MXI

Getting Started with Your PCI-Based MXI-2 Interface for Windows 2000/NT/Me/9*x*

NATIONAL INSTRUMENTS™

January 2001 Edition
Part Number 321712B-01

Worldwide Technical Support and Product Information

`ni.com`

National Instruments Corporate Headquarters

11500 North Mopac Expressway   Austin, Texas 78759-3504   USA   Tel: 512 794 0100

Worldwide Offices

Australia 03 9879 5166, Austria 0662 45 79 90 0, Belgium 02 757 00 20, Brazil 011 284 5011,
Canada (Calgary) 403 274 9391, Canada (Ottawa) 613 233 5949, Canada (Québec) 514 694 8521,
China (Shanghai) 021 6555 7838, China (ShenZhen) 0755 3904939, Denmark 45 76 26 00,
Finland 09 725 725 11, France 01 48 14 24 24, Germany 089 741 31 30, Greece 30 1 42 96 427,
Hong Kong 2645 3186, India 91805275406, Israel 03 6120092, Italy 02 413091, Japan 03 5472 2970,
Korea 02 596 7456, Mexico 5 280 7625, Netherlands 0348 433466, New Zealand 09 914 0488,
Norway 32 27 73 00, Poland 0 22 528 94 06, Portugal 351 1 726 9011, Singapore 2265886, Spain 91 640 0085,
Sweden 08 587 895 00, Switzerland 056 200 51 51, Taiwan 02 2528 7227, United Kingdom 01635 523545

For further support information, see the *Technical Support Resources* appendix. To comment on the documentation, send e-mail to `techpubs@ni.com`

Important Information

Compliance

FCC/Canada Radio Frequency Interference Compliance*

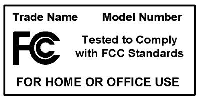
Determining FCC Class

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). Depending on where it is operated, this 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. By examining the product you purchased, you can determine the FCC Class and therefore which of the two FCC/DOC Warnings apply in the following sections. (Some products may not be labeled at all for FCC; if so, the reader should then assume these are Class A devices.)

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

FCC Class B products display either a FCC ID code, starting with the letters **EXN**, or the FCC Class B compliance mark that appears as shown here on the right.



Consult the FCC web site `http://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 Mark 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 National Instruments 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 will be required to correct the interference at his 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.

Class B

Federal Communications Commission

This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient or relocate the receiving antenna.
- Increase the separation between the equipment and receiver.

- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.
- Consult the dealer or an experienced radio/TV technician for help.

Canadian Department of Communications

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

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

European Union - Compliance to EEC Directives

Readers in the EU/EEC/EEA must refer to the Manufacturer's Declaration of Conformity (DoC) for information** pertaining to the CE Mark compliance scheme. The Manufacturer includes a DoC for most every hardware product except for those bought for OEMs, if also available from an original manufacturer that also markets in the EU, or where compliance is not required as for electrically benign apparatus or cables.

* Certain exemptions may apply in the USA, see FCC Rules §15.103 **Exempted devices**, and §15.105(c). Also available in sections of CFR 47.

** The CE Mark Declaration of Conformity will contain important supplementary information and instructions for the user or installer.

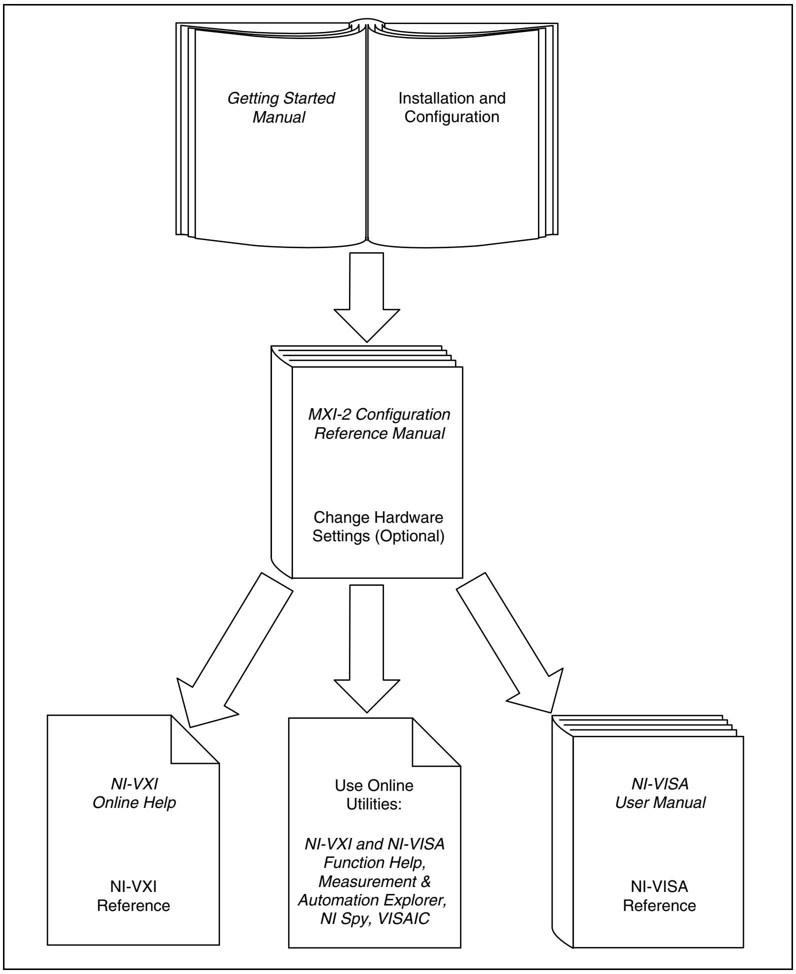
Contents

About This Manual

This manual contains instructions for installing and configuring the National Instruments PCI-based MXI-2 interface kit for Windows. This manual is meant to be used with the *MXI-2 Configuration Reference Manual*.

