
COMPREHENSIVE SERVICES
We offer competitive repair and calibration services, as well as easily accessible documentation and free downloadable resources.
SELL YOUR SURPLUS
We buy new, used, decommissioned, and surplus parts from every NI series. We work out the best solution to suit your individual needs.
Sell For Cash
Get Credit
Receive a Trade-In Deal
OBSOLETE NI HARDWARE IN STOCK & READY TO SHIP
We stock New, New Surplus, Refurbished, and Reconditioned NI Hardware.
APEX WAVES
Bridging the gap between the manufacturer and your legacy test system.
1-800-915-6216
www.apexwaves.com
sales@apexwaves.com
All trademarks, brands, and brand names are the property of their respective owners.
Request a Quote
CLICK HERE
VXI-MXI

MXI™

Getting Started with Your PCI-MXI-2 and the NI-VXI™/NI-VISA™ Software for Linux

October 2005
370551D-01

NATIONAL INSTRUMENTS™

Worldwide Technical Support and Product Information

ni.com

National Instruments Corporate Headquarters

11500 North Mopac Expressway Austin, Texas 78759-3504 USA Tel: 512 683 0100

Worldwide Offices

Australia 1800 300 800, Austria 43 0 662 45 79 90 0, Belgium 32 0 2 757 00 20, Brazil 55 11 3262 3599, Canada 800 433 3488, China 86 21 6555 7838, Czech Republic 420 224 235 774, Denmark 45 45 76 26 00, Finland 385 0 9 725 725 11, France 33 0 1 48 14 24 24, Germany 49 0 89 741 31 30, India 91 80 51190000, Israel 972 0 3 6393737, Italy 39 02 413091, Japan 81 3 5472 2970, Korea 82 02 3451 3400, Lebanon 961 0 1 33 28 28, Malaysia 1800 887710, Mexico 01 800 010 0793, Netherlands 31 0 348 433 466, New Zealand 0800 553 322, Norway 47 0 66 90 76 60, Poland 48 22 3390150, Portugal 351 210 311 210, Russia 7 095 783 68 51, Singapore 1800 226 5886, Slovenia 386 3 425 4200, South Africa 27 0 11 805 8197, Spain 34 91 640 0085, Sweden 46 0 8 587 895 00, Switzerland 41 56 200 51 51, Taiwan 886 02 2377 2222, Thailand 662 278 6777, United Kingdom 44 0 1635 523545

For further support information, refer to the *Technical Support and Professional Services* appendix. To comment on National Instruments documentation, refer to the National Instruments Web site at ni.com/info and enter the info code feedback.

Important Information

Warranty

The National Instruments MXIbus boards and accessories 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.

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

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

National Instruments believes that the information in this document 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

Patents

For patents covering National Instruments products, refer to the appropriate location: **Help»Patents** in your software, the `patents.txt` file on your CD, or `ni.com/patents`.

WARNING REGARDING USE OF NATIONAL INSTRUMENTS PRODUCTS

(1) NATIONAL INSTRUMENTS PRODUCTS ARE NOT DESIGNED WITH COMPONENTS AND TESTING FOR A LEVEL OF RELIABILITY SUITABLE FOR USE IN OR IN CONNECTION WITH SURGICAL IMPLANTS OR AS CRITICAL COMPONENTS IN ANY LIFE SUPPORT SYSTEMS WHOSE FAILURE TO PERFORM CAN REASONABLY BE EXPECTED TO CAUSE SIGNIFICANT INJURY TO A HUMAN.

(2) IN ANY APPLICATION, INCLUDING THE ABOVE, RELIABILITY OF OPERATION OF THE SOFTWARE PRODUCTS CAN BE IMPAIRED BY ADVERSE FACTORS, INCLUDING BUT NOT LIMITED TO FLUCTUATIONS IN ELECTRICAL POWER SUPPLY, COMPUTER HARDWARE MALFUNCTIONS, COMPUTER OPERATING SYSTEM SOFTWARE FITNESS, FITNESS OF COMPILERS AND DEVELOPMENT SOFTWARE USED TO DEVELOP AN APPLICATION, INSTALLATION ERRORS, SOFTWARE AND HARDWARE COMPATIBILITY PROBLEMS, MALFUNCTIONS OR FAILURES OF ELECTRONIC MONITORING OR CONTROL DEVICES, TRANSIENT FAILURES OF ELECTRONIC SYSTEMS (HARDWARE AND/OR SOFTWARE), UNANTICIPATED USES OR MISUSES, OR ERRORS ON THE PART OF THE USER OR APPLICATIONS DESIGNER (ADVERSE FACTORS SUCH AS THESE ARE HEREAFTER COLLECTIVELY TERMED "SYSTEM FAILURES"). ANY APPLICATION WHERE A SYSTEM FAILURE WOULD CREATE A RISK OF HARM TO PROPERTY OR PERSONS (INCLUDING THE RISK OF BODILY INJURY AND DEATH) SHOULD NOT BE RELIANT SOLELY UPON ONE FORM OF ELECTRONIC SYSTEM DUE TO THE RISK OF SYSTEM FAILURE. TO AVOID DAMAGE, INJURY, OR DEATH, THE USER OR APPLICATION DESIGNER MUST TAKE REASONABLY PRUDENT STEPS TO PROTECT AGAINST SYSTEM FAILURES, INCLUDING BUT NOT LIMITED TO BACK-UP OR SHUT DOWN MECHANISMS. BECAUSE EACH END-USER SYSTEM IS CUSTOMIZED AND DIFFERS FROM NATIONAL INSTRUMENTS' TESTING PLATFORMS AND BECAUSE A USER OR APPLICATION DESIGNER MAY USE NATIONAL INSTRUMENTS PRODUCTS IN COMBINATION WITH OTHER PRODUCTS IN A MANNER NOT EVALUATED OR CONTEMPLATED BY NATIONAL INSTRUMENTS, THE USER OR APPLICATION DESIGNER IS ULTIMATELY RESPONSIBLE FOR VERIFYING AND VALIDATING THE SUITABILITY OF NATIONAL INSTRUMENTS PRODUCTS WHENEVER NATIONAL INSTRUMENTS PRODUCTS ARE INCORPORATED IN A SYSTEM OR APPLICATION, INCLUDING, WITHOUT LIMITATION, THE APPROPRIATE DESIGN, PROCESS AND SAFETY LEVEL OF SUCH SYSTEM OR APPLICATION.

Compliance

Compliance with FCC/Canada Radio Frequency Interference Regulations

Determining FCC Class

The Federal Communications Commission (FCC) has rules to protect wireless communications from interference. The FCC places digital electronics into two classes. These classes are known as Class A (for use in industrial-commercial locations only) or Class B (for use in residential or commercial locations). All National Instruments (NI) products are FCC Class A products.

Depending on where it is operated, this Class A product could be subject to restrictions in the FCC rules. (In Canada, the Department of Communications (DOC), of Industry Canada, regulates wireless interference in much the same way.) Digital electronics emit weak signals during normal operation that can affect radio, television, or other wireless products.

All Class A products display a simple warning statement of one paragraph in length regarding interference and undesired operation. The FCC rules have restrictions regarding the locations where FCC Class A products can be operated.

Consult the FCC Web site at `www.fcc.gov` for more information.

FCC/DOC Warnings

This equipment generates and uses radio frequency energy and, if not installed and used in strict accordance with the instructions in this manual and the CE marking Declaration of Conformity*, may cause interference to radio and television reception. Classification requirements are the same for the Federal Communications Commission (FCC) and the Canadian Department of Communications (DOC).

Changes or modifications not expressly approved by NI could void the user's authority to operate the equipment under the FCC Rules.

Class A

Federal Communications Commission

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user is required to correct the interference at their own expense.

Canadian Department of Communications

This Class A digital apparatus meets all requirements of the Canadian Interference-Causing Equipment Regulations.

Cet appareil numérique de la classe A respecte toutes les exigences du Règlement sur le matériel brouilleur du Canada.

Compliance with EU Directives

Users in the European Union (EU) should refer to the Declaration of Conformity (DoC) for information* pertaining to the CE marking. Refer to the Declaration of Conformity (DoC) for this product for any additional regulatory compliance information. To obtain the DoC for this product, visit `ni.com/certification`, search by model number or product line, and click the appropriate link in the Certification column.

* The CE marking Declaration of Conformity contains important supplementary information and instructions for the user or installer.

Contents

About This Manual

Chapter 1
Introduction and Quick Start

Chapter 2
PCI-MXI-2 Configuration and Installation

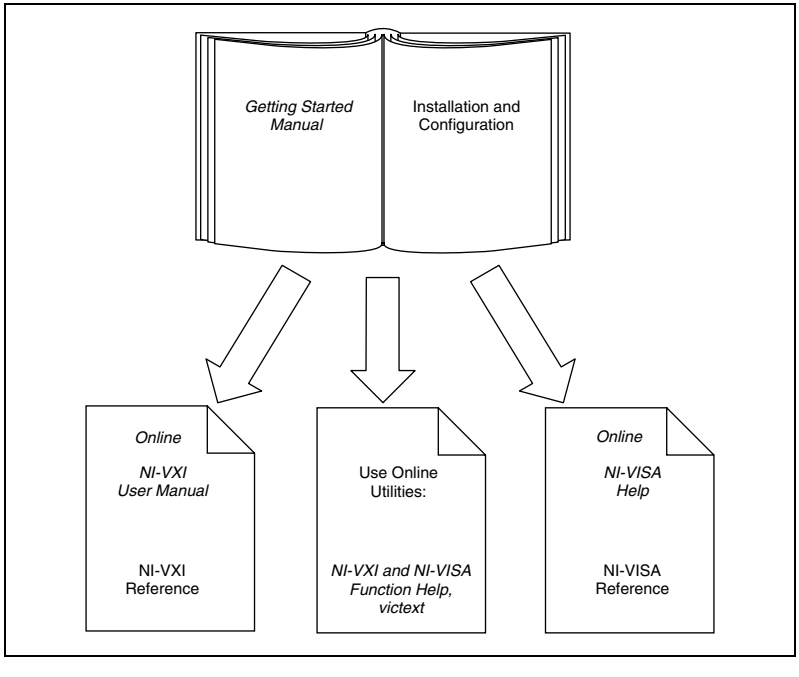
Chapter 3
VXI-MXI-2 Configuration and Installation

About This Manual

This manual contains instructions for installing and configuring the National Instruments PCI-MXI-2 kits for Linux. The VXI MXI-2 kit includes a VXI-MXI-2 module that plugs into a VXI mainframe and links your PCI-based computer to the VXIbus. The VME MXI-2 kit includes a VME-MXI-2 that plugs into a VME chassis and links your PCI-based computer to the VMEbus. Both kits include the PCI-MXI-2 or PCI-MXI-2 Universal interface board, which links your computer to the MXIbus, and the NI-VXI bus interface software. The VXI MXI-2 and VME MXI-2 kits are fully VXI*plug&play* compliant.

This manual uses the term *PCI-MXI-2 kit* when information applies to either kit and the term *VXI/VME-MXI-2* when information applies to either the VXI-MXI-2 or the VME-MXI-2. This manual also uses the term PCI-MXI-2 when information applies to either the PCI-MXI-2 or the PCI-MXI-2 Universal. The PCI-MXI-2 Universal has the same functionality as the PCI-MXI-2, except it supports 3.3 V PCI systems.

How To Use the Manual Set



Begin by reading this getting started manual to guide you through the installation and configuration of the hardware and software. You should install and configure the components of the PCI-MXI-2 kit in the order in which this manual describes them. Be sure to review the *Quick Start* and *Default Settings* sections in Chapter 1, *Introduction and Quick Start*. The material in those sections may be all you need to get up and running with your PCI-MXI-2 kit.

When you have successfully set up your system, you can begin to develop applications in NI-VXI and/or NI-VISA. The *NI-VXI User Manual* presents the concepts of VXI and prepares you for detailed explanations of the NI-VXI functions. Study the descriptions of each function given in the online help utility to fully understand the purpose and syntax of each function. This manual is available in the *NIVXI*`/manuals` directory (where *NIVXI* refers to the actual location where you have installed the NI-VXI software). Use the Adobe Acrobat Reader program to open this file.

We recommend the VISA API for new applications. Refer to the *NI-VISA User Manual* to learn about VISA and how to use it in your system. The NI-VISA online help describes the attributes, events, and operations you can use in NI-VISA. The user manual is available in the *VXIpnp*`/linux/NIvisa/manuals` directory (where *VXIpnp* refers to the actual location where you have installed the NI-VISA software). Use the Adobe Acrobat Reader program to open this file.

Conventions

The following conventions appear in this manual:

♦ — The ♦ symbol indicates that the following text applies only to a specific product, a specific operating system, or a specific software version.

 — This icon denotes a note, which alerts you to important information.

 — This icon denotes a caution, which advises you of precautions to take to avoid injury, data loss, or a system crash.

bold — Bold text denotes items that you must select or click in the software, such as menu items and dialog box options. Bold text also denotes parameter names.

italic — Italic text denotes variables, emphasis, a cross reference, or an introduction to a key concept. Italic text also denotes text that is a placeholder for a word or value that you must supply.

`monospace`	Text in this font denotes text or characters that you should enter from the keyboard, sections of code, programming examples, and syntax examples. This font is also used for the proper names of disk drives, paths, directories, programs, subprograms, subroutines, device names, functions, operations, variables, filenames, and extensions.
`monospace bold`	Bold text in this font denotes the messages and responses that the computer automatically prints to the screen. This font also emphasizes lines of code that are different from the other examples.
`monospace italic`	Italic text in this font denotes text that is a placeholder for a word or value that you must supply.

Related Documentation

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

- ANSI/IEEE Standard 1014-1987, *IEEE Standard for a Versatile Backplane Bus: VMEbus*
- ANSI/IEEE Standard 1155-1993, *IEEE VMEbus Extensions for Instrumentation: VXIbus*
- ANSI/VITA 1-1994, *VME64*
- *Multisystem Extension Interface Bus Specification*, Version 2.0, National Instruments Corporation
- *PCI Local Bus Specification*, Revision 2.0, PCI Special Interest Group
- *VXI-MXI-2 User Manual*, National Instruments Corporation
- *VME-MXI-2 User Manual*, National Instruments Corporation
- VXI-6, *VXIbus Mainframe Extender Specification*, Revision 2.0, VXIbus Consortium

1

Introduction and Quick Start

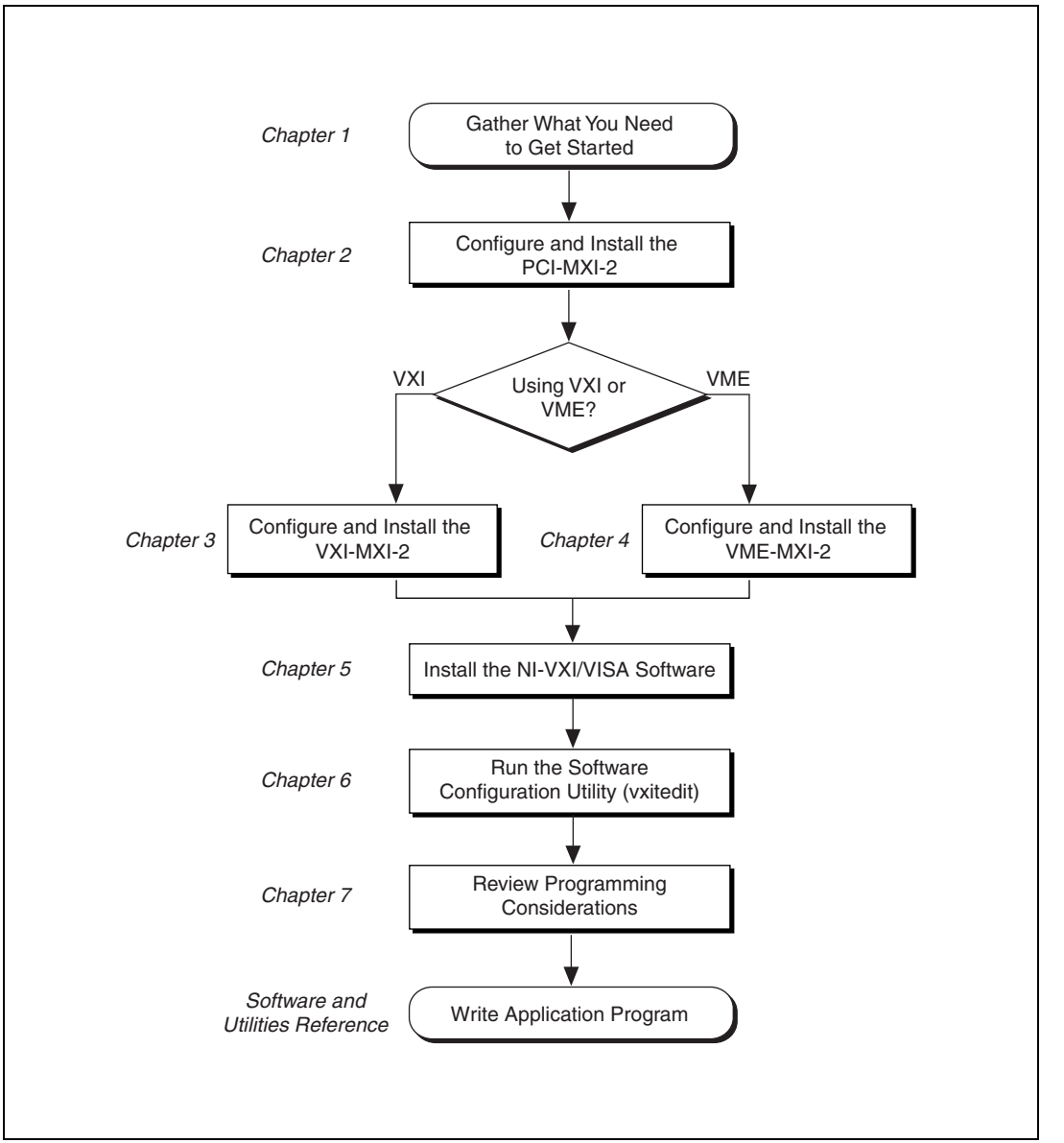
This chapter describes the PCI-MXI-2 kits, lists what you need to get started, introduces the concepts of MXI-2, and includes a brief description of the hardware and software.

This chapter also contains a *Quick Start* section, which has the basic information you need to install the PCI-MXI-2 kit with a simple configuration, and a *Default Settings* section, which lists the hardware and software default settings for easy reference. You may find that these sections contain as much information as you need to get started with your PCI-MXI-2 kit.

This manual uses the term *PCI-MXI-2 kit* when information applies to either the VXI MXI-2 kit, which contains a VXI-MXI-2 module, or the VME MXI-2 kit, which contains a VME-MXI-2 module. Similarly, the term *VXI/VME-MXI-2* means that information applies to either the VXI-MXI-2 or the VME-MXI-2.

How to Use This Manual

The following flowchart shows where to turn in this manual for more information about configuring and using the hardware and software.



PCI-MXI-2 Kit Overview

The PCI-MXI-2 kits link any computer with a PCI bus (hereafter referred to as a PCI-based computer) directly to the VXIbus or VMEbus using the high-speed Multisystem eXtension Interface bus (MXI-2).

A PCI-based computer equipped with a VXI MXI-2 kit can function as a VXI Commander and Resource Manager. A PCI-based computer equipped with a VME MXI-2 kit can function as a VMEbus master and/or slave device. The PCI-MXI-2 kit makes your PCI-based computer behave as if it were plugged directly into the VXI/VME backplane as an embedded CPU VXI/VME module.

The software included with the kits is for Intel *x*86-based computers.

What You Need to Get Started

- ❑ A PCI-based computer running a supported Linux distribution
- ❑ VXIbus or VMEbus mainframe
- ❑ PCI-MXI-2 or PCI-MXI-2 Universal interface board
- ❑ VXI-MXI-2 or VME-MXI-2 interface module
- ❑ MXI-2 cable
- ❑ NI-VXI/NI-VISA software media for the PCI-MXI-2

For supported distributions, refer to the *NI-VXI/NI-VISA for the PCI-MXI-2 for Linux Installation Guide*.

MXI-2 Description

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

Because MXI-2 incorporates all of these new signals into a single connector, the triggers, interrupts, and utility signals can be extended not only to other mainframes for multichassis VXI systems, but also to the local

CPU interface in all MXI-2 products using a single cable. Thus, MXI-2 lets CPU interface boards such as the PCI-MXI-2 have access to these signals as if they were plugged directly into the VXI/VME backplane.

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

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

Hardware Description

The PCI-MXI-2 is a half-size, PCI-compatible plug-in circuit board that plugs into one of the expansion slots in your PCI-based computer. It links your PCI-based computer directly to the MXIbus and vice versa. Because the PCI-MXI-2 uses the same communication register set that other VXIbus message-based devices use, other MXIbus devices view the PCI-MXI-2 as a VXIbus device. The PCI-MXI-2 can also function as the MXIbus System Controller and can terminate the MXIbus signals directly on the PCI-MXI-2. In addition, you can have onboard DRAM on the PCI-MXI-2 that can be shared with the MXIbus and VXI/VMEbus and used as a dedicated data buffer.

The VXI-MXI-2 module is an extended-class, register-based VXIbus device with optional VXIbus Slot 0 capability so that it can reside in any slot in a C-size or D-size chassis.

Note D-size VXI mainframes have connections for a P3 connector. The VXI-MXI-2, however, does not have this connector and, if configured as a Slot 0 controller, cannot provide the necessary control for VXI devices that need P3 support.

The VXI-MXI-2 uses address mapping to convert MXIbus cycles into VXIbus cycles and vice versa. By connecting to the PCI-MXI-2 board, the VXI-MXI-2 links the PCI bus to the VXIbus. The VXI-MXI-2 can automatically determine if it is located in VXI Slot 0 and/or if it is the MXIbus System Controller.

The VME-MXI-2 module is a single-slot, double-height VMEbus device with optional VMEbus System Controller functions. It uses address mapping to convert MXIbus cycles into VMEbus cycles and vice versa, just like the VXI-MXI-2. By connecting to the PCI-MXI-2 board, it links the PCI bus to the VMEbus. The VME-MXI-2 can automatically determine if it is located in the first slot of a VMEbus chassis and if it is the MXIbus System Controller.

Also, the VXI-MXI-2 and VME-MXI-2 automatically terminate the MXIbus if installed as the first or last device in the MXIbus. If installed in the middle of the MXIbus, both the VXI-MXI-2 and VME-MXI-2 automatically disable MXIbus termination. In addition, you can have up to 64 MB of onboard DRAM on the VXI-MXI-2 and VME-MXI-2 modules that can either be shared with the VXI/VMEbus and MXIbus or used as a dedicated data buffer.

The PCI-MXI-2, VXI-MXI-2, and VME-MXI-2 products achieve high-performance block transfer rates by integrating the MITE custom ASIC, a sophisticated dual-channel DMA controller with standard interfaces for VXI, VME, MXI, and PCI. By using MITE DMA to transfer data and commands to and from devices, the MITE frees up a computer's microprocessor to perform other tasks such as data analysis and presentation. In addition to DMA, the MITE incorporates both the new Synchronous MXI protocol and VME64 MBLT (8-byte block transfers in which both the address bus and data bus are used to transfer data) directly into the ASIC to perform the fastest transfer operation to instruments.

