven low by an optional daughter card installed on the two 96-pin daughter card connectors. They identify the daughter card and its capabilities. When a daughter card is not installed, these bits are all ones. The INTX daughter card has been assigned extended device type class hex E; therefore, when the VME-MXI includes this option, these bits are hex E. These bits are read only.
8r	ACCDIR	Access Direction Bit When this bit is set to one, the current access to the Status register originated from a device on the MXIbus. When this bit is cleared, the current access to the Status register originated from a device on the VMEbus. This bit is read only.

7-4r VERSION VME-MXI Version Number Bits

These bits specify the revision version number of the VME-MXI according the table below. These bits are read only.

Version Number	VME-MXI Revision
Hex F	Revision A
Hex E	Revision B
Hex D	Revision C
Hex C	Revision D

3r RDY Ready Bit

This bit is set to one in hardware to indicate that the device is ready to execute its full functionality. This bit is read only.

2r PASS Passed Bit

This bit is set to one in hardware to indicate that the device is functional. This bit is read only.

1r 0 This bit is reserved and reads back as zero.

0r/w RESET Reset Bit

When this bit is set, the VME-MXI is forced into the Soft Reset state. When this bit is cleared, the VME-MXI is in the normal operation state. This bit is readable and is cleared on a hard reset.

VMEbus Extender Registers

These registers are defined for VXIbus/VMEbus extender devices.

Logical Address Window Register

VMEbus Address: Base Address + A (hex)

Attributes: Read/Write

This register defines the range of logical addresses that are mapped into and out of the VME-MXI through the MXIbus. This register defines a configuration window in the upper 16 kilobytes of A16 space. These bits are cleared on a hard reset.

The CMODE bit in the MXIbus Control Register selects the format of this register. If the CMODE bit is 0 (default), a Base/Size window comparison is used to determine the range of addresses in the window. If the CMODE bit is set, an upper and lower bound is used to determine the range of addresses in the window.

The Logical Address Window Register has the following format when the CMODE bit is cleared:

15	14	13	12	11	10	9	8	R
0	LAEN	LADIR	1	1	LASIZE2	LASIZE1	LASIZE0	
0	LAEN	LADIR	0	0	LASIZE2	LASIZE1	LASIZE0	W
7	6	5	4	3	2	1	0	
LABASE7	LABASE6	LABASE5	LABASE4	LABASE3	LABASE2	LABASE1	LABASE0	R/W

Bit	Mnemonic	Description
15r/w	0	Reserved Bit This bit is reserved and reads back as zero. Write a zero when writing to this bit.
14r/w	LAEN	Logical Address Window Enable Bit When this bit is set, the logical address mapping window is enabled. When this bit is cleared, the logical address mapping window is disabled except for the logical address of this device. Access to the VME-MXI's own configuration space is always enabled.

13r/w LADIR Logical Address Window Direction Bit

When this bit is set, the logical address window applies to MXIbus cycles that are mapped into VMEbus cycles (inward cycles). When this bit is cleared, the logical address window applies to VMEbus cycles that are mapped out into MXIbus cycles (outward cycles). The complement of the defined range is mapped in the opposite direction.

LAEN	LADIR	Window Applies to
0	X	Disabled
1	0	VME cycles to MXI cycles
	1	MXI cycles to VME cycles

12-11r/w 1 Reserved Bits

These bits are reserved and read back as ones. Write a zero when writing to these bits.

10-8r/w LASIZE[2-0] Logical Address Window Size Bits

This 3-bit number specifies the number of significant address bits in the LABASE field that are compared when determining if an address is in the logical address window. The number of logical addresses in the window is 2^{8-i} where *i* is the value of LASIZE [2-0]. Because *i* can range from 0 to 7, the minimum size of a logical address window is 2, and the maximum size is 256.

7-0r/w LABASE[7-0] Logical Address Window Base Address Bits

These bits, in conjunction with the LASIZE bits, define the base address of the Logical Address window for the VME-MXI. The LASIZE bits indicate the number of LABASE bits that are significant. LABASE7 is the most significant, and LABASE0 is the least. The LABASE bits that are not significant can be replaced with zeros to provide the base address of the logical address window.

Logical Address Window Example:

LABASE	LASIZE	Logical Addresses in Window
any value	0	00 to FF
00	1	00 to 7F
08	2	00 to 3F
18	3	00 to 1F
3F	4	30 to 3F
55	5	50 to 57
88	6	88 to 8B
CC	7	CC to CD
AA	0	00 to FF
AA	1	80 to FF
AA	2	80 to BF
AA	3	A0 to BF
AA	4	A0 to AF
AA	5	A8 to AF
AA	6	A8 to AB
AA	7	AA to AB

The Logical Address Window Register has the following format when the CMODE bit is set:

15	14	13	12	11	10	9	8	
LAHIGH7	LAHIGH6	LAHIGH5	LAHIGH4	LAHIGH3	LAHIGH2	LAHIGH1	LAHIGH0	R/W

7	6	5	4	3	2	1	0	
LALOW7	LALOW6	LALOW5	LALOW4	LALOW3	LALOW2	LALOW1	LALOW0	R/W

Bit	Mnemonic	Description
15-8r/w	LAHIGH[7-0]	Logical Address Window Upper Bound Bits These bits define the upper limit of the range of MXIbus logical addresses that map into the VMEbus.
7-0r/w	LALOW[7-0]	Logical Address Window Lower Bound Bits These bits define the lower limit of the range of MXIbus logical addresses that map into the VMEbus.

This register defines the range of MXIbus logical addresses that map into the VMEbus where that range is:

$$LAHIGH > range \geq LALOW$$

The MXIbus logical addresses mapped out of the VME-MXI are the inverse of this range, that is, MXIbus logical addresses greater than or equal to the LAHIGH value or less than the LALOW value.

To map a consecutive range of MXIbus logical addresses out of the VME-MXI, the lower bound of the range must be placed in the LAHIGH field and the upper bound in the LALOW field. In this case, the range of MXIbus logical addresses mapped out of the VME-MXI is:

$$LALOW > range \geq LAHIGH$$

The MXIbus logical addresses mapped into the VMEbus are the inverse of this range, that is, MXIbus logical addresses greater than or equal to the LALOW value or less than the LAHIGH value.

The window is disabled whenever LAHIGH = LALOW = 0. All MXIbus logical addresses are mapped out to the MXIbus when:

$$FF\ (hex) \geq (LAHIGH = LALOW) \geq 80\ (hex)$$

All MXIbus logical addresses are mapped into the VMEbus when:

$$7F\ (hex) \geq (LAHIGH = LALOW) > 0$$

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

A16 Window Map Register

VMEbus Address: Base Address + C (hex)

Attributes: Read/Write

This register defines the range of addresses in the lower 48 kilobytes of A16 space that is mapped into and out of the VME-MXI through the MXIbus. These bits are cleared on a hard reset.

The CMODE bit in the MXIbus Control Register selects the format of this register. If the CMODE bit is 0 (default), a Base/Size window comparison is used to determine the range of addresses in the window. If the CMODE bit is set, an upper and lower bound is used to determine the range of addresses in the window.

The A16 Window Map Register has the following format when the CMODE bit is cleared:

15	14	13	12	11	10	9	8	
0	A16EN	A16DIR	1	1	A16SIZE2	A16SIZE1	A16SIZE0	R
0	A16EN	A16DIR	0	0	A16SIZE2	A16SIZE1	A16SIZE0	W

7	6	5	4	3	2	1	0	
A16BASE7	A16BASE6	A16BASE5	A16BASE4	A16BASE3	A16BASE2	A16BASE1	A16BASE0	R/W

Bit	Mnemonic	Description
15r/w	0	Reserved Bit This bit is reserved and reads back as zero. Write a zero when writing to this bit.
14r/w	A16EN	A16 Window Enable Bit When this bit is set, the A16 mapping window is enabled. When this bit is cleared, the A16 mapping window is disabled.
13r/w	A16DIR	A16 Window Direction Bit When this bit is set, the A16 window applies to MXIbus cycles that are mapped into VMEbus cycles (inward cycles). When this bit is cleared, the A16 window applies to VMEbus cycles that are mapped out into MXIbus cycles (outward cycles). The complement of the defined range is mapped in the opposite direction.

A16EN	A16DIR	Window Applies to
0	X	Disabled
1	0	VME cycles to MXI cycles
	1	MXI cycles to VME cycles

12-11r/w 1 Reserved Bits

These bits are reserved and read back as ones. Write a zero when writing to these bits.

10-8r/w A16SIZE[2-0] A16 Window Size Bits

This 3-bit number specifies the number of significant address bits in the A16BASE field that are compared when determining if an address is in the A16 window. The number of A16 addresses in the window is $256 * 2^{8-i}$ where *i* is the value of A16SIZE[2-0]. The minimum size of an A16 window is 512 bytes and the maximum size is 48 kilobytes (A16SIZE = 0).

7-0r/w A16BASE[7-0] A16 Window Base Address Bits

These bits, in conjunction with the A16SIZE bits, define the base address of the A16 window for the VME-MXI. The A16SIZE bits indicate the number of A16BASE bits that are significant. A16BASE7 is the most significant and A16BASE0 is the least. The A16BASE bits that are not significant can be replaced with zeros to provide the base address of the A16 window.

A16 Window Example:

A16BASE	A16SIZE	A16 Addresses in Window
any value	0	0000 to BFFF
3F	1	0000 to 7FFF
80	2	8000 to BFFF
26	3	2000 to 3FFF
49	4	4000 to 4FFF
98	5	9800 to 9FFF
42	6	4000 to 43FF
A1	7	A000 to A1FF
55	0	0000 to BFFF
55	1	0000 to 7FFF
55	2	4000 to 7FFF
55	3	4000 to 5FFF
55	4	5000 to 5FFF
55	5	5000 to 57FF
55	6	5400 to 57FF
55	7	5400 to 55FF

The A16 Window Map Register has the following format when the CMODE bit is set:

15	14	13	12	11	10	9	8	
A16HIGH7	A16HIGH6	A16HIGH5	A16HIGH4	A16HIGH3	A16HIGH2	A16HIGH1	A16HIGH0	R/W
7	6	5	4	3	2	1	0	
A16LOW7	A16LOW6	A16LOW5	A16LOW4	A16LOW3	A16LOW2	A16LOW1	A16LOW0	R/W

Bit	Mnemonic	Description
15-8r/w	A16HIGH[7-0]	A16 Window Upper Bound Bits These bits define the upper limit of the range of MXIbus A16 addresses that map into the VMEbus.
7-0r/w	A16LOW[7-0]	A16 Window Lower Bound Bits These bits define the lower limit of the range of MXIbus A16 addresses that map into the VMEbus.

This register defines the range of MXIbus A16 addresses that map into the VMEbus where that range is:

$$A16HIGH > range \geq A16LOW$$

The VMEbus A16 addresses mapped out of the VME-MXI are the inverse of this range, that is, MXIbus A16 addresses greater than or equal to the A16HIGH value or less than the A16LOW value.

To map a consecutive range of VMEbus A16 addresses out of the VME-MXI, the lower bound of the range must be placed in the A16HIGH field and the upper bound in the A16LOW field. In this case, the range of VMEbus A16 addresses mapped out of the VME-MXI is:

$$A16LOW > range \geq A16HIGH$$

The MXIbus A16 addresses mapped into the VMEbus are the inverse of this range, that is, VMEbus A16 addresses greater than or equal to the A16LOW value or less than the A16HIGH value.

The window is disabled whenever A16HIGH = A16LOW = 0. All VMEbus A16 addresses are mapped out to the MXIbus when:

$$FF\ (hex) \geq (A16HIGH = A16LOW) \geq 80\ (hex)$$

All MXIbus A16 addresses are mapped into the VMEbus when:

$$7F\ (hex) \geq (A16HIGH = A16LOW) > 0$$

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

A24 Window Map Register

VMEbus Address: Base Address + E (hex)

Attributes: Read/Write

This register defines the range of addresses in A24 space that are mapped into and out of the VME-MXI through the MXIbus. These bits are cleared on a hard reset.

The CMODE bit in the MXIbus Control Register selects the format of this register. If the CMODE bit is 0 (default), a Base/Size window comparison is used to determine the range of addresses in the window. If the CMODE bit is set, an upper and lower bound is used to determine the range of addresses in the window.

The A24 Window Map Register has the following format when the CMODE bit is cleared:

15	14	13	12	11	10	9	8	
0	A24EN	A24DIR	1	1	A24SIZE2	A24SIZE1	A24SIZE0	R
0	A24EN	A24DIR	0	0	A24SIZE2	A24SIZE1	A24SIZE0	W

7	6	5	4	3	2	1	0	
A24BASE7	A24BASE6	A24BASE5	A24BASE4	A24BASE3	A24BASE2	A24BASE1	A24BASE0	R/W

Bit	Mnemonic	Description
15r/w	0	Reserved Bit This bit is reserved and reads back as zero. Write a zero when writing to this bit.
14r/w	A24EN	A24 Window Enable Bit When this bit is set, the A24 mapping window is enabled. When this bit is cleared, the A24 mapping window is disabled.
13r/w	A24DIR	A24 Window Direction Bit When this bit is set, the A24 window applies to MXIbus cycles that are mapped into VMEbus cycles (inward cycles). When this bit is cleared, the A24 window applies to VMEbus cycles that are mapped out into MXIbus cycles (outward cycles). The complement of the defined range is mapped in the opposite direction.

A24EN	A24DIR	Window Applies to
0	X	Disabled
1	0	VME cycles to MXI cycles
	1	MXI cycles to VME cycles

12-11r/w 1 Reserved Bits

These bits are reserved and read back as ones. Write a zero when writing to these bits.

10-8r/w A24SIZE[2-0] A24 Window Size Bits

This 3-bit number specifies the number of significant address bits in the A24BASE field that are compared when determining if an address is in the A24 window. The number of A24 addresses in the window is $65536 * 2^{8-i}$ where *i* is the value of A24SIZE[2-0]. The minimum size of an A24 window is 128 kilobytes, and the maximum size is 16 megabytes.

7-0r/w A24BASE[7-0] A24 Window Base Address Bits

These bits, in conjunction with the A24SIZE bits, define the base address of the A24 window for the VME-MXI. The A24SIZE bits indicate the number of A24BASE bits that are significant. A24BASE7 is the most significant and A24BASE0 is the least. The A24BASE bits that are not significant can be replaced with zeros to provide the base address of the A24 window.

A24 Window Example:

A24BASE	A24SIZE	A24 Addresses in Window
any value	0	000000 to FFFFFF
4E	1	000000 to 7FFFFF
A7	2	800000 to BFFFFF
35	3	200000 to 3FFFFF
6C	4	600000 to 6FFFFF
81	5	800000 to 87FFFF
B4	6	B40000 to B7FFFF
02	7	020000 to 03FFFF
00	0	000000 to FFFFFF
00	1	000000 to 7FFFFF
00	2	000000 to 3FFFFF
00	3	000000 to 1FFFFF
00	4	000000 to 0FFFFF
00	5	000000 to 07FFFF
00	6	000000 to 03FFFF
00	7	000000 to 01FFFF

The A24 Window Map Register has the following format when the CMODE bit is set:

15	14	13	12	11	10	9	8	
A24HIGH7	A24HIGH6	A24HIGH5	A24HIGH4	A24HIGH3	A24HIGH2	A24HIGH1	A24HIGH0	R/W
7	6	5	4	3	2	1	0	
A24LOW7	A24LOW6	A24LOW5	A24LOW4	A24LOW3	A24LOW2	A24LOW1	A24LOW0	R/W

Bit	Mnemonic	Description
15-8r/w	A24HIGH[7-0]	A24 Window Upper Bound These bits define the upper limit of the range of MXIbus A24 addresses that map into the VMEbus.
7-0r/w	A24LOW[7-0]	A24 Window Lower Bound These bits define the lower limit of the range of MXIbus A24 addresses that map into the VMEbus.

This register defines the range of MXIbus A24 addresses that map into the VMEbus where that range is:

$$A24HIGH > range \geq A24LOW$$

The VMEbus A24 addresses mapped out of the VME-MXI are the inverse of this range, that is, MXIbus A24 addresses greater than or equal to the A24HIGH value or less than the A24LOW value.

To map a consecutive range of VMEbus A24 addresses out of the VME-MXI, the lower bound of the range must be placed in the A24HIGH field and the upper bound in the A24LOW field. In this case the range of VMEbus A24 addresses mapped out of the VME-MXI is:

$$A24LOW > range \geq A24HIGH$$

The MXIbus A24 addresses mapped into the VMEbus are the inverse of this range, that is, VMEbus A24 addresses greater than or equal to the A24LOW value or less than the A24HIGH value.

The window is disabled whenever A24HIGH = A24LOW = 0. All VMEbus A24 addresses are mapped out to the MXIbus when:

$$FF\ (hex) \geq (A24HIGH = A24LOW) \geq 80\ (hex)$$

All MXIbus A24 addresses are mapped into the VMEbus when:

$$7F\ (hex) \geq (A24HIGH = A24LOW) > 0$$

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

A32 Window Map Register

VMEbus Address: Base Address + 10 (hex)

Attributes: Read/Write

This register defines the range of addresses in A32 space that are mapped into and out of the VME-MXI through the MXIbus. These bits are cleared on a hard reset.

The CMODE bit in the MXIbus Control Register selects the format of this register. If the CMODE bit is 0 (default), a Base/Size window comparison is used to determine the range of addresses in the window. If the CMODE bit is set, an upper and lower bound is used to determine the range of addresses in the window.

The A32 Window Map Register has the following format when the CMODE bit is cleared:

15	14	13	12	11	10	9	8	
0	A32EN	A32DIR	1	1	A32SIZE2	A32SIZE1	A32SIZE0	R
0	A32EN	A32DIR	0	0	A32SIZE2	A32SIZE1	A32SIZE0	W

7	6	5	4	3	2	1	0	
A32BASE7	A32BASE6	A32BASE5	A32BASE4	A32BASE3	A32BASE2	A32BASE1	A32BASE0	R/W

Bit	Mnemonic	Description
15r/w	0	Reserved Bit This bit is reserved and reads back as zero. Write a zero when writing to this bit.
14r/w	A32EN	A32 Window Enable Bit When this bit is set, the A32 mapping window is enabled. When this bit is cleared, the A32 mapping window is disabled.
13r/w	A32DIR	A32 Window Direction Bit When this bit is set, the A32 window applies to MXIbus cycles that are mapped into VMEbus cycles (inward cycles). When this bit is cleared, the A32 window applies to VMEbus cycles that are mapped out into MXIbus cycles (outward cycles). The complement of the defined range is mapped in the opposite direction.

A32EN	A32DIR	Window Applies to
0	X	Disabled
1	0	VME cycles to MXI cycles
	1	MXI cycles to VME cycles

12-11r/w 1 Reserved Bits

These bits are reserved and read back as ones. Write a zero when writing to these bits.

10-8r/w A32SIZE[2-0] A32 Window Size Bits

This 3-bit number specifies the number of significant address bits in the A32BASE field that are compared when determining if an address is in the A32 window. The number of A32 addresses in the window is 16,777,216 * 2^{8-i} where *i* is the value of A32SIZE[2-0]. The minimum size of an A32 window is 32 megabytes, and the maximum size is 4 gigabytes.

7-0r/w A32BASE[7-0] A32 Window Base Address Bits

These bits, in conjunction with the A32SIZE bits, define the base address of the A32 window for the VME-MXI. The A32SIZE bits indicate the number of A32BASE bits that are significant. A32BASE7 is the most significant and A32BASE0 is the least. The A32BASE bits that are not significant can be replaced with zeros to provide the base address of the A32 window.

A32 Window Example:

A32BASE	A32SIZE	A32 Addresses in Window
any value	0	00000000 to FFFFFFFF
C8	1	80000000 to FFFFFFFF
3E	2	00000000 to 3FFFFFFF
49	3	40000000 to 5FFFFFFF
9A	4	90000000 to 9FFFFFFF
21	5	20000000 to 27FFFFF
75	6	74000000 to 77FFFFF
19	7	18000000 to 19FFFFF
FF	0	00000000 to FFFFFFFF
FF	1	80000000 to FFFFFFFF
FF	2	C0000000 to FFFFFFFF
FF	3	E0000000 to FFFFFFFF
FF	4	F0000000 to FFFFFFFF
FF	5	F8000000 to FFFFFFFF
FF	6	FC000000 to FFFFFFFF
FF	7	FE000000 to FFFFFFFF

The A32 Window Map Register has the following format when the CMODE bit is set:

15	14	13	12	11	10	9	8	
A32HIGH7	A32HIGH6	A32HIGH5	A32HIGH4	A32HIGH3	A32HIGH2	A32HIGH1	A32HIGH0	R/W
7	6	5	4	3	2	1	0	
A32LOW7	A32LOW6	A32LOW5	A32LOW4	A32LOW3	A32LOW2	A32LOW1	A32LOW0	R/W

Bit	Mnemonic	Description
15-8r/w	A32HIGH[7-0]	A32 Window Upper Bound These bits define the upper limit of the range of MXIbus A32 addresses that map into the VMEbus.
7-0r/w	A32LOW[7-0]	A32 Window Lower Bound These bits define the lower limit of the range of MXIbus A32 addresses that map into the VMEbus.

This register defines the range of MXIbus A32 addresses that map into the VMEbus where that range is:

$$A32HIGH > range \geq A32LOW$$

The VMEbus A32 addresses mapped out of the VME-MXI are the inverse of this range, that is, MXIbus A32 addresses greater than or equal to the A32HIGH value or less than the A32LOW value.

To map a consecutive range of VMEbus A32 addresses out of the VME-MXI, the lower bound of the range must be placed in the A32HIGH field and the upper bound in the A32LOW field. In this case, the range of VMEbus A32 addresses mapped out of the VME-MXI is:

$$A32LOW > range \geq A32HIGH$$

The MXIbus A32 addresses mapped into the VMEbus are the inverse of this range, that is, VMEbus A32 addresses greater than or equal to the A32LOW value or less than the A32HIGH value.

The window is disabled whenever A32HIGH = A32LOW = 0. All VMEbus A32 addresses are mapped out to the MXIbus when:

$$FF\ (hex) \geq (A32HIGH = A32LOW) \geq 80\ (hex)$$

All MXIbus A32 addresses are mapped into the VMEbus when:

$$7F\ (hex) \geq (A32HIGH = A32LOW) > 0$$

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

INTX Interrupt Configuration Register (on VME-MXIs with INTX only)

VMEbus Address: Base Address + 12 (hex)

Attributes: Read/Write

15	14	13	12	11	10	9	8	
0	EINT7EN	EINT6EN	EINT5EN	EINT4EN	EINT3EN	EINT2EN	EINT1EN	R/W
7	6	5	4	3	2	1	0	
0	EINT7DIR	EINT6DIR	EINT5DIR	EINT4DIR	EINT3DIR	EINT2DIR	EINT1DIR	R/W

This register on the INTX daughter card is used to configure the mapping of the seven VMEbus interrupts lines to and from the seven INTX interrupt lines.