Your kit contains either a VXI-MXI-2 or VME-MXI-2, which plugs into your VXI or VME mainframe and links your computer to the VXIbus or VMEbus, respectively. The kit also contains either a PCI-MXI-2 or a PXI-8320 interface board, which links your PCI-based or PXI/CompactPCI computer to the MXIbus. Your software consists of the NI-VXI bus interface software, which is fully VXI*plug&play* compliant, and the NI-VISA API, which is the National Instruments implementation of the VISA I/O software standard on which all VXI*plug&play* software components are based.

How to Use This Documentation Set


Getting Started Manual
Installation and Configuration
MXI-2 Configuration Reference Manual
Change Hardware Settings (Optional)
NI-VXI Online Help
NI-VXI Reference
Use Online Utilities:
NI-VXI and NI-VISA Function Help, Measurement & Automation Explorer, NI Spy, VISAIC
NI-VISA User Manual
NI-VISA Reference

This getting started manual contains an overview of the MXI-2 hardware and the NI-VXI/NI-VISA software, guides you through setting up your kit, and helps you get started with application development. You can also use this manual as a reference for the hardware and software default settings and to find the answers for commonly asked questions.

The *MXI-2 Configuration Reference Manual* contains information on configuring, installing, and cabling your MXI-2 hardware. You will need to use this manual in conjunction with the getting started manual.

When you have successfully set up your system, you can begin to develop applications in NI-VXI and/or NI-VISA. The NI-VXI online help presents the concepts of VXI and detailed explanations of the NI-VXI functions. Study the descriptions of each function in the online help to fully understand the purpose and syntax of each function. You can also access the NI-VXI online help in the `NIVXI` folder.

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 `c:\VXIpnp\`*`os`*`\NIvisa\manuals` directory (where *`os`* corresponds to the version of Windows you are using) under the name `NIVISAUM.pdf`. Use the Acrobat Reader program, Version 3 or later, to open this file.

Conventions

The following conventions appear in this manual:

»	The » symbol leads you through nested menu items and dialog box options to a final action. The sequence **File»Page Setup»Options** directs you to pull down the **File** menu, select the **Page Setup** item, and select **Options** from the last dialog box.
✎	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 on 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. This font 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, and code excerpts.

`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*
- *CompactPCI Specification*, Revision 2.0, PCI Industrial Computers Manufacturers Group
- *Multisystem Extension Interface Bus Specification*, Version 2.0, National Instruments Corporation
- *PCI Local Bus Specification*, Revision 2.1, PCI Special Interest Group
- *PXI Specification*, Revision 1.0, National Instruments Corporation
- *VME-MXI-2 User Manual*, National Instruments Corporation
- *VXI-MXI-2 User Manual*, National Instruments Corporation
- VXI-6, *VXIbus Mainframe Extender Specification*, Rev. 1.0, VXIbus Consortium

1

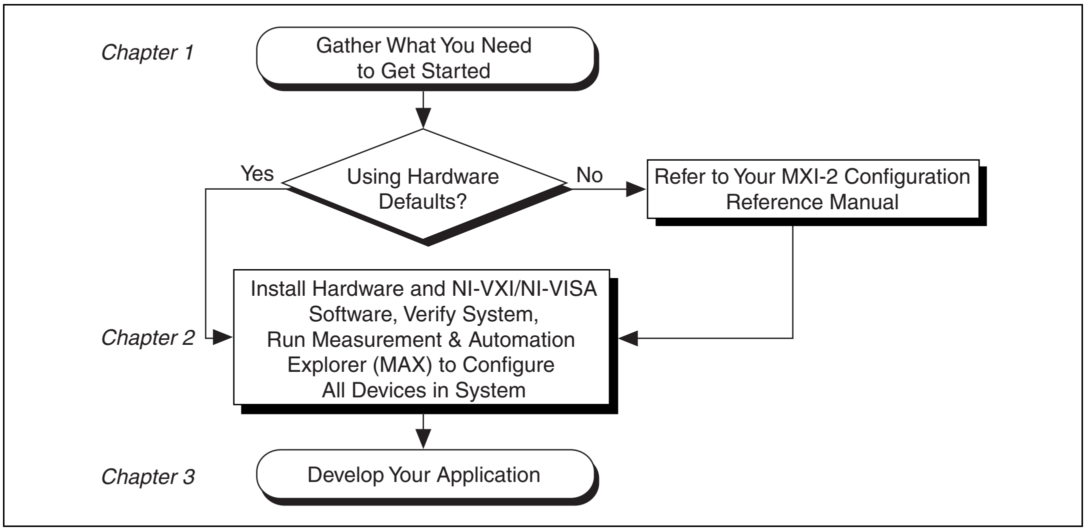
Introduction

This chapter describes your MXI-2 interface kit, lists what you need to get started, and includes a brief description of the hardware and software.