Software Description

The NI-VXI/NI-VISA bus interface software for the PCI-MXI-2 and Linux includes a Resource Manager, graphical and text-based interactive VXI resource editor programs, a comprehensive library of software routines for VXI/VME programming, and graphical and text-based interactive control programs for interacting with VXI/VME or VISA. You can use this software to seamlessly program multiple-mainframe configurations and have software compatibility across a variety of VXI/VME controller platforms.

NI-VISA has a comprehensive library of software routines not only for VXI/VME programming, but also for GPIB, GPIB-VXI, and serial. You can use this software to program instruments connected via different types of interfaces.

Optional Software

Your PCI-MXI-2 kit includes the NI-VXI/NI-VISA bus interface software. In addition, you can use National Instruments LabVIEW to ease your programming task. This standardized program matches the modular virtual instrument capability of VXI and can reduce your VXI/VMEbus software development time.

LabVIEW is a complete programming environment that departs from the sequential nature of traditional programming languages and features a graphical programming environment.

Quick Start

You can use this section as a guide to quickly configure and operate your VXI or VME system using the PCI-MXI-2 and the VXI-MXI-2 or VME-MXI-2.

This section assumes that you intend to perform a basic configuration as follows:

- You have one PCI-MXI-2 interface module, which you will install in your PCI-based computer as the Resource Manager (logical address 0).
- You have either one C-size VXI-MXI-2 or one 6U, B-size VME-MXI-2, which you will install in a VXI or VME chassis, respectively, and connect to the PCI-MXI-2.
- You will be using the NI-VXI/NI-VISA software for initialization, configuration, and device interaction.
- You will use the default hardware and software settings:
  - The PCI-MXI-2 is the main controller, the VXI/VME Resource Manager, and a message-based device.
  - Your system contains only one VXI or VME chassis.
  - There is no shared memory used on the PCI-based computer, the PCI-MXI-2, or the VXI/VME-MXI-2.

Refer to the *Default Settings* section for a complete listing of the hardware and software default settings. If you need more information or if you want to try a different configuration, refer to the appropriate hardware or software chapters in this manual, which describe the installation and configuration steps in greater detail.

Hardware Installation

To guard against electrostatic discharge, touch the antistatic plastic package to a metal part of your computer before removing the PCI-MXI-2 from the package. Install the PCI-MXI-2 in an available PCI slot in your PCI-based computer.

By default, the PCI-MXI-2 automatically detects whether it should be the system controller on the MXIbus. Verify that the correct cable end labeled *Connect This End To Device Closest To MXIbus Controller In This Daisy Chain* is attached securely to the PCI-MXI-2. You must connect the cable this way so that the PCI-MXI-2 can correctly detect whether it should be the system controller on the MXIbus. For more information, refer to Chapter 2, *PCI-MXI-2 Configuration and Installation*.

You received either a VXI-MXI-2 or a VME-MXI-2 in your PCI-MXI-2 kit. To guard against electrostatic discharge, touch the antistatic plastic package to a metal part of your computer before removing the VXI-MXI-2 or VME-MXI-2 from the package. Install the VXI-MXI-2 in the first slot of a VXI chassis, or install the VME-MXI-2 in the first slot of a VME chassis.

The VXI/VME-MXI-2 default configuration automatically detects whether it should be the VXI/VMEbus system controller. The VXI/VMEbus system controllers operate certain VXI/VMEbus lines as required for VXI/VME systems. Verify that any other VXI/VME devices with system controller capability that are located in the same chassis are not configured as system controller. Having more than one device configured as system controller will damage the VXI/VME system.

For VXI systems that include VME devices, ensure that the VME devices are not configured in the upper 16 KB (starting from 0xC000) of the A16 address space. This region is reserved for VXI device configuration registers, which are used for initializing, configuring, and interacting with VXI devices. The PCI-MXI-2 and VME-MXI-2 also use this region for this purpose.

Also ensure that no VXI devices in your system are configured for either logical addresses 0 or 1. These are the default configurations for the PCI-MXI-2 and the VXI-MXI-2, respectively.

For more information about the VXI-MXI-2 or VME-MXI-2 hardware, refer to either Chapter 3, *VXI-MXI-2 Configuration and Installation*, or Chapter 4, *VME-MXI-2 Configuration and Installation*.

Installing and Loading the NI-VXI/NI-VISA Software for Linux

For information on installing, uninstalling, or using the NI-VXI/NI-VISA software, refer to Chapter 5, *NI-VXI/NI-VISA Software Installation*.

VME Users

When used with a VXI-MXI-2, `resman` identifies and configures the VXI devices, including the VXI-MXI-2. When used with a VME-MXI-2, `resman` configures the VME-MXI-2 to allow the PCI-MXI-2 to access devices in the VME chassis. `resman` does not configure VME devices. The VME specification does not specify the initialization and configuration procedures that the VXI specification requires.

It is recommended that you enter the information about your VME devices into the `vxitedit` utility. `resman` can then properly configure the various device-specific VME address spaces and VME interrupt lines. For more information on configuring non-VXI devices in your VXI system, refer to Chapter 3, *VXI Text Resource Editor*, of the *NI-VXI Text Utilities Reference Manual*. For more details about installing the NI-VXI software, refer to Chapter 5, *NI-VXI/NI-VISA Software Installation*, in this manual.

Device Interaction

After `resman` has detected and configured all VXI/VME devices, you can view specific information on each device in your system by using the `vxiedit` utility or its text-mode counterpart, `vxitedit`. These utilities include a Resource Manager Display, which contains a description for each device, including each VXI device's logical address.

You can interact with your VXI/VME devices by using the `vic` and `victext` utilities for NI-VXI. These utilities let you interactively control your VXI/VME devices without using a conventional programming language, LabVIEW, or LabWindows™/CVI™.

Try the following in `vic`:

1. Click the **Bus Access** tab at the top of the window and select **InReg** as the **Operation** along the left side of the window.
2. Select the **VXI-MXI-2** or **VME-MXI-2** as the **Device Name** under **Input Parameters** and the **Id/Logical Address** register under **Input Parameters**.
3. Click **Go!**. If the CMPL light along the right side of the window is green, and the output value ends in an FF6, you have successfully read the manufacturer ID for National Instruments.

The same functionality is available in `victext` with the `vxiinreg` command. You can use `help vxiinreg` for the command's parameter description.

You may now want to read the configuration registers from other VXI devices in your system using the same procedure. The `InReg` operation accesses only the upper 16 KB of A16 space. Try reading the registers from one of the devices listed in the Resource Manager Display of `vxiedit`. In this way, you can verify that your PCI-MXI-2 can access each device in your VXI system successfully.

You can also access VXI and VME defines that are configured in A16, A24, and A32 address spaces by using the `In` and `Out` operations in `vic` or the `vxiin` or `vxiout` commands in `victext`.

Default Settings

This section summarizes the hardware and software default settings for the PCI-MXI-2 kit. If you need more information about a particular setting, or if you want to try a different configuration, please refer to the appropriate hardware or software chapters in this manual. The manual flowchart at the beginning of this chapter directs you to where to find the information you need.

PCI-MXI-2

This section summarizes the hardware and software default settings for the PCI-MXI-2.

Table 1-1. PCI-MXI-2 Hardware Default Settings

Hardware Component	Default Setting
U17 Switch 1 (FOV)	OFF: PCI-MXI-2 boots off the user-configured half of the EEPROM.
U17 Switch 2 (TST)	OFF: Factory configuration of the EEPROM is protected.
U17 Switch 3 (POS)	OFF: *Do not alter this setting*.
U17 Switch 4 (CT)	ON: *Do not alter this setting*.
DRAM SIMM Installed	Per customer order

Table 1-2. PCI-MXI-2 Universal Hardware Default Settings

Hardware Component	Default Setting
SW1 Switch 1 (FOV)	OFF: PCI-MXI-2 boots off the user-configured half of the EEPROM.
SW1 Switch 2 (TST)	OFF: Factory configuration of the EEPROM is protected.
SW1 Switch 3 (POS)	OFF: *Do not alter this setting*.
SW1 Switch 4 (CT)	ON: *Do not alter this setting*.
DRAM SODIMM Installed	Per customer order

Table 1-3. PCI-MXI-2 Logical Address Configuration Editor Default Settings

Editor Field	Default Setting
Logical Address	0
Device Type	MBD
Address Space	A16
VXI Shared RAM Size	0 KB
Shared RAM Pool	0 KB
Lower Half Window Byte Swapping	Disabled (non-swapped)
Lower Half Window Memory Select	System Memory
Upper Half Window Byte Swapping	Disabled (non-swapped)
Upper Half Window Memory Select	System Memory
Resource Manager Delay	5 s
Map Upper/Lower Halves to Same Address	Disabled

Table 1-4. PCI-MXI-2 Device Configuration Editor Default Settings

Editor Field	Default Setting
Default Controller (LA-1)	First Remote Controller
System IRQ Level	1
Servant Area Size	0
Number of Handlers	1
Number of Interrupters	0
Protocol Register	0xFF0
Read Protocol Response	0x8448

Table 1-5. PCI-MXI-2 Bus Configuration Editor Default Settings

Editor Field	Default Setting
MXI System Controller	Auto
MXI Bus Timeout Value	1 ms
MXI CLK10	Receive
MXI Transfer Limit	Unlimited
VXImove uses Synchronous MXI	Enabled
MXI-2 Auto Retry	Enabled
A24/A32 Write Posting	Disabled
User Window Size	64 KB
Driver Window Size	32 KB
Expansion ROM	Enabled

VXI/VME-MXI-2

This section summarizes the hardware and software default settings for the VXI-MXI-2 and VME-MXI-2.

Table 1-6. VXI-MXI-2 Hardware Default Settings

Hardware Component	Default Setting
Logical Address (U43)	1
VXIbus Slot 0/Non-Slot 0 (W2)	Automatic detection
VXIbus Local Bus (S8, S9)	Both OFF: Single VXI-MXI-2
VXIbus CLK10 Routing (W3)	From onboard oscillator
External Trigger Termination (S2)	OFF: Unterminated
SMB CLK10 Direction (S3)	OUT: Drive CLK10 signal
SMB CLK10 Termination (S4)	Ignored; effective only when S3 is set to IN.
Polarity of External SMB CLK10 (S5)	Inverted
MXIbus CLK10 Signal (S7)	Receive CLK10 from MXIbus
MXIbus Termination (U35 switches 1 and 2)	Automatic MXIbus termination: switch 2 set to YES; switch 1 ignored.
Configuration EEPROM (U35 switches 3 and 4)	User-modifiable; factory settings protected: both switches set to NO.
DRAM SIMMs Installed	Per customer order
SIMM Size Configuration (S6)	OFF if SIMMS are 4 M × 32 or larger; ON if smaller than 4 M × 32.

Table 1-7. VME-MXI-2 Hardware Default Settings

Hardware Component	Default Setting
A16 Base Address (U20)	Hex C040
VME-MXI-2 Intermodule Signaling (W2)	No user-defined pin selected
MXIbus Termination (U21 switches 3 and 4)	Automatic MXIbus termination: switch 3 OFF; switch 4 ignored.
Configuration EEPROM (U21 switches 1 and 2)	User-modifiable; factory settings protected: both switches OFF.
DRAM SIMMs Installed	Per customer order
SIMM Size Configuration (S2)	OFF if SIMMS are 4 M × 32 or larger; ON if smaller than 4 M × 32.

Table 1-8. VXI/VME-MXI-2 Configuration Editor Default Settings

Editor Field	Default Setting
Logical Address	1 (set by hardware switch)
LA Selection	Set by hardware switch
Address Space	A24*
Requested Memory	16 KB*
A16 Write Posting	Disabled
A24/A32 Write Posting	Disabled
Interlocked Mode	Disabled
VXI/VME System Controller	Auto
VXI/VME Bus Timeout Value	125 µs
VXI/VME Auto Retry	Disabled
VXI/VME Transfer Limit	256
VXI/VME Arbiter Type	Priority
VXI/VME Request Level	3
VXI/VME Fair Request	Enabled

Table 1-8. VXI/VME-MXI-2 Configuration Editor Default Settings (Continued)

Editor Field	Default Setting
VXI/VME Arbiter Timeout	Enabled
MXI System Controller	Auto
MXI Bus Timeout Value	1 ms
MXI Auto Retry	Disabled
MXI Transfer Limit	Unlimited
MXI Parity Checking	Enabled
MXI Fair Requester	Disabled
MXI CLK10	Set by hardware switch (VXI-MXI-2 only)
* Assumes no DRAM is installed. If DRAM is installed, the Address Space would be A32, and Requested Memory would match the amount of DRAM.	

2

PCI-MXI-2 Configuration and Installation

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

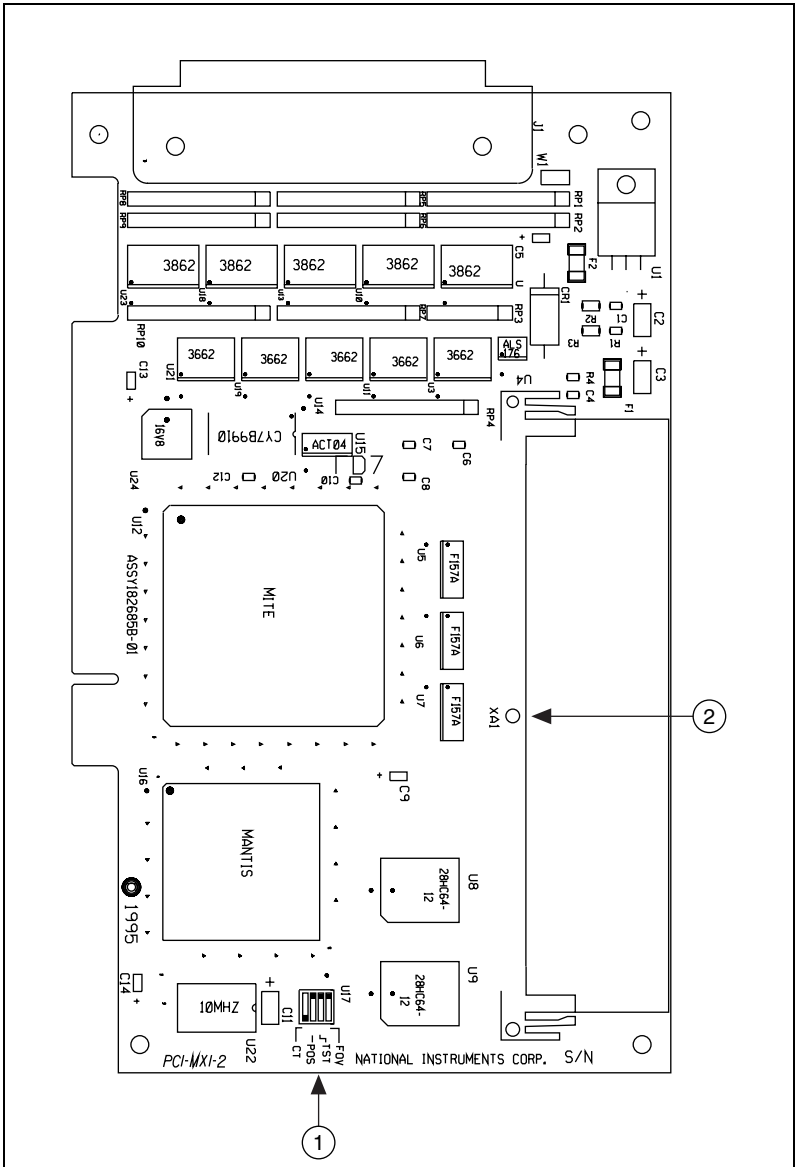
Caution Electrostatic discharge can damage several components on your PCI-MXI-2 module. To avoid such damage in handling the module, touch the antistatic plastic package to a metal part of your computer chassis before removing the PCI-MXI-2 from the package.

Configure the PCI-MXI-2

This section describes how to configure the configuration EEPROM on the PCI-MXI-2 and PCI-MXI-2 Universal.

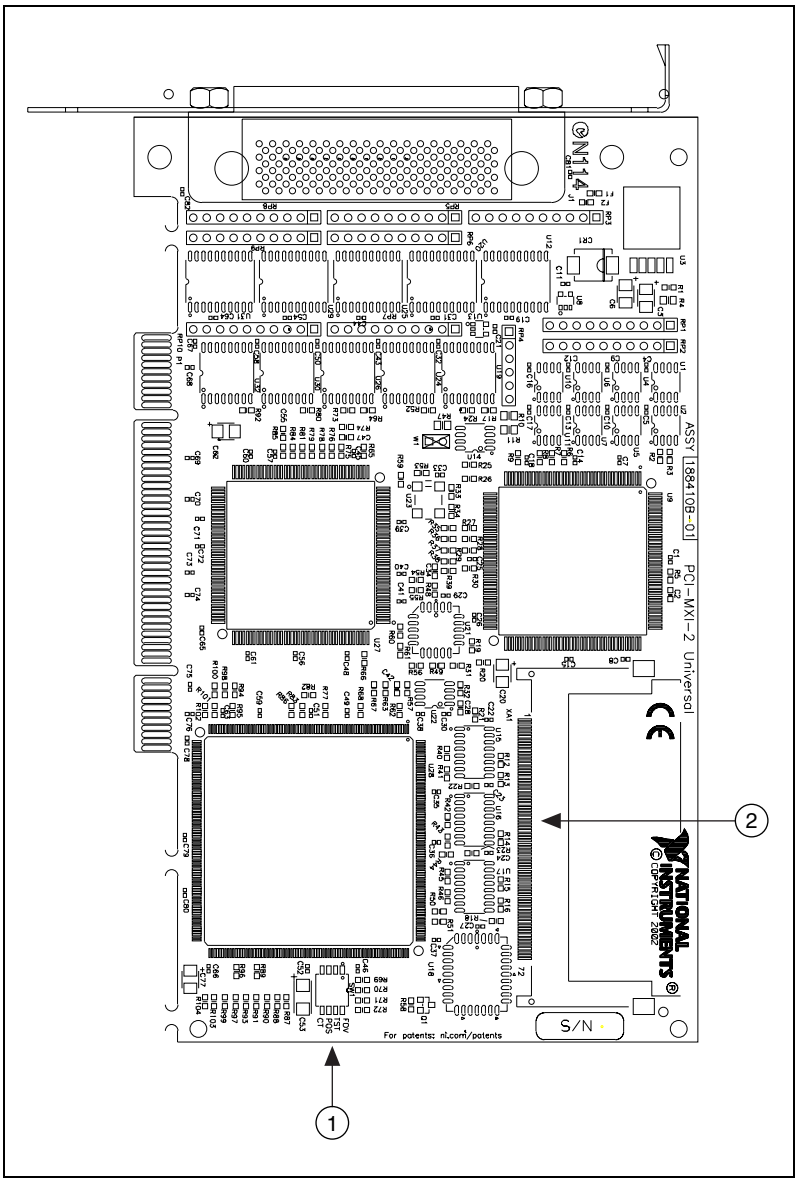
Figure 2-1 shows the PCI-MXI-2. The drawing shows the location and factory-default settings on the module. Your PCI-MXI-2 may look slightly different.

Figure 2-2 shows the PCI-MXI-2 Universal.



1 SW1	2 DRAM

Figure 2-1. PCI-MXI-2 Parts Locator Diagram



1	SW1	2	DRAM

Figure 2-2. PCI-MXI-2 Universal Parts Locator Diagram

Configuration EEPROM

The PCI-MXI-2 has an onboard EEPROM, which stores default register values that are loaded at power-on. The EEPROM is divided into two halves—a factory-configuration half, and a user-configuration half—so you can modify the user-configurable half, while the factory-configured half stores a back-up of the default user settings. The factory configuration is a minimal configuration that allows you to boot your PCI-MXI-2 regardless of the changes made to the user configuration.

For information on configuring the onboard EEPROM, refer to Appendix C, *Common Questions*.

Install the PCI-MXI-2

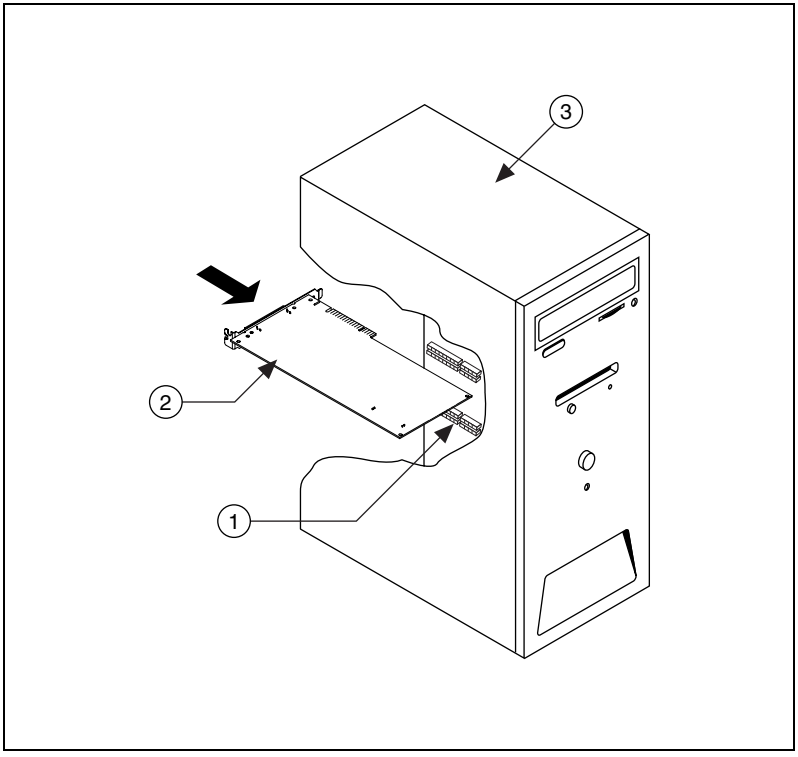
This section contains general installation instructions for the PCI-MXI-2. Consult your computer user manual or technical reference manual for specific instructions and warnings.

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

Caution To protect both yourself and the computer from electrical hazards, the computer should remain off until you are finished installing the PCI-MXI-2 module.

2. Remove the top cover or access port to the PCI bus.
3. Select any available PCI expansion slot.
4. Touch the metal part of the power supply case inside the computer to discharge any static electricity that might be on your clothes or body.
5. Line up the PCI-MXI-2 with the MXI-2 connector near the cut-out on the back panel. Slowly push down on the top of the PCI-MXI-2 until its card-edge connector is resting on the expansion slot receptacle. Using slow, evenly distributed pressure, press the PCI-MXI-2 straight down until it seats in the expansion slot.
6. Check the installation. Ensure that the PCI-MXI-2 is secure in its slot.
7. Replace the computer cover.

Figure 2-3 shows how to install the PCI-MXI-2.



1	PCI Bus Slot	2	PCI-MXI-2 Board	3	PC

Figure 2-3. PCI-MXI-2 Installed in a Computer

3

VXI-MXI-2 Configuration and Installation

This chapter contains the instructions to configure and install the VXI-MXI-2 module. This chapter applies only if you ordered the VXI MXI-2 kit. If you ordered the VME MXI-2 kit, skip this chapter and refer to Chapter 4, *VME-MXI-2 Configuration and Installation*.