Bit **Mnemonic** **Description**

15,7r/w 0 Zero Bits

These bits read back as zero to indicate that this register contains the extended interrupt mapping bits. Writes to these bits have no effect.

14-8r/w EINT[7-1]EN Extended Interrupt Enable Bits

Setting these bits individually enables the corresponding VMEbus IRQ lines to drive or receive the corresponding INTX interrupt line. The corresponding EINTDIR bits select whether the INTX interrupt line is driven or received by the VMEbus IRQ line. These bits are cleared on a hard reset.

6-0r/w EINT[7-1]DIR Extended Interrupt Direction Bits

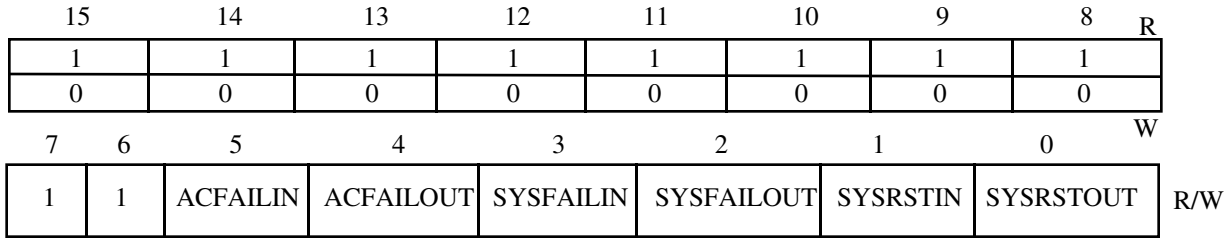
When the corresponding EINT*x*EN bits are clear, these bits have no meaning. When the corresponding EINT*x*EN bits are set, these bits control the routing of the INTX IRQ signals. When EINT*x*DIR is clear, the corresponding VMEbus IRQ line drives the INTX IRQ line. If the EINT*x*DIR bit is set, the INTX IRQ line drives the corresponding VMEbus IRQ line.

EINT*x*EN	EINT*x*DIR	Routing
0	X	Disabled
1	0	VME IRQ *X* drives INTX IRQ
	1	INTX IRQ drives VME IRQ *X*

INTX Utility Configuration Register (on VME-MXIs with INTX only)

VMEbus Address: Base Address + 18 (hex)

Attributes: Read/Write



This register on the INTX daughter card is used to configure the mapping of the three VMEbus reset signals to and from the three corresponding INTX reset signals.

Bit	Mnemonic	Description
15-6r/w,	1	Reserved Bits These bits are reserved and read back as ones. Write zeros to these bits when writing to this register.
5r/w	ACFAILIN	Extended ACFAIL Inward Bit Setting this bit enables the INTX ACFAIL line to be mapped onto the VMEbus ACFAIL line. Clearing this bit disables the mapping of the INTX ACFAIL line onto the VMEbus ACFAIL line. This bit is cleared on power-up.
4r/w	ACFAILOUT	Extended ACFAIL Outward Bit Setting this bit enables the VMEbus ACFAIL line to be mapped out onto the INTX ACFAIL line. Clearing this bit disables the mapping of the ACFAIL line onto the INTX ACFAIL line. This bit is cleared on power-up.
3r/w	SYSFAILIN	Extended SYSFAIL Inward Bit Setting this bit enables the INTX SYSFAIL line to be mapped in onto the VMEbus SYSFAIL line. Clearing this bit disables the mapping of the INTX SYSFAIL line onto the VMEbus SYSFAIL line. This bit is cleared on power-up.
2r/w	SYSFAILOUT	Extended SYSFAIL Outward Bit Setting this bit enables the VMEbus SYSFAIL line to be mapped out onto the INTX SYSFAIL line. Clearing this bit disables the mapping of the SYSFAIL line onto the INTX SYSFAIL line. This bit is cleared on power-up.

1r/w	SYSRSTIN	Extended SYSRESET Inward Bit Setting this bit enables the INTX SYSRESET line to be mapped in onto the VMEbus SYSRESET line. Clearing this bit disables the mapping of the INTX SYSRESET line onto the VMEbus SYSRESET line. This bit is cleared on power-up.
0r/w	SYSRSTOUT	Extended SYSRESET Outward Bit Setting this bit enables the VMEbus SYSRESET line to be mapped out onto the INTX SYSRESET line. Clearing this bit disables the mapping of the SYSRESET line onto the INTX SYSRESET line. This bit is cleared on power-up.

Subclass Register

VMEbus Address: Base Address + 1E (hex)

Attributes: Read only

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

These bits define the subclass of an extended device. The VME-MXI can be used as a VXIbus Mainframe Extender. Such devices are assigned the subclass code hex FFFC. Hard and soft resets have no effect on this register.

Bit	Mnemonic	Description
15-0r	SUBCLASS	Manufacturer Subclass These bits indicate the subclass code for the VME-MXI. These bits are configured in hardware as hex FFFC.

MXIbus Defined Registers

MXIbus Status/Control Register

VMEbus Address: Base Address + 20 (hex)

Attributes: Read/Write

15	14	13	12	11	10	9	8	
RMWMODE	CMODE	1	1	MXSCTO	INTLCK	DSYSFAIL	FAIR	R
RMWMODE	CMODE	0	0	0	0	DSYSFAIL	DSYSRST	W

7	6	5	4	3	2	1	0	
MXISC	0	MXSRSTINT	MXACFAILINT	LNGMXSCTO	MXBERR	MXSYSFINT	PARERR	R
0	0	MXSRSTEN	MXACFAILEN	LNGMXSCTO	0	0	0	W

This register contains status and control bits for various types of MXIbus operators.

Bit	Mnemonic	Description
15r/w	RMWMODE	Read/Modify Write Select Mode Bit This bit, along with the MXIbus Address Modifiers, selects how the VME-MXI will treat a MXIbus cycle when the MXIbus Address Strobe is held low for multiple data transfers. This bit is cleared on hard and soft resets. If the MXIbus address modifiers label the transfer for block mode, the MXIbus block-mode transfer is converted to a VMEbus block-mode transfer irrespective of the RMWMODE bit. If this bit is cleared and the MXIbus address modifiers do not label the transfer for block mode, the MXIbus cycle is interpreted as a RMW (Read/Modify/Write) cycle, which is then converted into a VMEbus RMW cycle. If this bit is set and the MXIbus address modifiers do not label the transfer for block mode, the MXIbus cycle is interpreted as a block transfer and is converted into single transfer VMEbus accesses. This mode should be used when transferring large amounts of data with MXIbus block mode to a VMEbus device that does not support block mode. The following table summarizes the RMWMODE function.

MXI Address Modifiers	RMWMODE Bit	Routing
Block	X	MXIbus block to VMEbus block
Non-Block	0	MXIbus RMW cycle to VMEbus RMW cycle
	1	MXIbus block to VMEbus single cycle

14r/w CMODE Comparison Mode Bit

This bit selects the range comparison mode for the logical address, A16, A24, and A32 Window Mapping Registers. If CMODE is cleared, a Base/Size range comparison is used to determine the range of addresses in the windows. If CMODE is set, an upper and lower bound is used to determine the range of addresses in the windows. This bit is cleared on hard and soft resets.

13-12r/w 1
11-10w,
7w, 2-0w Reserved Bits

These bits are reserved and read back as ones. Write a zero when writing to these bits.

11r MXSCTO MXIbus System Controller Timeout Status Bit

If this VME-MXI is the MXIbus System Controller, this bit is set if the VME-MXI sent a MXIbus BERR on the last MXIbus transfer in response to a MXIbus System Controller Timeout. This bit is cleared when this register is read and on hard and soft resets.

10r INTLCK VME-MXI Interlocked Bus Operation Status Bit

When this bit is set, the VME-MXI is configured to operate in interlocked bus mode. This mode of operation prevents deadlocks by allowing only one master of the entire system (VMEbus and MXIbus) at any given time. When this bit is cleared, the VME-MXI is configured to operate in normal mode. INTLCK is selected with slide switch S2. This bit is not affected by hard or soft resets.

9r/w DSYSFAIL Drive SYSFAIL Bit

When this bit is set, the VME-MXI is driving the VMEbus SYSFAIL line active. When this bit is cleared, the VME-MXI is not asserting the SYSFAIL line. This bit is cleared on hard and soft reset.

8r	FAIR	VME-MXI Fairness Status Bit When this bit is set, the VME-MXI is configured as a fair MXIbus requester. If this bit is cleared, the VME-MXI is configured as an unfair MXIbus requester. FAIR is selected with slide switch S3. This bit is not affected by hard or soft resets.
8w	DSYSRST	Drive SYSRESET line Bit Setting this bit will cause the VMEbus SYSRESET line to pulse asserted for a minimum of 200 msec. This bit is automatically cleared after the assertion of SYSRESET.
7r	MXISC	MXIbus System Controller Status Bit When this bit is set, the VME-MXI is configured as the MXIbus System Controller. When this bit is cleared, the VME-MXI is not configured as the MXIbus System Controller. MXISC is selected with slide switch S4. This bit is not affected by hard or soft resets.
6r/w	0	Reserved Bits This bit is reserved and reads back as zero. Write a zero when writing to this bit.
5r	MXSRSTINT	MXIbus SYSRESET Status Bit When this bit is set, the VMEbus SYSRESET line is active and is being driven across the MXIbus IRQ line. When this bit is cleared, the SYSRESET signal is not driving the MXIbus IRQ line. This bit is cleared on a hard reset.
5w	MXSRSTEN	MXIbus SYSRESET Enable Bit Setting this bit enables the VMEbus SYSRESET line to be driven across the MXIbus IRQ line. When this bit is cleared, the VMEbus SYSRESET line is not mapped to the MXIbus IRQ line. This bit is cleared on a hard reset.
4r	MXACFAILINT	MXIbus ACFAIL Status Bit When this bit is set, the VMEbus ACFAIL line is active and is being driven across the MXIbus IRQ line. When this bit is cleared, the ACFAIL signal is not driving the MXIbus IRQ line. This bit is cleared on a hard reset.
4w	MXACFAILEN	MXIbus ACFAIL Enable Bit Setting this bit enables the VMEbus ACFAIL line to be driven across the MXIbus IRQ line. When this bit is cleared, the VMEbus ACFAIL line is not mapped to the MXIbus IRQ line. This bit is cleared on a hard reset.

3r/w	LNGMXSCTO	Long MXIbus System Controller Timeout Bit When the VME-MXI powers on, this bit is cleared and, if the VME-MXI is the MXIbus System Controller, the MXIbus System Controller timeout is between 100 and 400 µsec (selected by jumper W7). When this bit is set, a longer MXIbus System Controller timeout value is used (a value between 10 and 40 msec) if the VME-MXI is the MXIbus System Controller. This bit is cleared on a hard reset.
2r	MXBERR	MXIbus Bus Error Bit If this bit is set, the VME-MXI terminated the previous MXIbus transfer by driving the MXIbus BERR line. This bit is cleared on hard and soft reset and on successful MXIbus transfers.
1r	MXSYSFINT	MXIbus SYSFAIL Status Bit When this bit is set, the VMEbus SYSFAIL line is active and is being driven across the MXIbus IRQ line. The VMEbus SYSFAIL line is enabled to drive the MXIbus IRQ line with the SYSFOUT bit in the MXIbus IRQ Configuration Register. When this bit is cleared, the SYSFAIL signal is not driving the MXIbus IRQ line. This bit is cleared on a hard reset.
0r	PARERR	Parity Error Bit 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 on hard and soft resets and on MXIbus transfers without a parity error.

MXIbus Lock Register

VMEbus Address: Base Address + 22 (hex)

Attributes: Read/Write

15	14	13	12	11	10	9	8	
1	1	1	1	1	1	1	1	R
0	0	0	0	0	0	0	0	W

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

The bit in this register performs differently depending on whether it was accessed by the VMEbus or the MXIbus. This register is cleared on hard and soft resets.

Bit	Mnemonic	Description
15-1r/w	1	Reserved Bits These bits are reserved and read back as ones. Write a zero when writing to these bits.
0r/w	LOCKED	Lock MXIbus or VMEbus Bit When this bit is set by a VMEbus device, the MXIbus is locked by that device as soon as the MXIbus is won by the VME-MXI. When the MXIbus is locked, indivisible operations to remote resources can be performed across the MXIbus. When this bit is set by a device from across the MXIbus, 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 one, it indicates that the MXIbus is locked. When a MXIbus device reads this bit as a one, it indicates that the VMEbus is locked.

MXIbus IRQ Configuration Register

VMEbus Address: Base Address + 24 (hex)

Attributes: Read/Write

15	14	13	12	11	10	9	8	
SYSFOUT	MIRQ7EN	MIRQ6EN	MIRQ5EN	MIRQ4EN	MIRQ3EN	MIRQ2EN	MIRQ1EN	R/W
7	6	5	4	3	2	1	0	
SYSFIN	MIRQ7DIR	MIRQ6DIR	MIRQ5DIR	MIRQ4DIR	MIRQ3DIR	MIRQ2DIR	MIRQ1DIR	R/W

This register either maps the MXIbus IRQ line onto a VMEbus IRQ line, or maps a VMEbus IRQ line onto the MXIbus IRQ line. These bits are cleared on a hard reset.

Bit	Mnemonic	Description
15r/w	SYSFOUT	SYSFAIL Output Enable Bit Setting this bit enables the VMEbus SYSFAIL line to be routed onto the MXIbus IRQ line. When this bit is cleared, the SYSFAIL line is not mapped to the MXIbus IRQ line.
14-8r/w	MIRQ[7-1]EN	MXIbus IRQ Enable Bits Setting these bits individually enables the corresponding VMEbus IRQ lines to drive or receive the MXIbus IRQ interrupt line. The corresponding MIRQDIR bits select whether the MXIbus IRQ interrupt line is driven or received by the VMEbus IRQ line.
7r/w	SYSFIN	SYSFAIL Input Enable Bit Setting this bit enables the MXIbus IRQ line to be driven on the VMEbus SYSFAIL line. When this bit is cleared, the MXIbus IRQ line is not mapped onto the SYSFAIL line.
6-0r/w	MIRQ[7-1]DIR	MXIbus IRQ Direction Bits When the corresponding MIRQ*x*EN bits are clear, these bits have no meaning. When the corresponding MIRQ*x*EN bits are set, these bits control the routing of the MXIbus IRQ signal. When MIRQ*x*DIR is clear, the corresponding VMEbus IRQ line drives the MXIbus IRQ line. If multiple VMEbus IRQ lines are enabled to drive the MXIbus IRQ line, the selected VMEbus IRQ lines are ORed together and the result drives the MXIbus IRQ line. If the MIRQ*x*DIR bit is set, the MXIbus IRQ line drives the corresponding VMEbus IRQ line.

MIRQ*x*EN	MIRQ*x*DIR	Routing
0	X	Disabled
1	0	VME IRQ *X* drives MXI IRQ
	1	MXI IRQ drives VME IRQ *X*

Read LA Register

VMEbus Address: Base Address + 26 (hex)

Attributes: Read Only

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

7	6	5	4	3	2	1	0	R
LADD7	LADD6	LADD5	LADD4	LADD3	LADD2	LADD1	LADD0	

This register provides the logical address of the VME-MXI. Hard and soft resets have no effect on this register.

Bit	Mnemonic	Description
15-8r	1	Reserved Bits These bits are reserved and read back as ones.
7-0r	LADD[7-0]	Logical Address Status Bits Reading these bits returns the logical address of this VME-MXI. The logical address is selected with the DIP switch located at U94.

Interrupt Status/Control Register

VMEbus Address: Base Address + 2A (hex)

Attributes: Read/Write

15	14	13	12	11	10	9	8	
LINT3	LINT2	LINT1	ACFAILINT	BKOFF	0	SYSFAIL	ACFAIL	R
LINT3	LINT2	LINT1	0	BKOFFIE	0	SYSFAILIE	ACFAILIE	W

7	6	5	4	3	2	1	0	
SYSFAILINT	IRQ7	IRQ6	IRQ5	IRQ4	IRQ3	IRQ2	IRQ1	R
0	DIRQ7	DIRQ6	DIRQ5	DIRQ4	DIRQ3	DIRQ2	DIRQ1	W

This register is used to configure local interrupts, drive the VMEbus IRQ lines individually, and reflect the status of the VMEbus IRQ lines. On an interrupt acknowledge cycle of the local interrupt, the upper byte (bits 15 through 8) of the register is returned along with the logical address of the VME-MXI in the lower byte (bits 7 through 0). The upper byte of this register is cleared on a hard reset. The lower byte is cleared on hard and soft resets.

Bit **Mnemonic** **Description**

15-13r/w LINT[3-1] Local Interrupt Line Bits

These bits select the VMEbus interrupt request line onto which the local VME-MXI interrupts are routed. The local interrupts are BKOFF, ACFAIL, and SYSFAIL.

LINT3	LINT2	LINT1	VMEbus Interrupt Request Line
0	0	0	Local Interrupt Disabled
0	0	1	Interrupt Request Line 1
0	1	0	Interrupt Request Line 2
0	1	1	Interrupt Request Line 3
1	0	0	Interrupt Request Line 4
1	0	1	Interrupt Request Line 5
1	1	0	Interrupt Request Line 6
1	1	1	Interrupt Request Line 7

12r	ACFAILINT	VMEbus ACFAIL Interrupt Status Bit If this bit is set, an interrupt is currently driven on the VMEbus interrupt line selected by the LINT[3-1] bits because the VMEbus ACFAIL line became set. This bit is cleared on an interrupt acknowledge cycle for the interrupt level selected by the LINT [3-1] bits.
12w, 10r/w, 7w	0	Reserved Bits These bits are reserved. Write a zero when writing to these bits.
11r	BKOFF	Backoff Status Bit This bit is set if a VMEbus transfer to the MXIbus could not complete because another MXIbus transfer to this module was already in progress; in other words, a deadlock condition occurred. The interrupt generated by this bit is cleared on an interrupt acknowledge cycle for the interrupt level selected by the LINT [3-1] bits. This bit is cleared when this register is read and on a hard reset.
11w	BKOFFIE	Backoff Interrupt Enable Bit If this bit is set, an interrupt is generated on the VMEbus interrupt line selected by the LINT[3-1] bits when a VMEbus Backoff condition occurs.
9r	SYSFAIL	VMEbus SYSFAIL Status Bit This bit reflects the status of the VMEbus SYSFAIL line.
9w	SYSFAILIE	VMEbus SYSFAIL Interrupt Enable Bit If this bit is set, an interrupt is generated on the VMEbus interrupt line selected by the LINT[3-1] bits when the VMEbus SYSFAIL line is set.
8r	ACFAIL	VMEbus ACFAIL Status Bit This bit reflects the status of the VMEbus ACFAIL line.
8w	ACFAILIE	VMEbus ACFAIL Interrupt Enable Bit If this bit is set, an interrupt is generated on the VMEbus interrupt line selected by the LINT[3-1] bits when the VMEbus ACFAIL line is set.

7r	SYSFAILINT	VMEbus SYSFAIL Interrupt Status Bit If this bit is set, an interrupt is currently driven on the VMEbus interrupt line selected by the LINT[3-1] bits because the VMEbus SYSFAIL line became set. This bit is cleared on an interrupt acknowledge cycle for the interrupt level selected by the LINT[3-1] bits.
6-0r	IRQ[7-1]	IRQ Status Bit These bits reflect the status of the corresponding VMEbus IRQ lines.
6-0w	DIRQ[7-1]	Drive IRQ Line Bits Setting these bits drives the corresponding VMEbus IRQ lines. When the VMEbus IRQ line driven by one of these bits is serviced by an VMEbus interrupt acknowledge cycle, the corresponding bit is cleared.

Status/ID Register

VMEbus Address: Base Address + 2C (hex)

Attributes: Read/Write

15	14	13	12	11	10	9	8	
S15	S14	S13	S12	S11	S10	S9	S8	R/W
7	6	5	4	3	2	1	0	
S7	S6	S5	S4	S3	S2	S1	S0	R/W

This register contains the Status/ID value returned to the Interrupt Handler acknowledging an interrupt request driven by one of the DIRQ bits in the Interrupt Control Register.

Bit	Mnemonic	Description
15-0r/w	S[15-0]	Status/ID Value This 16-bit value is the Status/ID data that is returned during a VMEbus interrupt acknowledge cycle used to handle a VMEbus interrupt request driven by one of the DIRQ bits in the Interrupt Control Register. This register powers up to an indeterminate value and is not cleared on either a hard or soft reset.

IRQ Acknowledge Registers

VMEbus Address: Base Address + 32 (hex) for IRQ1*
Base Address + 34 (hex) for IRQ2*
Base Address + 36 (hex) for IRQ3*
Base Address + 38 (hex) for IRQ4*
Base Address + 3A (hex) for IRQ5*
Base Address + 3C (hex) for IRQ6*
Base Address + 3E (hex) for IRQ7*

Attributes: Read Only

15	14	13	12	11	10	9	8	R
I15	I14	I13	I12	I11	I10	I9	I8	

7	6	5	4	3	2	1	0	R
I7	I6	I5	I4	I3	I2	I1	I0	

These registers generate a VMEbus interrupt acknowledge cycle when they are read from a MXIbus device.

Bit	Mnemonic	Description
15-0r	I[15-0]	Interrupt Acknowledge Status/ID Reading from these registers generates an interrupt acknowledge cycle on the VMEbus and returns the Status/ID value from the interrupt acknowledge cycle. These registers can be used to handle interrupts across the MXIbus. Each VMEbus IRQ line has a separate interrupt acknowledge register, as shown above in the VMEbus Address. The value returned when these registers are read by a VMEbus device is hex FFFF.

Chapter 5
Programming Considerations

This chapter explains important considerations for programming and configuring a VMEbus/MXIbus system using VME-MXIs.

Note: For detailed descriptions of all register bits, refer to Chapter 4, *Register Descriptions*.

System Configuration

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 chassis, VXIbus mainframes, personal computers (PCs), or stand-alone instruments. The MXIbus memory map is the same for all devices in the VMEbus/MXIbus system. Multiple VMEbus subsystems share one MXIbus Resource Manager (RM). This MXIbus RM configures all the MXIbus devices and VME-MXIs in the system to partition the MXIbus address space.

A VMEbus/MXIbus system can be connected 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 MXIbus RM. The root can be a VMEbus chassis or a stand-alone device (for example, 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 logical address space. Upon initialization, all windows are turned off, which isolates all MXIbus devices from each other. The MXIbus RM scans the MXIbus links for devices and configures the window registers on each MXIbus device in order to partition the MXIbus address space among all devices.