This manual uses 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 *NI-VXI/NI-VISA* when software information applies to both NI-VXI and NI-VISA, and the term *Windows 2000/NT/Me/9x* when information applies to all operating systems.

The following flowchart shows where to turn for more details on configuring and using the hardware and software.

How to Use This Manual


Chapter 1
Gather What You Need to Get Started
Yes
Using Hardware Defaults?
No
Refer to Your MXI-2 Configuration Reference Manual
Chapter 2
Install Hardware and NI-VXI/NI-VISA Software, Verify System, Run Measurement & Automation Explorer (MAX) to Configure All Devices in System
Chapter 3
Develop Your Application

Notice that for hardware configuration, you should refer to the *MXI-2 Configuration Reference Manual* for a full description of how to configure and install the MXI-2 boards and an overview of MXI-2 itself. Then return to this manual for further information.

What You Need to Get Started

You need the following items to set up your VXI or VME system:

❑ A PCI-based computer or PXI/CompactPCI chassis running Windows 2000/NT/Me/9*x*

❑ VXIbus or VMEbus mainframe

❑ PCI-MXI-2 or PXI-8320 interface board

❑ VXI-MXI-2, VXI-MXI-2/B, or VME-MXI-2 interface module

❑ MXI-2 cable

❑ National Instruments software media

PCI-Based MXI-2 Interface Kit Overview

The interface kits described in this manual link a PCI-based computer or a PXI or CompactPCI chassis directly to the VXIbus or VMEbus using the high-speed Multisystem eXtension Interface bus (MXI-2). The MXI-2 kits, which include the NI-VXI/NI-VISA software for Windows 2000/NT/Me/9*x*, are as follows:

- VXI-PCI8000, containing a PCI-MXI-2 and a C-size or B-size VXI-MXI-2
- VXI-PXI8000, containing a PXI-8320 and a C-size or B-size VXI-MXI-2
- VME-PCI8000, containing a PCI-MXI-2 and a VME-MXI-2
- VME-PXI8000, containing a PXI-8320 and a VME-MXI-2

A PCI-based computer equipped with a VXI-PCI8000 interface, or a PXI/CompactPCI chassis equipped with a VXI-PXI8000 interface, can function as a VXI Commander and Resource Manager. A PCI-based computer equipped with a VME-PCI8000 interface, or a PXI/CompactPCI chassis equipped with a VME-PXI8000, can function as a VMEbus master and/or slave device. The MXI-2 interface kit makes your computer or chassis behave as if it were plugged directly into the VXI/VME backplane as an embedded CPU VXI/VME module.

The CD included with the kits contains software for *x*86/Pentium-based computers.

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. The PXI-8320 is a 3U-size, PXI/CompactPCI-compatible circuit board that plugs into one of the peripheral slots in your PXI/CompactPCI chassis. Both boards link your computer directly to the MXIbus and vice versa.

Because the PCI-MXI-2 and PXI-8320 use the same communication register set that other VXIbus message-based devices use, other MXIbus devices view your board as a VXIbus device. The PCI-MXI-2 and PXI-8320 can also function as the MXIbus System Controller and can terminate the MXIbus signals directly. In addition, you can have up to 16 MB of onboard DRAM on your board 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. Optionally, you can use the VXI-MXI-2/B, which can reside in any B-size VXIbus slot.

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 or PXI-8320 board, the VXI-MXI-2 links the PCI bus or the PXI/CompactPCI bus to the VXIbus. The VXI-MXI-2 can automatically determine whether it is 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 or PXI-8320 board, it links the PCI bus or the PXI/CompactPCI 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/VME-MXI-2 automatically terminates the MXIbus if installed as the first or last device in the MXIbus. If installed in the middle of the MXIbus, the VXI/VME-MXI-2 automatically disables MXIbus termination. In addition, you can have up to 64 MB of onboard DRAM on the VXI/VME-MXI-2 module that either can be shared with the VXI/VMEbus and MXIbus or used as a dedicated data buffer.

VME Users

When used with a VXI-MXI-2, VXI Resource Manager (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 or PXI-8320 to access devices in the VME chassis. Resman does not configure VME devices. The VME specification does not include the initialization and configuration procedures that the VXI specification requires.

If you want to include VME devices in your system, you can enter information about your VME devices into the Measurement & Automation Explorer (MAX) utility using the **Add Device Wizard**. Resman can then properly configure the various device-specific VME address spaces and VME interrupt lines. Notice that the VME-MXI-2 itself does conform to the VXIbus register set and is therefore configured automatically by Resman.

For each address space in which your device has memory, you must create a separate pseudo-device entry with a logical address between 256 and 511. For example, a VME device with memory in both A24 and A32 spaces would require two entries. You can also specify which interrupt level(s) the device uses. Interrupt levels cannot be shared by VXI and VME devices. You can then access the device from NI-VXI/NI-VISA just as you would a VXI device, by specifying the address space and the offset from the base at which you have configured it. NI-VISA support for VME devices includes the register access operations (both high-level and low-level) and the block move operations, as well as the ability to receive interrupts.

Software Description

The NI-VXI/NI-VISA bus interface software includes a Resource Manager, an interactive configuration and troubleshooting program, a comprehensive library of software routines for VXI/VME programming, a logging utility you can use for debugging, and graphical interactive control programs for interacting with VISA. You can use this software to seamlessly program multiple-mainframe configurations and have software compatibility across a variety of 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.