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

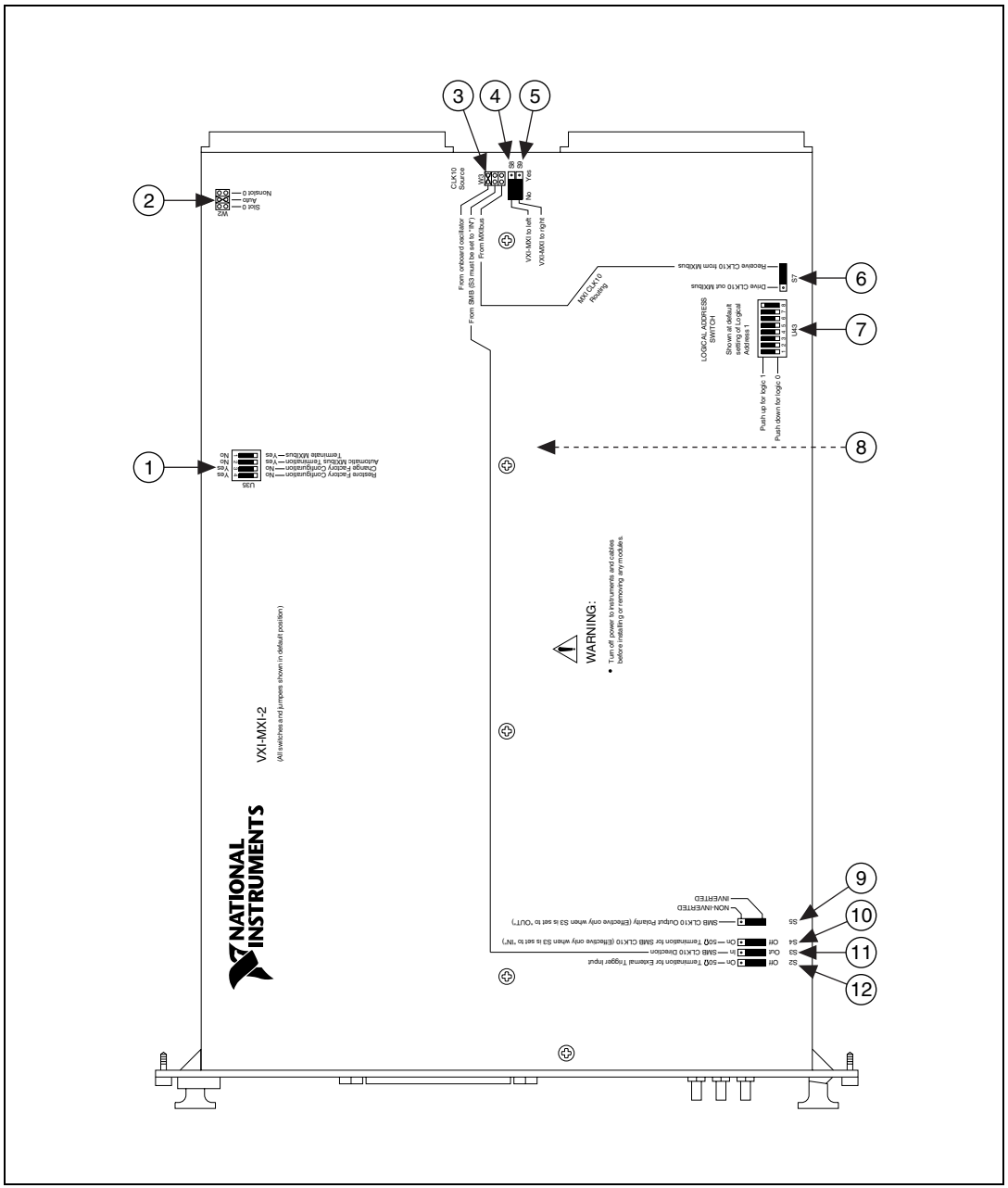
Configure the VXI-MXI-2

This section describes how to configure the following options on the VXI-MXI-2:

- VXIbus logical address
- VXIbus Slot 0/Non-Slot 0
- VXIbus local bus
- VXIbus CLK10 routing
- Trigger input termination
- MXIbus termination
- Configuration EEPROM
- Onboard DRAM

Note If you have only one VXI chassis in your system, you should leave the VXI-MXI-2 at Logical Address 1 and install it into Slot 0. To install the VXI-MXI-2, refer to the *Install the VXI-MXI-2* section.

Figure 3-1 shows the VXI-MXI-2 as it would appear when facing the right side cover. The drawing shows the location and factory-default settings of most of the configuration switches and jumpers on the module. Notice that switch S6 is accessible only by removing the front cover.



1	U35	3	W3	5	S9	7	U43	9	S5	11	S3
2	W2	4	S8	6	S7	8	S6	10	S4	12	S2

Figure 3-1. VXI-MXI-2 Right-Side Cover

Front Panel Features

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

- Three front panel LEDs
  - **SYSFAIL** LED indicates that the VMEbus SYSFAIL line is asserted.
  - **MXI** LED indicates when the VXI-MXI-2 is accessed from the MXIbus.
  - **VXI** LED indicates when the VXI-MXI-2 is accessed from the VXIbus.
- MXIbus connector
- Three SMB connectors
  - External clock
  - Trigger output
  - Trigger input
- System reset pushbutton

Removing the Metal Enclosure

The VXI-MXI-2 is housed in a metal enclosure to improve EMC performance and to provide easy handling. Because the enclosure includes cutouts to facilitate changes to the switch and jumper settings, it should not be necessary to remove it under normal circumstances.

However, it is necessary to remove the enclosure if you want to change the amount of DRAM installed on the VXI-MXI-2. Switch S6, which is directly related to the amount of DRAM you want to install, is also accessible only by removing the enclosure. If you will be making this change, remove the four screws on the top, the four screws on the bottom, and the five screws on the right side cover of the enclosure. Refer to the *Onboard DRAM* section for details about changing DRAM.

VXIbus Logical Address

Each device in a VXIbus/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 VXI configuration registers located on the device. With unique logical addresses, each VXIbus device in the system is assigned 64 bytes of configuration space in the upper 16 KB of A16 space.

Logical address 0 is reserved for the Resource Manager in the VXIbus system. Because the VXI-MXI-2 cannot act as a Resource Manager, do not configure the VXI-MXI-2 with a logical address of 0.

Some VXIbus devices have *dynamically configurable* logical addresses. These devices have an initial logical address of hex FF or 255, which indicates that they can be dynamically configured. While the VXI-MXI-2 does support dynamic configuration of VXI devices within its mainframe, it is itself a *statically configured* device and is preset at the factory with a VXI logical address of 1.

Ensure that no other statically configurable VXIbus devices have a logical address of 1. If they do, change the logical address setting of either the VXI-MXI-2 or the other device so that every device in the system has a unique associated logical address.

You can change the logical address of the VXI-MXI-2 by changing the setting of the 8-bit DIP switch labeled *LOGICAL ADDRESS SWITCH* (location designator U43) on the panel. The down position of the DIP switch corresponds to a logic value of 0 and the up position corresponds to a logic value of 1. Verify that the VXI-MXI-2 does not have the same logical address as any other statically configured VXIbus device in your system. Remember that logical addresses hex 0 and FF are not allowed for the VXI-MXI-2. Also, when setting logical addresses, keep in mind the grouping requirements set by the system hierarchy. Refer to VXI-6, *VXIbus Mainframe Extender Specification*, for more information about setting logical addresses on a multimainframe hierarchy.

Figure 3-2 shows switch settings for logical address hex 1 and C0.

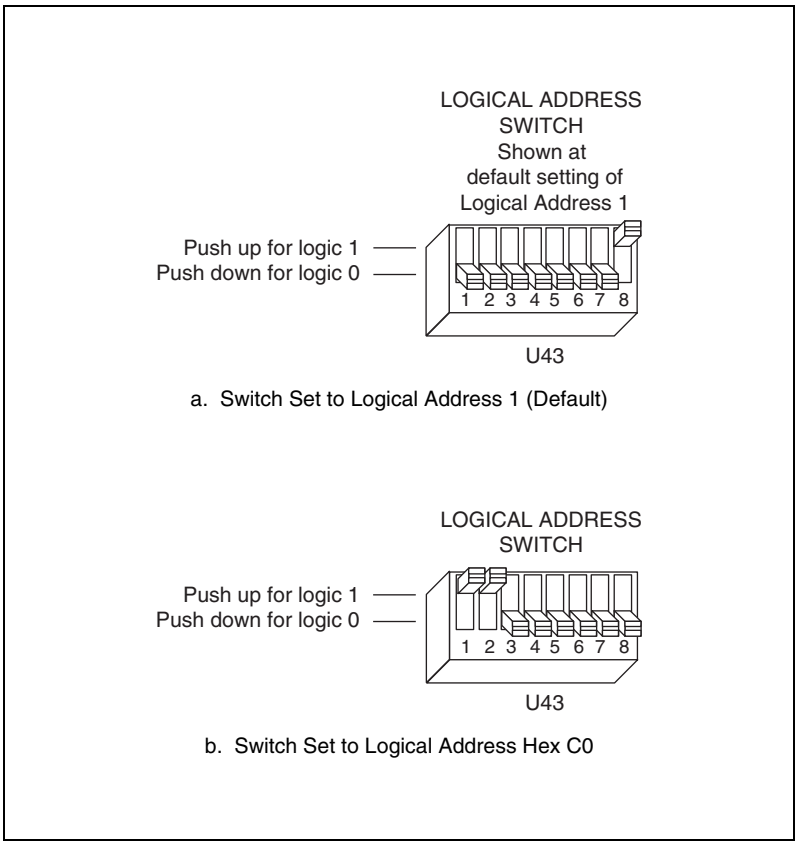


Figure 3-2. Logical Address Selection

VXIbus Slot 0/Non-Slot 0

The VXI-MXI-2 is configured at the factory to automatically detect if it is installed in Slot 0 of a VXIbus mainframe. With automatic Slot 0 detection, you can install the VXI-MXI-2 into any VXIbus slot.

You can manually configure the VXI-MXI-2 for either Slot 0 or Non-Slot 0 operation by defeating the automatic-detection circuitry. Use the three-position jumper W2 to select automatic Slot 0 detection, Slot 0, or Non-Slot 0 operation. Figure 3-3 shows these three settings.

Caution Do *not* install a device configured for Slot 0 into another slot without first reconfiguring it to either Non-Slot 0 or automatic configuration. Neglecting to do this could damage the device, the VXIbus backplane, or both.

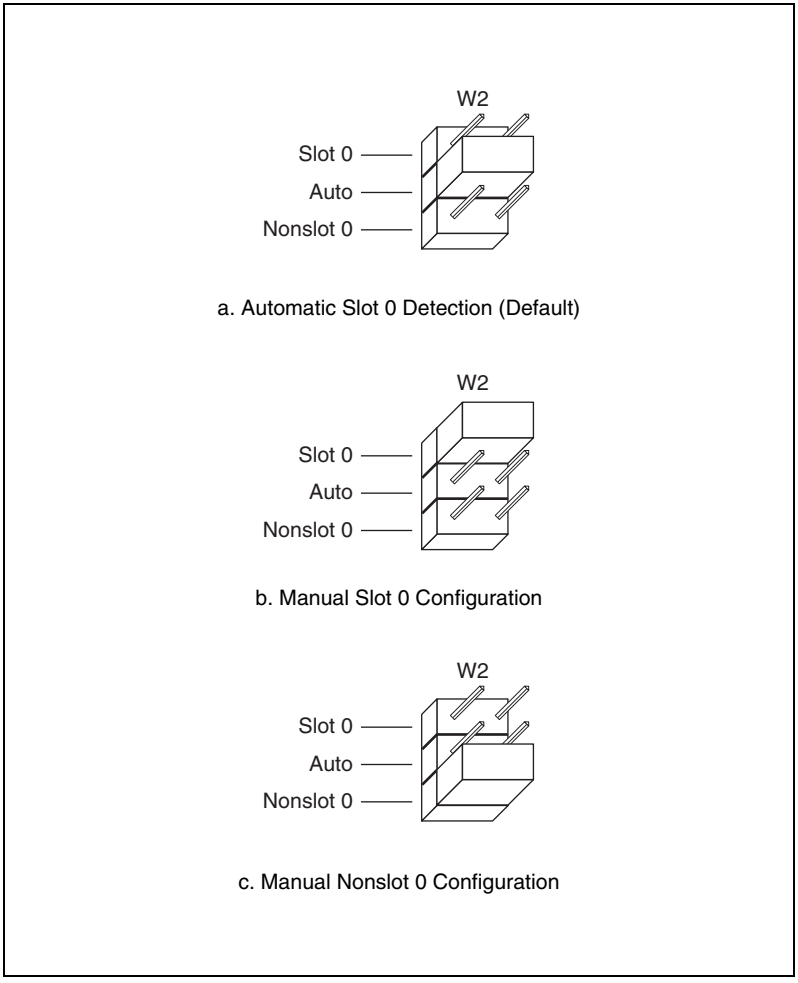


Figure 3-3. VXIbus Slot Configuration

When the VXI-MXI-2 is installed in Slot 0, it becomes the VMEbus System Controller. In this role, it has VMEbus Data Transfer Bus Arbiter circuitry that accepts bus requests on all four VMEbus request levels, prioritizes the requests, and grants the bus to the highest priority requester. As VMEbus System Controller, the VXI-MXI-2 also drives the 16 MHz VMEbus system clock by an onboard 16 MHz oscillator.

As required by the VXIbus specification, the VXI-MXI-2 drives the 10 MHz signal CLK10 on a differential ECL output when installed in Slot 0. When not installed in Slot 0, the VXI-MXI-2 only receives the CLK10 signal.

VXIbus Local Bus

If you will be installing more than one VXI-MXI-2 in a single VXIbus mainframe, you must configure the boards to use the local bus. The VXI-MXI-2 uses the local bus to pass a signal to the other VXI-MXI-2 modules in the mainframe to disable the VMEbus bus timeout unit (BTO) during cycles that map to the MXIbus. Because the local bus is used, you need to install all VXI-MXI-2 modules for a single mainframe in adjacent slots.

You will use two switches on the VXI-MXI-2 to select its position in relation to any other VXI-MXI-2 module in the mainframe. Use switch S9 when there is a VXI-MXI-2 to the right (higher numbered slot). Use S8 when there is a VXI-MXI-2 to the left (lower numbered slot).

Figure 3-4 shows four configuration settings for a VXI-MXI-2. Figure 3-4a illustrates the default setting, which is for a single VXI-MXI-2 in a mainframe. Use the setting in Figure 3-4b for the VXI-MXI-2 located to the left of all others. Figure 3-4c shows the setting to use if the VXI-MXI-2 is between two others. Use the setting of Figure 3-4d for the VXI-MXI-2 located to the right of all others.

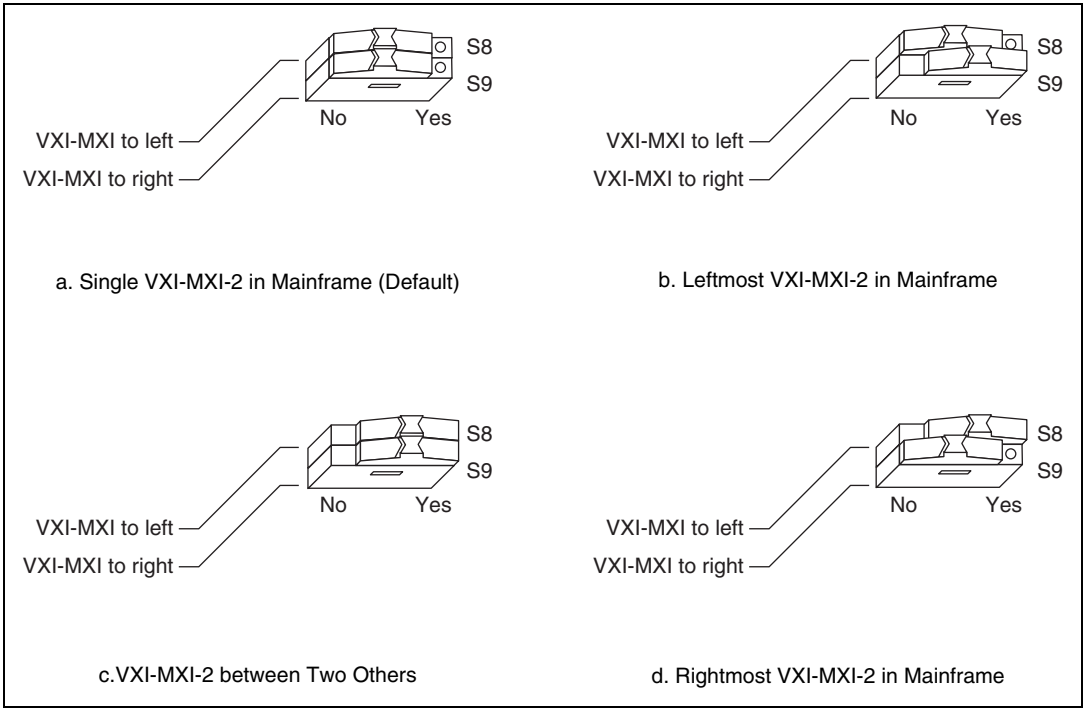


Figure 3-4. VXIbus Local Bus Configuration

VXIbus CLK10 Routing

When the VXI-MXI-2 is installed in Slot 0 of your mainframe, it supplies the VXIbus CLK10 signal. The VXI-MXI-2 can use three different sources to generate this signal: the onboard oscillator, the external CLK SMB connector, and the MXIbus CLK10 signal. Use the three-position jumper W3 to select these options, as shown in Figure 3-5.

Notice that Figure 3-5b and Figure 3-5c also show switches S3 and S7, respectively. You must configure these switches as shown when using the corresponding CLK10 source setting of W3.

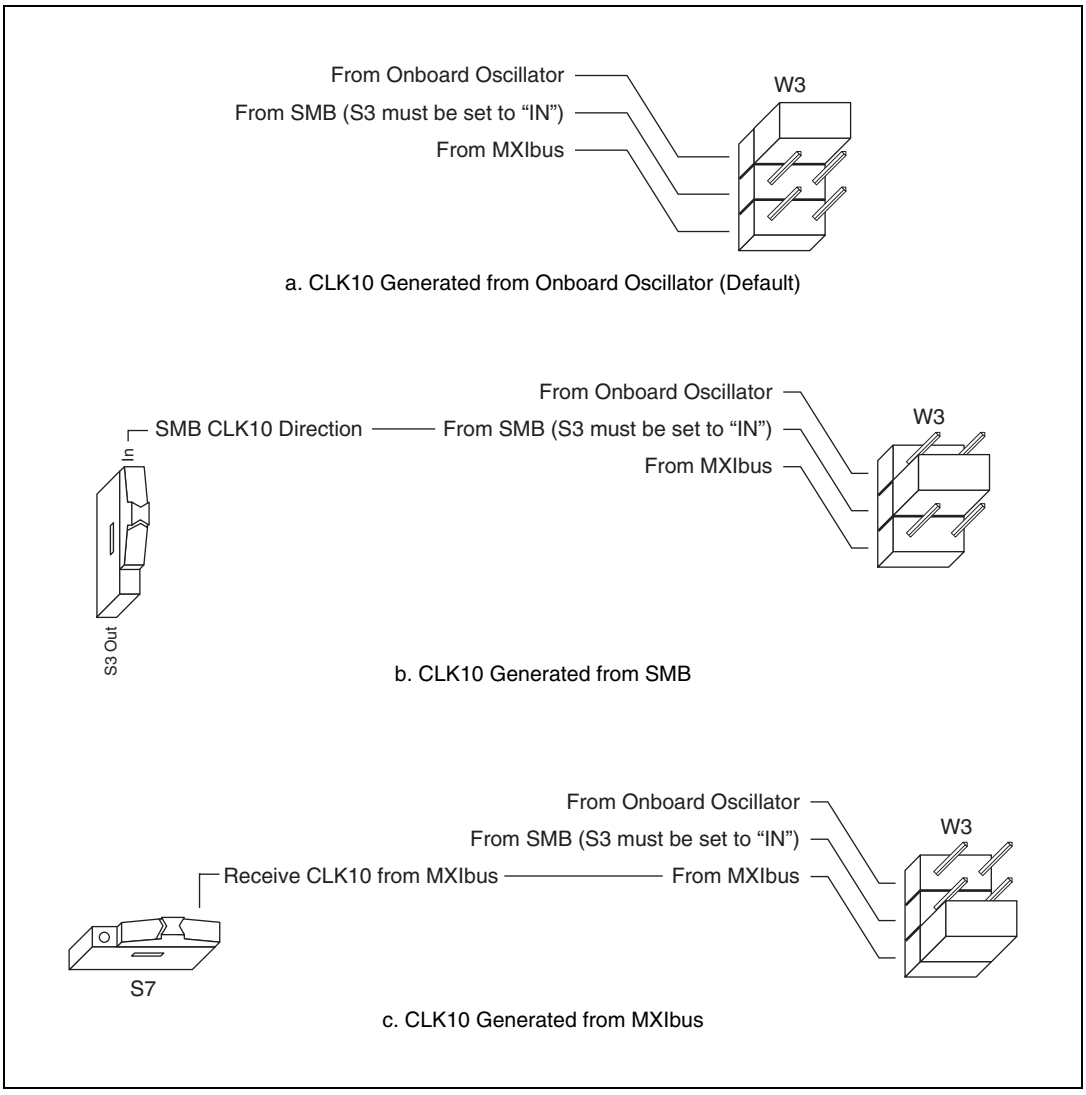


Figure 3-5. VXIbus CLK10 Routing

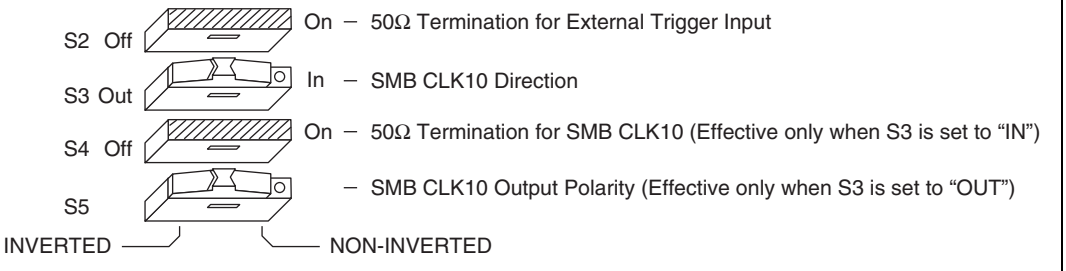
The VXI-MXI-2 can also be configured to drive the external CLK SMB signal from the VXIbus CLK10 signal. Switch S3 controls whether the VXI-MXI-2 drives or receives the external CLK SMB signal. If you change the S3 setting to drive CLK10 out the external CLK10 SMB connector, do not set the W3 jumper to receive the SMB CLK10 signal; instead use the settings shown in either Figure 3-5a or Figure 3-5c as appropriate.