Planning a MXIbus System Logical Address Map

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

Before you begin setting the logical addresses of the devices in your MXIbus system, you must determine the tree configuration of your system. The two basic configurations are the MXIbus RM as an external PC with a MXIbus interface, as shown in Figure 5-1, or the MXIbus RM in a VMEbus chassis, as shown in Figure 5-2. The location of the MXIbus RM constitutes the root of the system tree. MXIbus links connected to the root of the tree form levels of the tree.

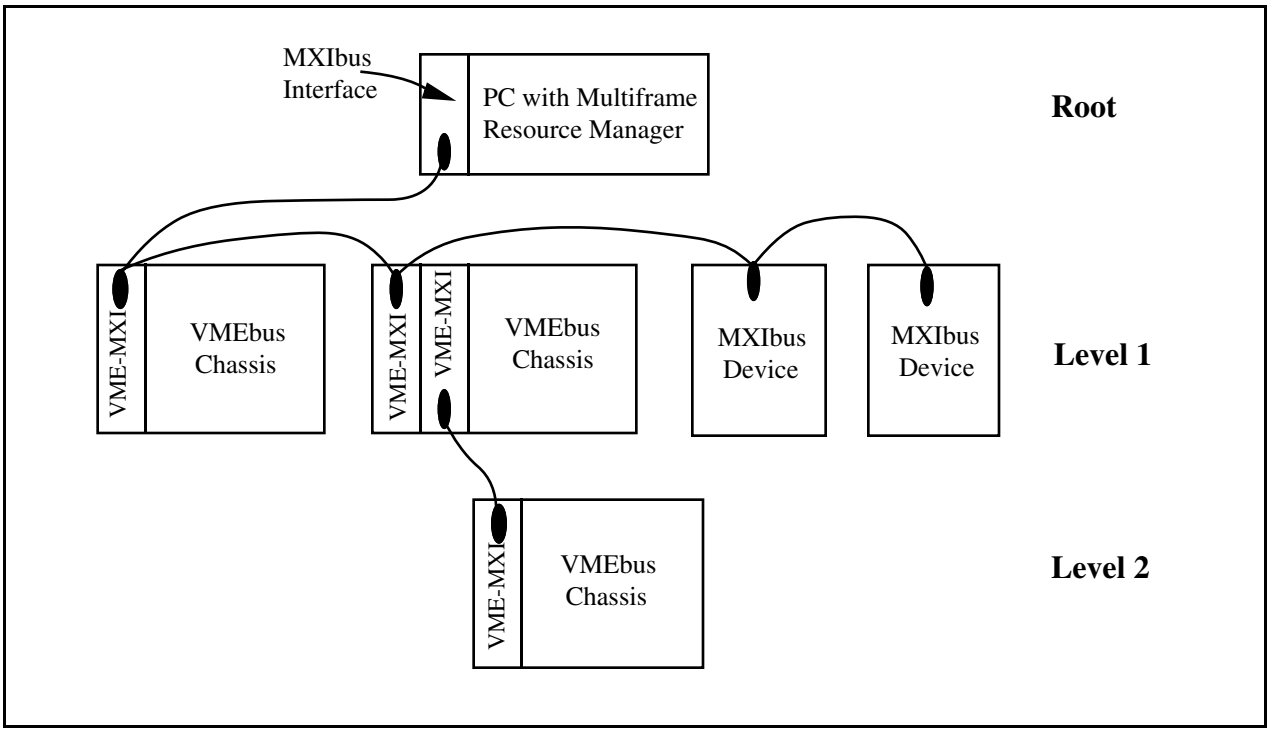


Figure 5-1. VMEbus/MXIbus System with MXIbus RM on a PC

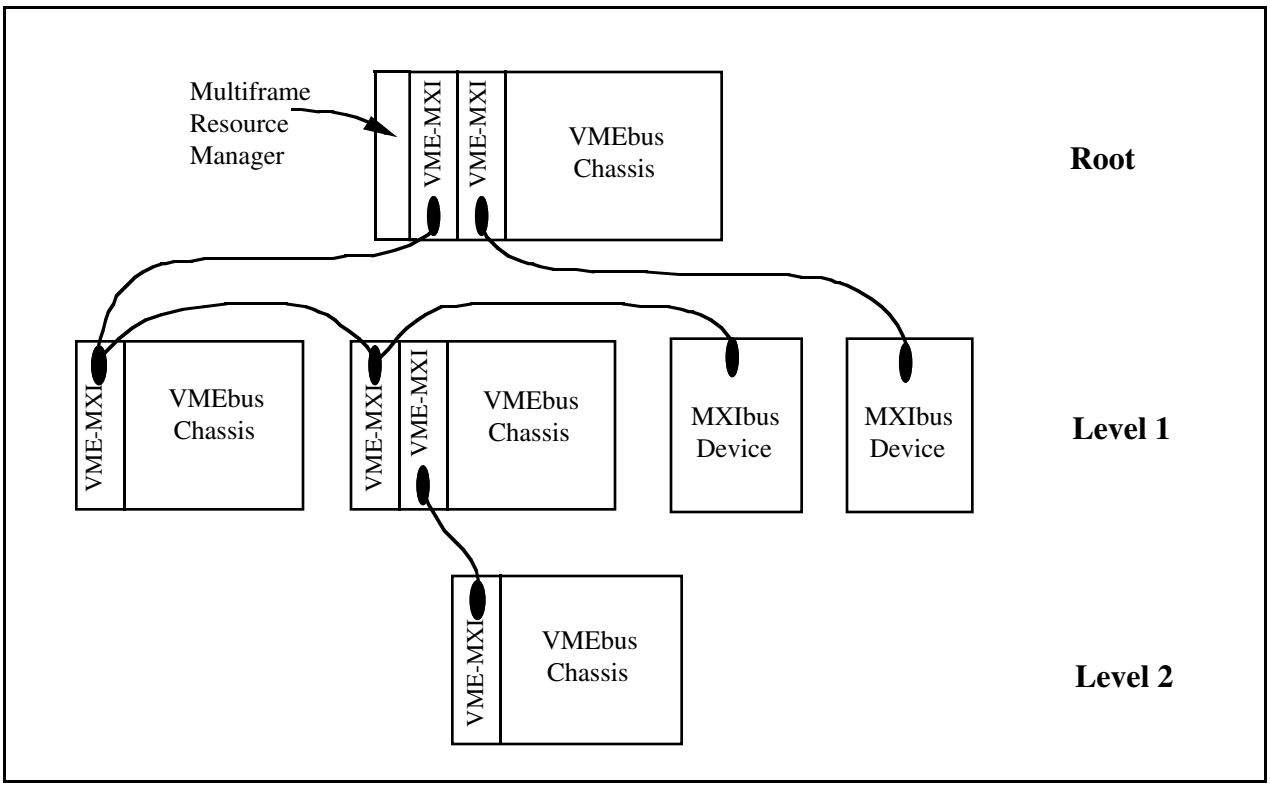


Figure 5-2. VMEbus/MXIbus System with MXIbus RM in a VMEbus Chassis

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

1. The VMEbus bus timeout unit must be on a VME-MXI.

2. If in interlocked bus mode, all VME-MXIs must be the highest priority VMEbus requesters in that frame, with the possible exception of one device in the MXIbus link.

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

Base/Size Configuration Format

Each address mapping window on a MXIbus interface has Base and Size parameters associated with it when the CMODE bit in the MXIbus Control Register 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 chassis.

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 hex 55. 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	*	*	*	*	*	*	*		0–1	54–55
6	*	*	*	*	*	*			0–3	54–57
5	*	*	*	*	*				0–7	50–57
4	*	*	*	*					0–F	50–5F
3	*	*	*						0–1F	40–5F
2	*	*							0–3F	40–7F
1	*								0–7F	00–7F
0									0–FF	00–FF

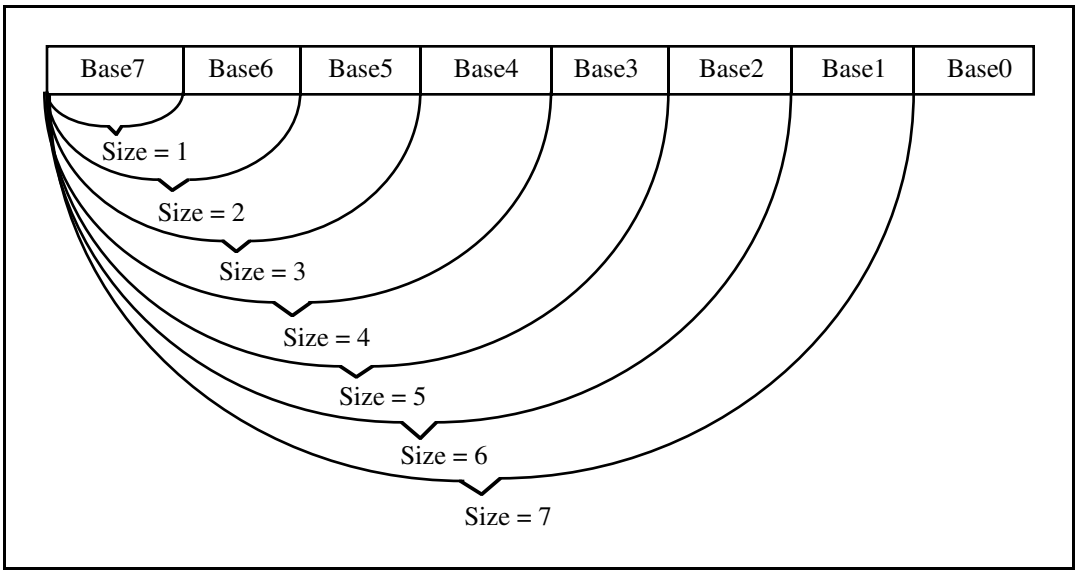


Figure 5-3. Base and Size Combinations

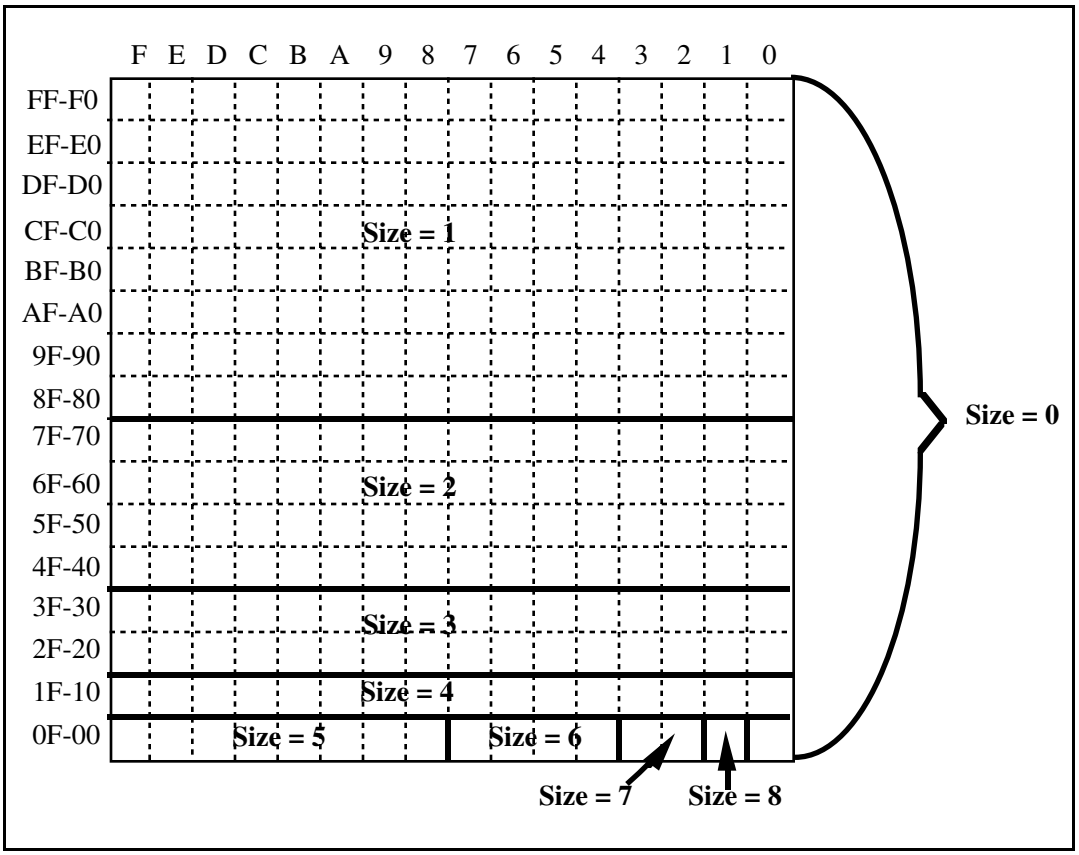


Figure 5-4. Address Range Allocation for Different Size Values

High/Low Configuration Format

Each address mapping window on the VME-MXI has High and Low address parameters associated with it when the CMODE bit in the MXIbus Control Register 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 chassis), 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 zero.

Steps to Follow When Planning a System Logical Address Map

As system integrator, when installing devices in the MXIbus system, you must assign a range of logical addresses for each MXIbus device, MXIbus link, and VXIbus mainframe. VXIbus and MXIbus devices have logical addresses assigned to them. The RM configures the logical address windows of each MXIbus device to include the static VXIbus logical addresses it finds in VXIbus mainframes and the logical addresses of MXIbus devices, and returns an error if these static logical address assignments prevent assignment of an entire system logical address map. VXIbus devices with dynamically configurable logical addresses are assigned logical addresses within the range of addresses defined by the static devices in the same VXIbus mainframe.

The example system in Figure 5-5 has two levels. The VMEbus RM is in VMEbus Chassis #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 device up to the next power of 2. 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 VXIbus mainframe and MXIbus device. See Figure 5-5 and Table 5-2 for examples. Identify the MXIbus 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.

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 chassis, the total number of logical addresses required is equal to the number of VME-MXIs in the chassis. If the RM is in a VXIbus mainframe, determine the number of logical addresses required by all VXIbus devices in that mainframe. 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 2 (unless the number is already a power of 2) 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 chassis and/or VXIbus mainframes on Level 1, you must change the logical addresses of both VME-MXI interfaces so that they are not at the default of 1. For VXIbus mainframe devices, 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 MXIbus RM is installed in VMEbus Chassis #1 and that frame requires two logical addresses. As the value 2 is already a power of 2, 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. Notice that only one MXIbus link can be in the first level if the MXIbus RM is a PC and that multiple MXIbus links can be in the first level if the RM is installed in a VMEbus chassis or VXIbus mainframe. 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 2 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 2 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 2 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 Chassis #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 Chassis #3. MXIbus #3 needs two logical addresses and the devices in VMEbus Chassis #3 need two logical addresses. The sum of these numbers is 4, which is already a power of 2 and does not need to be rounded.

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 highest power of 2 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 2 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 2. Refer to Table 5-1 and Figure 5-4 for example logical address allocations for different Size values.

   The RM by definition is located at Logical Address 0; therefore, the host device of the RM must be assigned a range of logical addresses that includes Logical Address 0. Starting with the MXIbus link on Level 1 that 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 Chassis #1, the host to the 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 2 is 40 to 7F hex. The other first-level MXIbus link, MXIbus #2, only needs 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 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 device, VXIbus Mainframe #2, needs 32 logical addresses and is assigned the next lowest available range of 32: 40 to 5F hex. The next largest one is VMEbus Chassis #3, which requires four logical addresses. This device has a second-level MXIbus link that needs two logical addresses, and the chassis needs two logical addresses for its own VME-MXIs. First, assign the devices in the chassis to the lowest available range within the allotted 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 address. They are assigned 64 to 67, and 68, respectively.

10. Determine the range of addresses that will be occupied by each second-level device and MXIbus link. Remember that the range of 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 address range 62 to 63 hex. MXIbus #3 has two devices: VMEbus Chassis #4 and VMEbus Chassis #5. Each requires one logical address; therefore, we assigned them addresses 62 hex and 63 hex, respectively.

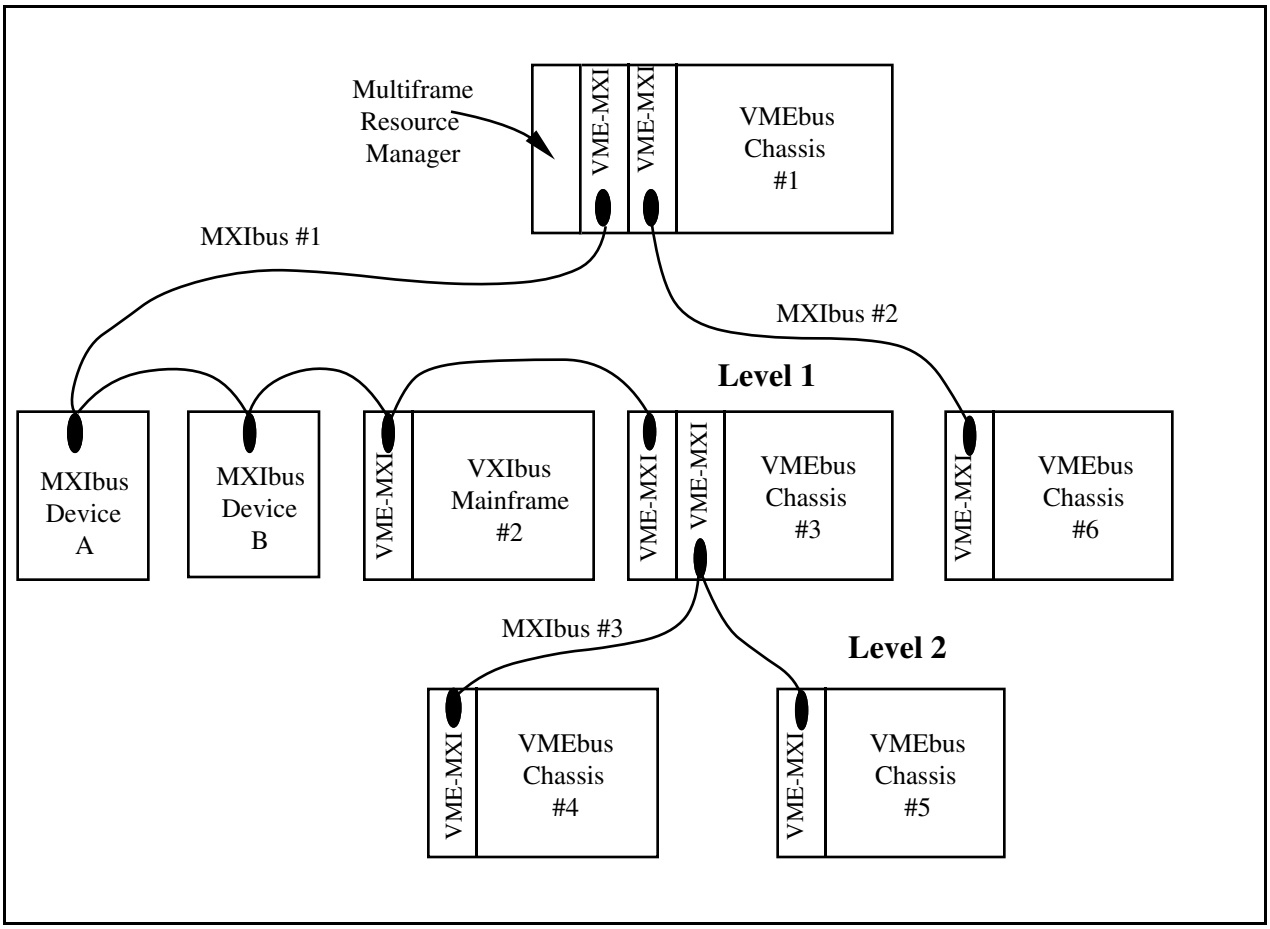


Figure 5-5. Example VMEbus/MXIbus System

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

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

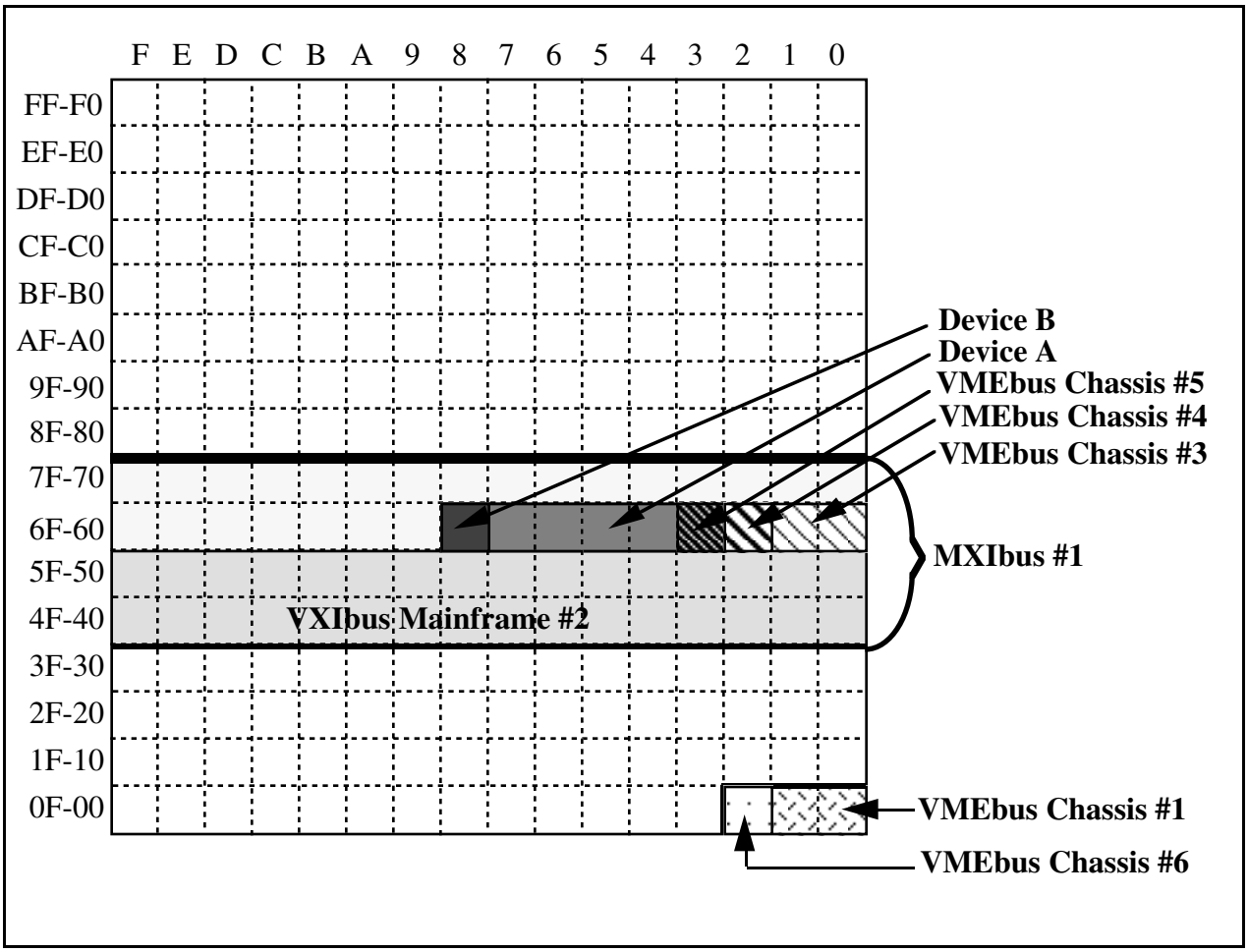


Figure 5-6. Logical Address Map Diagram for Example VMEbus/MXIbus System

Resource Manager Mainframe: *VMEbus Chassis #1*				
Total number of logical addresses required by this device:		*2*	Range =	*0 – 1*
Round total number up to the next power of 2:	*	*$2(2^1)$*	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 2:	*	*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 2:	*	*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 2:	*		Size =	
Total Number of Logical Addresses Required:		*67*		
(Add numbers after the "*")			Range =	*0 – 7F*
Round total number up to next power of 2:		*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 for 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 2:		*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 2:	*	*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 2:		*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 2:	*	*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 2:		*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 2:	*	*32 (2^5)*	Size =	*8-5 = 3*

Device: *VMEbus Chassis #3*

Number of logical addresses required by device:		*2*	Range =	*60 – 61*
Round total number up to the next power of 2:		*2 (2^3)*	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 2:	*	*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 2:			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 2:	*		Size =	

Total Number of Logical Addresses Required:		*41*	
(Add numbers after the "*")		Range =	*40 – 7F*
Round total number up to next power of 2:	*64 (2^6)*	Size =	*8-6 = 2*

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

MXIbus Link: *MXIbus #2*				
Device: *VMEbus Chassis #6*				
Number of logical addresses required by device:		*1*	Range =	*2*
Round total number up to the next power of 2:		*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 2:	*	*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 2:			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 2:	*		Size =	
Total Number of Logical Addresses Required:		*1*		
(Add numbers after the "*")			Range =	*2*
Round total number up to next power of 2:		*1 (2^0)*	Size =	*8-0 = 8*

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

MXIbus Link: *MXIbus #3*				
Device: *VMEbus Chassis #4*				
Number of logical addresses required by device:		*1*	Range =	*62*
Round total number up to the next power of 2:		*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 2:	*	*1 (2^0)*	Size =	*8-0 = 8*
Device: *VMEbus Chassis #5*				
Number of logical addresses required by device:		*1*	Range =	*63*
Round total number up to the next power of 2:		*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 2:	*	*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 2:		*2 (2^1)*	Size =	*8-1 = 7*

Figure 5-10. Worksheet 4 for 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 sample VMEbus/MXIbus system.