Measurement & Automation Explorer (MAX) helps merge NI-VXI and NI-VISA closer together. You use MAX to view your entire T&M system and configure various components, whether they are VXI, GPIB, or Serial devices. This utility not only takes the place of the NI-VXI T&M Explorer utility, but it also adopts the functionality of the NI-DAQ Configuration utility so you can configure National Instruments VXI-DAQ cards. You can also add VME devices to your system easily with MAX and view them on a screen display along with the rest of your system.

MAX also features various options for running Resman. You can still execute Resman independently to configure your instruments after a power cycle. But you can also perform resource manager operations directly from MAX or configure it to run Resman automatically at startup.

The NI Spy utility tracks the calls your application makes to National Instruments T&M drivers, including NI-VXI, NI-VISA, and NI-488.2. NI Spy helps you debug your application by clearly highlighting the functions that return errors. You can let NI Spy keep a log of your program's calls to these drivers so that you can check them for errors at your convenience.

Software Configurations

There are two software configurations described in this manual:

- NI-VXI for Windows 2000/NT/ME/9*x*—This is a fully 32-bit native Plug and Play driver for Windows 2000/NT/ME/9*x*. You can run *only* 32-bit applications with this driver.
- NI-VISA for Windows 2000/NT/Me/9*x*—This is a 32-bit driver designed for Windows 2000/NT/Me/9*x*. Note that for VXI support, the NI-VXI driver must be installed. You can use this driver to develop and run 32-bit applications for Windows.

National Instruments Application Software

In addition to the NI-VXI/NI-VISA software, you can use the National Instruments LabVIEW and Measurement Studio application programs and instrument drivers to ease your programming task. These standardized programs match the modular virtual instrument capability of VXI and can reduce your VXI/VME software development time. These programs are fully VXI*plug&play* compliant and feature extensive libraries of VXI instrument drivers written to take full advantage of direct VXI control.

LabVIEW and Measurement Studio include all the tools needed for instrument control, data acquisition, analysis, and presentation.

LabVIEW is an easy-to-use, graphical programming environment you can use to acquire data from thousands of different instruments, including IEEE 488.2 devices, VXI devices, serial devices, PLCs, and plug-in data acquisition boards. After you have acquired raw data, you can convert it into meaningful results using the powerful data analysis routines in LabVIEW. LabVIEW also comes with hundreds of instrument drivers, which dramatically reduce software development time, because you do not have to spend time programming the low-level control of each instrument.

Measurement Studio bundles LabWindows/CVI for C programmers, ComponentWorks for Microsoft Visual C++ programmers, and ComponentWorks++ for Microsoft Visual C++ programmers. Measurement Studio is designed for building measurement and automation applications with the programming environment of your choice:

- LabWindows/CVI is an interactive ANSI C programming environment designed for building virtual instrument applications. LabWindows/CVI delivers a drag-and-drop editor for building user interfaces, a complete ANSI C environment for building your test program logic, and a collection of automated code generation tools, as well as utilities for building automated test systems, monitoring applications, or laboratory experiments.
- ComponentWorks for Visual Basic is a collection of ActiveX controls designed for building virtual instrumentation systems. Based on ActiveX technology, ComponentWorks controls are configured through simple property pages. You can use the ComponentWorks GPIB, Serial, and VISA I/O controls and property pages to set up communication with your instruments.
- ComponentWorks++ for Visual C++ takes advantage of integrated C++ libraries and ActiveX to help you build measurement and automation applications. With the ComponentWorks++ instrument classes, you can use the IEEE 488.2 library and VISA, an industry-standard I/O library, to communicate with GPIB, VXI, or Serial devices using the same set of components.

If you want to use either of these application programs, install them during the NI-VXI/NI-VISA software installation. Both LabVIEW and Measurement Studio integrate the VXI and VISA libraries that are required to support your PCI-based MXI-2 products. You also get hundreds of complete instrument drivers, which are modular, source-code programs that handle the communication with your instrument to speed your application development.

2 Setup

This chapter contains the instructions to set up your VXI or VME system using the MXI-2 hardware and NI-VXI/NI-VISA software.

Configuring the Hardware

This section contains basic information about configuring your MXI-2 hardware.

Note Install the NI-VXI software for Windows first, followed by the hardware.

The default settings for your MXI-2 hardware are acceptable for most typical applications. Refer to Appendix A, *Default Settings*, for a complete listing of the hardware and software default settings.

The *MXI-2 Configuration Reference Manual* fully describes the configuration and installation of each MXI-2 board discussed in this getting started manual. Refer to the *MXI-2 Configuration Reference Manual* if you want to try a different hardware configuration or would like more information on a particular setting.

Use the MAX utility in NI-VXI/NI-VISA to change PCI-MXI-2 or PXI-8320 configuration settings. For information on the software, including optional settings, use MAX and its online help. Access MAX from the MAX icon on the desktop. To access the MAX online help, open the **Help** menu and select **Help Topics**.

Installing the Hardware

This section summarizes how to install your MXI-2 hardware.

Installing Your PCI-Based MXI-2 Interface

You received either a PCI-MXI-2 or a PXI-8320 in your kit.

Caution To guard against electrostatic discharge, touch the antistatic plastic package to a metal part of your computer or chassis before removing the board from the package. Your computer or chassis should be plugged in but powered off.