When switch S3 is set so that the VXI-MXI-2 receives the SMB CLK10 signal, you have the option to add a 50Ω termination to the signal by setting switch S4. S4 is unused—its setting does not matter—when S3 is configured to drive the external CLK SMB signal.

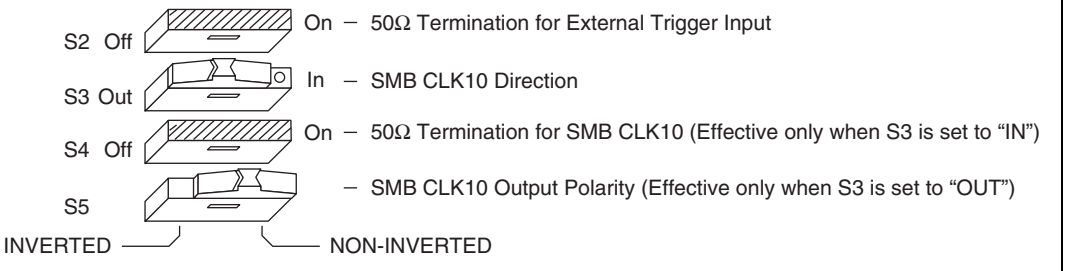
You can use an additional switch, S5, to control the polarity of the external CLK SMB signal when S3 is configured to drive it. S5 is unused—its setting does not matter—when S3 is configured to receive the external CLK SMB signal.

Figure 3-6 shows four configuration settings for the VXI-MXI-2. Figure 3-6a shows the default configuration, which is for driving the inverted external CLK SMB. Use the settings of Figure 3-6b to drive the noninverted external CLK SMB signal. Figure 3-6c illustrates the setting for receiving the external CLK SMB signal. Finally, you can configure the switches as shown in Figure 3-6d to receive the external CLK SMB signal with a 50 Ω termination.

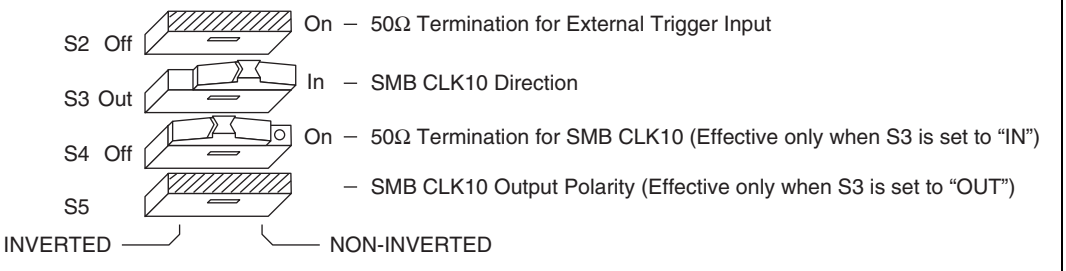
Note The settings of any switches shown with this pattern (▨) have no bearing on the configuration described in any of the following figures.



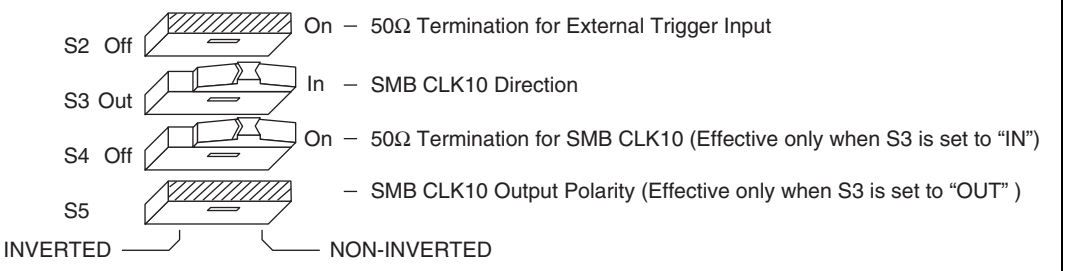
a. Drive Inverted External CLK SMB (Default)



b. Drive Non-Inverted External CLK SMB



c. Receive External CLK SMB



d. Receive External CLK SMB with 50Ω Termination

Figure 3-6. SMB CLK10 Settings

The VXI-MXI-2 can also drive or receive the MXIbus CLK10 signal. Switch S7 controls whether the VXI-MXI-2 drives MXIbus CLK10 from the VXIbus CLK10 or receives MXIbus CLK10. As shown in Figure 3-5c, if W3 is configured to use the MXIbus CLK10 to generate the VXIbus CLK10 signal, switch S7 must be configured to receive MXIbus CLK10. This is shown again in Figure 3-7a. If you change the S7 setting to drive CLK10 out the MXIbus, do not set the W3 jumper to receive the MXIbus CLK10; instead use the settings shown in Figure 3-5a or Figure 3-5b as appropriate.

Caution Do *not* configure more than one MXIbus device to drive the MXIbus CLK10. Having a second device driving MXIbus CLK10 could damage the device.

Figure 3-7 shows the configuration settings for receiving and driving MXIbus CLK10, respectively.

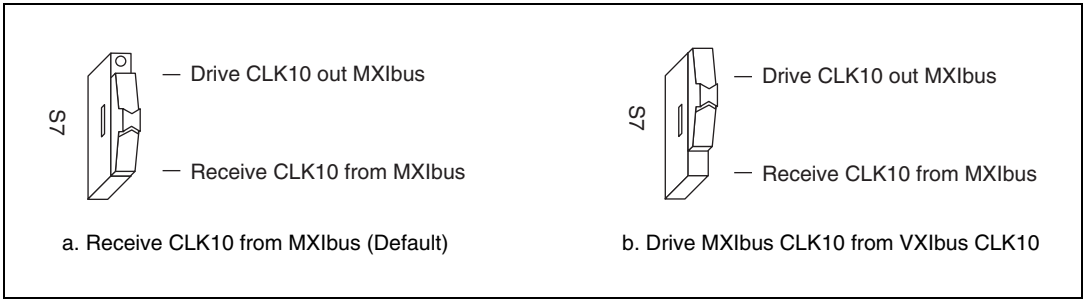


Figure 3-7. Receiving or Driving MXIbus CLK10

Trigger Input Termination

You can use switch S2 to terminate the external trigger input SMB with 50 Ω. Figure 3-8a shows the default setting for a nonterminated trigger input SMB. Use the setting of Figure 3-8b to terminate the trigger input SMB. Switch S2 is located above switches S3, S4, and S5, which have no effect on this configuration.

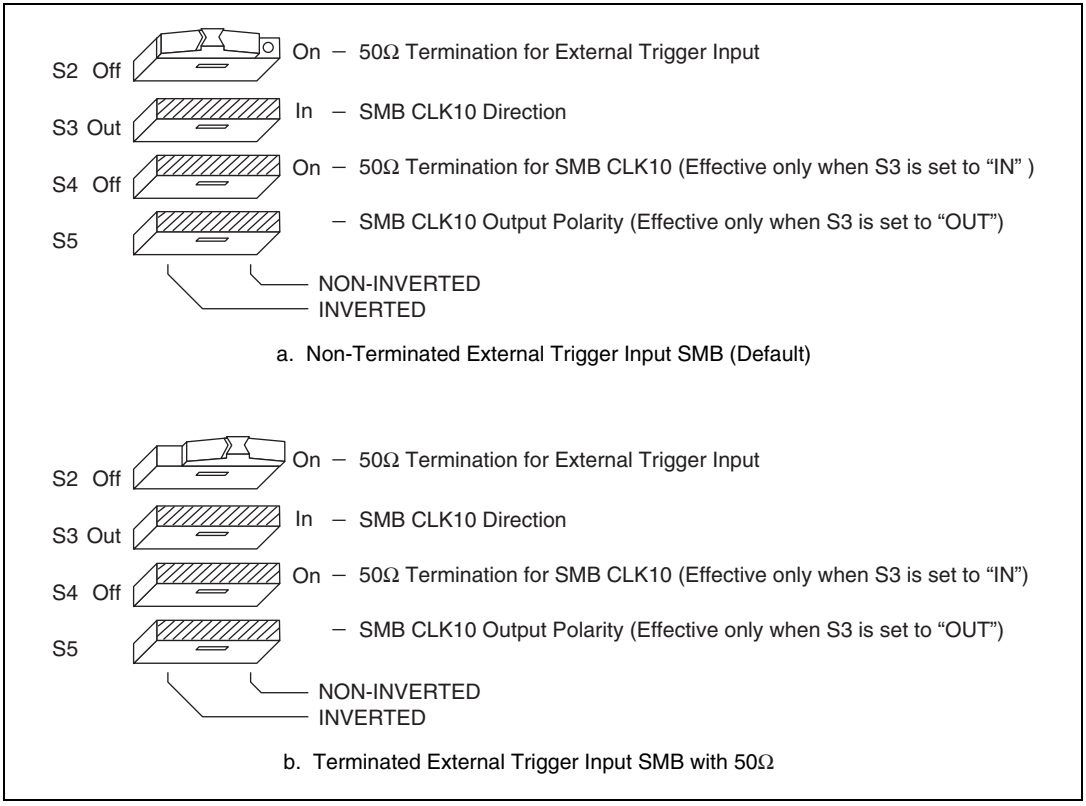


Figure 3-8. SMB Trigger Input Termination

MXIbus Termination

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

The VXI-MXI-2 automatically senses whether it is at either end of the MXIbus cable to terminate the MXIbus. You can manually control MXIbus termination by defeating the automatic circuitry. Use switches 1 and 2 of the four-position switch at location U35 to control whether MXIbus termination is automatic (Figure 3-9a), on (Figure 3-9b), or off (Figure 3-9c). The settings of switches 3 and 4 have no effect on MXIbus termination.

Use switch 2 of U35 to select whether you want the VXI-MXI-2 to automatically control termination of the MXIbus. Switch 1 of U35 lets

you manually control whether to terminate the MXIbus when automatic termination is turned off. Switch 1 has no effect when switch 2 is set for automatic MXIbus termination; you must turn off automatic termination if you want to manually control termination.

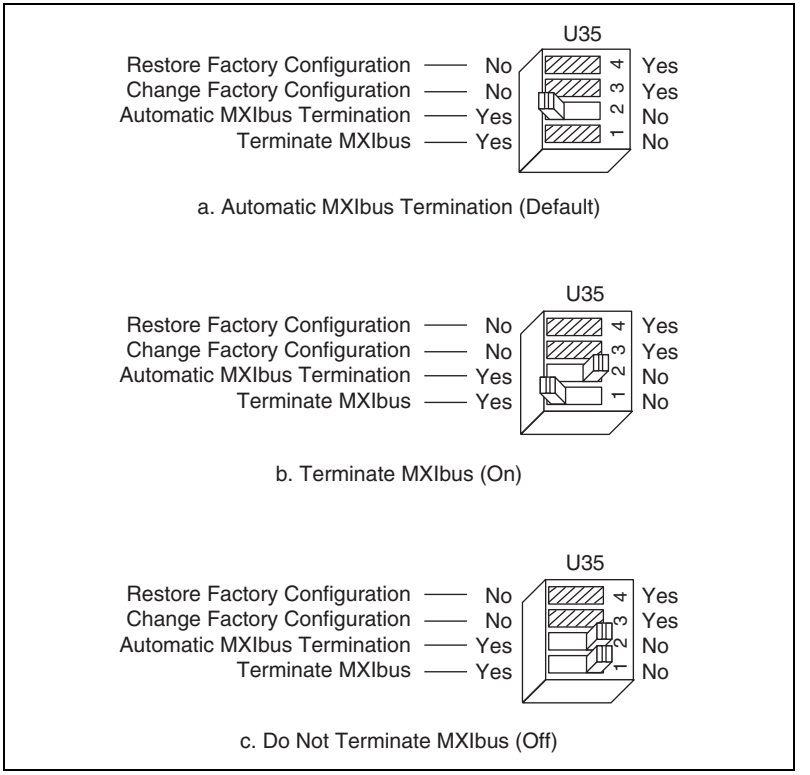


Figure 3-9. MXIbus Termination

Configuration EEPROM

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

Use switches 3 and 4 of the four-position switch at location U35 to control the operation of the EEPROM. The Restore Factory Configuration switch (switch 4) causes the VXI-MXI-2 to boot off the factory-configured half instead of the user-modified settings. This is useful in the event that the

user-configured half of the EEPROM becomes corrupted in such a way that the VXI-MXI-2 boots to an unusable state.

The Change Factory Configuration switch (switch 3 of U35) lets you change the factory-default configuration settings by permitting writes to the factory settings section of the EEPROM. This switch serves as a safety measure and should not be needed under normal circumstances. When this switch is off (its default setting) the factory configuration of the EEPROM is protected, so any writes to the factory area will be ignored. The factory area is protected regardless of the setting of switch 4 of U35.

Figure 3-10 shows the configuration settings for EEPROM operation. The settings of switches 1 and 2 have no effect on EEPROM configuration.

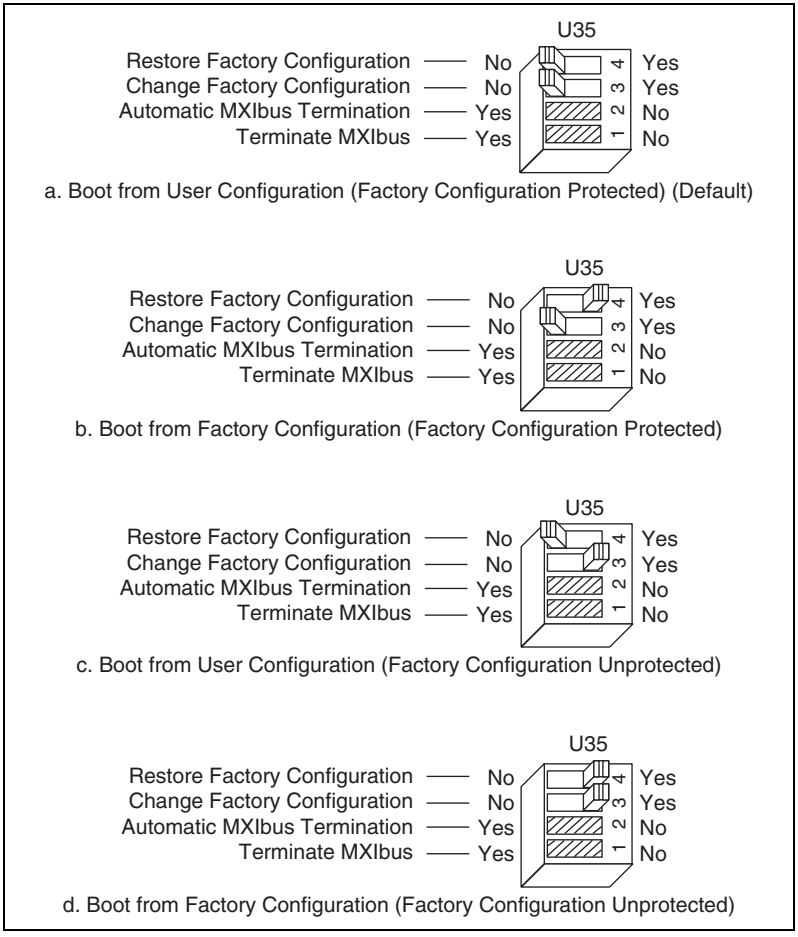


Figure 3-10. EEPROM Operation

Onboard DRAM

The VXI-MXI-2 can accommodate up to two 1.35 in. DRAM SIMMs. Table 3-1 lists the SIMMs you can use.

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

Bank 0	Bank 1	Total DRAM	National Instruments Option	Switch Setting of S6
—	—	0	—	—
256 K × 32 or 256 K × 36	—	1 MB	—	ON
256 K × 32 or 256 K × 36	256 K × 32 or 256 K × 36	2 MB	—	ON
512 K × 32 or 512 K × 36	—	2 MB	—	ON
512 K × 32 or 512 K × 36	512 K × 32 or 512 K × 36	4 MB	—	ON
1 M × 32 or 1 M × 36	—	4 MB	YES	ON
1 M × 32 or 1 M × 36	1 M × 32 or 1 M × 36	8 MB	—	ON
2 M × 32 or 2 M × 36	—	8 MB	YES	ON
2 M × 32 or 2 M × 36	2 M × 32 or 2 M × 36	16 MB	—	ON
4 M × 32 or 4 M × 36	—	16 MB	YES	OFF
4 M × 32 or 4 M × 36	4 M × 32 or 4 M × 36	32 MB	—	OFF
8 M × 32 or 8 M × 36	—	32 MB	YES	OFF
8 M × 32 or 8 M × 36	8 M × 32 or 8 M × 36	64 MB	YES	OFF

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

Switch S6 is used to select the size of each SIMM. The SIMM sockets and S6 are accessible only by removing the right-side cover. To access these components, remove the four screws on the top, the four screws on the bottom, and the five screws on the right-side cover of the metal enclosure. If the SIMMs are 4 M × 32 or larger, S6 should be in the OFF setting as shown in Figure 3-11a. For SIMMs *smaller* than 4 M × 32, use the ON setting as shown in Figure 3-11b.

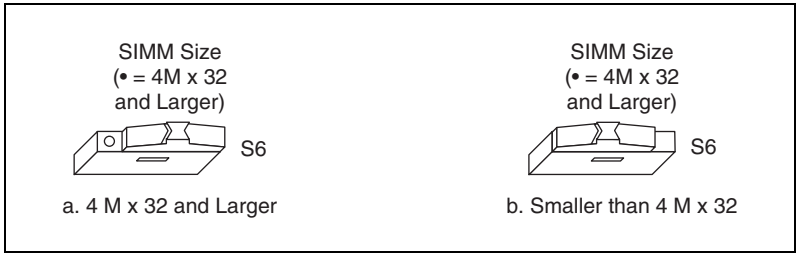


Figure 3-11. SIMM Size Configuration

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

Install the VXI-MXI-2

This section contains general installation instructions for the VXI-MXI-2. Refer to your VXIbus mainframe user manual or technical reference manual for specific instructions and warnings.

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

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

2. Remove or open any doors or covers blocking access to the mainframe slots.

3. If you are installing the VXI-MXI-2 into a D-size mainframe, install a support designed for installing C-size boards in D-size mainframes. The VXI-MXI-2 has no P3 connector and cannot provide P3 Slot 0 control to VXI devices requiring this capability.

Caution If the VXI-MXI-2 is not configured for automatic Slot 0 detection, be certain that the slot you select in your VXIbus mainframe matches the VXI-MXI-2 configuration as either a Slot 0 device or a Non-Slot 0 device. If you install your VXI-MXI-2 into a slot that does not correspond with the jumper setting, you risk damage to the VXI-MXI-2, the VXIbus backplane, or both.

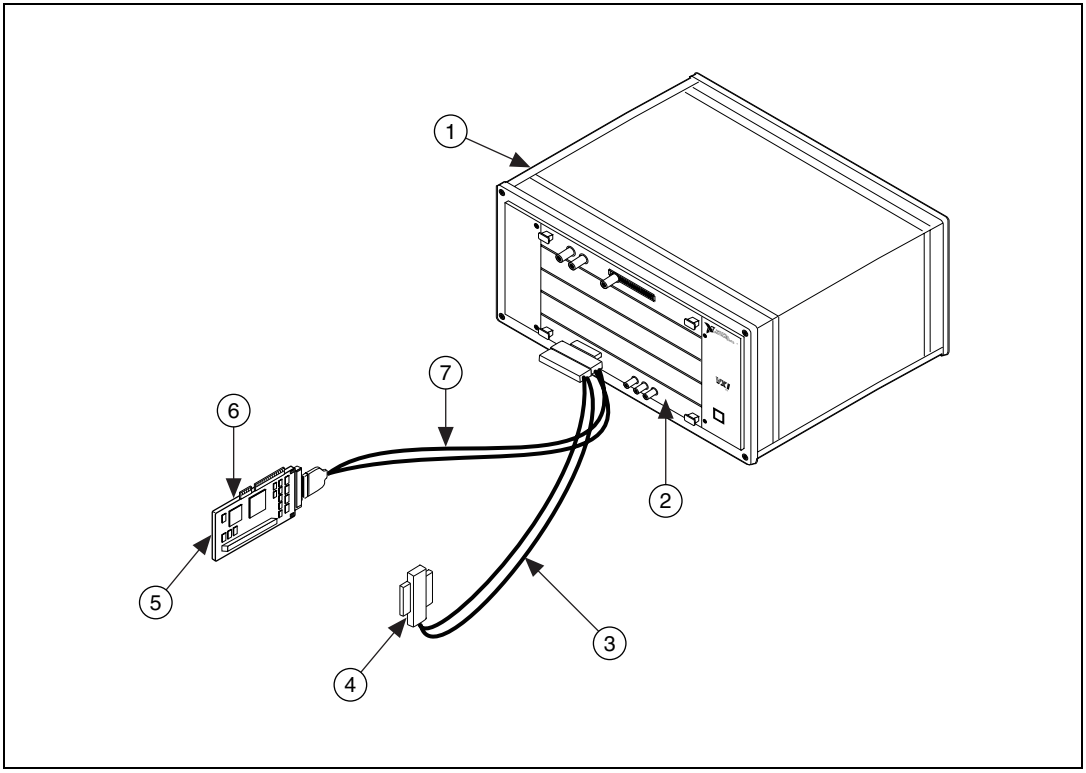
4. Insert the VXI-MXI-2 in the slot you have selected by aligning the top and bottom of the board with the card-edge guides inside the mainframe. Slowly push the VXI-MXI-2 straight into the slot until its plug connectors are resting on the backplane receptacle connectors. Using slow, evenly distributed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

Connect the MXIbus Cable

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

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

Figure 3-12 shows the correct cabling for a VXI system containing a PCI-MXI-2 board in a PCI-based computer cabled to a VXI-MXI-2 module residing in Slot 0 of a VXIbus mainframe. Notice that you can expand your system to include other devices by using an additional MXI-2 cable. However, in such a case the first cable needs to have a double connector on one end. You can use a cable with a single connector on each end to connect the last device on the MXIbus.



1 VXI Mainframe	5 Connection to External Computer
2 VXI-MXI-2 Slot 0 Module	6 PCI-MXI-2 Interface Module
3 Additional MXI-2 Cable	7 MXI-2 Cable
4 Connection to Other Mainframes	

Figure 3-12. MXI-2 Cable Configuration Using a PCI-MXI-2 and a VXI-MXI-2

When you have properly connected the MXI-2 cable, power on the VXIbus mainframe and then the computer.

Note Always turn on the mainframe first. Doing so makes it possible for your external computer to access the VXI boards in the mainframe upon startup.

4

VME-MXI-2 Configuration and Installation

This chapter contains the instructions to configure and install the VME-MXI-2 module. This chapter applies only if you ordered the VME MXI-2 kit. If you ordered the VXI MXI-2 kit, you should refer to Chapter 3, *VXI-MXI-2 Configuration and Installation*.