	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
FF-F0																
EF-E0																
DF-D0																
CF-C0																
BF-B0																
AF-A0																
9F-90																
8F-80																
7F-70																
6F-60																
5F-50																
4F-40																
3F-30																
2F-20																
1F-10																
0F-00																

Resource Manager Mainframe: ____________________

Total number of logical addresses required by this device: _________ Range = _________

Round total number up to the next power of 2: * _________ 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 2: * _________ 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 2: * _________ 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 2: * _________ 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 2: * _________ Size = __________

Total Number of Logical Addresses Required: ________

(Add numbers after the "*") Range = _________

Round total number up to next power of 2: _________ Size = _________

(If this number is greater than 256, you need to reorganize devices and try again.)

MXIbus Link:

Device: ____________________________

Number of logical addresses required by device: _________ Range = _________
Round total number up to the next power of 2: _________ 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 2: * _________ Size = _________

Device: ____________________________

Number of logical addresses required by device: _________ Range = _________
Round total number up to the next power of 2: _________ 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 2: * _________ Size = _________

Device: ____________________________

Number of logical addresses required by device: _________ Range = _________
Round total number up to the next power of 2: _________ 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 2: * _________ Size = _________

Device: ____________________________

Number of logical addresses required by device: _________ Range = _________
Round total number up to the next power of 2: _________ 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 2: * _________ Size = _________

Device: ____________________________

Number of logical addresses required by device: _________ Range = _________
Round total number up to the next power of 2: _________ 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 2: * _________ Size = _________

Device: ____________________________

Number of logical addresses required by device: _________ Range = _________
Round total number up to the next power of 2: _________ 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 2: * _________ Size = _________

Total Number of Logical Addresses Required: _________
(Add numbers after the "*") Range = _________
Round total number up to next power of 2: _________ Size = _________

MXIbus Link:

Device: ________________________________

Number of logical addresses required by device:		_________	Range =	_________
Round total number up to the next power of 2:		_________	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 2:	*	_________	Size =	_________

Device: ________________________________

Number of logical addresses required by device:		_________	Range =	_________
Round total number up to the next power of 2:		_________	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 2:	*	_________	Size =	_________

Device: ________________________________

Number of logical addresses required by device:		_________	Range =	_________
Round total number up to the next power of 2:		_________	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 2:	*	_________	Size =	_________

Device: ________________________________

Number of logical addresses required by device:		_________	Range =	_________
Round total number up to the next power of 2:		_________	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 2:	*	_________	Size =	_________

Device: ________________________________

Number of logical addresses required by device:		_________	Range =	_________
Round total number up to the next power of 2:		_________	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 2:	*	_________	Size =	_________

Device: ________________________________

Number of logical addresses required by device:		_________	Range =	_________
Round total number up to the next power of 2:		_________	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 2:	*	_________	Size =	_________

Total Number of Logical Addresses Required:	________		
(Add numbers after the "*")		Range =	_________
Round total number up to next power of 2:	_________	Size =	_________

MXIbus Link:

Device: ____________________

Number of logical addresses required by device:		________	Range =	________
Round total number up to the next power of 2:		________	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 2:	*	________	Size =	________

Device: ____________________

Number of logical addresses required by device:		________	Range =	________
Round total number up to the next power of 2:		________	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 2:	*	________	Size =	________

Device: ____________________

Number of logical addresses required by device:		________	Range =	________
Round total number up to the next power of 2:		________	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 2:	*	________	Size =	________

Device: ____________________

Number of logical addresses required by device:		________	Range =	________
Round total number up to the next power of 2:		________	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 2:	*	________	Size =	________

Device: ____________________

Number of logical addresses required by device:		________	Range =	________
Round total number up to the next power of 2:		________	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 2:	*	________	Size =	________

Device: ____________________

Number of logical addresses required by device:		________	Range =	________
Round total number up to the next power of 2:		________	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 2:	*	________	Size =	________

Total Number of Logical Addresses Required:	________		
(Add numbers after the "*")		Range =	________
Round total number up to next power of 2:	________	Size =	________

Alternative Worksheets for Planning Your MXIbus Logical Address Map

For most 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. 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 5-11 is one of these worksheets filled out for the example 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 2. With High/Low configuration, each VME-MXI window can be configured for exactly the amount of address space the chassis needs.


Device _____
Device LA's ___
Range IN _____
Lower LA's ___
Range OUT____
MXI#1
MXI#2
MXI#3
MXI#4
MXI#5
MXI#6
MXI#7
MXI#8
Device _____
Device LA's ___
Lower LA's ___
Total LA's ____
Range IN _____
Range OUT____

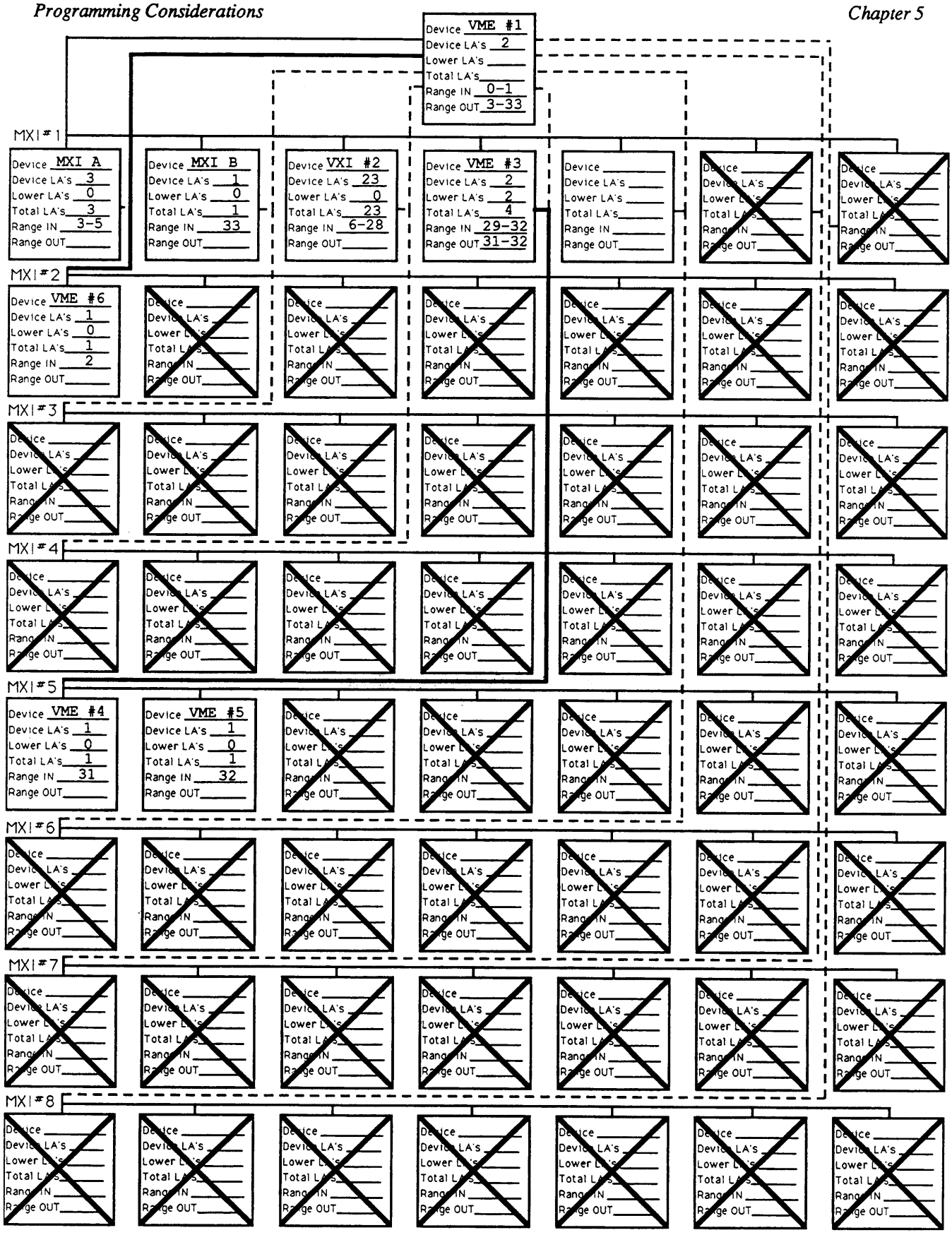


Figure 5-11. Logical Address Map Example with Alternative Worksheet

Planning a VMEbus/MXIbus System A16 Address Map

In a MXIbus system, A16 space is defined as the lower 48 kilobytes of VMEbus A16 space. The top 16 kilobytes is reserved for MXIbus device configuration registers. The lower 48 kilobytes of A16 space should be used for the A16 requirements of VMEbus devices installed in the VMEbus system. As system integrator, you must determine the A16 address requirements for your VMEbus/MXIbus system and define the A16 space ranges needed to the system RM.

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

When using Base/Size windowing formats, the minimum size of an A16 window is 512 bytes and the maximum size is 48 kilobytes (window size = 0). Setting an A16 window address range in the upper 16 kilobytes 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 bytes
6	1 kilobyte
5	2 kilobytes
4	4 kilobytes
3	8 kilobytes
2	16 kilobytes
1	32 kilobytes
0	48 kilobytes (All of A16 space)

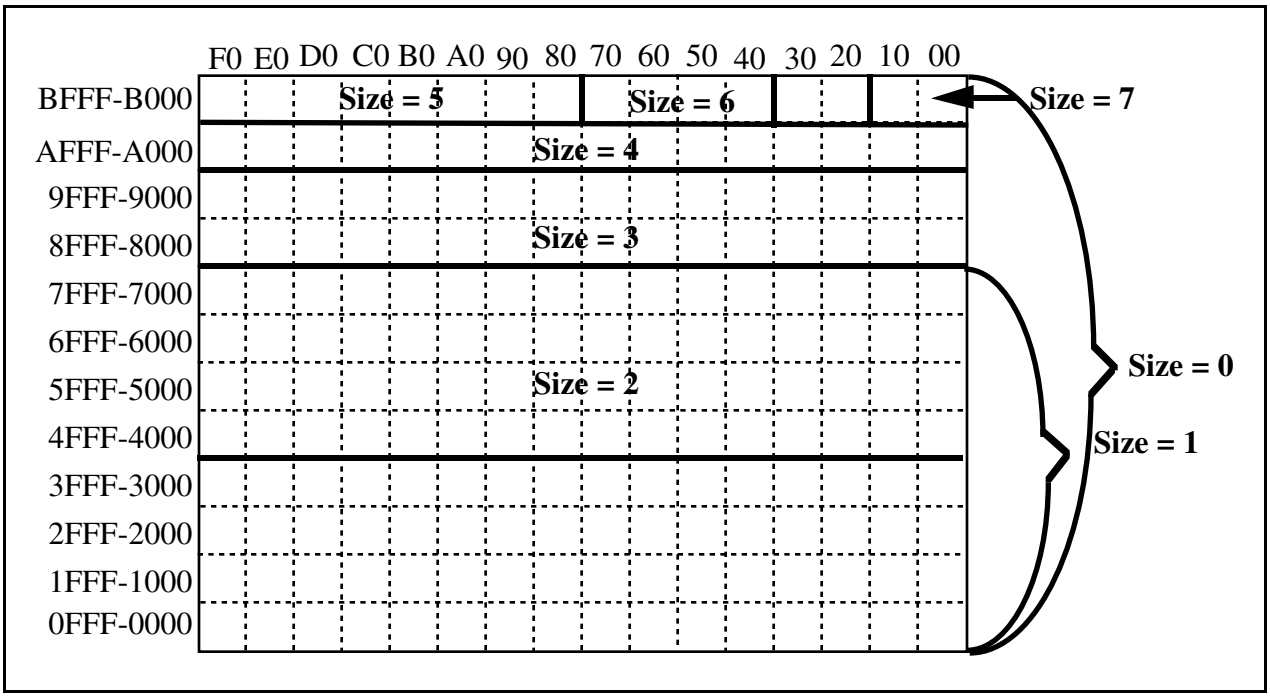


Figure 5-12. A16 Space Allocations for all Size Values

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, unless it already equals an amount in the table. Next, assign the A16 space, starting with the root device and working down the MXIbus system tree. To assist you in configuring the A16 windows on the VME-MXI 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 5-13 and Table 5-4.

2. Fill out the RM Mainframe information in Figure 5-15, Worksheet 1. In this example, the RM Mainframe needs 16 kilobytes of A16 space, which is already rounded to the address break of 16 kilobytes.

3. Next, analyze the first-level MXIbus links and complete a worksheet for each link. For our example, we fill out Figure 5-16, Worksheet 2, for MXIbus #1, which includes MXIbus Device A, MXIbus Device B, VXIbus Mainframe #2, and VMEbus Chassis #3. MXIbus Device A needs 512 bytes of A16 space, which is already rounded to the address break of 512. 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 Chassis #3 needs 4 kilobytes of A16, in addition to the amount of A16 required by MXIbus link #3 connected to it on Level 2.

4. Figure 5-17 is the worksheet for MXIbus #3, which includes VMEbus Chassis #4 and #5. VMEbus Chassis #4 needs 2 kilobytes and VMEbus Chassis #5 needs 1 kilobyte of A16 space. We fill in the appropriate spaces on the worksheet.

5. Now we return to Figure 5-16 and fill in the MXIbus #3 information in the space for a second-level MXIbus link connected to VMEbus Chassis #3. MXIbus #3 needs 2 kilobytes for Chassis #4 and 1 kilobyte for Chassis #5. The sum is 3 kilobytes, which rounds up to the next address break of 4 kilobytes. The amount of A16 space required for the window into VMEbus Chassis #3 is 4 kilobytes plus the 3 kilobytes required by MXIbus #3, which rounds up to the next address break of 8 kilobytes. We enter all of these numbers into the worksheet.

6. We now fill in Figure 5-15 for MXIbus #1. MXIbus #1 requires 512 bytes for MXIbus Device A and 8 kilobytes for VMEbus Chassis #3. The sum of these values rounds up to the nearest address break of 16 kilobytes. We record this information on the worksheet.

7. Figure 5-15 is now completed for MXIbus #2. The only device on MXIbus #2 is VMEbus Chassis #6, which needs 2 kilobytes 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 kilobytes, which rounds up to the next address break of 48 kilobytes. This number does not exceed the maximum of 48 kilobytes, 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 in the system. We first assign A16 space to the VMEbus RM Chassis. From Figure 5-15, we see it needs 16 kilobytes of A16 space, so we assign it the bottom 16 kilobytes of A16 space, addresses 0 through 3FFF hex. See Figure 5-14 for a pictorial representation of this assignment.

10. Each first-level MXIbus link is connected to the RM through a VME-MXI. The A16 window for MXIbus link #1 is 16 kilobytes in size. We assign the next lowest available 16-kilobyte portion of A16 space to MXIbus link #1, which is address range 4000 to 7FFF hex. (See Figure 5-14.) The base address of this window is 4000, which we enter into Figure 5-15. The Size field for the window is *i* where the size of the window = $256 * 2^{8-i}$. To determine the Size value, 16 kilobytes = $256 * 2^{8-2}$, 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 kilobytes of A16 space. The next lowest available 2-kilobyte 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 kilobytes = $256 * 2^{8-5}$, 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 5-15.

12. The VME-MXI 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 in VMEbus Chassis #3 should be assigned the lowest available 8 kilobytes of space assigned to MXIbus #1. Therefore, the base should be 4000 hex, and because 8 kilobytes = $256 * 2^{8-3}$, Size = 3. The direction of the window is in relation to the chassis; therefore, it is *In.* The VME-MXI connected to MXIbus #3 must be assigned a window within the range of addresses assigned to Chassis #3. Devices in Chassis #3 need 4 kilobytes of the 8 kilobytes assigned to the chassis. The other 4 kilobytes can be assigned to

MXIbus #3. Therefore, we assign addresses 4000 to 4FFF hex to devices in Chassis #3, and addresses 5000 through 5FFF to MXIbus #3. For the VME-MXI connected to MXIbus #3, we set Base = 5000, Size = 4 because 4 kilobytes = 256 * 2^{8-4}, and the direction toward MXIbus #3, or *Out*.

14. The 4 kilobytes assigned to MXIbus #3 is further divided between VMEbus Chassis #4 and #5. We assigned the bottom portion, 5000 to 57FF, to VMEbus Chassis #4, and the next portion, 5800 to 5BFF, to VMEbus Chassis #5. Therefore, for VMEbus Chassis #4, we assign Base = 5000, Size = 5 because 2 kilobytes = 256 * 2^{8-5}, and Direction = *In.* For VMEbus Chassis #5, Base = 5800, Size = 6 because 1 kilobyte = 256 * 2^{8-6}, and Direction = *In*.