Install the PCI-MXI-2 or PXI-8320 in an available peripheral slot in your PCI-based computer or PXI/CompactPCI chassis. For more information, refer to the PCI-MXI-2 or PXI-8320 chapter in the *MXI-2 Configuration Reference Manual*.

Installing Your Mainframe Extender

You also received either a VXI-MXI-2, VXI-MXI-2/B, or VME-MXI-2 in your kit.

Caution To guard against electrostatic discharge, touch the antistatic plastic package to a metal part of your chassis before removing the module from the package. Your VXI or VME chassis should be plugged in but powered off.

Install the VXI-MXI-2 or VXI-MXI-2/B 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 in the same chassis are not configured as system controller.

Caution Having more than one device configured as system controller can 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 or PXI-8320 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 or PXI-8320 and the VXI-MXI-2, respectively.

For more information, refer to the VXI-MXI-2, VXI-MXI-2/B, or VME-MXI-2 chapter in the *MXI-2 Configuration Reference Manual*.

Connecting the MXI-2 Cable Properly

By default, the PCI-MXI-2 or PXI-8320 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 or PXI-8320. The cable must be connected in this manner so that the MXI board can correctly detect whether it should be the system controller on the MXIbus. Attach the other end of the cable to the VXI/VME-MXI-2.

Installing the Software for Windows 2000/NT/Me/9*x*

Use the Setup program that came with your NI-VXI/NI-VISA software to install the entire software package or a software update, or to reinstall software in the event that your files were accidentally erased. The Setup program works in essentially the same way for Windows 2000/NT/Me/9*x*. You can install NI-VXI with or without NI-VISA.

Some of the utilities rely on the LabWindows/CVI Run-Time Engine. This software is installed, if necessary, during the NI-VXI/NI-VISA installation.

Depending on the type of installation you choose, you may need up to 40 MB of free space available to accommodate the NI-VXI/NI-VISA software. If you choose the **Custom** installation method, Setup displays the amount of memory required for the options you select.

To be compliant with VXI*plug&play* specifications, a VXI controller must provide the VISA I/O driver library standardized by VXI*plug&play*. VISA ensures that your controller can run all VXI*plug&play*-compatible software now and in the future.

The NI-VISA software in this kit is compatible with the WIN95/GWIN95 and WINNT/GWINNT frameworks. With NI-VISA installed on your computer, you can run any VXI*plug&play* software compatible with these frameworks. This includes instrument drivers and executable soft front panel software included with VXI*plug&play*-compatible instruments from a variety of vendors.

Preparing Your System (PCI-MXI-2 Only)

If you are currently using either the NI-VXI software for DOS/Windows 3.*x* or the NI-VXI Windows 95 Upgrade, you must remove it before installing the new software. You cannot have both the 16-bit and the 32-bit versions of NI-VXI installed at the same time.

Note If you plan to run both 16-bit and 32-bit applications, you should use the NI-VXI Windows 95 Upgrade version instead.

If you have been using your PCI-MXI-2 under Windows 95 with either the NI-VXI software for DOS/Windows 3.*x* or the NI-VXI Windows 95 Upgrade, you need to remove the Plug and Play information from the Windows 9*x* Device Manager before installing the new NI-VXI software.

Follow these steps to remove the PCI-MXI-2 information.

1. Double-click on the **System** icon under **Start»Settings»Control Panel**.
2. Select the **Device Manager** tab from the **System Properties** dialog that appears.
3. Click on the **View devices by type** button and double-click on the **Other Devices** icon.
4. Select the PCI-MXI-2 from the list of devices under **Other Devices**. It will appear under the name **PCI Card** and will have a circled exclamation point through the **?** (question mark) icon.
5. Click on the **Remove** button.
6. Click **OK** to exit the Device Manager after removing the device information.

Installing the Software

This section describes how to install the 32-bit NI-VXI/NI-VISA software. The Setup program works in essentially the same way for Windows 2000/NT/Me/9*x*. Please carefully read these directions along with any messages on the screen before making your selections.

You can quit the Setup program at any time by pressing the **Cancel** button.

Setup is an interactive, self-guiding program that installs the NI-VXI/NI-VISA software and configures your system to use the software with the PCI-MXI-2 or PXI-8320. Follow these steps to perform the installation:

1. Insert the CD-ROM labeled *NI-VXI/NI-VISA for PCI-MXI-2 for Windows xx* (where *xx* is your version of Windows).
2. Select **Run...** from the **Start** menu and enter the following text, where *X* is your CD-ROM (usually D or E)

   ```
   X:\setup.exe
   ```

   and press <Enter>.
3. Click on the **Next** button at the **Welcome** screen to start the installation and accept the license agreement.

Note If Setup detects a 16-bit (DOS or Windows 3.*x*) version of the NI-VXI software, it prompts you to remove it. Setup will quit so you can uninstall the old software. If you have a previous 32-bit (Windows 9*x* or Windows NT) version of the NI-VXI software installed, Setup installs the new version over the previous version.

Caution If you want to keep the manufacturer/model name tables or the VME device configuration from a previous installation, be sure to back them up before starting Setup.