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

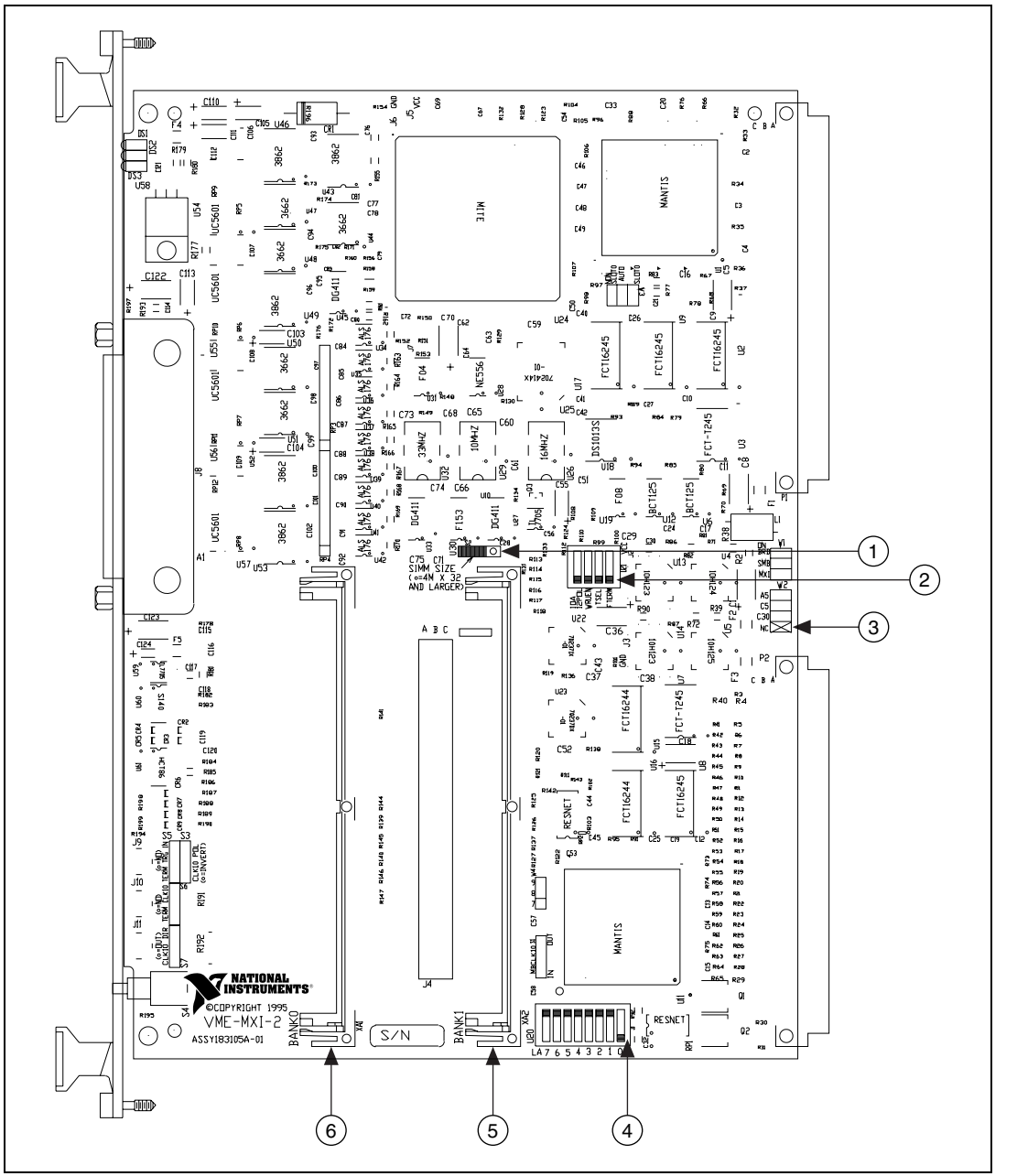
Configure the VME-MXI-2

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

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

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

Figure 4-1 shows the VME-MXI-2. The drawing shows the location and factory-default settings of the configuration switches and jumpers on the module.



1	S2	3	W2	5	DRAM Bank 1
2	U21	4	U20	6	DRAM Bank 0

Figure 4-1. VME-MXI-2 Parts Locator Diagram

Front Panel Features

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

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

VMEbus A16 Base Address

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

$$base\ address = C000\ hex + (logical\ address) \times 40\ hex$$

The factory-default logical address for the VME-MXI-2 is 1, which locates the registers in the range C040 hex to C07F hex. You can change the logical address of the VME-MXI-2 by changing the setting of the 8-bit DIP switch at location designator U20. The ON position of the DIP switch corresponds to a logic value of 0, and the OFF position corresponds to a logic value of 1.

Allowable logical addresses for the VME-MXI-2 range from 1 to 254 (hex FE). Verify that no other devices in your system use the A16 address space for the VME-MXI-2. If possible, configure all other VMEbus A16 devices to be located within the lower three quarters of A16 space. Also, when setting base addresses, keep in mind the grouping requirements set by the system hierarchy. Refer to VXI-6, *VXIbus Mainframe Extender Specification*, for more information on setting base addresses on a multimainframe hierarchy.

Figure 4-2 shows switch settings for A16 base address hex C040 and F000.

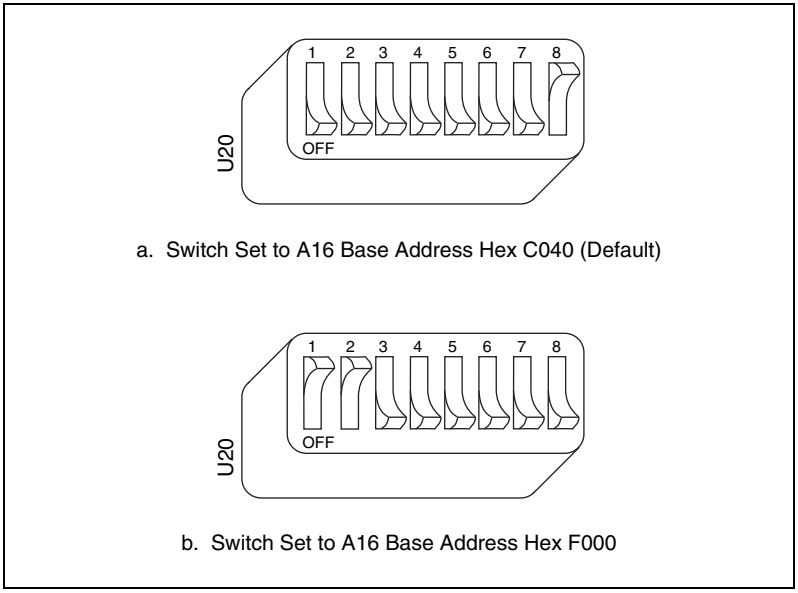


Figure 4-2. Base Address Selection

VME-MXI-2 Intermodule Signaling

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

You can choose from three user-defined pins on J2/P2. The pin you select must be bused on the VMEbus backplane between all slots that will have a VME-MXI-2 installed. Use jumper W2 to select pin A5, C5, or C30 of J2/P2, as shown in Figure 4-3.

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

Figure 4-3 shows the four intermodule signaling settings.

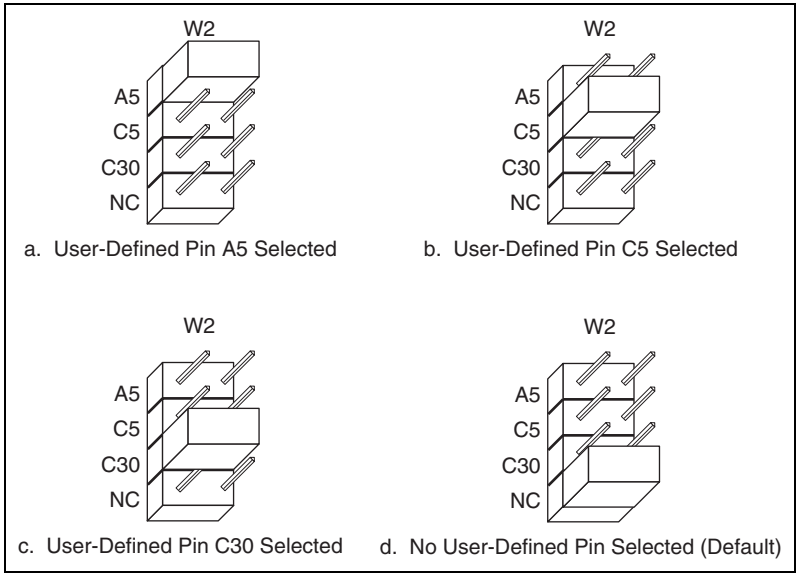


Figure 4-3. VME-MXI-2 Intermodule Signaling Settings

MXIbus Termination

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

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

Use switch 3 to select whether you want the VME-MXI-2 to automatically control termination of the MXIbus. Switch 4 lets you manually control whether to terminate the MXIbus when automatic termination is turned off.

Switch 4 has no effect when switch 3 is set for automatic MXIbus termination; you must turn off automatic termination if you want to manually control termination.

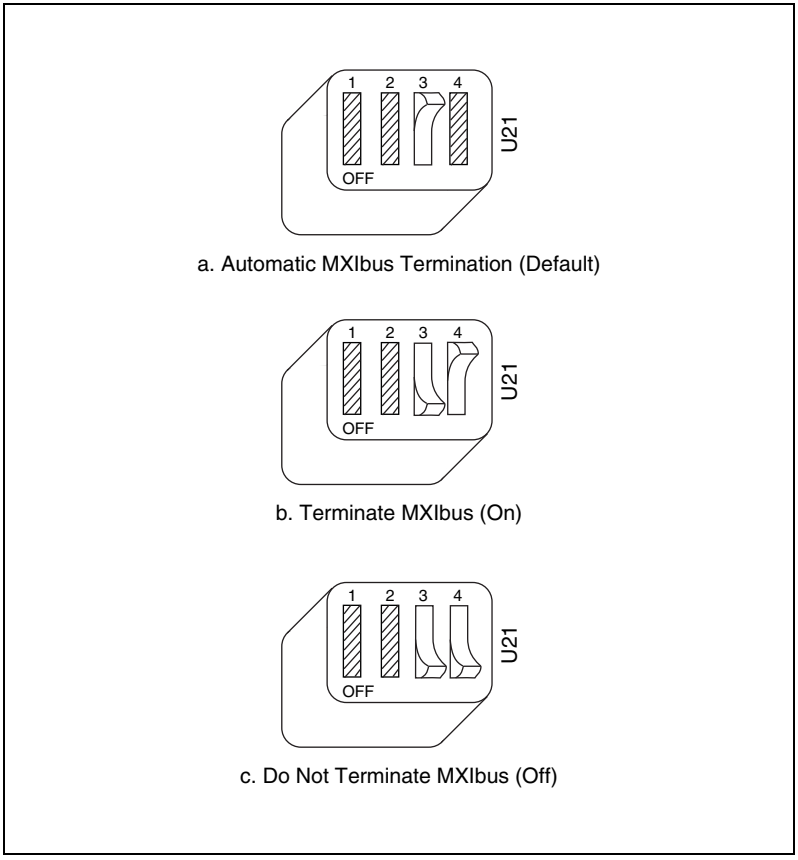


Figure 4-4. MXIbus Termination

Configuration EEPROM

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

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

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

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

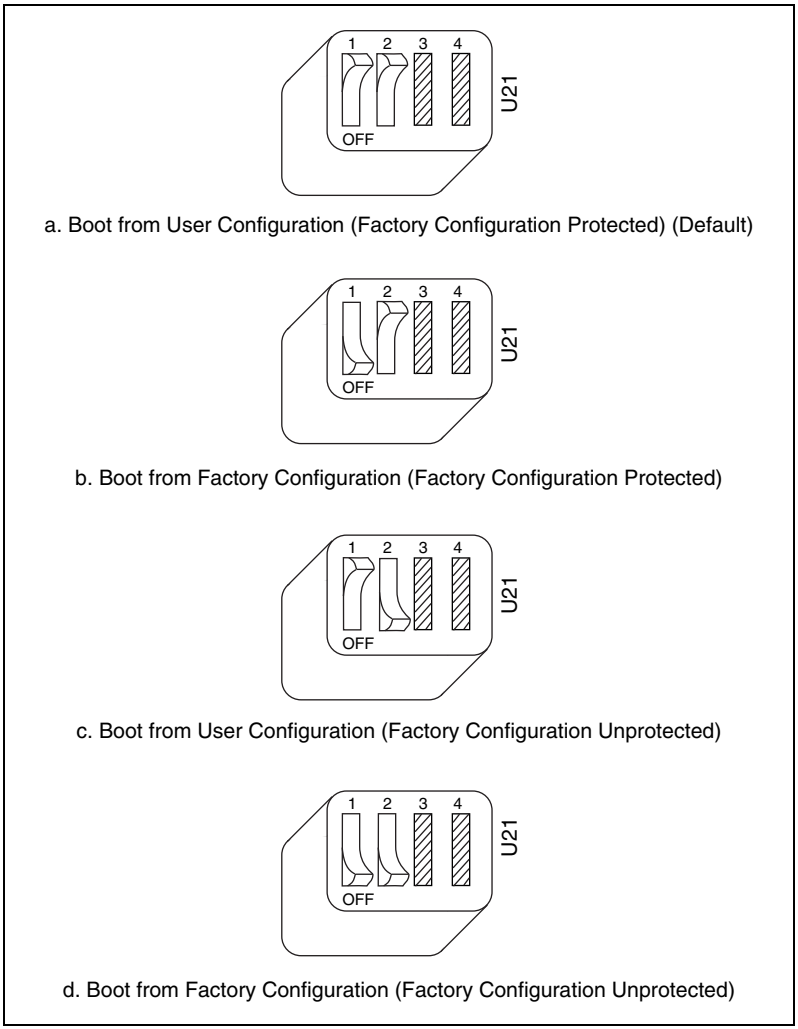


Figure 4-5. EEPROM Operation

Onboard DRAM

The VME-MXI-2 can accommodate up to two 1.35 in. DRAM SIMMs. Table 4-1 lists the SIMMs you can use. You can use 32- or 36-bit SIMMs since DRAM parity is not required. Because the VME-MXI-2 supports only one organization at a time, all SIMMs installed must be of the same type. Use Bank 0 first when installing SIMMs. This allows you to install up to 64 MB. The VME-MXI-2 supports DRAM speeds of 80 ns or faster.

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

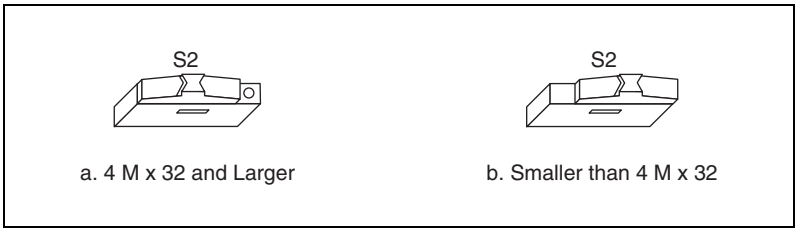


Figure 4-6. SIMM Size Configuration

Refer to Table 4-1 for how to adjust the switch (ON or OFF) for all supported DRAM configurations. Many of the DRAM options are available from National Instruments.

Table 4-1. VME-MXI-2 DRAM Configurations

Bank 0	Bank 1	Total DRAM	National Instruments Option	Switch Setting of S6
—	—	0	—	—
256 K × 32 or 256 K × 36	—	1 MB	—	ON
256 K × 32 or 256 K × 36	256 K × 32 or 256 K × 36	2 MB	—	ON
512 K × 32 or 512 K × 36	—	2 MB	—	ON
512 K × 32 or 512 K × 36	512 K × 32 or 512 K × 36	4 MB	—	ON

Table 4-1. VME-MXI-2 DRAM Configurations (Continued)

Bank 0	Bank 1	Total DRAM	National Instruments Option	Switch Setting of S6
1 M × 32 or 1 M × 36	—	4 MB	YES	ON
1 M × 32 or 1 M × 36	1 M × 32 or 1 M × 36	8 MB	—	ON
2 M × 32 or 2 M × 36	—	8 MB	YES	ON
2 M × 32 or 2 M × 36	2 M × 32 or 2 M × 36	16 MB	—	ON
4 M × 32 or 4 M × 36	—	16 MB	YES	OFF
4 M × 32 or 4 M × 36	4 M × 32 or 4 M × 36	32 MB	—	OFF
8 M × 32 or 8 M × 36	—	32 MB	YES	OFF
8 M × 32 or 8 M × 36	8 M × 32 or 8 M × 36	64 MB	YES	OFF

Install the VME-MXI-2

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

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

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

2. Remove or open any doors or covers blocking access to the mainframe slots.

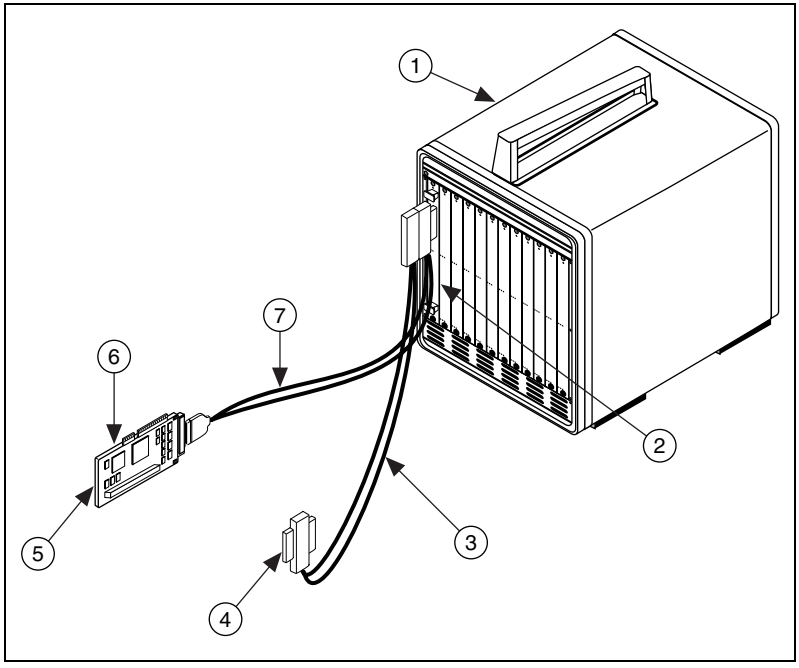
3. Insert the VME-MXI-2 in the slot you have selected by aligning the top and bottom of the board with the card-edge guides inside the mainframe. Slowly push the VME-MXI-2 straight into the slot until its plug connectors are resting on the backplane receptacle connectors. Using slow, evenly distributed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

Connect the MXIbus Cable

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

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

Figure 4-7 shows the correct cabling for a VME system containing a PCI-MXI-2 board in a PCI-based computer cabled to a VME-MXI-2 module residing in Slot 1 of a VMEbus mainframe. Notice that you can expand your system to include other devices by using an additional MXI-2 cable. However, in such a case the first cable needs to have a double connector on one end. You can then use a cable with a single connector on each end to connect the last device on the MXIbus.



1	VME Chassis	5	Connection to External Computer
2	VME-MXI-2 Module	6	PCI-MXI-2 Interface Module
3	Additional MXI-2 Cable	7	MXI-2 Cable
4	Connection to Other Mainframes		

Figure 4-7. MXI-2 Cable Configuration Using a PCI-MXI-2 and a VME-MXI-2

When you have properly connected the MXI-2 cable, power on the VMEbus mainframe and then the computer.

Note Always turn on the mainframe first. Doing so makes it possible for your external computer to access the VME boards in the mainframe upon startup.

5

NI-VXI/NI-VISA Software Installation

This chapter describes how to install and uninstall the NI-VXI/NI-VISA software for Linux.

Installing the NI-VXI/NI-VISA Software for Linux

Before you begin, you may need to install Linux on your computer. Refer to the Linux documentation for instructions. After your computer is booted into Linux, you are ready to install the NI-VXI/NI-VISA software.

To install NI-VXI/NI-VISA for the PCI-MXI-2 for Linux, perform the following steps:

1. Insert the *NI-VXI/NI-VISA for Linux* CD.
2. Login to your system as `root`.
3. Mount the CD-ROM.
4. To change the current directory to the mounted CD-ROM, type the following command:

   ```
   cd /media/cdrom
   ```

Note This command may vary depending on your Linux distribution.

5. To run the installation script, type the following command:

   ```
   ./INSTALL
   ```

The `INSTALL` places NI-VXI and NI-VISA in their default locations. The script uses `rpm` to install the packages. The script also optionally installs support for NI-VXI in LabVIEW.

Note Refer to the `README` file on the CD-ROM for additional important information and instructions.

Removing the NI-VXI Driver for Linux

To uninstall the driver, you must meet the following requirements:

- You must have superuser privileges.
- The driver must not be in use.

To remove the NI-VXI/NI-VISA software, use the uninstall script on the CD-ROM. Follow these steps to uninstall the software:

1. Follow steps 1–4 in the *Installing the NI-VXI/NI-VISA Software for Linux* section.
2. To run the uninstallation script, type the following command:

   `./UNINSTALL`

Using the NI-VXI/NI-VISA Software

The NI-VXI software is configured to be loaded in the `/usr/local/nivxi` directory.

The NI-VISA software is configured to be loaded in the `/usr/local/vxipnp` directory.

Completing the Software Installation

After the software is installed, run `resman`, the National Instruments Resource Manager. You must run `resman` every time the computer or chassis power is cycled so that your application can access devices in the VXI/VME chassis.

After you run `resman`, you are ready to use the NI-VXI Resource Editor program `vxiedit` to interactively configure the hardware in your system. Refer to Chapter 6, *NI-VXI Configuration Utility*, for instructions on using the configuration editors in `vxiedit`.

6

NI-VXI Configuration Utility

This chapter contains instructions for using the VXI Resource Editor utility of the NI-VXI software to configure the PCI-MXI-2 and the VXI-MXI-2 or VME-MXI-2.

`vxiedit` is the VXI resource editor program that you use to configure the system and to edit the manufacturer name and ID numbers, the model names of VXI and non-VXI devices in the system, and the system interrupt configuration information. This program also displays the system configuration information generated by the Resource Manager.

Note A text-based version, `vxitedit`, is also available as an alternative. Although this chapter focuses only on the graphical `vxiedit` program, the two programs are functionally equivalent. For information on `vxitedit`, refer to the *NI-VXI Text Utilities Reference Manual*.

Running the VXIedit Configuration Utility

To run `vxiedit`, type `vxiedit` at the command prompt. You can run `vxiedit` from any directory, but make sure that both the `PATH` and `NIVXIPATH` environment variables have the destination directory of the NI-VXI software added to them. `NIVXIPATH` is used by the application to find the different configuration files (`*.cfg`), table files (`*.tbl`), and help files (`*.hlp`) during its execution. The default pathname used by the program if `NIVXIPATH` is not set is `/usr/local/nivxi`.

Most of the features on the PCI-MXI-2, VXI-MXI-2, and VME-MXI-2 are configurable through software, using `vxiedit`, rather than through hardware switches or jumpers on the boards themselves. In addition, the `vxiedit` utility can override some of the hardware settings.

Figure 6-1 shows the main menu of the `vxiedit` resource editor.