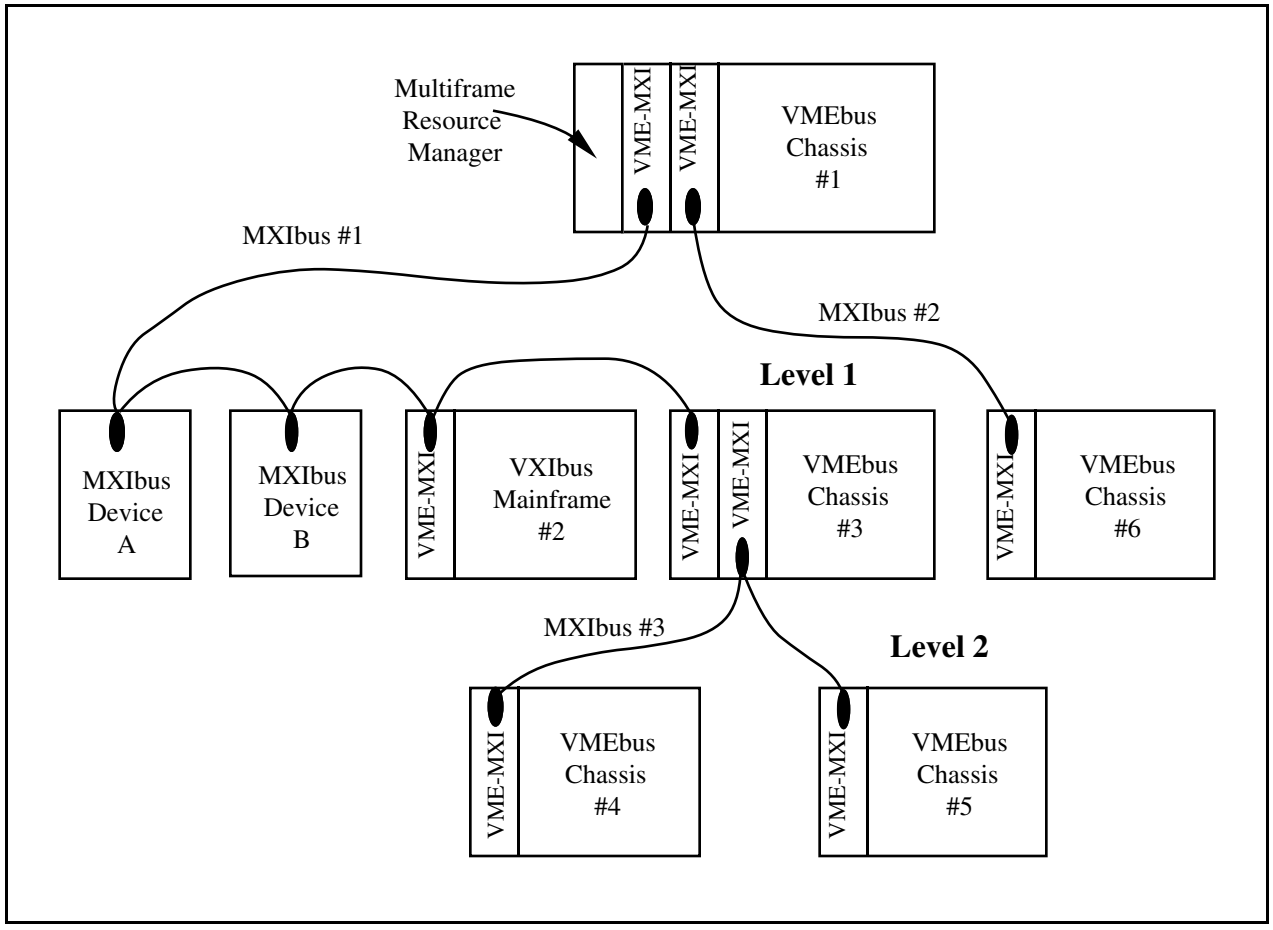


Figure 5-13. Example VMEbus/MXIbus System

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

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

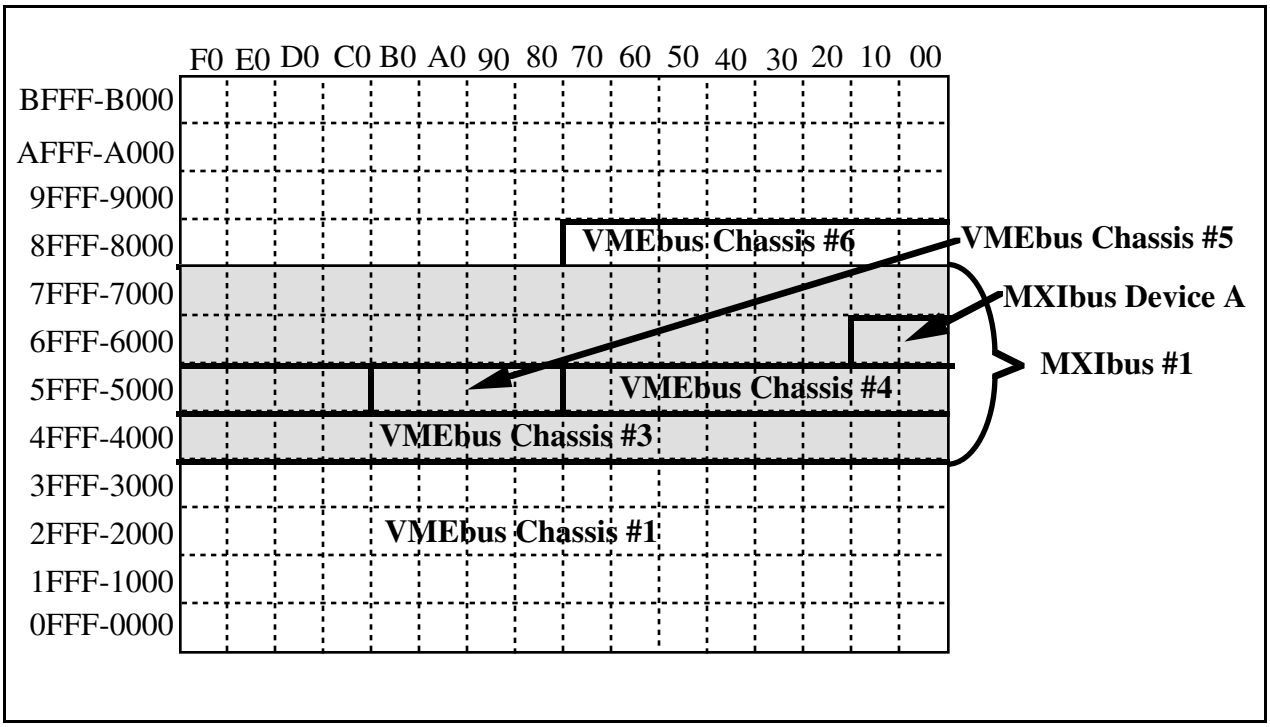


Figure 5-14. Example A16 Space Address Map

Resource Manager Chassis: *VMEbus Chassis #1*

Amount of A16 space required for this mainframe: *16K*

Round up to next address break: * *16K*

First Level MXIbus Link: *MXIbus #1*

Amount of A16 space required for devices connected to this VME-MXI: *8K + 512*

Round up to next address break: * *16K*

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: *2K*

Round up to next address break: * *2K*

A16 Window: Base: *8000* Size: *5* Direction: *Out*

First Level MXIbus Link: _____

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

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: _____

Round up to next address break: * _____

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

Total Amount of A16 Space Required by System: *34K*

(Add numbers after the "*")

Round up to next address break: *48K*

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

Figure 5-15. Worksheet 1 for 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: ____ ____ = *512*
Total amount of A16 space required for this window: * *512*
Round up total amount to the next address size break:
First Level VME-MXI:
A16 Window: Base: ____ Size: ____ Direction: ____
Second Level VME-MXI #1: ____
A16 Window: Base: ____ Size: ____ Direction: ____
Second Level VME-MXI #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: ____ ____ = *0*
Total amount of A16 space required for this window: * *0*
Round up total amount to the next address size break:
First Level VME-MXI:
A16 Window: Base: ____ Size: ____ Direction: ____
Second Level VME-MXI #1: ____
A16 Window: Base: ____ Size: ____ Direction: ____
Second Level VME-MXI #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: ____ ____ = *0*
Total amount of A16 space required for this window: * *0*
Round up total amount to the next address size break:
First Level VME-MXI:
A16 Window: Base: *0000* Size: *0* Direction: *Out*
Second Level VME-MXI #1: ____
A16 Window: Base: ____ Size: ____ Direction: ____
Second Level VME-MXI #2: ____
A16 Window: Base: ____ Size: ____ Direction: ____

Device: *VMEbus Chassis #3*
Amount of A16 space required by this device: *4K*
A16 space requirement for each second level MXIbus link connected to this device:
#1 *2K + 1K* + #2 ____ = *3K*
Round up to next address break: *4K* ____ = *7K*
Total amount of A16 space required for this window: * *8K*
Round up total amount to the next address size break:
First Level VME-MXI:
A16 Window: Base: *4000* Size: *3* Direction: *In*
Second Level VME-MXI #1: *MXIbus #3*
A16 Window: Base: *5000* Size: *4* Direction: *Out*
Second Level VME-MXI #2: ____
A16 Window: Base: ____ Size: ____ Direction: ____

Figure 5-16. Worksheet 2 for A16 Map Example

MXIbus Link: *MXIbus #3*

Device: *VMEbus Chassis #4*

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

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: = *2K*

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

First Level VME-MXI:

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

Second Level VME-MXI #1: ____

A16 Window: Base: ____ Size: ____ Direction: ____

Second Level VME-MXI #2: ____

A16 Window: Base: ____ Size: ____ Direction: ____

Device: *VMEbus Chassis #5*

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

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: = *1K*

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

First Level VME-MXI:

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

Second Level VME-MXI #1: ____

A16 Window: Base: ____ Size: ____ Direction: ____

Second Level VME-MXI #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:

A16 Window: Base: ____ Size: ____ Direction: ____

Second Level VME-MXI #1: ____

A16 Window: Base: ____ Size: ____ Direction: ____

Second Level VME-MXI #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:

A16 Window: Base: ____ Size: ____ Direction: ____

Second Level VME-MXI #1: ____

A16 Window: Base: ____ Size: ____ Direction: ____

Second Level VME-MXI #2: ____

A16 Window: Base: ____ Size: ____ Direction: ____

Figure 5-17. Worksheet 3 for A16 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 sample VMEbus/MXIbus system.

	F0	E0	D0	C0	B0	A0	90	80	70	60	50	40	30	20	10	00
BFFF-B000																
AFFF-A000																
9FFF-9000																
8FFF-8000																
7FFF-7000																
6FFF-6000																
5FFF-5000																
4FFF-4000																
3FFF-3000																
2FFF-2000																
1FFF-1000																
0FFF-0000																

Resource Manager Chassis: ______________________

Amount of A16 space required for this chassis: ________

Round up to next address break: * ________

First Level MXIbus Link: ______________________

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

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: ________

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: ________

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: ________

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 48K bytes, reorganize devices and try again.)

MXIbus Link: _____

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:

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

Second Level VME-MXI #1: _____

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

Second Level VME-MXI #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:

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

Second Level VME-MXI #1: _____

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

Second Level VME-MXI #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:

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

Second Level VME-MXI #1: _____

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

Second Level VME-MXI #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:

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

Second Level VME-MXI #1: _____

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

Second Level VME-MXI #2: _____

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

MXIbus Link: _____

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:

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

Second Level VME-MXI #1: _____

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

Second Level VME-MXI #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:

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

Second Level VME-MXI #1: _____

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

Second Level VME-MXI #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:

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

Second Level VME-MXI #1: _____

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

Second Level VME-MXI #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:

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

Second Level VME-MXI #1: _____

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

Second Level VME-MXI #2: _____

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

MXIbus Link: _____

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:

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

Second Level VME-MXI #1: _____

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

Second Level VME-MXI #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:

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

Second Level VME-MXI #1: _____

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

Second Level VME-MXI #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:

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

Second Level VME-MXI #1: _____

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

Second Level VME-MXI #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:

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

Second Level VME-MXI #1: _____

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

Second Level VME-MXI #2: _____

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

MXIbus Link: _____

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:

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

Second Level VME-MXI #1: _____

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

Second Level VME-MXI #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:

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

Second Level VME-MXI #1: _____

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

Second Level VME-MXI #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:

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

Second Level VME-MXI #1: _____

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

Second Level VME-MXI #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:

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

Second Level VME-MXI #1: _____

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

Second Level VME-MXI #2: _____

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

MXIbus RM Operation

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

- Identifies all MXIbus and 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 VXIbus commander/servant system hierarchy, if any
- Initiates normal system operation

Configuring the Logical Address Window

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

1. If the RM resides in a PC, it scans all logical addresses from 1 to FE (the RM is at Logical Address 0) to find all VXIbus and MXIbus devices. For each logical address, it reads the 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 VXI-MXI and VME-MXI found, it performs Step 2.

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

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

   A. Scans all logical addresses (0 to FF) in the mainframe to find all static configuration (SC) and VXIbus dynamic configuration (DC) devices, skipping over logical addresses occupied by previously encountered devices and any upper A16 space occupied by VMEbus devices (the RM must be informed of any such VMEbus devices before configuration begins). If the VME-MXI is in a VXIbus mainframe, the RM finds the Slot 0 device and uses it to move all DC devices in the mainframe to the lowest unused logical addresses. The RM 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 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 VME-MXI found in the mainframe or chassis, starting with the lowest addressed VME-MXI:

      i. Sets the VME-MXI 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 and any upper A16 space occupied by VMEbus devices.

iii. For each VME-MXI found in Step ii, starting with the lowest addressed VME-MXI:

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

iv. Sets the VME-MXI 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 MXIbus 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. The RM will find all MXIbus devices, and all MXIbus devices must have their logical addresses statically configured. Therefore, each device must have a logical address that was configured before the RM is executed.

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

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

The RM performs the following steps:

1. Scans logical addresses (0 to FF) and identifies all devices in VMEbus Chassis #1. Finds the VME-MXIs at Logical Addresses 0 and 1.

2. Enables the logical address window of the VME-MXI 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 in VMEbus Chassis #3, the VME-MXI in VXIbus Mainframe #2, MXIbus Device A, and MXIbus Device B.

3. Enables the logical address window of the VME-MXI in VMEbus Chassis #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 at Logical Address 61.

4. Enables the logical address window of the VME-MXI 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 in VMEbus Chassis #4 and the VME-MXI in VMEbus Chassis #5.

5. Enables the logical address window of the VME-MXI in VMEbus Chassis #4 for the entire inward mapping range of 0 to FF. Scans all logical addresses, skipping all previously encountered devices. No more VME-MXI interfaces are found. Because VMEbus Chassis #4 has no VXIbus or MXIbus devices, the logical address window does not need programming.

6. Enables the logical address window of the VME-MXI in VMEbus Chassis #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. No more VME-MXI interfaces are found. Because VMEbus Chassis #4 has no VXIbus or MXIbus devices, the logical address window does not need programming.

7. Sets the logical address window of the VME-MXI found at Logical Address 61 to cover the ranges of the VME-MXI in VMEbus Chassis #4 (62) and the VME-MXI in VMEbus Chassis #5 (63). Enables the logical address window of the VME-MXI 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 found in VMEbus Chassis #3 at Logical Address 60 to cover the devices in that chassis (60 to 61) and the ranges required of its Level 2 devices: the VME-MXI in VMEbus Chassis #4 (62) and the VME-MXI in VMEbus Chassis #5 (63). Enables the logical address window of the VME-MXI 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 VME-MXI 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 VME-MXI interfaces are found. The RM enables the logical address window for the VME-MXI 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 found in VMEbus Chassis #1 at Logical Address 0 to cover the devices connected to that extender: the VME-MXI in VMEbus Chassis #3 (60 to 63), the VME-MXI 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 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 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 in VMEbus Chassis #6.

12. Enables the logical address window of the VME-MXI in VMEbus Chassis #6 for the entire inward mapping range of 0 to FF. Scans all logical addresses, skipping all previously encountered devices and defined ranges, and finds the VME-MXI at Logical Address 2. No more VME-MXI interfaces are found. Because VMEbus Chassis #6 has no VXIbus or MXIbus devices, the logical address window does not need programming.

13. Sets the logical address window of the VME-MXI found in VMEbus Chassis #1 at Logical Address 1 to cover the devices connected to that extender: the VME-MXI in VMEbus Chassis #6 (2). Enables the logical address window of the VME-MXI 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 configuring the logical address space for the system, the MXIbus RM configures the A16, A24 and A32 address spaces. 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 chassis and the devices on levels below it. That amount is then rounded up to the next power of 2 (unless it already is a power of 2) if the Base/Size format is used.

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

Chapter 6
Theory of Operation

A brief description of the VME-MXI is given in Chapter 2 along with a functional block diagram (see Figure 2-1). The major elements of the VME-MXI are discussed in more detail in this chapter. For a detailed discussion of the VMEbus, refer to the VMEbus specification. For a description of MXIbus, refer to the MXIbus specification.

VMEbus Address and Address Modifier Transceivers

The VMEbus address transceivers and associated logic control the direction of the VMEbus address lines and latch incoming address lines on the rising edge of the VMEbus Address Strobe (AS) signal. The VMEbus Address Modifier lines are also controlled by this circuitry and are also latched on the rising edge of AS.

VMEbus System Controller Functions

If the VME-MXI is configured as the VMEbus System Controller, this circuitry provides the 16-MHz VMEbus system clock and the VMEbus data transfer bus arbiter.

The 16-MHz system clock driver is derived from an onboard clock with an accuracy of 100 ppm and a duty cycle of 50% (± 5%). The data transfer bus arbiter capability (PRI ARBITER) accepts bus requests from all four VMEbus requester levels, prioritizes the requests, and grants the bus to the highest priority requester.

VMEbus Data Transceivers

The VMEbus data transceivers control the sending and receiving of data on the 32-bit data bus from the VMEbus.

VMEbus Control Signal Transceivers

The VMEbus control signal transceivers control the sending and receiving of the VMEbus control signals such as address strobe (AS*), the data strobes (DS1*, DS0*), longword (LWORD*), read/write (WRITE*), data transfer acknowledge (DTACK*), and bus error (BERR*). These signals indicate the beginning and end of a transfer, the size of data involved in the transfer (8, 16, or 32 bits), whether the transfer is a read or a write, and whether the transfer was successful.

VMEbus Requester and Arbiter Circuitry

Through the VMEbus requester and arbiter circuitry, a remote MXIbus device can access main memory in the VMEbus system via the VMEbus. The arbiter circuitry is only active on the VME-MXI if it is configured as the VMEbus System Controller.

The VME-MXI requests use of the VMEbus when it detects a MXIbus address that maps through one of the mapping windows to the VMEbus. The VME-MXI drives its VMEbus request line active to initiate arbitration for the VMEbus. When the VME-MXI is the highest priority requesting device, the VMEbus System Controller sends a bus grant signal indicating that the VMEbus is granted to the VME-MXI. The VME-MXI drives the VMEbus BBSY* signal indicating that it owns the VMEbus, and then releases its bus request line.

A remote MXIbus device can lock the VMEbus so that it can perform indivisible operations across the VMEbus. When the LOCKED bit in the Local Bus Lock Register is set by a MXIbus device, the VME-MXI interface will not release the VMEbus the next time it is granted the bus (on the next transaction) until the LOCKED bit is cleared by a MXIbus device.

SYSFAIL, ACFAIL, and SYSRESET

The VMEbus signals SYSFAIL*, ACFAIL*, and SYSRESET* can be individually monitored and driven by the VME-MXI card. These three signals can also be used individually to generate an interrupt across the MXIbus IRQ line and/or one of the VMEbus interrupt request lines.

Interrupt Circuitry

The MXIbus has one interrupt line, IRQ. This IRQ line can be mapped to or driven by one of the VMEbus interrupt lines IRQ[7–1]* or driven by VMEbus signals SYSFAIL*, ACFAIL*, and/or SYSRESET*. Registers in the MXIbus configuration space are used to configure the MXIbus IRQ* line operation. Three local VME-MXI conditions can be enabled to drive the VMEbus interrupt lines: SYSFAIL* asserted, ACFAIL* asserted, and the Backoff condition.

The Backoff condition occurs when the VME-MXI is a MXIbus master arbitrating for the MXIbus and a MXIbus transfer requesting the VMEbus is received. This situation results in a deadlock condition. The MXIbus master circuitry must send a BERR* to the VMEbus master initiating the MXIbus transfer so that the incoming transfer can complete. The VMEbus master can monitor the backoff interrupt. If the interrupt occurs, the master should retry its last MXIbus operation because it did not complete due to the deadlock condition.

VMEbus interrupt requests can be handled by an interrupt handler on another VMEbus device in the VMEbus chassis or by an external device on the MXIbus. The VME-MXI has IACK daisy-chain driver circuitry that passes interrupt acknowledge cycles not meant for the VME-MXI to other interrupters in the VMEbus chassis. Similarly, the MXIbus has an IACK daisy-chain mechanism that converts and passes interrupt acknowledge cycles from VMEbus to MXIbus to VMEbus, making transparent interrupt acknowledge cycles possible between VMEbus chassis. Because multiple VMEbus IRQ lines can be mapped onto the single MXIbus IRQ line, interrupt acknowledge sequences for MXIbus IRQ requests cannot be completely transparent. You can have completely transparent interrupt handling through the use of the Interrupt and Timing Extension (INTX) daughter card option, in which each VMEbus interrupt line is mapped on a separate signal.

When multiple VMEbus IRQ lines are mapped onto the single shared MXIbus IRQ line, the interrupt handler routine can acknowledge the interrupts in one of two ways.

1. If the interrupt handler cannot perform MXIbus IACK cycles, it must poll all MXIbus devices to determine the source of the MXIbus IRQ signal. The interrupt handler polls the MXIbus IRQ Configuration Register and the Interrupt Status Register of each VME-MXI on the MXIbus link to determine which VMEbus IRQ line is being sourced onto the MXI IRQ line. The interrupt handler can then read from the corresponding IRQ acknowledge register on the VME-MXI driving the MXIbus IRQ line to acknowledge the interrupt request.

2. If the interrupt handler can generate MXIbus IACK cycles, it is not necessary to poll MXIbus devices to find the source of the MXIbus IRQ signal. The interrupt handler can perform an IACK cycle for the VMEbus line onto which the MXIbus IRQ line is mapped in that frame. The VME-MXI driving the MXIbus IRQ line responds with a Status/ID value in which the lower byte is the logical address of the VME-MXI. The interrupt handler then polls the MXIbus IRQ Configuration Register and the Interrupt Status Register on the VME-MXI at the logical address specified by the Status/ID value received to determine which VMEbus IRQ line is routed onto the MXIbus IRQ line. The interrupt handler can then read from the corresponding VME-MXI IRQ acknowledge register to acknowledge the interrupt request.

When this process is completed and if another VMEbus IRQ is also driving the MXIbus IRQ, the interrupt handler module is interrupted again, and should follow the same procedure described above.

MXIbus defines a special interrupt acknowledge (IACK) cycle, which is denoted with a special MXIbus address modifier code, hex 12. When a VMEbus interrupt handler generates a VMEbus IACK cycle for an active interrupt request line that is mapped into its VMEbus chassis from the MXIbus IRQ line, the VMEbus IACK cycle is converted into a MXIbus IACK cycle. The VME-MXI driving the interrupt request initiates a VMEbus IACK cycle when it detects the MXIbus IACK cycle, and responds by driving its Status/ID on the data bus and asserting DTACK*. The interrupt handler receives the Status/ID and DTACK* from across the MXIbus as if it had been in the same chassis as the VME-MXI.

The Status/ID information returned by the remote VME-MXI indicates its logical address. With this information, the interrupt handler can poll the remote VME-MXI to determine which interrupt lines are mapped onto the MXIbus IRQ line and which interrupt lines are active. The interrupt is then acknowledged by reading the corresponding register shown in Table 6-1.

Multiple MXIbus devices can interrupt on the same interrupt line; therefore, a MXIbus interrupt acknowledge daisy-chain is required. The MXIbus GIN and GOUT signals are normally used for the arbitration bus grant in/bus grant out daisy-chain. However, when a MXIbus device initiates a MXIbus IACK cycle and drives the MXIbus address modifier code hex 12, the MXIbus GIN and GOUT lines are used as the interrupt acknowledge daisy-chain. The MXIbus System Controller starts the interrupt acknowledge daisy-chain when it detects the address modifier code hex 12. The interrupt acknowledge signal propagates down the daisy-chain to each MXIbus device. If the device is not interrupting the MXIbus, it passes the signal down the daisy-chain to the next device. If the VME-MXI is interrupting, the cycle is converted into a VMEbus IACK cycle.

A MXIbus device not capable of generating a MXIbus IACK cycle can service an interrupt in a remote VMEbus chassis by reading from a designated address in the MXIbus configuration space on the remote VME-MXI. The external device must know which VMEbus interrupt level it is servicing and read from the appropriate address. Table 6-1 shows the designated addresses for VMEbus IRQ[7–1].

Table 6-1. VME-MXI Addresses for VMEbus Interrupt Levels

VMEbus IRQ Line	VME-MXI Configuration Register to Read
VMEbus IRQ1	Interrupt Acknowledge 1 (VME-MXI offset = 32)
VMEbus IRQ2	Interrupt Acknowledge 2 (VME-MXI offset = 34)
VMEbus IRQ3	Interrupt Acknowledge 3 (VME-MXI offset = 36)
VMEbus IRQ4	Interrupt Acknowledge 4 (VME-MXI offset = 38)
VMEbus IRQ5	Interrupt Acknowledge 5 (VME-MXI offset = 3A)
VMEbus IRQ6	Interrupt Acknowledge 6 (VME-MXI offset = 3C)
VMEbus IRQ7	Interrupt Acknowledge 7 (VME-MXI offset = 3E)

Reading from one of the addresses listed in Table 6-1 initiates a VMEbus IACK cycle. The information sent back from the read is the VMEbus Status/ID information. The lower byte of the Status/ID is the logical address of the responding interrupter. The upper byte is undefined.