4. Select the type of installation from the **Choose Setup** screen.
   - **Express** setup is the fastest and simplest installation option. This option installs all the NI-VXI/NI-VISA software in default directories without prompting you to make any further choices.
   - **Typical** setup prompts you to make high-level choices of which driver(s) to install and the destination directories.
   - **Custom** setup gives you complete control over which files and utilities you want installed on your system. This option is recommended for advanced users.
5. The **Express** setup completes without further questions. Follow the prompts if you select either the **Typical** or the **Custom** setup options. The final prompt displays the choices you made concerning applications, support, and destination directories. Click on the **Next** button to begin the installation.
6. Setup now copies the necessary files to your hard drive and creates program icons.

Completing the Software Installation

1. Review the information in any `README` files that Setup prompts you to read.
2. When the installation process completes, you must reboot your computer for the changes to take effect. The NI-VXI driver is loaded at this time.
3. If you backed up the manufacturer and model name files, restore them to the `TBL` subdirectory of your NI-VXI directory before running MAX.
4. After you install the NI-VXI/NI-VISA software, run MAX. It will prompt you to run Resman. You must run Resman every time the chassis power is cycled so that your application can access devices in the VXI/VME chassis. You can also configure MAX to run Resman automatically at every computer startup.
5. After you run Resman, you are ready to use MAX to interactively configure the National Instruments hardware in your system. Use the right-click help for information about the various configuration options.

Verifying Your System Configuration

After you finish configuring the system through MAX, verify the system configuration through one of the interactive control utilities. Use VIC under NI-VXI or VISAIC under NI-VISA.

For more details about the utilities in NI-VXI/NI-VISA, refer to Chapter 3, *Developing Your Application*.

3 Developing Your Application

This chapter discusses the software utilities you can use to start developing applications that use the NI-VXI/NI-VISA driver.

After installing the driver software, you can begin to develop your VXI or VISA application software. Be sure to check the `README.txt` file for the latest application development notes and changes.

Your software includes several utilities to assist you in your system development. These include MAX, Resman, NI Spy, VISAIC, and VIC. You can also access several examples to learn how to use NI-VISA or NI-VXI for certain tasks. Each component assists you with one of four steps of development: configuration, device interaction, programming, and debugging.

After installation, you can access these utilities through the Windows **Start** menu. Open the **National Instruments** or **VXIpnp** program group, then open either the **VXI** or **VXIpnp** (if you are using the VISA 2.5 driver) subgroup to access the utilities you want to use.

Configuration

The configuration utilities in your kit are MAX and Resman. Resman performs VXI Resource Manager functions as described in the VXIbus specification. Its most important functions include configuring all devices on the VXI backplane for operation and allocating memory for devices that request it.

Note Power cycling resets all devices, so you need to run Resman to reconfigure them every time chassis power is cycled.

MAX presents a graphical display of your entire test and measurement system to help you configure various components. When you launch MAX, you see all your VXI, GPIB, GPIB-VXI, and serial devices on the screen. You can view the properties (such as logical address, address space used, primary address, and so on) of each device by right-clicking on the device in the tree. When you view the properties of most National Instruments devices, you can configure the hardware settings directly in the property pages.

MAX and Resman are designed to work together. You can run the Resource Manager through MAX by either clicking on the **Run VXI Resource Manager** button on the toolbar or selecting **NI-VXI»VXI Resource Manager** from the **Tools** menu. From the **User Preferences** dialog in the **Tools** menu, you can also configure MAX to run Resman automatically when the computer boots up. Resman reports all errors it finds in your system to MAX. When you view your system through MAX, you can easily spot any errors in your system that Resman found.

You can find more information about MAX by using its online help. From MAX, open the **Help** menu and select **Help Topics**.

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 MAX utility. This utility includes a configuration view, 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 or VISAIC utility (VIC for NI-VXI or VISAIC for NI-VISA). You can use these utilities to interactively control your VXI/VME devices without using a conventional programming language, LabVIEW, or Measurement Studio.

Note The VIC utility is not part of the basic NI-VXI installation. However, if you want to use VIC, run the NI-VXI installer and select a "custom" installation instead of "typical." In the **Select Features** dialog box, enable the **NI-VXI API Development** option and continue as prompted. After you reboot, VIC will be ready for use.

Note You can launch VIC or VISAIC from the **Tools** menu in MAX.

Try the following in VIC. In the **Command** entry field, type:

```
help vxiinreg
```

This help file shows you the syntax for this command, which reads VXI device configuration registers. The first argument is a logical address, and the second is the offset of the VXI device configuration register to be read.

Type:

```
vxiinreg 1,0
```

The **History** window shows the result of the command execution, such as:

```
Return Status (0): Success.
Value = 0x4ff6
```

If the value ends with `ff6`, you have successfully read the National Instruments manufacturer ID from the VXI/VME-MXI-2 ID register.

You may now want to read the configuration registers from other VXI devices in your system using the command `vxiinreg`. This command accesses only the upper 16 KB of A16 space. Try reading a register from each device listed in the MAX configuration view. In this way, you can verify that your PCI-MXI-2 or PXI-8320 can access each device in your VXI system successfully.

You can also access VXI and VME devices configured in A16, A24, and A32 address space by using the `vxiin` or `vxiout` commands. For more information regarding VIC operation and commands, refer to the VIC online help.

Alternately, you can use VISAIC to interact with your devices. VISAIC lists the available devices, similar to what MAX displays. By double-clicking on a given device, you can open a VISA session and access the device through it. For more information regarding VISAIC, use the right-click help available from all panels.