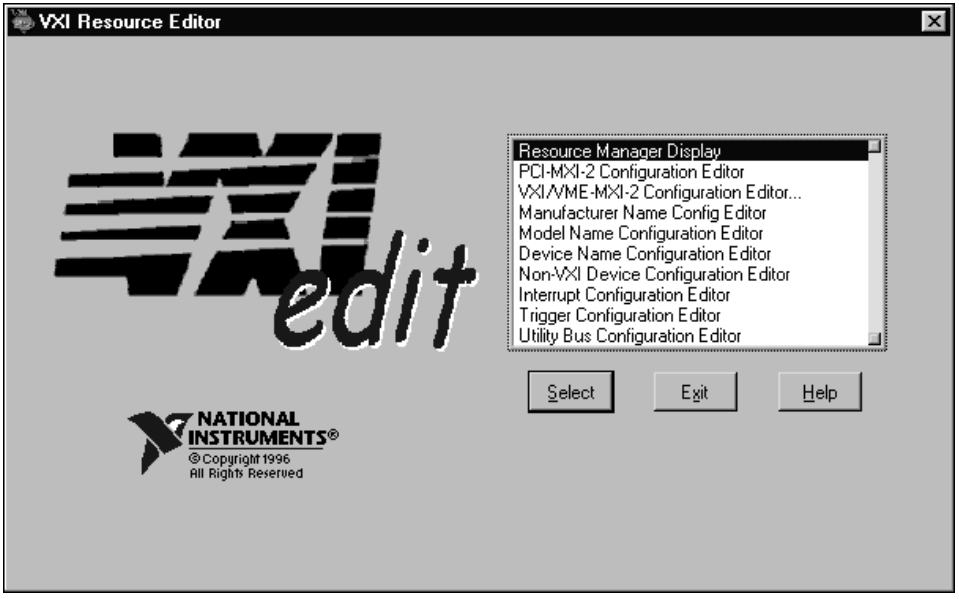


Figure 6-1. VXIedit Main Screen

The rest of this chapter describes only the features of the PCI-MXI-2 Configuration Editor and the VXI/VME-MXI-2 Configuration Editor. For instructions on using the other editors, refer to your software utility reference manual—either the *NI-VXI Graphical Utilities Reference Manual* or the *NI-VXI Text Utilities Reference Manual*.

PCI-MXI-2 Configuration Editor

Figure 6-2 shows the opening screen of the PCI-MXI-2 Configuration Editor. Notice that the screen displays the serial number and hardware revision of the PCI-MXI-2 board in addition to several configuration options.

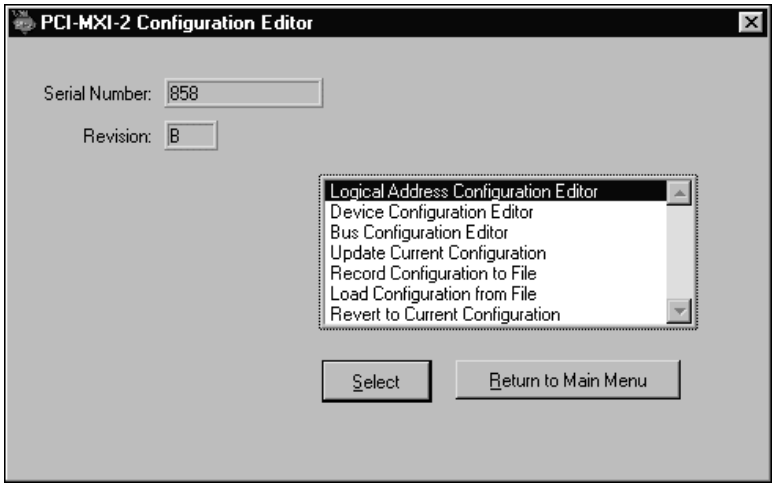


Figure 6-2. PCI-MXI-2 Configuration Editor

The first three options under the PCI-MXI-2 Configuration Editor are as follows:

- **Logical Address Configuration Editor**
- **Device Configuration Editor**
- **Bus Configuration Editor**

When making changes to the PCI-MXI-2 through these editors, remember that the changes do not take effect until you commit them by selecting the **Update Current Configuration** option.

Before proceeding to a description of each field in these editors, review the remaining four options of the PCI-MXI-2 Configuration Editor. These options directly relate to how you can use the changes you make using the configuration editors, which are described after the options.

Update Current Configuration

Use this option to write the configuration settings to the PCI-MXI-2 EEPROM and files used by NI-VXI. Notice that some of the configuration settings cannot take effect until you reset the machine, either by using the reset button or by turning the power off and on again.

Record Configuration to File

With this option you can save your configuration settings to a file. Notice that this option does *not* write the configuration settings to the PCI-MXI-2 configuration EEPROM.

If you want to update the PCI-MXI-2 configuration settings, use the **Update Current Configuration** option instead.

Load Configuration from File

You can use this option to load your configuration settings from a file. This action only updates the configuration settings in your editor. This does *not* write the configuration settings to the PCI-MXI-2 configuration EEPROM. To update the configuration use the **Update Current Configuration** option for the changes to take effect.

Revert to Current Configuration

If you made changes to the configuration settings without committing those changes (writing to configuration EEPROM), you can revert the configuration settings to the values they had before you made the changes.

Note You can successfully revert only if you have *not* yet selected the **Update Current Configuration** option.

Logical Address Configuration Editor

Figure 6-3 shows the Logical Address Configuration Editor. Notice that the options are arranged into three groups—**Device Settings**, **VXI Shared Memory**, and **Resource Manager**. The following sections describe the options you can select for each of the fields.

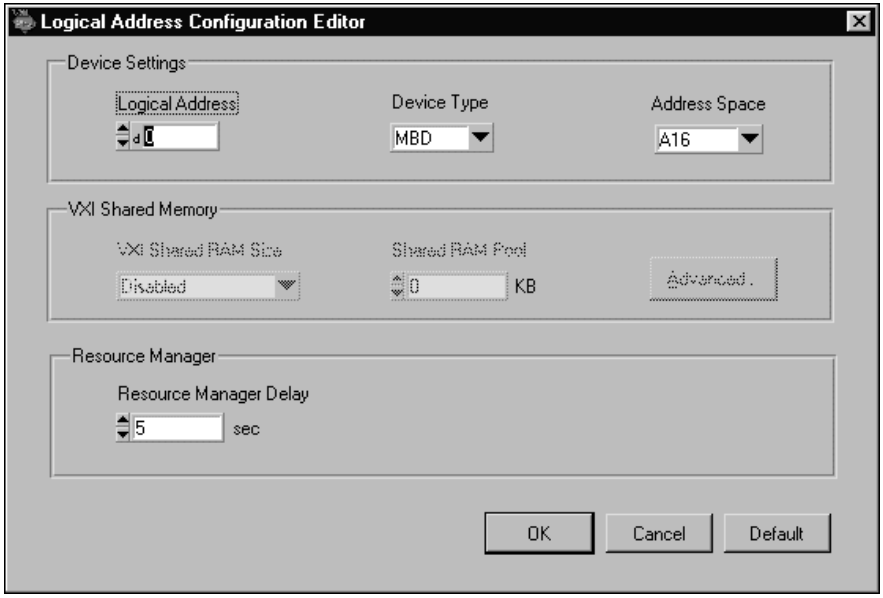


Figure 6-3. PCI-MXI-2 Logical Address Configuration Editor

Device Settings

The **Device Settings** group contains the controls to set the logical address, device type, and address space of the PCI-MXI-2.

Logical Address

This parameter sets the MXI logical address of the PCI-MXI-2. The following table shows the allowable range of values and the default value.

Logical Address Range	Default Value
0 to 254	0

Device Type

This field indicates the classification of the PCI-MXI-2. The default value is **MBD**, designating a message-based device. The following table shows the available options.

Classification	Setting
Extended Device	**EXT**
Message-Based Device	**MBD**
Register-Based Device	**RBD**

The device type affects only the contents of the Device Class field in the Device Type register. The functionality of the other registers does not change.

Address Space

This field indicates the addressing mode(s) of the device's operational registers. The PCI-MXI-2 can be configured in one of three ways. The default addressing mode is for A16 space only. Your other options are A16/A24 and A16/A32.

Notice that several other controls on the configuration editor panel are disabled when the addressing mode is A16, as shown in Figure 6-3. Only if you select A16/A24 or A16/A32 are the following controls relevant:

- **VXI Shared RAM Size**
- **Shared RAM Pool**
- **Advanced**
  - **Byte Swapping**
  - **Memory Select**
  - **Mapping**

VXI/VME Shared Memory

The **VXI Shared Memory** group contains the controls to set the VXI and VME shared RAM size and the shared RAM pool. The **Advanced** button leads to additional options that configure the upper and lower half of the shared RAM area.

VXI/VME Shared RAM Size

This field indicates the amount of RAM (in bytes) that is shared in either A24 or A32 space. This determines the *total* shared RAM size, which is then divided into two equal halves that you can set up independently of one another.

Note When the Address Space field is in the default setting of A16 only, this field is ignored.

Shared RAM Pool

This field indicates the size of memory (in kilobytes) that is allocated at startup. This memory is used by the lower/upper half window when the **Memory Select** control—accessible through the Advanced popup field—is set to **System memory**.

Memory Range	Default Value
0 to 65535 KB	0 KB

The following table indicates how the **Shared RAM Pool** relates to the **VXI Shared RAM Size** depending on the setting of the **Memory Select** control for the upper and lower half windows.

Lower Half Window	Upper Half Window	Shared RAM Pool (Window)
System memory	System memory	Equal to VXI/VME Shared RAM Size
System memory	Onboard memory	Half the VXI/VME Shared RAM Size
Onboard memory	System memory	Half the VXI/VME Shared RAM Size
Onboard memory	Onboard memory	0

The shared RAM pool is used by `VXImemAlloc()` function calls. For information on the `VXImemAlloc()` function, refer to the *NI-VXI User Manual* and the *NI-VXI Programmer Reference Manual.*

Note When the Address Space field is in the default setting of A16 only, this field is ignored. This field is also ignored if the Memory Select fields for both the lower and upper half windows are set to **Onboard memory**.

Advanced Shared RAM Settings

Clicking the **Advanced** button displays a dialog box to configure the destination of MXIbus cycles that map into the PCI-MXI-2 through the A24/A32 shared RAM.

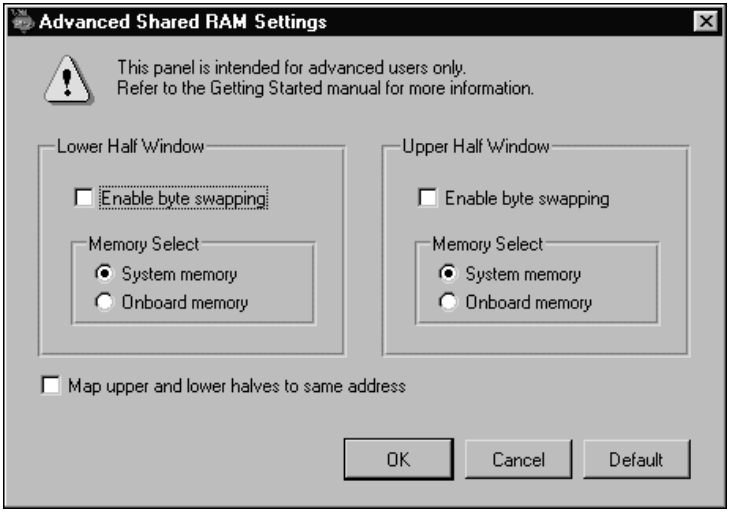


Figure 6-4. Advanced Shared RAM Settings

Note When the Address Space field is in the default setting of A16 only, these fields are ignored, and cannot be accessed.

The VXI/VME shared RAM is divided into two halves, or *windows*. You can select the byte order for each half independently. You can map each half of the VXI/VME shared RAM independently into system memory on the motherboard or into onboard memory on the PCI-MXI-2.

Because each half is independent of the other, you can choose from any of the following mapping options:

- Half the VXI/VME shared RAM mapped to system memory; the other half mapped to PCI-MXI-2 onboard memory
- Both halves mapped to PCI-MXI-2 onboard memory
- Both halves mapped to system memory

Enable Byte Swapping

This checkbox indicates whether byte swapping should be performed for slave accesses to this half of the VXI/VME shared RAM space. For example, if the native byte order of the shared RAM is Motorola (big-endian), and you want to present data to the VXI/VMEbus in Intel (little-endian) byte order, you will need to enable byte swapping. The default value is non-swapped. Click on the checkbox if you want to enable byte swapping.

This field is ignored if the Memory Select field is set to **Onboard memory**.

Memory Select

This option determines where this half of the VXI/VME shared RAM is mapped. By default, the shared RAM is mapped to **System memory**. If you want to use the RAM on the PCI-MXI-2, choose the **Onboard memory** option.

Window Mapping

You can click on the checkbox at the bottom of the screen if you want to map both halves of the inward window to the same address. When both halves of the inward window are mapped to the same destination with the same byte order, the windows essentially form one continuous window. If the windows are mapped to different destinations, the base of each inward window maps to the base of each destination.

If the windows both map to the shared RAM destination but the byte order is different, the base of each inward window maps to the base of the shared RAM destination. This results in one half of the window accessing the system RAM in little-endian byte order and the other half accessing it in big-endian byte order.

Caution There is a potential problem when opening up a shared memory region to point to system RAM. The PCI bus may return a retry on any cycle into system RAM. As a consequence, an external VXI/VME device accessing the system RAM may get a VXI/VME retry back. If the external VXI/VME device does not support VXI/VME retry, the VXI/VME device will falsely detect the retry condition as a bus error condition.

VXI/VME devices that support retries will not have this problem, because they can handle VXI/VME retry conditions correctly by automatically retrying the access. For example, the National Instruments VXI-DAQ boards handle VXI/VME retry conditions properly, and do not exhibit this problem.

Resource Manager Delay

The only option under the Resource Manager portion of the Logical Address Configuration Editor is the **Resource Manager Delay** control.

Note This field is effective only when the PCI-MXI-2 is at its default logical address of 0. The PCI-MXI-2 is the Resource Manager only if its logical address is 0.

This field specifies the time in seconds that the Resource Manager (RM) waits before accessing any other VXI/VMEbus device's A16 configuration registers.

RM Delay Range	Default Value
0 to 65535 s	5

Figure 6-5 shows the Device Configuration Editor. The following paragraphs describe the options you can select for each of the fields.

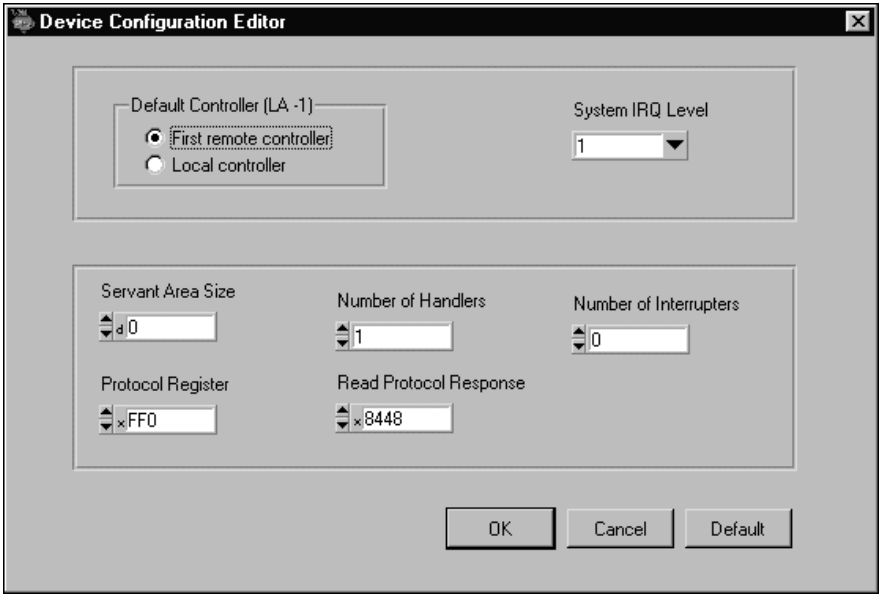


Figure 6-5. PCI-MXI-2 Device Configuration

Default Controller (LA –1)

Many NI-VXI functions use **controller** as a parameter with –1 accepted as a valid value. You use this selection to determine which controller you are referring to when you use –1 in these NI-VXI functions. Review the descriptions of the NI-VXI functions to determine which are applicable for this control.

By default, the **Default Controller (LA –1)** option is set to **First remote controller**, so that any NI-VXI functions that are passed the value of –1 for the **controller** parameter will be executed on the first VXI/VME-MXI-2 in the MXI-2 chain. If you select the **Local controller** option, the NI-VXI functions execute on the PCI-MXI-2.

System IRQ Level

The remote controllers—in this case the VXI/VME-MXI-2—can report events such as triggers and DMA to the PCI-MXI-2 through a VXI IRQ line. This field selects which VXI IRQ level the remote controllers should use to report events to the PCI-MXI-2.

Interrupt Request Levels	Default Value
1 to 7	1

Servant Area Size

This field designates the servant area size, which is supplied to the Resource Manager in response to the Read Servant Area command (if the PCI-MXI-2 is *not* the Resource Manager in your system). The servant area size is an 8-bit value (0 through 255) that indicates the servant area. The servant area begins at the logical address following the PCI-MXI-2 logical address, and includes *N* contiguous logical addresses, where *N* is the value of the servant area size. This field is meaningful only when the PCI-MXI-2 is configured as a message-based device.

Servant Area Range	Default Value
0 to 255	0

Note If the PCI-MXI-2 is the Resource Manager (Logical Address 0), this setting is irrelevant.

Number of Handlers

This field gives the number of interrupt handlers that the PCI-MXI-2 supports.

Interrupt Handlers	Default Value
0 to 7	1

Number of Interrupters

This field gives the number of interrupters that the PCI-MXI-2 supports.

Interrupters	Default Value
0 to 7	0

Protocol Register

This field specifies the contents of the Protocol register, indicating which protocols the device supports. This field is meaningful only when the PCI-MXI-2 is configured as a message-based device. The default value is 0x0ff0 (Commander, Signal Register, Master).

Read Protocol Response

This field specifies the response value to a Read Protocol command received by the PCI-MXI-2 from the Resource Manager (if the PCI-MXI-2 is *not* the Resource Manager in your system). This field is meaningful only when the PCI-MXI-2 is configured as a message-based device. The default value is 0x8448 (Response Generation, Event Generation, Programmable Handler, Word Serial Trigger, Instrument, Extended Longword Serial, Longword Serial).

Bus Configuration Editor

Figure 6-6 shows the Bus Configuration Editor. The following paragraphs describe the options you can select for each of the fields.

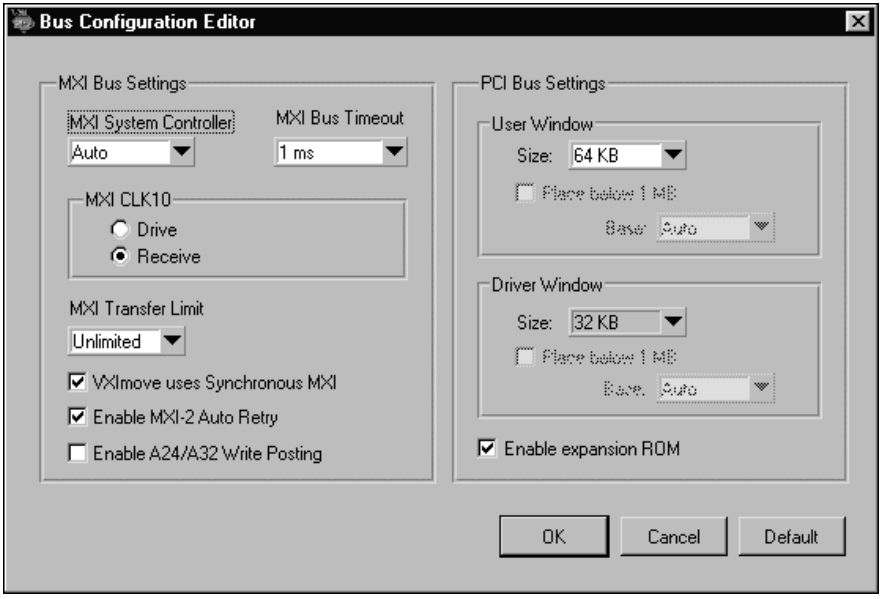


Figure 6-6. PCI-MXI-2 Bus Configuration Editor

MXI Bus

The following paragraphs describe the options for the **MXI Bus** portion of this editor.

MXI System Controller

You can use the **MXI System Controller** control to determine whether the PCI-MXI-2 acts as the MXIbus System Controller. The three options are **Auto**, **Yes**, and **No**.

When the **Auto** setting is active (the default setting), the PCI-MXI-2 automatically can sense from the MXIbus cable whether it should be the controller. This setting requires that the cable is attached properly before making any NI-VXI function calls or attempting to use the VXI/VME-MXI-2 Configuration Editor. Refer to the *Connect the MXIbus Cable* section of Chapter 3, *VXI-MXI-2 Configuration and Installation*, or the *Connect the MXIbus Cable* section of Chapter 4, *VME-MXI-2 Configuration and Installation*.

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

Note Make sure the MXI-2 cable is connected to the PCI-MXI-2 when you power on or reboot the computer. This is required for the PCI-MXI-2 to automatically detect whether it will be the MXIbus System Controller.

MXI Bus Timeout

The MXIbus Bus Timeout (BTO) is a watchdog timer for transfers on the MXIbus. The MXIbus BTO operates only when the PCI-MXI-2 is acting as the MXIbus System Controller.

After the specified amount of time has elapsed, the BTO circuitry terminates a MXIbus cycle if no slave has responded. The BTO is also disabled when the current MXIbus cycle maps to the VXI/VMEbus through a VXI/VME-MXI-2.

The default timeout value is 1 ms. If the **Enable MXI-2 Auto Retry** checkbox option is enabled, you should use a **MXI Bus Timeout** of 1 ms or greater.

MXI CLK10

The PCI-MXI-2 is capable of either receiving or driving the MXIbus CLK10 signal.

You can use the **Drive** or **Receive** options of the **MXI CLK10** feature to control the direction of the MXIbus CLK10 signal. By default all MXI-2 boards receive MXI CLK10 (the **Receive** option is active); therefore, you must choose one board on your MXI-2 bus to drive CLK10 by changing the value of the control to **Drive**. For most configurations, it is recommended to choose the System Controller as the CLK10 source for simplicity. The only exception you may want to make is if you want your triggers synchronous to the VXI 10 MHz clock.

Caution Do *not* configure more than one MXIbus device to drive MXIbus CLK10. Having a second device driving MXIbus CLK10 could damage the device.

MXI Transfer Limit

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

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

Synchronous MXI

The MXIbus has a special high-speed block protocol for transferring large blocks of data. This protocol, *SYNC-MXI*, locks the MXIbus during the transfer, which prevents anyone else from using the bus. `VXImove()` uses this protocol to transfer data.

If a VXI interrupt or signal comes in while a synchronous MXI move is underway, there can be a problem. Using the SYNC-MXI protocol will prohibit access to the MXI bus, which will prevent you from responding to an interrupt, and prevent bus monitoring devices from accessing the bus at regular intervals.