When one of the local VME-MXI interrupt conditions is serviced by an interrupt handler, the Status/ID information returned is as follows:

15	14	13	12	11	10	9	8
LINT3	LINT2	LINT1	ACFAILINT	BKOFF	TRIGINT	SYSFAIL	ACFAIL
7	6	5	4	3	2	1	0
LADD7	LADD6	LADD5	LADD4	LADD3	LADD2	LADD1	LADD0

The VMEbus interrupt lines can be individually driven by writing to the Interrupt Status/Control Register. When one of these interrupt requests is serviced by an interrupt handler, the information in the Status/ID Register is returned during the IACK cycle and the interrupt request is cleared.

Parity Check and Generation

All MXIbus devices are required to generate even parity. The VME-MXI always generates and checks parity on all 32 MXIbus address and data lines. If upper bytes of the address or data are not driven, these lines are pulled high by the MXIbus termination circuitry and do not affect the parity generation.

A32, A24, A16, and LA Windows

Four addressing windows map in and out of the VMEbus chassis. These windows represent the three VMEbus address spaces (A32, A24, and the lower 48 kilobytes of A16) plus a dedicated window for mapping the MXIbus configuration space (the upper 16 kilobytes of A16). For each window, the range that maps into the chassis from the MXIbus to the VMEbus is whatever is left over from the window that maps out of the chassis from the VMEbus to the MXIbus. VME-MXI configuration registers are used to program these windows to indicate which addresses in each window are mapped onto the MXIbus.

VME-MXI Configuration Registers

The VME-MXI configuration registers are accessible from both the VMEbus and the MXIbus and are used to configure the VME-MXI. These registers are described in detail in Chapter 4, *Register Descriptions*.

When the VME-MXI interface decodes a VMEbus address specifying the configuration space on the card, the least significant VMEbus address lines are used to specify the registers in configuration space and the VMEbus operation does not need to request control of the MXIbus. Similarly, when the VME-MXI interface decodes a MXIbus address specifying configuration space on the card, the least significant MXIbus address lines are used to specify the registers in configuration space and the access does not need to request control of the VMEbus. Onboard circuitry automatically arbitrates between the MXIbus and VMEbus for use of the dual-ported configuration space and prevents deadlock conditions on configuration space accesses.

MXIbus Master Mode State Machine

The VME-MXI continuously compares VMEbus addresses and address modifiers to the four MXIbus addressing windows. When a VMEbus transfer involving an address corresponding to one of the outward mapping windows is detected, the VME-MXI begins arbitrating for the MXIbus. The VME-MXI can translate A32, A24, A16, D32, D16, and D08(EO) VMEbus transfers into corresponding MXIbus master mode transfers.

When the VME-MXI wins ownership of the MXIbus, a MXIbus cycle is initiated and the VMEbus transfer is converted into a MXIbus transfer. The MXIbus address and address strobe are sent, followed by the data (if the transfer is a write) and a data strobe. The transfer is complete when the responding device sends DTACK* and the VME-MXI releases the data strobe and address strobe. The VME-MXI interface supports 8-bit, 16-bit, and 32-bit reads and writes across the MXIbus. The least significant data bit maps to MXIbus data line AD00 and the byte orientation on the MXIbus is standard 68000 format.

Communication across the MXIbus between devices in separate VMEbus chassis appears as normal transfers to the devices. The bus cycles are mapped from one device through the addressing windows, across the MXIbus, and through address windows on the second device. The first device initiates the transfer with an address strobe and data strobe, and the second device responds by asserting DTACK* or BERR*. Figure 6-1 illustrates that a master device initiates a transfer on the VMEbus, which is converted into a MXIbus transfer, then back into a VMEbus transfer to reach the target slave.

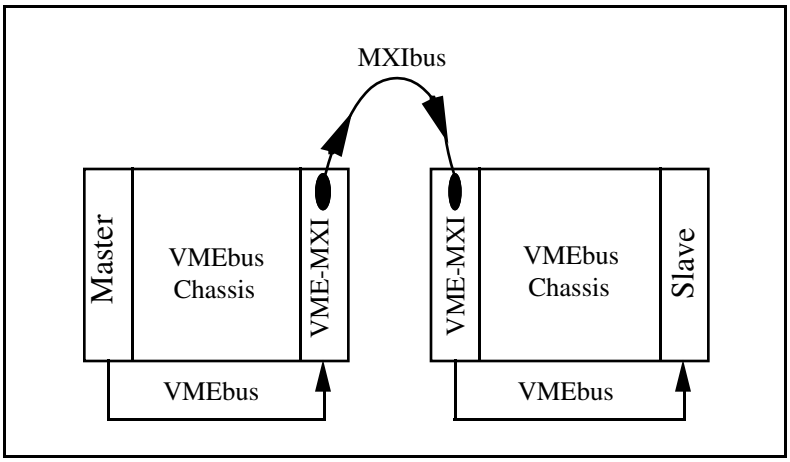


Figure 6-1. Master to Slave VMEbus/MXIbus Transfers

The 32 VMEbus address lines map directly to the 32 MXIbus address lines. The VMEbus requires six address modifier lines, while MXIbus only defines five. The VMEbus address modifier lines map to the MXIbus address modifier lines as shown in Table 6-2. The VME-MXI responds to the VMEbus address modifier codes shown in Table 6-3.

Information specifying the number of bytes and which bytes are involved in a VMEbus transfer is sent on data strobe lines DS1* and DS0*, address line A01, and the LWORD* line. During MXIbus transfers, the same information is transferred on the Size line, and Address/Data lines 1 and 0 (AD01 and AD00). The VMEbus transfer size information is converted into MXIbus transfer size information during a MXIbus master transfer. Table 6-4 compares this information for the VMEbus and the MXIbus.

Table 6-2. VMEbus to MXIbus Address Modifier Line Map

VMEbus Address Modifier Line	**MXIbus Address Modifier Line**
VMEbus AM5	MXIbus AM4
VMEbus AM4	MXIbus AM3
VMEbus AM2	MXIbus AM2
VMEbus AM1	MXIbus AM1
VMEbus AM0	MXIbus AM0

Table 6-3. Transfer Responses for VMEbus Address Modifiers

AM5	AM4	AM3	AM2	AM1	AM0	Transfer Type
H	H	H	H	H	H	A24 supervisory block transfer
H	H	H	H	H	L	A24 supervisory program access
H	H	H	H	L	H	A24 supervisory data access
H	H	H	L	H	H	A24 nonprivileged block transfer
H	H	H	L	H	L	A24 nonprivileged program access
H	H	H	L	L	H	A24 nonprivileged data access
H	L	H	H	L	H	A16 supervisory access
H	L	H	L	H	L	MXIbus transparent IACK cycle
H	L	H	L	L	H	A16 nonprivileged access
L	L	H	H	H	H	A32 supervisory block transfer
L	L	H	H	H	L	A32 supervisory program access
L	L	H	H	L	H	A32 supervisory data access
L	L	H	L	H	H	A32 nonprivileged block transfer
L	L	H	L	H	L	A32 nonprivileged program access
L	L	H	L	L	H	A32 nonprivileged data access

Table 6-4. VMEbus/MXIbus Transfer Size Comparison

	VMEbus				MXIbus			Byte Locations			
	DS1*	DS0*	A01	LWORD*	Size	AD01	AD00	D24-31	D16-23	D08-15	D00-07
8-bit Transfers											
Byte(0)	0	1	0	1	0	0	0			Byte(0)	
Byte(1)	1	0	0	1	0	0	1				Byte(1)
Byte(2)	0	1	1	1	0	1	0			Byte(2)	
Byte(3)	1	0	1	1	0	1	1				Byte(3)
16-bit Transfers											
Byte(0-1)	0	0	0	1	1	0	0			Byte(0)	Byte(1)
Byte(2-3)	0	0	1	1	1	1	0			Byte(2)	Byte(3)
32-bit Transfers											
Byte(0-3)	0	0	0	0	1	1	1	Byte(0)	Byte(1)	Byte(2)	Byte(3)

The VME-MXI generates and checks the parity of the address and data portions of all MXIbus cycles. The MXIbus PAR* signal is generated and sent during the address portion of all MXIbus cycles initiated by the VME-MXI. It is also generated and sent during the data portion of MXIbus master write cycles. When the VME-MXI detects a parity error in the data transfer portion of a MXIbus read cycle, it asserts the VMEbus BERR* signal to indicate to the VMEbus host that the data read contains an error.

Deadlock occurs when a VMEbus master is arbitrating for the MXIbus at the same time that a remote MXIbus device is requesting the same VMEbus. This situation is shown in Figure 6-2 where the VMEbus master arbitrating for the MXIbus is the VME-MXI in VMEbus Chassis #2 and the remote MXIbus device requesting the VMEbus is in VMEbus Chassis #1.

To overcome the deadlock condition, the VME-MXI that is arbitrating for the MXIbus terminates the transfer request by sending a BERR* to the VMEbus. The remote MXIbus transfer to the VMEbus can then arbitrate for the VMEbus and complete the transfer. In the situation in Figure 6-2, the VME-MXI in VMEbus Chassis #2 will send the VMEbus BERR* signal to resolve the deadlock condition. A Backoff condition occurs when a MXIbus master must terminate a transfer before acquiring the MXIbus in order to prevent deadlock. A VMEbus interrupt can be generated on this condition.

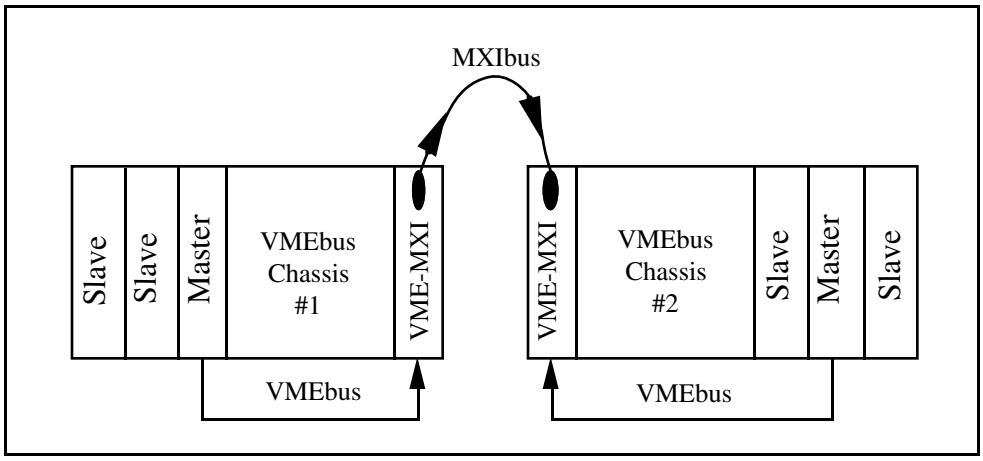


Figure 6-2. Deadlock Situation

If the VME-MXI responds with a VMEbus BERR* to a transfer initiated by a VMEbus device, the transfer was not completed successfully. The following situations are possible reasons for an unsuccessful transfer:

- A MXIbus timeout occurred.
- A local timeout occurred.
- A parity error occurred in the address or data portion of the transfer.
- The transfer attempted to access non-existent memory.

The MXIbus has a built-in block mode capability for high-speed transfers. VMEbus block mode transfers, which are identified by an address modifier code, and which are directed to outward windows through the VME-MXI to the MXIbus, are transparently converted into MXIbus block transfers. Block mode MXIbus operations improve MXIbus performance because a single address is sent at the beginning of a block mode cycle. As block mode transfers can span over the address range of several MXIbus devices, all MXIbus slave devices are responsible for latching the initial address broadcast and for generating the successive addresses to determine if any of the remaining transfers of the block mode operation are directed to the slave. The amount of increment between successive addresses depends on whether the block mode transfer is 8 bits, 16 bits, or 32 bits wide.

MXIbus Slave Mode State Machine

When the VME-MXI is addressed by a remote MXIbus device, the VME-MXI translates MXIbus 8-bit, 16-bit, and 32-bit read and write cycles into VMEbus read and write cycles in A16, A24 or A32 space. The VME-MXI interface responds to 8-bit or 16-bit reads and writes to onboard registers located in MXIbus configuration space.

The VME-MXI is continuously comparing MXIbus addresses and address modifiers to the four mapping windows. The VME-MXI only responds to the address modifier codes listed in Table 6-3. When a transfer involving an address in one of the inward windows is detected, the VME-MXI begins arbitrating for the VMEbus. When the VME-MXI wins the VMEbus, the MXIbus transfer is converted into a VMEbus transfer. The data transfer size information is converted from MXIbus signals to VMEbus signals as shown in Table 6-4. The transfer is complete when the responding VMEbus device sends a DTACK or BERR signal across the MXIbus and the remote MXIbus device releases the address strobe and data strobe.

The VME-MXI circuitry generates and checks parity during the address and data portions of all MXIbus cycles. The VMEbus is not requested if the MXIbus address received has a parity error. Parity is also checked when MXIbus data is written to the VME-MXI slave circuitry. If a parity error occurs, the bad data is not written to the VMEbus and a BERR is sent back to the MXIbus master. The MXIbus PAR* signal is generated and sent during a MXIbus slave read cycle.

When the MXIbus address strobe remains low during multiple data transfers, the VME-MXI interprets the transfer in one of three ways, depending upon the information sent on the address modifier lines and the state of the RMWMODE bit in the MXIbus Status/Control Register:

1. If the address modifiers indicate a VMEbus block mode transfer, the MXIbus transfer is converted directly into a VMEbus block mode transfer, regardless of the state of the RMWMODE bit. MXIbus does not limit the length of the block transfer in any way; however, the VMEbus specification limits VMEbus block transfers to a maximum of 256 bytes in length. The VME-MXI, therefore, will initiate a new block transfer after every 256 bytes of the MXIbus block transfer.

2. If the RMWMODE bit is 0 and the address modifiers sent across the MXIbus indicate a non-VMEbus block mode transfer, the MXIbus transfer is interpreted as a read/modify/write (RMW) cycle and is converted into a VMEbus RMW cycle.

3. If the RMWMODE bit is 1 and the address modifiers sent across the MXIbus indicate a non-VMEbus block mode transfer, the VME-MXI uses onboard 32-bit counters to convert the MXIbus block mode transfer into many VMEbus single cycle transfers. All MXIbus slaves are required to latch the MXIbus address into onboard address counters on the assertion edge of AS* and increment the counters on each trailing edge of DS*.

The length of a MXIbus block transfer is not limited to the address space of a single MXIbus device. A MXIbus master can perform a single block mode transfer to multiple address-consecutive MXIbus slaves. For this reason, each MXIbus slave must continually monitor the address count of all MXIbus block mode transfers and decode the output of the address counters to determine if the block transfer crosses into its inward address window. At any time the transfer can cross into one of the VME-MXI's inward windows, requiring the circuitry to respond to the transfer.

MXIbus Address/Data and Address Modifier Transceivers

The MXIbus address/data transceivers and the associated circuitry multiplex and de-multiplex the MXIbus address and data information from the MXIbus AD[31-0] lines and control the direction of address and data flow. Address and address modifier information from the MXIbus is latched on the rising edge of the MXIbus address strobe. The address is latched into address counters that are incremented on each falling edge of the MXIbus data strobe.

MXIbus specifies trapezoidal bus transceivers to reduce noise and crosstalk in the MXIbus transmission system. These transceivers have open collector drivers that generate precise trapezoidal waveforms with typical rise and fall times of 9 nsec. The trapezoidal shape, due to the constant rise and fall times, reduces noise coupling to adjacent lines. The receiver uses a low pass filter to remove noise in conjunction with a high-speed comparator that differentiates the trapezoidal-shaped signal from the noise.

MXIbus cables are matched impedance cables. Each MXIbus signal line is twisted with a ground line and the impedance is controlled by the thickness of the insulation around the wires. This impedance matching minimizes skew between signals because they travel down the cable at the same speed. Signal reflections are also minimized because the signals travel through the same impedance as they daisy-chain through multiple cables. Termination resistor networks are placed at the first and last MXIbus devices to minimize reflections at the ends of the cable.

MXIbus System Controller Functions

An onboard slide switch sets whether or not the VME-MXI interface board is the MXIbus System Controller. If it is the system controller, the VME-MXI must be the first device in the MXIbus daisy-chain. Onboard arbitration circuitry transparently performs the MXIbus arbitration for the MXIbus chain.

The MXIbus System Controller is also responsible for the MXIbus system timeout. This timeout, typically 100 µsec, begins when a MXIbus data strobe is received and stops when a MXIbus DTACK or BERR is detected. When the timeout expires, the MXIbus System Controller sends a MXIbus BERR to clear the MXIbus system. The VME-MXI powers up with the MXIbus system timeout between 100 and 400 µsec, enabling the system Resource Manager to scan all logical addresses in a reasonable amount of time. When the Resource Manager has finished scanning and configuring the MXIbus system, it should set the LNGMXSCTO bit in the

MXIbus Control Register. When this bit is set, the MXIbus system timeout will be between 10 and 40 msec.

MXIbus Control Signals Transceivers

The MXIbus control signal transceivers control the sending and receiving of the MXIbus control signals address strobe (AS*), data strobe (DS*), transfer size (SIZE*), read/write (WR*), data transfer acknowledge (DTACK*), bus error (BERR*), and parity (PAR*). These signals indicate the beginning and end of a transfer, the size of data involved in the transfer (8, 16, or 32 bits), whether the transfer is a read or write, and whether the transfer was successful.

MXIbus Requester and Arbiter Circuitry

The MXIbus requester and arbiter circuitry is used to request and grant the MXIbus to MXIbus devices. The arbiter circuitry is only active on the VME-MXI if it is configured as the MXIbus System Controller.

All MXIbus masters must have bus request logic for requesting the MXIbus, and the MXIbus System Controller must have bus arbiter logic to grant the bus to requesting masters. Four signals are used for arbitration: bus request (BREQ*), bus grant in (BGIN*), bus grant out (BGOUT*), and bus busy (BUSY*). The MXIbus has a serial, release-on-request arbitration with fairness and bus lock options.

In a serial arbitration scheme, devices request the bus by asserting the BREQ* line. This signal is a wired-OR signal that indicates when one or more MXIbus devices are requesting use of the bus. When the System Controller detects BREQ* active, it grants the bus by driving the bus grant daisy-chain line BGOUT* active. BGOUT* propagates down the daisy-chain to the next device's BGIN* signal. If that device is not driving the BREQ* line, it passes the BGIN* signal on to the next device in the daisy-chain via its BGOUT* line. The first device that is driving BREQ* and receives an active low level on its BGIN* line is the device that is granted the bus. That device does not pass the bus grant signal on the daisy-chain to the next device.

When a requester is granted control of the bus, it drives the BUSY* line active and unasserts BREQ*. The BUSY* signal indicates to the other MXIbus devices that the bus is busy. The master in control of the bus holds BUSY* low until it is finished with the bus. At that time, if no other MXIbus device is driving BREQ*, the master can continue to drive BUSY* until it detects the BREQ* line active.

A VMEbus device can lock the MXIbus so that the device can perform indivisible operations across the MXIbus. When the LOCKED bit in the Local Bus Lock Register is set by a VMEbus device, the VME-MXI interface will not release the MXIbus the next time it is granted the bus (on the next transaction) until the LOCKED bit is cleared by a VMEbus device.

A fairness feature ensures that all requesting devices will be granted use of the bus. If fairness is enabled, a master must refrain from driving BREQ* active after releasing it until it detects BREQ* inactive.

When the VME-MXI is arbitrating for the MXIbus and a remote MXIbus transfer requesting the VMEbus is received, deadlock occurs. The VME-MXI cannot win the MXIbus because another MXIbus device owns it, and that device wants to arbitrate for the VMEbus, which is currently owned by another device. To resolve the conflict, the VME-MXI in the process of arbitrating for the MXIbus terminates its VMEbus transfer by sending a BERR to the VMEbus. The remote MXIbus transfer to the VMEbus can then arbitrate for the VMEbus and complete.

Unless the optional interlocked arbitration mode is used, VME modules must be able to handle the BERR exceptions that occur because of deadlock conditions. In interlocked arbitration mode, only one device owns the VMEbus/MXIbus system at a time. Deadlocks are prevented because there is only one master of the entire system (VMEbus and MXIbus) at a time.

In interlocked arbitration mode, the VMEbus arbiter and the MXIbus arbiter are synchronized so that both buses are tightly coupled at all times. When the VME-MXI receives a VMEbus BGIN* signal, it cannot drive the daisy-chain VMEbus BGOUT* signal until it owns the MXIbus (is driving the MXIbus BUSY* signal). Similarly, when the VME-MXI receives a MXIbus BGIN* signal, it cannot drive the MXIbus BGOUT* lines until it owns the VMEbus (is driving the VMEbus BBSY* signal). When the VME-MXI is driving the VMEbus BBSY* signal, it cannot release the line until it owns the MXIbus. Similarly, when the VME-MXI is driving the MXIbus BUSY* signal, it cannot release the line until it owns the VMEbus. In other words, the VME-MXI cannot release the bus it owns until it gains ownership of the other bus.

For example, if the VME-MXI owns the VMEbus and it receives a VMEbus bus request from another VMEbus device, the VME-MXI continues holding the VMEbus and arbitrating for the MXIbus. When it wins the MXIbus, the VME-MXI can then release the VMEbus so that another VMEbus requester can gain ownership of the VMEbus. Likewise, if the VME-MXI owns the MXIbus and receives a MXIbus request from another device, the VME-MXI continues to hold the MXIbus BUSY* line while it arbitrates for its VMEbus. Once it wins the VMEbus, it can release the MXIbus.

Transparent interoperability between VMEbus chassis is an advantage of interlocked arbitration mode; however, this mode of operation does have disadvantages. In normal operation mode, the VMEbus activity within each chassis is independent of the activity in other chassis except when a device in one chassis accesses a device in another chassis. In interlocked arbitration mode, there can be only one master of the entire VMEbus/MXIbus system at a time. Devices in separate chassis, therefore, cannot run operations in parallel. The global arbitration scheme required by interlocked arbitration mode also adds considerable overhead to each VMEbus access.