Programming with VXI

National Instruments provides two different programming interfaces for accessing your instruments: NI-VISA and NI-VXI. NI-VISA is the National Instruments implementation of the VISA API as defined by the VXI*plug&play* standard. It is very useful in situations where you have different types of instruments in your system (such as VXI, GPIB, and serial devices) because the NI-VISA functions have the same interface.

NI-VXI is the National Instruments proprietary interface for programming VXI instruments. Both NI-VXI and NI-VISA grant you register-level access of VXI instruments as well as messaging capability to message-based devices. With either interface you can service asynchronous events, such as triggers and signals, and also assert them.

The best way to learn how to program with NI-VXI or NI-VISA is by reviewing the example programs included in your software. In the examples directory are examples for many different types of applications. If you are just getting started, you should first learn how to access registers with high-level calls and send messages with word serial functions. The NI-VISA examples of these tasks are called `VISAhigh.c` and `VISAws.c`. The NI-VXI examples are called `VXIhigh.c` and `VXIws.c`. You should use the other examples as you try more advanced techniques. Consult the *NI-VISA User Manual* or NI-VXI online help for additional information on these topics.

Note The *NI-VISA User Manual* is in the `VXIpnp\`*`os`*`\NIvisa\manuals` directory, where *`os`* would be your version of Windows. Use the Acrobat Reader program to open and navigate this manual.

Table 3-1 summarizes the topics addressed by the example programs.

Table 3-1. NI-VXI/NI-VISA Examples

Coverage	NI-VISA Example	NI-VXI Example
Message-Based Access	`VISAws.c`	`VXIws.c`
High-Level Register Access	`VISAhigh.c`	`VXIhigh.c`
Low-Level Register Access	`VISAlow.c`	`VXIlow.c`
Sharing Memory	`VISAmem.c`	`VXImem.c`
Interrupt Handling	`VISAint.c`	`VXIint.c`
Trigger Handling	`VISAtrig.c`	`VXItrig.c`

Note MAX includes special settings that you must use for low-level functions and memory sharing. Consult the MAX online help for information on setting these up.

Compiler Symbols for NI-VXI

You may need to define some symbols so that the NI-VXI library can work properly with your program.

Note Skip this section if you are programming with NI-VISA only. NI-VISA neither requires nor uses these symbols.

You can define the symbols using `#define` statements in the source code or you can use either the `/D` or `-D` option in your compiler (both the Microsoft and Borland compilers support the `/D` and `-D` options). If you use `#define` statements, you must define the symbols before including the NI-VXI header file `nivxi.h`. If you use the makefiles to compile the sample program, the makefile already defines the necessary symbols.

The following symbol is usually required. You must define it when using the Microsoft C or Borland C compiler.

- `VXINT` designates the application as a Windows 2000/NT/Me/9*x* application.

Note LabWindows/CVI automatically defines the correct symbol. You do not need to define `VXINT` when using LabWindows/CVI.

The `BINARY_COMPATIBLE` is optional. This symbol makes the application binary compatible with embedded VXI controllers, such as the National Instruments VXIpc series of embedded controllers. This option may cause a slight performance degradation when you use low-level VXIbus access functions.

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

```
#define VXINT

#define BINARY_COMPATIBLE

.

.

.

#include <nivxi.h>
```

If you define these symbols using the `/D` or `-D` compiler options, you should specify the following when invoking the compiler.

For the Microsoft C compiler:

```
/DVXINT /DBINARY_COMPATIBLE
```

For the Borland C compiler:

```
-DVXINT; BINARY_COMPATIBLE;
```

You also need to link in the appropriate import library for your code. If you are using a Microsoft C compiler, use the `nivxint.lib` in the `nivxi\win32\msc\` directory. If you are using a Borland C compiler, use the `nivxint.lib` in the `nivxi\win32\borlandc\` directory.

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.

Window Mapping

The MITE architecture allows a lot of flexibility in low-level mapping of VXI address spaces. In particular, the CPU interface of the MITE has windows that can be dynamically resized and relocated from CPU space to VXI space. The low-level functions have extensions that reflect this feature. Refer to the NI-VXI online help for information about these functions in NI-VXI. The NI-VISA online help and the *NI-VISA Programmer Reference Manual* cover this information for NI-VISA applications. As mentioned earlier, you can use the Acrobat Reader 3.0 to view and navigate through this manual.

The functions `MapVXIAddress()` and `viMapAddress()` check whether a window that can be shared already maps to the desired address space and location. If so, they return a pointer to that window. If the desired space is not already mapped, they set up a new MITE window to the VXI address and return a pointer to the new window.

The `MapVXIAddressSize()` function is the standard mechanism for specifying how large a window the driver should map on a call to `MapVXIAddress()`. The default size of a mapped window when using NI-VXI is 64 KB. In VISA, you specify the window size directly in `viMapAddress()`.

The success of this allocation depends on the availability of three factors:

- Address space in the User Window
- Number of MITE windows
- Memory for allocating data structures for the map

Address Space

The PCI-MXI-2 can decode any 32-bit address on the PCI bus as a VXI cycle, giving 4 GB of addressability, which can be used for windows on the PCI-MXI-2. The operating system or computer architecture may limit which addresses can be assigned to the PCI-MXI-2.