By default, this option is enabled—the **VXImove uses Synchronous MXI** checkbox is checked. You can also disable SYNC-MXI programmatically in `VXImove()`. However, if you have an older NI-VXI application that does not disable SYNC-MXI programmatically, you can deselect the checkbox to force your application to not use SYNC-MXI, if necessary.

MXI-2 Auto Retry

The PCI-MXI-2 has an automatic retry feature for cycles that map from the MXI bus to the PCI bus. By default this option is enabled—the **Enable MXI-2 Auto Retry** checkbox is checked.

Normally, when a cycle maps from the MXI bus to the PCI bus, any retry response received on the PCI bus is passed to the MXI bus. When the MXI-2 auto retry feature is enabled, the PCI-MXI-2 automatically retries any PCI cycle when the PCI host responds to a cycle with a retry. The PCI-MXI-2 automatically continues to retry the PCI cycle until it receives either a *DTACK* or *BERR* response, which it then passes to the MXI bus. This is the default situation because many external masters do not support VXI/MXI retries. If the external master does support retries, it may be beneficial to disable the MXI-2 auto retry feature. With this feature

disabled, you can lower the **MXI Bus Timeout** because there will be no delay due to the inward cycles being retried.

Note The PCI-MXI-2 has a limit on the number of automatic retries they will perform on any one cycle. If the limit is exceeded and the PCI-MXI-2 receives another retry, it will pass a retry back to the MXIbus even though the **Enable MXI-2 Auto Retry** checkbox is checked.

A24/A32 Write Posting

This field determines whether to enable write posting for incoming slave accesses to A24/A32 VXI/VME shared RAM. By default this option is disabled—the **Enable A24/A32 Write Posting** checkbox is cleared.

Enabling write posting will increase the throughput of your inward cycles. However, you should not enable write posting unless the destination of your inward A24/A32 cycles is onboard RAM, because cycles to onboard RAM will always complete successfully.

PCI Bus

The following sections describe the options for the **PCI Bus** portion of this editor.

User Window and Driver Window

The PCI-MXI-2 driver requires the use of two PCI windows: a user window and a driver window. Calling the `MapVXIAddress()` function allocates regions of the user window to your application. `VXIpeek()` and `VXIpoke()` accesses are performed through this window. NI-VXI uses the driver window to perform high-level functions such as `VXIin()` and `VXIout()`, and to access registers on the PCI-MXI-2 and VXI/VME-MXI-2.

The windows are mapped to PCI base address registers and determine the amount of PCI memory space the PCI-MXI-2 requests from the PCI system during initialization.

Window Size

The amount of space you can allocate for the user window is system dependent. You can use the **Size** control to select the size of the user window (minimum of 4 KB, maximum of 2 GB). The more you increase the size of the user window, the larger the window you can map in `MapVXIAddress()`.

You also can disable this option. Disabling the user window causes the PCI-MXI-2 to request the minimum amount of address space on the PCI bus. With the window disabled, you will be unable to perform any low-level function calls such as `VXIpeek()`, `VXIpoke()`, and `MapVXIAddress()`.

It is recommended to have a user window of at least 64 KB, and the default setting for the user window is set at this value. If you are going to be initiating transfers to a wide variety of addresses in both A24 and A32, you should increase the size of the user window. Any change you make to the size of the user window requires that you reboot your computer.

The size of the driver window, however, is system defined and is not user configurable.

Note Neither the user window nor the driver window can be placed below 1 MB with the Linux NI-VXI driver. Therefore, the **Place below 1 MB** option and the **Window Base** option will always be disabled.

Expansion ROM

Use the **Enable expansion ROM** control to enable or disable the PCI expansion ROM. The expansion ROM is enabled by default. It is recommended to retain the default setting. This option is included in `vxiedit` in case future versions of the PCI-MXI-2 do not implement a PCI expansion ROM.

VXI/VME-MXI-2 Configuration Editor

Before running the VXI/VME-MXI-2 Configuration Editor, you must run `resman`.

Note Throughout this section, the term *VXI/VME-MXI-2* denotes that the information applies to both the VXI-MXI-2 and the VME-MXI-2.

Upon entering the VXI/VME-MXI-2 Configuration Editor, the program displays a list of VXI/VME-MXI-2 boards that `resman` detected in your system, as shown in Figure 6-7.

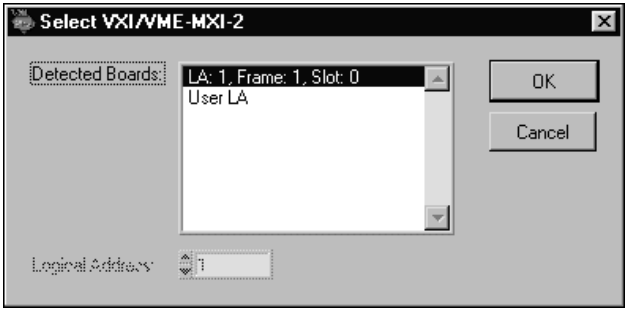


Figure 6-7. VXI/VME-MXI-2 Selection Dialog Box

Select the device you want to configure from the **Detected Boards** pull-down list, or you can select **User LA** and type in the board's logical address in the Logical Address field. Click **OK** to enter the editor or **Cancel** to return to the main menu.

After finding a VXI/VME-MXI-2, the VXI/VME-MXI-2 Configuration Editor displays a panel, as shown in Figure 6-8, that you can use to modify its configuration settings. The panel displays the current settings of the module. Notice that it also shows the hardware revision and serial number of the VXI/VME-MXI-2.

The title of the screen will reflect the model of the device you have. For instance, if you have a VXI-MXI-2, the title will read **VXI-MXI-2 Configuration Editor** as shown in the following screen.

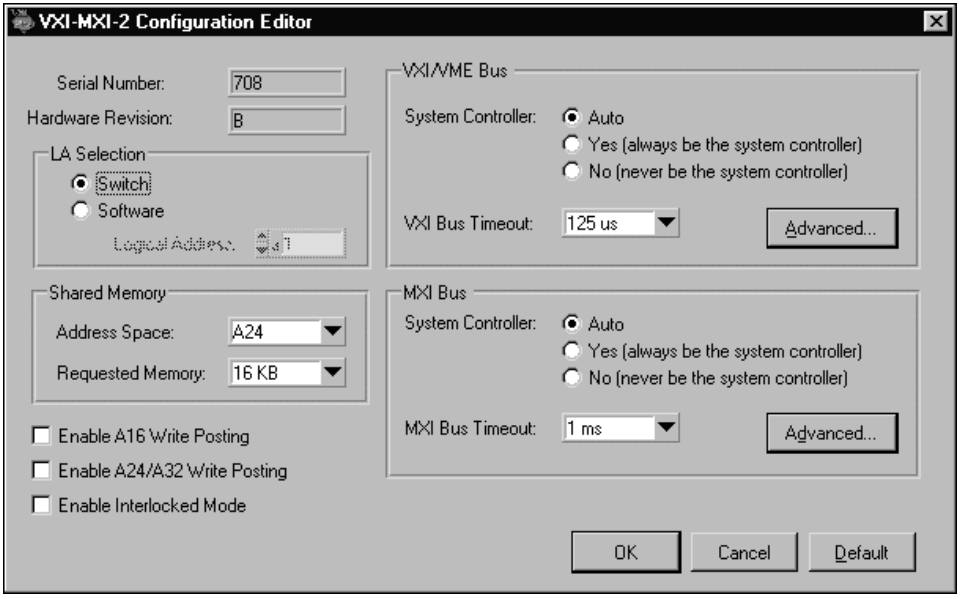


Figure 6-8. VXI/VME-MXI-2 Configuration Editor

LA Selection and Logical Address

You can set or modify the logical address of the VXI/VME-MXI-2 either within the VXI/VME-MXI-2 Configuration Editor itself or with the onboard 8-position DIP switch. To select the configuration method you prefer, use the **LA Selection** controls.

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

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

Address Space and Requested Memory

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

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

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

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

A16 and A24/A32 Write Posting

The VXI/VME-MXI-2 can increase performance with its capability to post write cycles from both the MXIbus and the VXI/VMEbus. Write cycles should be posted only to devices that cannot return a *BERR* signal, because the *BERR* will not be reported to the originating master.

Click on the checkbox control(s) if you want to use either A16 or A24/A32 write posting. By default, both options are disabled.

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

The A24/A32 write posting control affects write cycles that map through the A24 window and A32 window from the VXI/VMEbus to the MXIbus and vice-versa. This control also affects write cycles to the

VXI/VME-MXI-2 itself via its requested memory space from both the VXI/VMEbus and the MXIbus. For more information on the A16, A24, and A32 windows, refer to VXI-6, the *VXIbus Mainframe Extender Specification*.

Interlocked Mode

Interlocked arbitration mode is an optional mode of operation in which at any given moment the system can perform as if it were one large VXI/VMEbus mainframe with only one master of the entire system—VXI/VMEbus and MXIbus. This mode of operation prevents deadlocks by interlocking all arbitration in the VXI/VMEbus/MXIbus system. By default, this option is disabled, which puts the VXI/VME-MXI-2 in normal operating mode.

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

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

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

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

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

VXI/VME Bus Options

Use the options in this group to control features of the VXI/VMEbus interface on the VXI/VME-MXI-2.

VMEbus System Controller

You can use the **System Controller** control to override the jumper setting on the VXI-MXI-2. The VME-MXI-2 does not have an onboard jumper setting for this option. When the **Auto** setting (the default setting) is active, the onboard jumper setting determines if the VXI-MXI-2 is the VXI Slot 0 device. For more information, refer to the *VXIbus Slot 0/Non-Slot 0* section of Chapter 3, *VXI-MXI-2 Configuration and Installation*.

Otherwise, choose either the **Yes** or **No** option. Notice that selecting either of these options overrides the onboard jumper setting on the VXI-MXI-2, so it will not matter how the jumper is set. You would need to run the VXI/VME-MXI-2 Configuration Editor again if you decide to change the VMEbus System Controller (VXI Slot 0) setting at a later time.

Caution Do *not* install a VXI/VME-MXI-2 configured for VMEbus System Controller (VXI Slot 0) into another slot without first reconfiguring it to either Non-Slot 0 or automatic configuration. Neglecting to do this could damage the VXI/VME-MXI-2, the VXI/VMEbus backplane, or both.

This means that you should use either the **No** option or the **Auto** option of this control. For the VXI-MXI-2, you also have the option of changing the hardware jumper setting.

VXI/VME Bus Timeout Value

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

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

Advanced VXI Settings

Click the **Advanced** button to reach additional configuration options for the VXI/VME Bus portion of this editor, as shown in Figure 6-9. These options are intended for more advanced users.

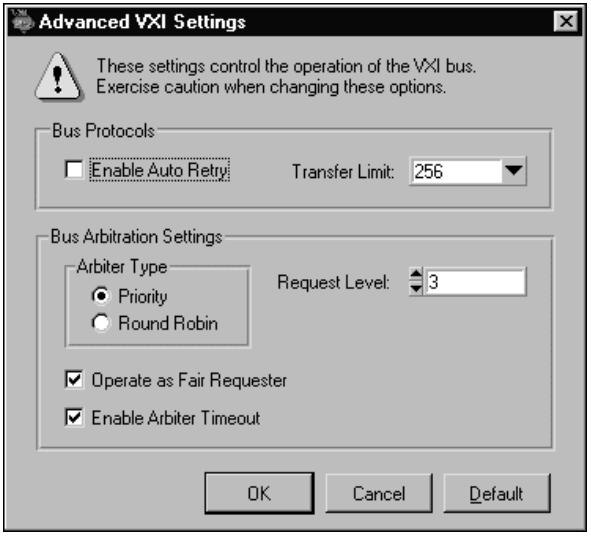


Figure 6-9. Advanced VXI Settings

VXI/VME Auto Retry

The VXI/VME-MXI-2 has an automatic retry feature for cycles that map from the VXI/VMEbus to the MXIbus. By default this option is disabled.

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

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

Transfer Limit

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

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

Arbiter Type

You can use the **Arbiter Type** feature to configure the VXI/VME-MXI-2 as either a Priority or Round Robin VMEbus arbiter. This control is applicable only if the VXI/VME-MXI-2 you are configuring is a VMEbus System Controller (VXI Slot 0) device. The default value is **Priority**.

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

Request Level

The VXI/VME-MXI-2 uses one of the four VMEbus request levels (0 to 3) to request use of the VME Data Transfer Bus (DTB). The VXI/VME-MXI-2 requests use of the DTB whenever an external MXIbus device, such as a PCI-based computer with a PCI-MXI-2 interface, attempts a transfer that maps into the VXI/VMEbus mainframe.

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

VXI/VME Fair Requester

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

Arbiter Timeout

An arbitration timeout feature is available on the VXI/VME-MXI-2 when it is acting as the VMEbus arbiter. This feature applies only to a VXI Slot 0 (VMEbus System Controller) VXI/VME-MXI-2. By default this option is enabled.

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

MXI Bus Options

Use the options in this group to control features of the MXIbus interface on the VXI/VME-MXI-2 module.

MXI Bus System Controller

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

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

MXI Bus Timeout Value

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

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

Advanced MXI Settings

Click the **Advanced** button to reach additional configuration options for the MXI Bus portion of this editor, as shown in Figure 6-10. These options are intended for more advanced users.

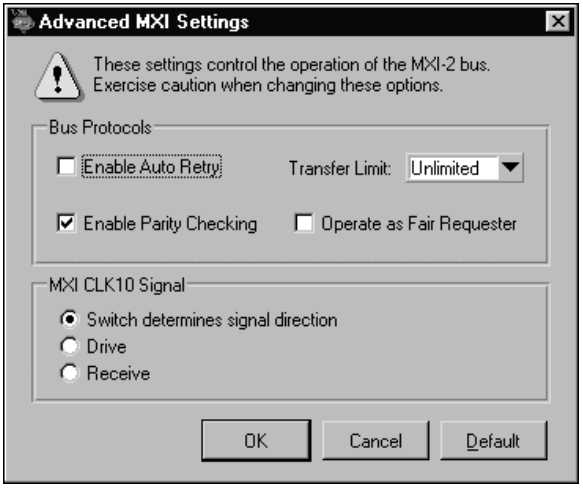


Figure 6-10. Advanced MXI Settings

MXI Auto Retry

The VXI/VME-MXI-2 has an automatic retry feature for cycles that map from the MXIbus to the VXI/VMEbus. This feature works in the same manner as the **Auto Retry** control described in the *VXI/VME Bus Options* section. By default, this option is disabled.

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

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

Transfer Limit

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

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

Parity Checking

By default, MXIbus parity checking is enabled and should not be disabled under normal circumstances. MXIbus parity is always generated regardless if checking is enabled or disabled.

MXI Fair Requester

You can use the **Operate as Fair Requester** checkbox control to configure the VXI/VME-MXI-2 as either a fair or unfair requester on the MXIbus. In its default setting (disabled), the VXI/VME-MXI-2 will request the bus at any time. If you enable this option, the VXI/VME-MXI-2 will request the MXIbus only when there are no requests pending from other MXIbus masters. This prevents other MXIbus masters from being starved of bandwidth.

MXI CLK10 Signal

The VXI-MXI-2 can either receive or drive the MXIbus CLK10 signal. In its default setting, the VXI-MXI-2 uses the switch setting of S7 to determine the signal direction.

- **VME users**—This option is not applicable to the VME-MXI-2.

  You can use the **Drive** or **Receive** options to override the setting of S7 and control the direction of the MXIbus CLK10 signal. When receiving the MXIbus CLK10 signal, configure the W3 jumper setting to use the MXIbus as the source for generating the VXIbus CLK10 (applicable only if the VXI-MXI-2 is a Slot 0 device). When driving the MXIbus CLK10, the VXIbus CLK10 is used as the source. In this case, change the jumper setting so that it does *not* use the MXIbus CLK10 as the source for the VXIbus CLK10.

Caution Do *not* configure more than one MXIbus device to drive MXI CLK10. Setting up a second device to drive MXI CLK10 could damage the device.

7

Using the NI-VXI/NI-VISA Software

This chapter discusses programming information for you to consider when developing applications that use the NI-VXI/NI-VISA driver.

After installing the driver software, you can begin to develop your VXI/VME application software. Be sure to check the `README` file for the latest application development notes.

You must run `resman` each time the computer or chassis power is cycled so that your application can access devices in the VXI or VME chassis.

The NI-VXI software was designed for use in VXI systems. Because VXI is a superset of VME, you can also use the NI-VXI functions as a comprehensive set of programming tools for VME systems. Refer to the *NI-VXI User Manual* and the NI-VXI online help for overviews of NI-VXI and detailed descriptions of the NI-VXI functions. The user manual is available in the *`NIVXI`*`/manuals` directory (where *`NIVXI`* refers to the actual location where you have installed the NI-VXI software). Use the Adobe Acrobat Reader program to open and navigate through this manual. Notice that the function descriptions indicate whether they apply to VXI only or both VXI and VME. The following function classes apply only to VXI:

- Commander Word Serial Protocol functions
- Servant Word Serial Protocol functions
- VXI Signal functions
- VXI Trigger functions

Refer to the *NI-VISA User Manual* to learn about VISA and how to use it in your system. The NI-VISA online help describes the attributes, events, and operations you can use in NI-VISA. The user manual is available in the *`VXIpnp`*`/linux/NIvisa/Manuals` directory. Use the Adobe Acrobat Reader program to open and navigate through this manual.

Interactive Control of NI-VXI/NI-VISA

The easiest way to learn how to communicate with your instruments is by controlling them interactively. Use the VXI/VME interactive control utility (`vic` or its text mode counterpart, `victext`) to write to and read from your instruments. This utility displays the status of your VXI/VME transactions and informs you of any errors that occur.

Refer to the online help for instructions on how to use `victext` and to learn about the features of each.

Example Programs

The `NIVXI/examples` subdirectory contains various example programs along with a makefile that show how to use various functions in the NI-VXI software and how to develop application programs using these functions. Make certain that the environment variable `NIVXIPATH` is set correctly as described in Chapter 5, *NI-VXI/NI-VISA Software Installation*. Also refer to your software utilities reference manual for additional examples.

For NI-VISA programming examples, look in *`VXIpnp`*`/linux/NIvisa/Examples`.

Programming Considerations

The following sections contain information for you to consider when developing Linux applications that use the NI-VXI/VISA bus interface software.

Multiple Applications Using the NI-VXI and VISA Libraries

Multiple-application support is another feature in the NI-VXI/NI-VISA libraries. You can have several applications that use the NI-VXI and/or NI-VISA libraries running simultaneously. In addition, you can have multiple instances of the same application running simultaneously. The NI-VXI/NI-VISA functions perform in the same manner whether you have only one application or several applications (or several instances of a single application) all using the NI-VXI/VISA libraries.

However, you do need to be careful in certain cases as described in the *Low-Level Access Functions* section.

Low-Level Access Functions

The memory windows used to access the VXI/VMEbus are a limited resource. You should follow the protocol of calling the `viMapAddress()` or `MapVXIAddress()` function with Access Only mode first before attempting to perform low-level VXI/VMEbus access with `viPeekX()`/`viPokeX()` or `VXIpeek()`/`VXIpoke()`. Your application should always call the `viUnmapAddress()` or `UnMapVXIAddress()` function immediately after the accesses are complete so that you free up the memory window for other applications.

The functions `viMapAddress()` and `MapVXIAddress()` return a pointer for use with low-level access functions. It is strongly recommended that you use the functions to access the memory instead of directly dereferencing the pointer. Using these functions makes the NI-VXI/VISA software more portable between platforms. Refer to the *Compiling Your C Program for NI-VXI/NI-VISA* section for more information on portability issues, and to your NI-VXI or NI-VISA software reference manual for more information on low-level VXIbus or VMEbus access functions.

Local Resource Access Functions

By using `vxiedit`, you can set up the PCI-MXI-2 to share either the system memory on the motherboard or the onboard memory on the PCI-MXI-2 with the VXI/VME system. Refer to the *NI-VXI Graphical Utilities Reference Manual* for more information on setting these parameters.

Notice that sharing the system memory with the VXI/VME system does not mean that the entire range of shared system memory is available to be used for VXI/VME transfers. You need to be cautious in specifying the portion of memory you want to share, as some areas are already used for other purposes.

Caution Use `viMemAlloc()` or `VXImemAlloc()` to allocate a buffer in the system memory that is reserved for your use only. Using any range of addresses that was not returned from `viMemAlloc()` or `VXImemAlloc()` to receive data may cause your computer to crash or behave incorrectly.

The onboard memory on your PCI-MXI-2, on the other hand, is entirely available to you. You can obtain the VXI address of your onboard memory using the `GetDevInfo()` function. When you have the VXI/VME address, you can access that memory using high-level or low-level VXI/VMEbus access functions.

System Configuration Functions

The System Configuration functions provide the lowest-level initialization of your VXI controller. For NI-VXI, use the `InitVXIlibrary()` function at the start of each application and the `CloseVXIlibrary()` function at the end of each application. For NI-VISA, use `viOpenDefaultRM()` at the start of each application and the `viClose()` function at the end of each application.

Compiling Your C Program for NI-VXI/NI-VISA

You can use the sample programs included with the NI-VXI software as a starting point to develop your own C program that uses NI-VXI functions. First, look over and compile the sample programs using the makefile provided to get familiar with how the functions operate. The example programs are broken into multiple files, and each one shows how to use different groups of functions. You can then modify the sample programs to try out different aspects of the NI-VXI software.

The sample programs for the Linux gcc compiler are in the `/usr/local/nivxi/examples` directory for NI-VXI and `/usr/local/vxipnp/linux/NIvisa/Examples` directory for NI-VISA.

The easiest way to compile the sample programs is to use the makefile included with the NI-VXI/NI-VISA software. For example, go to the `examples` directory and type `make`.

Symbols

You may need to define a symbol so that the NI-VXI library can work properly with your program. The `VXILINUX` symbol is required for NI-VXI applications, but not NI-VISA applications. `VXILINUX` designates the application as a Linux NI-VXI application. You can define this symbol using a `#define` statements in the source code or you can use the `-D` option in your compiler. If you use a `#define` statement, you must define the symbol before including the NI-VXI header file `nivxi.h`. If you use the makefiles to compile the sample program, the makefile defines the necessary symbol for you.

If you define this symbol in your source code, your source code should look something like the following sample code:

```
#define VXILINUX

.

.

.

#include <nivxi.h>
```

Refer to the documentation that came with your compiler package for detailed instructions about using the compiler and the various tools (linker, debugger, and so on). Your compiler documentation is an important and useful source of information for writing, compiling, and debugging C programs.

A

NI-VXI/NI-VISA Software Overview

This appendix lists and describes the main programs and files that make up the NI-VXI/NI-VISA software.

Main Programs and Files

This section lists the main programs and files that you can use for controlling your VXI/VME interface.

Note Any executable not listed in this section is used by the driver and should not be executed by the user directly.