In a VMEbus/MXIbus system, some VME-MXIs can be configured for normal operation mode and others for interlocked arbitration mode. The VMEbus chassis configured in interlocked arbitration mode are interlocked with each other and the chassis configured for normal operation can perform transfers in parallel. If no bus masters are in a VMEbus chassis, or if the bus masters communicate only with the slaves in their chassis and never attempt transfers across the MXIbus, a deadlock cannot occur. These VMEbus chassis can be configured for normal operation in a VMEbus/MXIbus system with VMEbus chassis configured for interlocked arbitration mode. Even though PCs with MXIbus interfaces do not support interlocked arbitration mode, they can be installed in a VMEbus/MXIbus system with VMEbus chassis running in interlocked mode.

Appendix A
Specifications

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

Capability Codes

VMEbus

Capability Code	Description
MA32, MA24, MA16	Master Mode A32, A24, and A16 addressing
SA32, SA24, SA16	Slave Mode A32, A24, and A16 addressing
MD32, MD16, MD08(EO)	Master Mode D32, D16, and D08 data sizes
SD32, SD16, SD08(EO)	Slave Mode D32, D16, and D08 data sizes
MBLOCK	Master Mode block transfers
SBLOCK	Slave Mode block transfers
MRMW	Master Mode Read/Modify/Write
SRMW	Slave Mode Read/Modify/Write
PRI	Prioritized arbitration
ROR	Release on Request bus requester
IH	Interrupt Handler
IR	Interrupt Requester
ROAK	Release on Acknowledge interrupter
BTO	Bus Timeout
SC	Optional VMEbus System Controller
IACK	IACK daisy-chain driver

MXIbus

Capability Code	Description
MA32, MA24, MA16	Master Mode A32, A24, and A16 addressing
SA32, SA24, SA16	Slave Mode A32, A24, and A16 addressing
MD32, MD16, MD08(EO)	Master Mode D32, D16, and D08 data sizes
SD32, SD16, SD08(EO)	Slave Mode D32, D16, and D08 data sizes
MBLOCK	Master Mode block transfers
SBLOCK	Slave Mode block transfers
SC	Optional MXIbus System Controller
FAIR	Optional MXIbus fair requester
TERM	Can accept MXIbus termination resistors
IH	Interrupt Handler
IR	Interrupt Requester

Electrical

Source	Typical	Maximum
+5 VDC	5.25 A	7.0 A

Environmental

Component temperature	0° to 70° C operating; -40° to 85° C storage
Relative humidity	10% to 90% noncondensing operating; 0% to 95% noncondensing storage
Emissions	FCC Class A
Safety	Not applicable
Shock and Vibration	Not applicable

Physical

Board size	9.187 in. by 6.299 in. (233.35 mm by 160 mm)
Connectors	Single fully implemented MXIbus connector Single INTX connector (on boards equipped with optional INTX daughter card)
Slot Requirements	Single slot
VXI Keying Class	Class 1 TTL

Fully compatible with VMEbus specification

Reliability

MTBF	Contact Factory

Requirements

A16 Space	64 bytes

Timing

Master Mode

Transfer Type	Transfer Rate
Write	675.5 nsec
Read	625.5 nsec
Block Write	320 nsec
Block Read	270 nsec

Slave Mode

Transfer Type	Transfer Rate
Write	381 nsec
Read	381 nsec
Block Write	238 nsec
Block Read	238 nsec

Other

Daisy-Chain Delay 120 nsec max
(Passing GIN to GOUT or GOUT generation from System Controller)

Appendix B
Mnemonics Key

This appendix contains a mnemonics key that defines the mnemonics (abbreviations) used throughout this manual for signals and terminology specific to MXIbus, VMEbus, VXIbus, and the daughter card, and for bit descriptions.

The mnemonic types in the key that follows are abbreviated to mean the following:

B	Bit
MBS	MXIbus Signal
MXI	MXIbus Terminology
VBS	VMEbus Signal
VME	VMEbus Terminology

Mnemonic	Type	Definition
A		
A01	VBS	VME Address Line 1
A16BASE[7-0]	B	A16 Window Base Address
A16DIR	B	A16 Window Direction
A16EN	B	A16 Window Enable
A16HIGH[7-0]	B	A16 Window Upper Bound
A16LOW[7-0]	B	A16 Window Lower Bound
A16SIZE[2-0]	B	A16 Window Size
A24BASE[7-0]	B	A24 Window Base Address
A24DIR	B	A24 Window Direction
A24EN	B	A24 Window Enable
A24HIGH[7-0]	B	A24 Window Upper Bound
A24LOW[7-0]	B	A24 Window Lower Bound
A24SIZE[2-0]	B	A24 Window Size
A32BASE[7-0]	B	A32 Window Base Address
A32DIR	B	A32 Window Direction
A32EN	B	A32 Window Enable
A32HIGH[7-0]	B	A32 Window Upper Bound
A32LOW[7-0]	B	A32 Window Lower Bound
A32SIZE[2-0]	B	A32 Window Size
A[31-1]	VBS	VME Address Lines 31 through 1
ACCDIR	B	Access Direction
ACFAIL	B	VMEbus ACFAIL Status
ACFAIL*	VBS	VME ACFAIL Signal
ACFAILIE	B	VMEbus ACFAIL Interrupt Enable
ACFAILIN	B	Extended ACFAIL Inward
ACFAILINT	B	VMEbus ACFAIL Interrupt Status
ACFAILOUT	B	Extended ACFAIL Outward
AD[31-0]	MBS	MXIbus Address/Data Lines 31 through 0
ADDR	B	Address Space
AM[4-0]	MBS	MXIbus Address Modifier Lines
AM[5-0]	VMS	VMEbus Address Modifer Lines
AS*	VBS/MBS	Address Strobe
B		
BBSY*	VBS	VMEbus Busy Line
BERR*	VBS/MBS	Bus Error
BG[3-0]IN*	VBS	VMEbus Bus Grant In
BG[3-0]OUT*	VBS	VMEbus Bus Grant Out
BKOFF	B	Backoff Status
BKOFFIE	B	Backoff Interrupt Enable
BR[3-0]*	VME	VMEbus Bus Request Lines 3 through 0
BREQ*	VBS/MBS	Bus Request
BTO	VBS/MBS	Bus Timeout
BUSY*	MBS	Bus Busy
C		
CMODE	B	Comparison Mode

Mnemonic	Type	Definition
D		
D[31-0]	VBS	VMEbus Data Lines 31 through 0
DEVCLASS	B	VXIbus Device Class
DIRQ[7-1]	B	Drive IRQ Lines
DS*	MBS	MXIbus Data Strobe
DS0*	VBS	VMEbus Data Strobe 0
DS1*	VBS	VMEbus Data Strobe 1
DSYSFAIL	B	Drive SYSFAIL
DSYSRST	B	Drive SYSRESET
DTACK*	VBS/MBS	Data Transfer Acknowledge
E		
EDTYPE	B	Extended Device Type
EINT[7-1]DIR	B	Extended Interrupt Direction
EINT[7-1]EN	B	Extended Interrupt Enable
EO	VME	Even and Odd Transfers
F		
FAIR	B	VME-MXI Fairness Status
G		
GIN	MBS	Daisy-chain Grant In
GOUT	MBS	Daisy-chain Grant Out
I		
I[15-0]	B	Interrupt Acknowledge Status/ID
IACK*	VME	VMEbus Interrupt Acknowledge
IACKIN*	VME	VMEbus Interrupt Acknowledge Daisy-Chain Input
IACKOUT*	VME	VMEbus Interrupt Acknowledge Daisy-Chain Output
INTLCK	B	Interlocked Bus Operation
INTX	MXI	Interrupt and Timing Extension Option
IRQ*	MBS	MXIbus Interrupt Request
IRQ[7-1]*	VBS	VMEbus Interrupt Request Lines
L		
LABASE[7-0]	B	Logical Address Window Base Address
LADD[7-0]	B	Logical Address Status
LADIR	B	Logical Address Window Direction
LAEN	B	Logical Address Window Enable
LAHIGH[7-0]	B	Logical Address Window Upper Bound
LALOW[7-0]	B	Logical Address Window Lower Bound
LASIZE[2-0]	B	Logical Address Window Size
LINT[3-1]	B	Local Interrupt Line
LNGMXSCTO	B	Long MXIbus System Controller Timeout

Mnemonic	Type	Definition
LOCKED	B	Lock MXIbus or VMEbus
LWORD*	VBS	VMEbus Longword

M

MANID	B	Manufacturer ID
MIRQ[7-1]DIR	B	MXIbus IRQ Direction
MIRQ[7-1]EN	B	MXIbus IRQ Enable
MODEL	B	Model Code
MXACFAILEN	B	MXIbus ACFAIL Enable
MXACFAILINT	B	MXIbus ACFAIL Status
MXBERR	B	MXIbus Bus Error Status
MXISC	B	MXIbus System Controller Status
MXSCTO	B	MXIbus System Controller Timeout Status
MXSRSTEN	B	MXIbus SYSRESET Enable
MXSRSTINT	B	MXIbus SYSRESET Status
MXSYSFINT	B	MXIbus SYSFAIL Status

P

PAR*	MBS	MXIbus Parity Line
PARERR	B	Parity Error
PASS	B	Passed
PRI ARBITER	VME	VMEbus Prioritized Arbiter

R

RDY	B	Ready
RESET	B	Soft Reset
RMWMODE	B	Read/Modify/Write Select Mode
ROAK	VME	Release on Interrupt Acknowledge

S

S[15-0]	B	Status/ID
SIZE	MBS	MXIbus Size Signal
Status/ID	VME	VMEbus Interrupt Status/Identification Data
SUBCLASS	B	Manufacturer Subclass
SYSCLK	VME	VMEbus System Clock
SYSFAIL	B	VMEbus SYSFAIL Status
SYSFAIL*	VME	System Failure
SYSFAILIE	B	VMEbus SYSFAIL Interrupt Enable
SYSFAILIN	B	Extended SYSFAIL Inward
SYSFAILINT	B	VMEbus SYSFAIL Interrupt Status
SYSFAILOUT	B	Extended SYSFAIL Outward
SYSFIN	B	SYSFAIL Input Enable
SYSFOUT	B	SYSFAIL Output Enable
SYSRESET*	VME	System Reset
SYSRSTIN	B	Extended SYSRESET Inward
SYSRSTOUT	B	Extended SYSRESET Outward

Mnemonic	Type	Definition
T		
TERMPWR	MXI	Terminator Power
V		
VERSION	B	VME-MXI Version Number
W		
WR	MBS	MXIbus Write
WRITE*	VBS	VMEbus Write

Appendix C
Connector Descriptions

This appendix describes the connector pin assignments for the MXIbus connector and the INTX connector.

MXIbus Connector

The MXIbus signals are assigned to the device connector as shown in Figure C-1 and Table C-1.

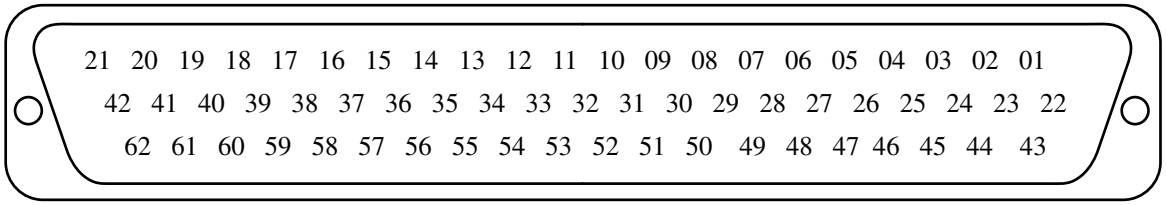


Figure C-1. MXIbus Connector

Table C-1. MXIbus Connector Signal Assignments

Pin	Signal Name	Pin	Signal Name	Pin	Signal Name
1	*AM4**	22	*AD15**	43	*PAR**
2	*AM3**	23	*AD14**	44	*SIZE**
3	*AM2**	24	*AD13**	45	*BREQ**
4	*AM1**	25	*AD12**	46	*BUSY**
5	*AM0**	26	*AD11**	47	*GND*
6	*AD31**	27	*AD10**	48	*GND*
7	*AD30**	28	*AD09**	49	*GND*
8	*AD29**	29	*AD08**	50	*GND*
9	*AD28**	30	*AD07**	51	*GND*
10	*AD27**	31	*AD06**	52	*GND*
11	*AD26**	32	*AD05**	53	*GND*
12	*AD25**	33	*AD04**	54	*GND*
13	*AD24**	34	*AD03**	55	*GND*
14	*AD23**	35	*AD02**	56	*GND*
15	*AD22**	36	*AD01**	57	*GND*
16	*AD21**	37	*AD00**	58	*GND*
17	*AD20**	38	*DS**	59	*GOUT**
18	*AD19**	39	*AS**	60	*GIN**
19	*AD18**	40	*WR**	61	*IRQ**
20	*AD17**	41	*DTACK**	62	*TERMPWR*
21	*AD16**	42	*BERR**		

The MXIbus defines 49 active signals, 12 ground lines, and 1 line for terminator power. Table C-2 describes the signals on the MXIbus connector and groups them in five categories.

Table C-2. MXIbus Signal Groupings

Category	Description	Signal Name	Lines	Pin Numbers
Address/Data	Address/Data	*AD[31-00]**	32	6-37
	Address Modifier	*AM[4-0]**	5	1-5
	Address Strobe	*AS**	1	39
	Transfer Size	*SIZE**	1	44
	Read/Write	*WR**	1	40
	Data Strobe	*DS**	1	38
	Data Acknowledge	*DTACK**	1	41
	Parity	*PAR**	1	43
Arbitration	MXIbus Busy	*BUSY**	1	46
	MXIbus Request	*BREQ**	1	45
	MXIbus Grant In	*GIN**	1	60
	MXIbus Grant Out	*GOUT**	1	59
Interrupt	Interrupt Request	*IRQ**	1	61
Utility	MXIbus Error	*BERR**	1	42
Power	Ground	*GND*	12	47-58
	Terminator Power	*TERMPWR*	1	62

Notice that there are 12 ground contacts on the connector. The 48 twisted ground lines in the cable are generated from these lines under the cable connector hood. Also notice that although there are two connector contacts required for the GIN*–GOUT* daisy-chain, only one signal line is required in the cable assembly.

For more information, refer to the MXIbus specification.

INTX Connector

The INTX connector is used only on VME-MXIs with the INTX daughter card option. The INTX signals are assigned to the device connector as shown in Figure C-2 and Table C-3.

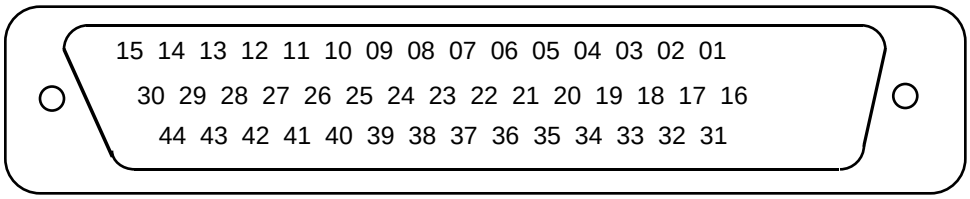


Figure C-2. INTX Connector

Table C-3. INTX Connector Signal Assignments

Pin	Signal Name	Pin	Signal Name	Pin	Signal Name
1	*Reserved*	16	*Reserved*	31	*Reserved*
2	*SYSFAIL**	17	*Reserved*	32	*IRQ3**
3	*GND*	18	*IRQ7**	33	*GND*
4	*ACFAIL**	19	*GND*	34	*IRQ2**
5	*GND*	20	*IRQ6**	35	*GND*
6	*SYSRESET**	21	*GND*	36	*IRQ1**
7	*GND*	22	*IRQ5**	37	*GND*
8	*Reserved*	23	*GND*	38	*Reserved*
9	*Reserved*	24	*IRQ4**	39	*Reserved*
10	*Reserved*	25	*GND*	40	*Reserved*
11	*Reserved*	26	*Reserved*	41	*Reserved*
12	*Reserved*	27	*Reserved*	42	*Reserved*
13	*Reserved*	28	*Reserved*	43	*Reserved*
14	*Reserved*	29	*Reserved*	44	*TERMPWR*
15	*Reserved*	30	*Reserved*		

Table C-4. INTX Signal Groupings

Category	Description	Signal Name	Lines	Type §
Interrupts	INTX Interrupt	*IRQ7-1**	7	O.C.
Utility Lines	INTX SYSRESET INTX SYSFAIL INTX ACFAIL	*SYSRESET** *SYSFAIL** *ACFAIL**	1 1 1	O.C. O.C. O.C.
Power	Ground Termination Power	*GND* *TERMPWR*	10 1	 ---
Reserved	Reserved Pin	*Reserved*	23	---

§ The type of signal grouping is abbreviated in the preceding table as follows:

O.C. represents an Open Collector (trapezoidal) DS3862 or DS3662.
Diff represents a Differential RS485.

Appendix D
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.

Corporate Headquarters
(512) 795-8248
Technical support fax: (800) 328-2203
(512) 794-5678

Branch Offices	Phone Number	Fax Number
Australia	(03) 879 9422	(03) 879 9179
Austria	(0662) 435986	(0662) 437010-19
Belgium	02/757.00.20	02/757.03.11
Denmark	45 76 26 00	45 76 71 11
Finland	(90) 527 2321	(90) 502 2930
France	(1) 48 14 24 00	(1) 48 14 24 14
Germany	089/741 31 30	089/714 60 35
Italy	02/48301892	02/48301915
Japan	(03) 3788-1921	(03) 3788-1923
Netherlands	03480-33466	03480-30673
Norway	32-848400	32-848600
Spain	(91) 640 0085	(91) 640 0533
Sweden	08-730 49 70	08-730 43 70
Switzerland	056/20 51 51	056/27 00 25
U.K.	0635 523545	0635 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. Use additional pages if necessary.

Name _____

Company _____

Address _____

_____

Fax (___) _____ Phone (_) _____

Computer brand _____ Model _____ Processor _____

Operating system _____

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 _____

_____

_____

_____

VME-MXI Hardware and Software Configuration Form

Record the settings and revisions of your hardware and software on the line located to the right of each item. Complete this form each time you revise your hardware or software 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 Module Part Number 180960 -____
- Serial Number _____
- Revision Number _____
- VMEbus System Controller? (S5) _____
- MXIbus Logical Address (U94) _____
- VMEbus Request Level (3, 2, 1, or 0) (W2, W3, W4, W5 and W6) _____
- VMEbus Timeout Value (W8) _____
- VMEbus Timeout Chain Position (W1) _____
- Interlocked Bus Cycle Mode/Normal Operating Mode (S2) _____
- MXIbus System Controller? (S4) _____
- MXIbus System Controller Timeout Value (W7) _____
- MXIbus Fair Requester? (S3) _____
- Reset Signal Select (SYSRESET* or ACFAIL*) (S1) _____
- MXIbus Terminators Installed? _____
- INTX Terminators Installed? (INTX only) _____

Other Products

- Other MXIbus Devices in System

Manufacturer	Model	Function	Slot	Logical Address

- Other VXIbus or VMEbus Devices

Manufacturer	Model	Function	Slot	Logical Address

- Address Space(s) and Size(s) of Other Devices _____

  _____

  _____

  _____

- VMEbus Interrupt Level(s) of Other Devices _____

  _____

  _____

- VMEbus Chassis Make and Model _____

  _____

  _____

- VXIbus/MXIbus Resource Manager
  (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 User Manual**

Edition Date: **November 1993**

Part Number: **320330-01**

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, MS 53-02
Austin, TX 78730-5039

Fax to: Technical Publications
National Instruments Corporation
MS 53-02
(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
%	percent

A

A	amperes
A16 Space	VXIbus address space equivalent to the VME 64 kilobyte *short* address space. In VXI, the upper 16 kilobytes of A16 space is allocated for use by VXI devices configuration registers. This 16 kilobyte region is referred to as VXI Configuration space.
A24 Space	VXIbus address space equivalent to the VME 16 megabyte *standard* address space.
A32 Space	VXIbus address space equivalent to the VME 4 gigabyte *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 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 and mode (supervisory/nonprivileged, data/ program/block) in which a data transfer is to take place.

Address Space	A set of 2^n memory locations differentiated from other such sets in VXI/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.
Address Strobe	A VMEbus or MXIbus signal used to indicate that valid addressing information exists and may be decoded.
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 of a bus.
Asynchronous	Not synchronized; not controlled by time signals.

B

Backplane	An assembly, typically a printed circuit board, with 96-pin connectors and signal paths that bus the connector pins. A C-size VXIbus system will have two sets of bused connectors called J1 and J2. A D-size VXIbus system will have three sets of bused connectors called J1, J2, and J3.
Base Address	A specified address that is combined with a *relative* address to determine the *absolute* address of a data location. All VXI address windows have an associated base address for their assigned VXI 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). The BERR* signal is also used in VXI for certain protocol implementations such as writes to a full Signal register and synchronization under the Fast Handshake Word Serial Protocol.
Binary	A numbering system with a base of 2.
Bit	Binary digit. The smallest possible unit of data: a two-state, true/false, 1/0 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
Chassis Extender	A device such as the VME-MXI that interfaces a VMEbus chassis to an interconnect bus. It routes bus transactions from the VMEbus to the interconnect bus or vice versa. A chassis extender has a set of registers that defines the routing mechanisms for data transfers, interrupts, and utility bus signals, and has optional VMEbus Slot 1 capability.
Clearing	Replacing the information in a register, storage location, or storage unit with zeros or blanks.
Commander	A Message-Based device which is also a bus master and can control one or more Servants.
Command	A directive to a device. In VXI, three types of commands exists: In Word Serial Protocol, a 16-bit imperative to a servant from its Commander (written to the Data Low register); In Shared Memory Protocol, a 16-bit imperative from a client to a server, or vice versa (written to the Signal register); In Instrument devices, an ASCII-coded, multi-byte directive.
Communications Registers	In Message-Based devices, a set of registers that are accessible to the device's Commander and are used for performing Word Serial Protocol communications. Not present in Register-Based devices such as the VME-MXI.
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, all accessible from the P1 connector on the VMEbus.
Configuration Space	The upper 16 kilobytes of A16 space in which the configuration registers for VXI and MXIbus devices exist.

Controller	An intelligent device (usually involving a CPU) that is capable of controlling other devices.

D

DACK	DMA Acknowledge
Daisy-Chain	A signal line used to propagate a signal level from board to board on a priority basis, starting with the first slot and ending with the last slot.
Data Strobe	A signal used to inform a slave that valid data exists on the bus or used to request that a slave place data 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.
Daughter Card	A board that plugs directly into the expansion connectors of another board. The VME-MXI is available with or without a daughter card option called the INTX card.
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.
DRQ	DMA Request
DTACK	Data Transfer Acknowledge
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 VME device initially responds at Logical Address 255 when its MODID line is asserted. A MXIbus device responds at Logical Address 255 when its Bus Grant In line is asserted. The Resource Manager subsequently assigns it a new logical address, which the device responds to until powered down. The VME-MXI cannot be dynamically configured.

E

EMI	electromagnetic interference

Extended Class Device	A class of VXIbus device defined for future expansion of the VXIbus specification. These devices have a subclass register within their configuration space that defines the type of extended device. The VME-MXI is an extended class mainframe extender device.

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.
FCC	Federal Communications Commission

G

GIN	Daisy-chain Grant In signal
GOUT	Daisy-chain Grant Out signal
GPIB	General Purpose Interface Bus; the industry standard IEEE 488 bus.

H

Hard Reset	Occurs when the mainframe is powered on and when the VMEbus SYSRESET signal is active. A hard reset clears all the registers on the VME-MXI.
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
IC	Integrated Circuit
IEEE	Institute of Electrical and Electronics Engineers
IEEE 1014	The VMEbus specification.
in.	inches
I/O	input/output; the techniques, media, and devices used to achieve communication between entities.

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.
INTX	Interrupt and Timing Extension; a daughter card option that plugs into the two daughter card connectors on the VME-MXI. It extendes the seven VMEbus interrupt lines and the VMEbus reset signals SYSRESET*, SYSFAIL*, and ACFAIL*.
ISA	Industry Standard Architecture

L

Latch	To sample a signal and remember its value.
LED	light-emitting diode
Logical Address	An 8-bit number that uniquely identifies the location of a MXIbus device's configuration registers in a system. The A16 base address of a device is C000h + Logical Address * 40h.
LSB	Least Significant Bit (bit 0)

M

MB	megabytes of memory
m	meters
Mapping	Establishing a range of addresses for a one-to-one correspondence between each address in the window and an access in VME address space.
Master	A functional part of a MXI/VME/VXIbus device that initiates data transfers on the bus. A transfer can be either a read or a write.
Master-Mode Operation	A device is in master mode if it has initiated and is performing a bus cycle.

Memory Device	A device that not only has configuration registers, but also has memory that is accessible through addresses on the VME/VXI 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.
MODID	Module Identification lines; a set of signal lines on the VXI backplane that VXI systems use to identify which modules are located in which slots in the mainframe.
MSB	Most Significant Bit (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 Slot 0 devices but only one global Resource Manager.
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

NI-VXI	The National Instruments bus interface software for VME/VXIbus systems.
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. The VME-MXI should not be used as a VXIbus Slot 0 device.
Normal Operating Mode	Contrasted with *Interlocked Arbitration Mode*; in this mode there can be multiple masters operating simultaneously in the VXIbus/MXIbus system. Vulnerable to deadlock situations.

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. In VXI, the P2 connector adds trigger, MODID, and CLK10 signals.
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.
PC	Personal Computer
PRI	Priority
Privileged Access	See *Supervisory Access.*
Propagation	The transmission of signals 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. The VME-MXI is a Register-Based device.
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 VXI interrupter which always deasserts its interrupt line in response to an IACK cycle on the VXIbus.
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

sec	seconds
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.
Slave	A functional part of a MXI/VME/VXIbus 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, arbitration for data transfers across the backplane, and MODID functions. Additional Slot 0 services include trigger control. Installing such a device into any other slot can damage the device, the VXIbus backplane, or both. The VME-MXI cannot function as a VXIbus Slot 0 device.
SMB	Sub-miniature BNC; a miniature connector for coaxial cable connections.
Soft Reset	Occurs when the RESET bit in the Status/Control Register of the VME-MXI 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.
Statically Configured Device	A device whose logical address cannot be set through software; that is, it is not dynamically configurable. The VME-MXI is a statically configured device.
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.
SYSFAIL*	System Fail; a signal that indicates when a failure has occurred in the system; can be generated by any board in the system.
SYSRESET*	System Reset; a signal that indicates 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.

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.

V

V	volts
VDC	volts direct current
VMEbus	Versa Module Eurocard or IEEE 1014; the IEEE Standard for a Versatile Backplane Bus.
VMEbus System Controller	A device configured for installation in Slot 1 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 VMEbus backplane, or both.
VXIbus	VMEbus Extensions for Instrumentation

W

Write	Copying data to a storage device.
Word Serial Protocol	The simplest required communication protocol supported by Message-Based devices in the VXIbus system. It utilizes the A16 communication registers to transfer data using a simple polling handshake method.

Index

A

A16 address window, 2-6
A16 resources, 3-16
A16 address map, 5-21 to 5-34
    amount of A16 space allocated for all size values, 5-21
    example VMEbus/MXIbus system, 5-24
    worksheets, 5-29 to 5-34
        examples, 5-26 to 5-28
A16 Window Map register, 4-13
A16BASE[7-0] bits, 4-14
A16DIR bit, 4-13
A16EN bit, 4-13
A16HIGH[7-0] bits, 4-15
A16LOW[7-0] bits, 4-15
A16SIZE[2-0] bits, 4-14
A24 address window, 2-6
A24 and A32 addressing windows, 5-38
A24 Window Map register, 4-17
A24BASE[7-0] bits, 4-18
A24DIR bit, 4-17
A24EN bit, 4-17
A24HIGH[7-0] bits, 4-19
A24LOW[7-0] bits, 4-19
A24SIZE[2-0] bits, 4-18
A32 address window, 2-6
A32 Window Map register, 4-21
A32BASE[7-0] bits, 4-22
A32DIR bit, 4-21
A32EN bit, 4-21
A32HIGH[7-0] bits, 4-23
A32LOW[7-0] bits, 4-23
A32SIZE[2-0] bits, 4-22
ACCDIR bits, 4-7
ACFAIL, 2-6, 2-9, 6-2
    reset signal selection, 3-15
ACFAIL bit, 4-38
ACFAILIE bit, 4-38
ACFAILIN bit, 4-26
ACFAILINT bit, 4-38
ACFAILOUT bit, 4-26
ADDR bits, 4-4
address, logical, MXIbus, 3-5 to 3-6
    switch settings, 3-6
address maps. *See* A16 address map; logical address map.
address modifier line map, 6-6
address modifier transceivers
    MXIbus, 2-7, 6-10
    VMEbus, 2-6, 6-1
address modifiers, VMEbus, transfer responses, 6-7
address transceivers, VMEbus, 2-6, 6-1
address windows, 2-6
address/data transceivers, MXIbus, 2-7, 6-10
addresses, for VMEbus interrupt levels, 6-4
addressing windows, 6-5
arbiter circuitry
    MXIbus, 2-7, 6-11 to 6-12
    VMEbus, 2-6, 6-2

B

bit descriptions
    0, 4-9, 4-17, 4-21, 4-25, 4-31, 4-38
    1, 4-7, 4-10, 4-14, 4-18, 4-22, 4-26, 4-30, 4-33, 4-36
    A16BASE[7-0], 4-14
    A16DIR, 4-13
    A16EN, 4-13
    A16HIGH[7-0], 4-15
    A16LOW[7-0], 4-15
    A16SIZE[2-0], 4-14
    A24BASE[7-0], 4-18
    A24DIR, 4-17
    A24EN, 4-17
    A24HIGH[7-0], 4-19
    A24LOW[7-0], 4-19
    A24SIZE[2-0], 4-18
    A32BASE[7-0], 4-22
    A32DIR, 4-21
    A32EN, 4-21
    A32HIGH[7-0], 4-23
    A32LOW[7-0], 4-23
    A32SIZE[2-0], 4-22
    ACCDIR, 4-7
    ACFAIL, 4-38
    ACFAILIE, 4-38
    ACFAILIN, 4-26
    ACFAILINT, 4-38
    ACFAILOUT, 4-26
    ADDR, 4-4
    BKOFF, 4-38
    BKOFFIE, 4-38

CMODE, 4-30
DEVCLASS, 4-4
DIRQ[7-1], 4-39
DSYSFAIL, 4-30
DSYSRST, 4-31
EINT[7-1]DIR, 4-25
EINT[7-1]EN, 4-25
FAIR, 4-31
I[15-0], 4-41
INTLCK, 4-30
IRQ[7-1], 4-39
LABASE[7-0], 4-10
LADD[7-0], 4-36
LADIR, 4-10
LAEN, 4-9
LAHIGH[7-0], 4-11
LALOW[7-0], 4-11
LASIZE[2-0], 4-10
LINT[3-1], 4-37
LNGMXSCTO, 4-32
LOCKED, 4-33
MANID, 4-5
MIRQ[7-1]DIR, 4-34
MIRQ[7-1]EN, 4-34
MODEL, 4-6
MXACFAILEN, 4-31
MXACFAILINT, 4-31
MXBERR, 4-32
MXISC, 4-31
MXSCTO, 4-30
MXSRSTEN, 4-31
MXSRSTINT, 4-31
MXSYSFINT, 4-32
PARERR, 4-32
PASS, 4-8
RDY, 4-8
RESET, 4-8
RMWMODE, 4-29
S[15-0], 4-40
SUBCLASS, 4-28
SYSFAIL, 4-38
SYSFAILIE, 4-38
SYSFAILIN, 4-26
SYSFAILINT, 4-39
SYSFAILOUT, 4-27
SYSFIN, 4-34
SYSFOUT, 4-34
SYSRSTIN, 4-27
SYSRSTOUT, 4-27
VERSION, 4-8
BKOFF bit, 4-38
BKOFFIE bit, 4-38
block diagram, VME-MXI, 2-5
BTO, 3-7
bus extender registers. *See* registers.
bus signals, VME-MXI, 2-1 to 2-2
bus timeout unit. *See* BTO.

C

cable, MXIbus
dual-ended, 3-25
single-ended, 3-24
capability codes, A-1 to A-2
MXIbus, A-2
VMEbus, A-1
CMODE bit, 4-9, 4-13, 4-17, 4-21
description, 4-30
configuration
interlocked arbitration mode, 3-12 to 3-13
switch settings, 3-13
MXIbus
fairness option, 3-14 to 3-15
logical address, 3-5 to 3-6
System Controller, 3-13
reset signal switch settings, 3-15
VME-MXI, 3-1 to 3-7
VMEbus
request level, 3-6 to 3-7
System Controller, 3-4
timeout value, 3-7
configuration registers, 2-6, 6-5.
See registers.
connectors, C-1 to C-4
INTX, C-3 to C-4
signal assignments, C-3
signal groupings, C-4
MXIbus, C-1 to C-2
signal assignments, C-1
signal groupings, C-2
contents of VME-MXI kit, 1-6
control signal transceivers
MXIbus, 2-7, 6-11
VMEbus, 2-6, 6-1
customer communication, *xii*, D-1

D

daisy-chain
example, MXIbus and INTX, 3-21
MXIbus, 3-16 to 3-17
data transceivers, VMEbus, 2-6, 6-1
daughter card. *See also* INTX.
block diagram, 2-8

connection, 2-7
deadlock, 6-8
DEVCLASS bits, 4-4
Device Type register, 4-6
DIRQ[7-1] bits, 4-39
DSYSFAIL bit, 4-30
DSYSRST bit, 4-31
dual-ended cable, MXIbus, 3-25

E

EINT[7-1]DIR bits, 4-25
EINT[7-1]EN bits, 4-25
electrical characteristics of VME-MXI, 2-1 to 2-2
electrical specifications, A-2
environmental specifications, A-2

F

FAIR bit, 4-31
fair requester, MXIbus, switch settings, 3-15
fairness option, MXIbus, 3-14 to 3-15
front panel, 1-5

H

hard reset, 4-1
how to use this manual, *xii*

I

I[15-0] bits, 4-41
ID register, 4-4
installation, VME-MXI, 3-16 to 3-26
  A16 resources, 3-16
  daisy-chain example, MXIbus and INTX, 3-21
  INTX, 3-20
    cable, connecting, 3-22
    removing the card, 3-20
    termination, 3-19 to 3-21
  MXIbus
    cable, connecting, 3-23 to 3-25
    termination, 3-16 to 3-19
  step-by-step, 3-21 to 3-22
  system power cycling requirements, 3-25 to 3-26
interlocked arbitration mode, 3-12 to 3-13
  switch settings, 3-13
interrupt circuitry, 2-6, 6-2 to 6-4
interrupt control, 2-9
interrupt levels, VMEbus, addresses, 6-4
Interrupt Status/Control register, 4-37
INTLCK bit, 4-30
INTX
  cable, connecting, 3-22
  connector, C-3 to C-4
    signal assignments, C-3
    signal groupings, C-4
  installing, 3-20
  parts locator diagram, 3-19
  registers, 2-8
  removing the card, 3-20
  termination, 3-19 to 3-21
INTX Interrupt Configuration register, 4-25
INTX Utility Configuration register, 4-26
IRQ Acknowledge registers, 4-41
IRQ[7-1] bits, 4-39

J

jumper settings
  System Controller
    timeout value, 3-14
  VMEbus timeout, 3-9
    value, 3-8

L

LA address window, 2-6
LABASE[7-0] bits, 4-10
LADD[7-0] bits, 4-36
LADIR bit, 4-10
LAEN bit, 4-9
LAHIGH[7-0] bits, 4-11
LALOW[7-0] bits, 4-11
LASIZE[2-0] bits, 4-10
line map, address modifier, 6-6
LINT[3-1] bits, 4-37
LNGMXSCTO bit, 4-32
LOCKED bit, 4-33
logical address map, MXIbus, 5-1 to 5-20
  base/size configuration format, 5-3
    address range allocation for different size values, 5-4
    base and size combinations, 5-3
    high/low, 5-5
  example VMEbus/MXIbus system, 5-8
    diagram, 5-9

required logical addresses, 5-8
MXIbus RM in a VMEbus chassis, 5-2
MXIbus RM on a PC, 5-2
planning, 5-5
worksheets, 5-13 to 5-20
examples, 5-10 to 5-12
logical address, MXIbus, 3-5 to 3-6
switch settings, 3-6
Logical Address Window register, 4-9
logical address window, RM, 5-35 to 5-38
example, 5-36 to 5-38

M

MANID bits, 4-5
master mode state machine, MXIbus, 2-6, 6-5 to 6-9
MIRQ[7-1]DIR bits, 4-34
MIRQ[7-1]EN bits, 4-34
mnemonics key, B-1 to B-5
MODEL bits, 4-6
MXACFAILEN bit, 4-31
MXACFAILINT bit, 4-31
MXBERR bit, 4-32
MXIbus
A16 address window, 2-6
A24 address window, 2-6
A32 address window, 2-6
arbiter circuitry, 2-7, 6-11 to 6-12
cable, connecting, 3-23 to 3-25
dual-ended, 3-25
single-ended, 3-24
capability codes, A-2
configuration registers, 4-4
connectors, C-1 to C-2
signal assignments, C-1
signal groupings, C-2
daisy-chain, 3-16 to 3-17
fair requester switch settings, 3-15
fairness option, 3-14 to 3-15
interrupt circuitry, 2-6
LA address window, 2-6
logical address, 3-5 to 3-6
switch settings, 3-6
parity check and generation, 2-6, 6-4
registers, 4-29. *See also* registers.
requester circuitry, 2-7, 6-11 to 6-12
RM operation, 5-35 to 5-38
A24 and A32 addressing windows, 5-38
logical address window, 5-35 to 5-38
state machines
master mode, 2-6, 6-5 to 6-9
slave mode, 2-7, 6-9 to 6-10
System Controller, 3-13, 6-10 to 6-11
functions, 2-7
switch settings, 3-13
timeout, 3-13
system power cycling requirements, 3-25 to 3-26
termination, 3-16 to 3-19
networks, 3-16 to 3-19
SIP resistor networks, 3-17
TERMPWR connection, 3-17
transceivers
address modifier, 2-7, 6-10
address/data, 2-7, 6-10
control signals, 2-7, 6-11
requirements, 2-2
MXIbus IRQ Configuration register, 4-34
MXIbus Lock register, 4-33
MXIbus Status/Control register, 4-29
MXISC bit, 4-31
MXSCTO bit, 4-30
MXSRSTEN bit, 4-31
MXSRSTINT bit, 4-31
MXSYSFINT bit, 4-32

O

optional equipment, 1-6 to 1-7
organization of this manual, *xi*

P

PARERR bit, 4-32
parity check and generation, MXIbus, 2-6, 6-4
parts locator diagram
INTX, 3-19
VME-MXI, 3-2
with INTX, 3-3
PASS bit, 4-8
physical specifications, A-3
pin assignments, C-1 to C-4
programming considerations, 5-1 to 5-38
system configuration, 5-1 to 5-35

R

RDY bit, 4-8
Read LA register, 4-36

register maps, 4-1, 4-2 to 4-3
register sizes, 4-1
registers, 4-1 to 4-41
  configuration, 2-6
  INTX, 2-8
  MXIbus configuration, 4-4 to 4-8
    Device Type, 4-6
    ID, 4-4
    Status/Control, 4-7
  MXIbus defined, 4-29 to 4-41
    Interrupt Status/Control, 4-37
    IRQ Acknowledge, 4-41
    MXIbus IRQ Configuration, 4-34
    MXIbus Lock, 4-33
    MXIbus Status/Control, 4-29
    Read LA, 4-36
    Status/ID, 4-40
  VMEbus extender, 4-9 to 4-28
    A16 Window Map, 4-13
    A24 Window Map, 4-17
    A32 Window Map, 4-21
    INTX Interrupt Configuration, 4-25
    INTX Utility Configuration, 4-26
    Logical Address Window, 4-9
    Subclass, 4-28
related documentation, *xii*
reliability specification, A-3
request level, VMEbus, 3-6 to 3-7
requester circuitry
  MXIbus, 2-7, 6-11 to 6-12
  VMEbus, 2-6, 6-2
requester jumper settings, VMEbus, 3-7
requirements specification, A-3
reserved bits. *See* 0 bit; 1 bits.
reset
  hard, 4-1
  soft, 4-1
RESET bit, 4-8
reset signal switch settings, 3-15
  ACFAIL, 3-15
  SYSRESET, 3-15
RM operation, MXIbus. *See* MXIbus, RM operation.
RMWMODE bit, 4-29

S

S[15-0] bits, 4-40
signal assignments
  INTX, C-3
  MXIbus, C-1
signal groupings
  INTX, C-4
  MXIbus, C-2
single-ended cable, MXIbus, 3-24
SIP resistor networks, 3-17
slave mode state machine, MXIbus, 2-7, 6-9 to 6-10
soft reset, 4-1
specifications, A-1 to A-3
  capability codes, A-1 to A-2
  electrical, A-2
  environmental, A-2
  physical, A-3
  reliability, A-3
  requirements, A-3
  timing, A-3
state machines, MXIbus
  master mode, 2-6, 6-5 to 6-9
  slave mode, 2-7, 6-9 to 6-10
Status/Control register, 4-7
Status/ID register, 4-40
SUBCLASS bits, 4-28
Subclass register, 4-28
switch settings
  interlocked arbitration mode, 3-13
  MXIbus
    fair requester, 3-15
    logical address, 3-6
    System Controller, 3-13
  reset signal, 3-15
    ACFAIL, 3-15
    SYSRESET, 3-15
  VMEbus System Controller, 3-4
SYSFAIL, 2-6, 2-9, 6-2
SYSFAIL bit, 4-38
SYSFAILIE bit, 4-38
SYSFAILIN bit, 4-26
SYSFAILINT bit, 4-39
SYSFAILOUT bit, 4-27
SYSFIN bit, 4-34
SYSFOUT bit, 4-34
SYSRESET, 2-6, 2-9, 6-2
  reset signal selection, 3-15
SYSRSTIN bit, 4-27
SYSRSTOUT bit, 4-27
system configuration, programming considerations, 5-1 to 5-35
  A16 address map, 5-21 to 5-34
  logical address map, MXIbus, 5-1 to 5-20
System Controller
  MXIbus, 3-13, 6-10 to 6-11
    functions, 2-7

switch settings, 3-13
timeout, 3-13 to 3-14
VMEbus, 3-4, 6-1
functions, 2-6
switch settings, 3-4
system logcial address map. *See* logical address map, MXIbus.
system power cycling requirements, MXIbus, 3-25 to 3-26
distributed, 3-26
master power switch, 3-26
system resets control, VMEbus, 2-9

T

termination
INTX, 3-19 to 3-21
MXIbus, 3-16 to 3-19
SIP resistor networks, 3-17
networks, MXIbus, 3-16 to 3-19
TERMPWR connection, MXIbus, 3-17
timeout, MXIbus, System Controller, 3-13
timeout settings, VMEbus, 3-9
multiple VME-MXIs in VMEbus chassis, 3-10
no timeout, 3-11
multiple VME-MXIs in VXIbus mainframe, 3-10
no timeout, 3-11
one VME-MXI in chassis, 3-9
timeout value, VMEbus, 3-7
jumper settings, 3-8
timing specifications, A-3
transceivers
MXIbus
address modifier, 2-7, 6-10
address/data, 2-7, 6-10
control signals, 2-7, 6-11
requirements, 2-2
VMEbus
address, 2-6, 6-1
address modifier, 2-6, 6-1
control signal, 2-6, 6-1
data, 2-6, 6-1
transfer responses for VMEbus address modifiers, 6-7
transfer size comparison, 6-7

U

unpacking VME-MXI, 1-7

V

VERSION bits, 4-8
VME-MXI, 1-1 to 1-7, 2-1 to 2-9
block diagram, 2-5
bus signals, 2-1 to 2-2
configuration, 3-1 to 3-7
electrical characteristics, 2-1 to 2-2
front panel features, 1-5
functional description, 2-5 to 2-9
installing, 3-16 to 3-26
interface module, 1-2
with INTX option, 1-3
INTX daughter card, 2-8
kit contents, 1-6
optional equipment, 1-6 to 1-7
parts locator diagram, 3-2
with INTX, 3-3
registers, 4-1 to 4-41. *See also* registers.
specifications, A-1 to A-3
theory of operation, 6-1 to 6-12
unpacking, 1-7
VMEbus
A16 address window, 2-6
A24 address window, 2-6
A32 address window, 2-6
ACFAIL, 2-6
arbiter circuitry, 2-6, 6-2
capability codes, A-1
compliance levels, 2-3 to 2-4
extender registers, 4-9. *See also* registers.
interrupt circuitry, 2-6
interrupt levels, addresses, 6-4
LA address window, 2-6
modules, 2-2 to 2-4
request level, 3-6 to 3-7
requester circuitry, 2-6, 6-2
requester jumper settings, 3-7
signals, 2-1 to 2-2
SYSFAIL, 2-6
SYSRESET, 2-6
System Controller, 3-4
functions, 2-6, 6-1
switch settings, 3-4
timeout settings, 3-9
timeout value, 3-7
jumper settings, 3-8
transceivers
address, 2-6, 6-1
address modifier, 2-6, 6-1
control signal, 2-6, 6-1
data, 2-6, 6-1

W

worksheets
for A16 address map, 5-29 to 5-34
examples, 5-26 to 5-28
for logical address map, 5-13 to 5-20
examples, 5-10 to 5-12