- `resman` is the National Instruments multiple-mainframe Resource Manager.
- `vic` is a graphical interactive control program. This program is described in detail in the *NI-VXI Graphical Utilities Reference Manual*.
- `victext` is a text-based interactive control program with all the power of `vic`. This program is described in detail in the *NI-VXI Text Utilities Reference Manual*.
- `vxiedit` is a graphical VXI resource editor program. This program is described in detail in the *NI-VXI Graphical Utilities Reference Manual*.
- `vxitedit` is the text-based VXI resource editor program. This program is described in detail in the *NI-VXI Text Utilities Reference Manual*.
- `vxiinit` is the driver initialization program. Depending on the version of the driver installed on your system, it may or may not be present.
- The `README` file contains the latest updates and corrections to the manual when appropriate.

Header Files for NI-VXI

The *NIVXI*/include directory (where *NIVXI* is the actual location where you installed the NI-VXI software package) contains the following include files for the C language interface:

- nivxi.h is the main header file containing the C prototypes for the NI-VXI functions.
- datasize.h contains data size specifications.
- busacc.h contains parameter and return values for the bus access functions.
- devinfo.h contains parameter and return values for the device information and system configuration functions.
- vxiint.h contains parameter and return values for the interrupt and signal functions.
- sysint.h contains parameter and return values for the system interrupt functions.
- trig.h contains parameter and return values for the trigger functions. This file is useful in VXI systems but is not applicable for VME systems.
- ws.h contains parameter and return values for the Commander and Servant Word Serial functions. This file is useful in VXI systems but is not applicable for VME systems.

Header Files for NI-VISA

The *VXIpnp*/linux/include directory (where *VXIpnp* is the actual location where you installed the NI-VISA software package) contains the following include files for the C language interface:

- visa.h is the main header file containing the C prototypes for the NI-VISA functions.
- visatype.h contains VXI*plug&play* data type specifications.
- vpptype.h contains useful definitions for instrument drivers.

B

EEPROM Configuration

This appendix describes how to control the operation of the PCI-MXI-2 onboard EEPROM and how to fix an invalid EEPROM setting.

The EEPROM stores default registers values that are loaded at power-on. The EEPROM is divided into two halves so that you can modify one half, while the factory-configured half retains a back-up of the default user settings.

Controlling the EEPROM Operation

Use switch 1 (FOV) of the four-position switch at location U17 (SW1 on Universal) to control the operation of the EEPROM. Switch 1 determines whether the PCI-MXI-2 boots off the factory-configured half or the user-configurable half. In its default setting, the PCI-MXI-2 boots off the user-configurable half. This switch is useful in the event that the configuration becomes corrupted in such a way that the PCI-MXI-2 boots to an unusable state.

The TST switch (switch 2 of U17 on the PCI-MXI-2, or SW1 on the PCI-MXI-2 Universal) lets you change the factory-default configuration settings by permitting writes to the factory settings section of the EEPROM. This switch serves as a safety measure and should not be needed under normal circumstances. When this switch is off (its default setting) the factory configuration of the EEPROM is protected so any writes to the factory area will be ignored. The factory area is protected regardless of the setting of switch 1 of U17 or SW1.

Figure B-1 shows the default settings for EEPROM operation.

Caution Do *not* alter the settings of switches 3 and 4 of U17. Leave these switches as shown in Figure B-1 unless specifically directed by National Instruments.

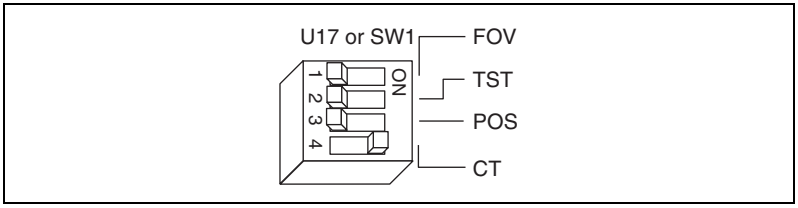


Figure B-1. EEPROM Operation Default Settings

Fixing an Invalid EEPROM Configuration

Certain EEPROM configurations can cause your PCI computer to lock up while in its boot process. Generally, only the size and location of the memory windows can cause problems with the PCI-MXI-2 locking up your system. For example, many PCI-based computers will not boot if a board in its system requests more memory space than the computer can allocate. If you encounter this situation you should reduce the size of the PCI-MXI-2 user window.

If this situation occurs after changing the configuration on the PCI-MXI-2, follow these steps to reconfigure the PCI-MXI-2.

1. Turn your computer off.

Caution To protect both yourself and the computer from electrical hazards, the computer should remain off while changing the settings on the PCI-MXI-2 module.

2. Remove the top cover or access port to the PCI bus.
3. Change switch 1 (FOV) on U17 (SW1 on the PCI-MXI-2 Universal) to the ON position as shown in Figure B-2 to restore the factory configuration.

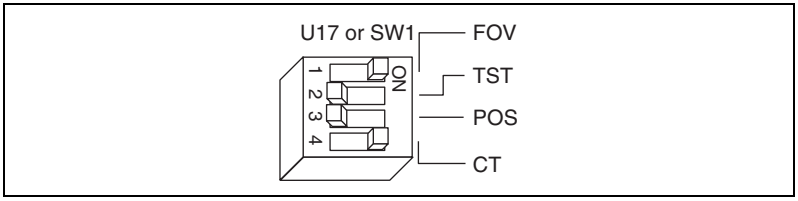


Figure B-2. Restoring the Factory Configuration

Note If you have to remove the PCI-MXI-2 module to access switch 1, follow the installation instructions given in Chapter 2, *PCI-MXI-2 Configuration and Installation*, to reinstall the PCI-MXI-2 module.

4. Replace the computer cover.
5. Turn on the computer. The computer should boot this time because the factory-default configuration is being used to initialize the PCI-MXI-2 module.
6. Run `vxiedit` to re-adjust the PCI-MXI-2 configuration. Refer to Chapter 6, *NI-VXI Configuration Utility*, for instructions on using this utility.
7. After saving the configuration, reboot the computer.
8. Remove the top cover or access port to the PCI bus.
9. Change switch 1 (FOV) on U17 (SW1 on the PCI-MXI-2 Universal) to the OFF position.
10. Replace the computer cover.
11. Turn on the computer. If the computer does not boot with this configuration, you will have to repeat these steps, modifying your configuration until a final configuration is reached.

C

Common Questions

This appendix addresses common questions you may have about using the NI-VXI bus interface software on the PCI-MXI-2 platform.

How can I determine which version of the NI-VXI software I have installed?

Run the NI-VXI utility program `vic`. Under the **Text** tab, type `ver` in the command field, and the utility will display the versions of `vic` and NI-VXI, and the latest PCI-MXI-2 board revision that this NI-VXI driver supports.

How can I determine the revision of the PCI-MXI-2 board that my NI-VXI software supports?

Running the NI-VXI utility program `vic` as described above will display the versions of `vic` and NI-VXI, and the hardware revision of the PCI-MXI-2 that the NI-VXI software supports.

How can I determine the serial number and hardware revision of the PCI-MXI-2 board?

Run the NI-VXI utility program `vxiedit`. Choose the **PCI-MXI-2 Configuration Editor**. The editor window displays the serial number and hardware revision of the PCI-MXI-2 board.

How can I determine the serial number and hardware revision of the VXI-MXI-2 or VME-MXI-2?

Run the NI-VXI utility program `vxiedit`. Choose the **VXI/VME-MXI-2 Configuration Editor**. The opening screen displays the serial number and hardware revision of the VXI-MXI-2 or VME-MXI-2.

Which NI-VXI utility program must I use to configure the PCI-MXI-2?

Use the VXI Resource Editor program `vxiedit` or its text-mode counterpart, `vxitedit`, to configure the PCI-MXI-2. The Resource Editor program is located in the *`NIVXI`*`/bin` directory (`/usr/local/nivxi/bin` by default).

Which NI-VXI utility program must I use to perform startup Resource Manager operations?

Use the `resman` program to perform startup Resource Manager operations. It is located in the *`NIVXI`*`/bin` directory (`/usr/local/nivxi/bin` by default). `resman` uses the settings in the Configuration Editor of `vxitedit`. It initializes your VXI/VMEbus system and stores the information that it collects to the `resman.tbl` file in the `tbl` subdirectory of the *`NIVXI`* directory.

What can I do to make sure that my system is up and running?

The fastest method for testing the system is to run `resman`. This program attempts to access memory in the upper A16 address space of each device in the system. If `resman` does not report any problems, the VXI/MXI communication system is operational.

To test individual devices, you can use the `vic` program or it text-mode counterpart, `victext`, to interactively issue NI-VXI functions. You can use the `VXIin()` and `VXIout()` functions or the `VXIinReg()` and `VXIoutReg()` functions to test register-based devices by programming their registers. If you have any message-based devices, you can send and receive messages with the `WSwrt()` and `WSrd()` functions. Notice that `VXIinReg()` and `VXIoutReg()` are for VXI devices only.

Finally, if you are using LabVIEW or LabWindows/CVI and you have instrument drivers for the devices in your chassis, you can use the interactive features of these programs to quickly test the functionality of the devices.

What do the LEDs on the front of the VXI-MXI-2 or VME-MXI-2 mean?

The **SYSFAIL** LED shows the state of the VXIbus/VMEbus SYSFAIL line. This line is asserted whenever any device in the chassis has not yet passed its self test, if it has failed its self test, or if it has detected a failure after originally passing its self test. The **MXI** LED indicates that the VXI-MXI-2 or VME-MXI-2 is acting as a slave to another device on the MXIbus, such as when the PCI-MXI-2 communicates with either the VXI-MXI-2 or VME-MXI-2 or with another device in the chassis. The **VXI** (**VME**) LED, when lit, indicates that the VXI-MXI-2 or VME-MXI-2 is acting as a slave to another device in the VXI (VME) chassis, such as when a bus master inside the chassis wants to talk to either the VXI-MXI-2 or VME-MXI-2 or another device outside the chassis.

Are the PCI-MXI-2 and the VXI-MXI-2 two devices or one with respect to the VXIbus?

Both the PCI-MXI-2 and the VXI-MXI-2 are unique VXIbus devices with their own logical addresses. However, the MXIbus allows the computer to behave as if it is inside the chassis with the VXI-MXI-2 by transparently converting PCI bus cycles to MXIbus cycles to VXIbus cycles, and vice versa.

I have a system that requires rugged chassis and bulkhead cables. Can I still use MXIbus?

Yes, National Instruments sells MXIbus bulkhead cables. Contact National Instruments for further information.

What kind of signal is CLK10 and what kind of signal do I need for an external CLK10?

CLK10 is a differential ECL signal on the VXIbus backplane. However, the oscillator for the VXI-MXI-2 and the EXTCLK input from the front panel use TTL. Therefore, you need to supply a TTL level signal for EXTCLK and our voltage converters will convert the signal to differential ECL.

CLK10 is not applicable to VME.

What is the accuracy of the CLK10 signal?

The CLK10 generated by the VXI-MXI-2 is 100 ppm (0.01%) as per the VXIbus specification. If you need a more accurate CLK10 signal, you can use the EXTCLK input at the front of the VXI-MXI-2.

CLK10 is not applicable to VME.

Whenever I try to execute any of the NI-VXI utilities, I receive a message similar to this one:

```
error in loading shared libraries: libnivxi.so: cannot
open shared object file: no such file or directory
```

What does this error message mean?

This usually means that the application could not load the NI-VXI library. Check the environment variable `NIVXIPATH` and your `/etc/ld.so.conf` file. `NIVXIPATH` should be set only to the directory where you installed NI-VXI (`/usr/local/nivxi` by default). `/etc/ld.so.conf` should include the directory where you installed the

NI-VXI library (/usr/local/nivxi/lib by default). Run ldconfig to reread this file.

Whenever I try to execute any of the NI-VXI utilities, I receive a message that it could not find a particular file even though the file does exist. What is wrong?

When a NI-VXI utility cannot find a file that it needs, it usually means that one of the environment variables is set incorrectly. Check the environment variable NIVXIPATH and your /etc/ld.so.conf file. NIVXIPATH should be set only to the directory where you installed NI-VXI (/usr/local/nivxi by default). /etc/ld.so.conf should include the directory where you installed the NI-VXI library (/usr/local/nivxi/lib by default). Run ldconfig to reread this file.

You can also receive this error message if you do not have full permissions to some of the NI-VXI files and directories. Users who will be using NI-VXI should have full permissions to the tbl and examples directories. They should also have read/write permissions for all the files contained in those directories.

Whenever I try to run resman without the MXI-2 cable plugged into my PCI-MXI-2, it hangs. This also happens if I have the wrong end of my cable plugged into the PCI-MXI-2. Why?

The PCI-MXI-2 uses the MXI-2 cable to find out if it is the MXIbus System Controller. If the correct end of the cable is not securely attached to the PCI-MXI-2, resman and other NI-VXI applications can hang. The MXI-2 cable has a label indicating which end should be plugged into the system controller. If you need to run resman with the MXI-2 cable unattached, you can force the PCI-MXI-2 to be system controller by setting the **MXI System Controller** field to **yes** in vxitedit.

D

Technical Support and Professional Services

Visit the following sections of the National Instruments Web site at ni.com for technical support and professional services:

- **Support**—Online technical support resources at ni.com/support include the following:
  - **Self-Help Resources**—For answers and solutions, visit the award-winning National Instruments Web site for software drivers and updates, a searchable KnowledgeBase, product manuals, step-by-step troubleshooting wizards, thousands of example programs, tutorials, application notes, instrument drivers, and so on.
  - **Free Technical Support**—All registered users receive free Basic Service, which includes access to hundreds of Application Engineers worldwide in the NI Developer Exchange at ni.com/exchange. National Instruments Application Engineers make sure every question receives an answer.

    For information about other technical support options in your area, visit ni.com/services or contact your local office at ni.com/contact.
- **Training and Certification**—Visit ni.com/training for self-paced training, eLearning virtual classrooms, interactive CDs, and Certification program information. You also can register for instructor-led, hands-on courses at locations around the world.
- **System Integration**—If you have time constraints, limited in-house technical resources, or other project challenges, National Instruments Alliance Partner members can help. To learn more, call your local NI office or visit ni.com/alliance.

If you searched ni.com and could not find the answers you need, contact your local office or NI corporate headquarters. Phone numbers for our worldwide offices are listed at the front of this manual. You also can visit the Worldwide Offices section of ni.com/niglobal to access the branch office Web sites, which provide up-to-date contact information, support phone numbers, email addresses, and current events.

Glossary

Prefix	Meanings	Value
p	pico	10^{-12}
n	nano	10^{-9}
µ	micro	10^{-6}
m	milli	10^{-3}
k	kilo	10^{3}
M	mega	10^{6}
G	giga	10^{9}
T	tera	10^{12}

Symbols

°	Degrees.
Ω	Ohms.
%	Percent.

A

A	Amperes.
A16 space	VXIbus address space equivalent to the VME 64 KB short address space. In VXI, the upper 16 KB of A16 space is allocated for use by VXI devices configuration registers. This 16 KB region is referred to as VXI configuration space.
A24 space	VXIbus address space equivalent to the VME 16 MB *standard* address space.
A32 space	VXIbus address space equivalent to the VME 4 GB *extended* address space.

ACFAIL	A VMEbus backplane signal that is asserted when a power failure has occurred (either AC line source or power supply malfunction), or if it is necessary to disable the power supply (such as for a high temperature condition).
address	Character code that identifies a specific location (or series of locations) in memory.
address modifier	One of six signals in the VMEbus specification used by VMEbus masters to indicate the address space in which a data transfer is to take place.
address space	A set of 2^n memory locations differentiated from other such sets in 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. In VME/VXI, because there are six address modifiers, there are 64 possible address spaces.
address window	A portion of address space that can be accessed from the application program.
ANSI	American National Standards Institute.
arbitration	A process in which a potential bus master gains control over a particular bus.
asynchronous	Not synchronized; not controlled by time signals.

B

B	Bytes.
backplane	An assembly, typically a printed circuit board, with 96-pin connectors and signal paths that bus the connector pins. A C-size 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.
BERR*	Bus error signal.
binary	A numbering system with a base of 2.
BIOS	Basic Input/Output System. BIOS functions are the fundamental level of any PC or compatible computer. BIOS functions embody the basic operations needed for successful use of the computer's hardware resources.

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.

C

C	Celsius.
CLK10	A 10 MHz, ± 100 ppm, individually buffered (to each module slot), differential ECL system clock that is sourced from Slot 0 of a VXIbus mainframe and distributed to Slots 1 through 12 on P2. It is distributed to each slot as a single-source, single-destination signal with a matched delay of under 8 ns.
CMOS	Complementary Metal Oxide Semiconductor; a process used in making chips.
Commander	A message-based device which is also a bus master and can control one or more Servants.
configuration registers	A set of registers through which the system can identify a module device type, model, manufacturer, address space, and memory requirements. In order to support automatic system and memory configuration, the VXIbus specification requires that all VXIbus devices have a set of such registers.

D

daisy-chain	A method of propagating signals along a bus, in which the devices are prioritized on the basis of their position on the bus.
Data Transfer Bus	DTB; one of four buses on the VMEbus backplane. The DTB is used by a bus master to transfer binary data between itself and a slave device.
DIP	Dual Inline Package.
DMA	Direct Memory Access; a method by which data is transferred between devices and internal memory without intervention of the central processing unit.
DRAM	Dynamic RAM.
driver window	A region of PCI address space that is decoded by the PCI-MXI-2 for use by the NI-VXI software.
DTACK*	Data Acknowledge signal.
DTB	*See* Data Transfer Bus.
dynamically configured device	A device that has its logical address assigned by the Resource Manager. A VXI device initially responds at Logical Address 255 when its MODID line is asserted. A MXIbus device responds at Logical Address 255 during a priority select cycle. The Resource Manager subsequently assigns it a new logical address, which the device responds to until powered down.
dynamic configuration	A method of automatically assigning logical addresses to VXIbus devices at system startup or other configuration times.

E

ECL	Emitter-Coupled Logic.
EEPROM	Electronically Erasable Programmable Read Only Memory.
embedded controller	An intelligent CPU (controller) interface plugged directly into the VXI backplane, giving it direct access to the VXIbus. It must have all of its required VXI interface capabilities built in.
EMC	Electromechanical Compliance.

EMI	Electromagnetic Interference.
expansion ROM	An onboard EEPROM that may contain device-specific initialization and system boot functionality.
external controller	In this configuration, a plug-in interface board in a computer is connected to the VXI mainframe via one or more VXIbus extended controllers. The computer then exerts overall control over VXIbus system operations.

F

fair requester	A MXIbus master that will not arbitrate for the MXIbus after releasing it until it detects the bus request signal inactive. This ensures that all requesting devices will be granted use of the bus.

H

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

I/O	Input/output; the techniques, media, and devices used to achieve communication between machines and users.
IC	Integrated Circuit.
IEEE	Institute of Electrical and Electronics Engineers.
in.	Inches.
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.
interrupt level	The relative priority at which a device can interrupt.
IRQ*	Interrupt signal.

K

KB	Kilobytes of memory.

L

LED	Light Emitting Diode.
logical address	An 8-bit number that uniquely identifies each VXIbus device in a system. It defines the A16 register address of a device, and indicates Commander and Servant relationships.

M

m	Meters.
master	A functional part of a MXI/VME/VXIbus device that initiates data transfers on the backplane. A transfer can be either a read or a write.
master-mode operation	A device is in master mode if it is performing a bus cycle which it initiated.
MB	Megabytes of memory.
MBLT	Eight-byte block transfers in which both the Address bus and the Data bus are used to transfer data.
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.
MITE	A National Instruments custom ASIC, a sophisticated dual-channel DMA controller that incorporates the Synchronous MXI and VME64 protocols to achieve high-performance block transfer rates.
MODID	Module Identification lines.
MTBF	Mean Time Between Failure.
MXI-2	The second generation of the National Instruments MXIbus product line. MXI-2 expands the number of signals on a standard MXIbus cable by including VXI triggers, all VXI interrupts, CLK10, SYSFAIL*, SYSRESET*, and ACFAIL*.

MXIbus	Multisystem eXtension Interface Bus; a high-performance communication link that interconnects devices using round, flexible cables.
MXIbus System Controller	A functional module that has arbiter, daisy-chain driver, and MXIbus cycle timeout responsibility. Always the first device in the MXIbus daisy-chain.

N

NI-VXI	The National Instruments bus interface software for VME/VXIbus systems.
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.

O

Onboard RAM	The optional RAM installed into the SIMM slots of the PCI-MXI-2 board or VXI/VME-MXI-2 module.

P

PCI	Peripheral Component Interconnect. The PCI bus is a high-performance 32-bit or 64-bit bus with multiplexed address and data lines.
propagation	The transmission of signal through a computer system.

R

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.
resman	The name of the National Instruments Resource Manager in NI-VXI bus interface software. *See also* Resource Manager.
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.

retry	An acknowledge by a destination that signifies that the cycle did not complete and should be repeated.
RPM	RPM Package Manager, a widely-used software distribution tool that you can use to install, upgrade, or remove software from your system.

S

s	Seconds.
Servant	A device controlled by a Commander; there are message-based and register-based Servants.
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.
SIMM	Single In-line Memory Module.
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 and arbitration for data transfers across the backplane. Installing such a device into any other slot can damage the device, the VXIbus backplane, or both.
statically configured device	A device whose logical address cannot be set through software; that is, it is not dynamically configurable.
SYSFAIL	A VMEbus signal that is used by a device to indicate an internal failure. A failed device asserts this line. In VXI, a device that fails also clears its PASSed bit in its Status register.
SYSRESET	A VMEbus signal that is used by a device to indicate a system reset or power-up condition.
System RAM	RAM installed on your personal computer and used by the operating system, as contrasted with onboard RAM, which is installed on the PCI-MXI-2 or VXI/VME-MXI-2.

T

trigger	Either TTL or ECL lines used for intermodule communication.
TTL	Transistor-Transistor Logic.

U

user window	A region of PCI address space reserved by the PCI-MXI-2 for use via the NI-VXI low-level function calls. `MapVXIAddress()` uses this address space to allocate regions for use by the `VXIpeek()` and `VXIpoke()` macros.

V

V	Volts.
VDC	Volts direct current.
VICtext	VXI Interactive Control Program, a part of the NI-VXI bus interface software package. Used to program VXI devices, and develop and debug VXI application programs.
VME	Versa Module Eurocard or IEEE 1014.
VMEbus System Controller	A device configured for installation in Slot 0 of a VXIbus mainframe or 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/VXIbus backplane, or both.
VXIbus	VMEbus Extensions for Instrumentation.
VXIinit	A program in the NI-VXI bus interface software package that initializes the board interrupts, shared RAM, VXI register configurations, and bus configurations.

VXItedit	VXI Resource Editor program, a part of the NI-VXI bus interface software package. Used to configure the system, edit the manufacturer name and ID numbers, edit the model names of VXI and non-VXI devices in the system, as well as the system interrupt configuration information, and display the system configuration information generated by the Resource Manager.

W

Word Serial Protocol	The simplest required communication protocol supported by message-based devices in a VXIbus system. It utilizes the A16 communication registers to transfer data using a simple polling handshake method.
write posting	A mechanism that signifies that a device will immediately give a successful acknowledge to a write transfer and place the transfer in a local buffer. The device can then independently complete the write cycle to the destination.

Index

D

E

F

G

H

I

K

L

M

N

P

Q

R

S

T

U

V

W