To change the address space, edit the **User window size** field in the **PCI** tab of the PCI-MXI-2 **Hardware Configuration** settings in MAX. This setting limits the total amount of memory you can map with `MapVXIAddress()` or `viMapAddress()`. If the User Window is disabled, the `MapVXIAddress()` function returns `NO_HARDWARE_SUPPORT (-1)`.

The NI-VXI online help implies that the error code `MAP_TIMEOUT (-8)` is returned when the window is in use. Because the MITE-based products have multiple hardware windows of variable size, the meaning of this error has been modified. `MapVXIAddress()` now returns the error code `MAP_TIMEOUT (-8)` whenever there are not enough resources to map the window.

For example, if you use `MapVXIAddressSize()` and `MapVXIAddress()` to request a 1 MB window to A32 space, and you request a User Window in MAX of only 64 KB, `MapVXIAddress()` returns the error code `MAP_TIMEOUT` because there are not enough resources to complete the request.

Number of MITE Windows

The MITE has eight CPU windows. NI-VXI uses four of these windows, leaving four for user applications.

Memory for Allocating Data Structures

You need to have sufficient memory available to set up the necessary page tables. If you request a very large window—hundreds of megabytes, for example—you may run out of memory.

MITE DMA

The MITE has two DMA channels to improve the throughput of block transfers to and from the VXI system. The DMA channels can use various high-speed bus protocols, such as the following:

- MXI block
- MXI synchronous

- Burst mode (on the PCI bus)
- VME64 (on the VXI bus)

The DMA channels can transfer data between a VXI device and local memory, or between VXI devices. The DMA channel can handle contiguous or noncontiguous local memory. If it is handling noncontiguous memory, it can perform scatter-gather operations on the noncontiguous memory.

The `VXImove()` and `viMove XX()` functions automatically use appropriate bus protocols and transfer types to efficiently perform the data transfer specified in the function. You can also use configuration options in MAX to instruct the NI-VXI/VISA software to use DMA channels for particular types of operations and to designate what protocols the channel should use. In addition, you can programmatically control which protocols to use in NI-VXI. See the NI-VXI online help for complete descriptions of `VXImove()` and other high-level functions. Notice that previously written NI-VXI and NI-VISA code uses the DMA capabilities of the MITE without modification.

To take full advantage of the throughput of the DMA channels, you should perform 32-bit transfers where both the source and the destination are longword aligned. If you need to transfer character data between devices of different byte orders—for example, between a big-endian device and an Intel 80*x*86-based Windows NT PC—transfer the data as longwords but adjust the byte-ordering parameters in `VXImove()` to get the correct data in the most efficient manner.

NI-VXI examples:

```
/* Transferring 32-bit data to a big-endian A32 device */
VXImove(0x0, userBuffer, 0x3, deviceOffset,
numDataPoints, 4);
```

```
/* Transferring 8-bit data to a big-endian A32 device */
VXImove(0x80, userBuffer, 0x3, deviceOffset,
numDataPoints / 4, 4);
```

Shared Memory

In the **Hardware Configuration** settings of the PCI-MXI-2 in MAX, you can share memory on your computer or from DRAM added to the PCI-MXI-2. Right click on any of the settings or consult the online help in MAX for more information. You can access shared memory on your computer using `VXImemAlloc()` in NI-VXI and `viMemAlloc()` in VISA.

Remote Controllers

Remote controllers, when configured to detect asynchronous events such as a VXI interrupt or VXI trigger, need to inform the local controller that such an asynchronous event has occurred. The remote controllers report these events back to the local controller via a VXI IRQ line. This IRQ line is called the *system IRQ line*. You can use MAX to select which VXI interrupt line the remote controller uses to report remote events to the local controller. You need to map the system IRQ line back to the local controller to receive remote controller interrupts. This mapping is performed automatically by the Resource Manager in the parent-side VXI-MXI-2 controllers, but not in other mainframe extenders. You can map interrupts through MAX, or with the `MapVXIint()` function, which is described in the NI-VXI online help.

The system IRQ line is treated differently than other IRQ lines used by NI-VXI:

- The system IRQ line is always acknowledged by the Resource Manager (Logical Address 0).
- The system IRQ line cannot be disabled on the Resource Manager. Calling `DisableVXIint()` on the system IRQ line does *not* disable it.
- Devices other than remote controllers can also interrupt on the system IRQ line, provided that the device at Logical Address 0 is the handler for the interrupt.
- Routing the system IRQ line to the signal queue is not recommended. Because the system IRQ line cannot be disabled, this routing could lose interrupts.

Passing the value `-1` as the logical address of a controller in NI-VXI causes NI-VXI to select the first *remote* controller in your system. Notice that on embedded controllers such as the VXIpc-850, `-1` refers to the *local* controller. This is to maintain compatibility with older systems where the external controller needed an extender to assert and receive interrupts.

Enhancements to the NI-VXI Software

The following sections describe the additional options beyond what is documented in the NI-VXI online help.

Compatibility

NI-VXI applications that follow the guidelines documented in the NI-VXI online help will work with NI-VXI for the PCI-MXI-2.

System Configuration Functions

The `InitVXIlibrary()` function has a new return value of `INIT_RET_OK_RMERROR (2)`. If this value is returned, it means the NI-VXI library successfully initialized, but the Resource Manager has not been run successfully. Always run the Resource Manager before using the NI-VXI library.

Low-Level VXIbus Access Functions

Do not make any assumptions about the size and features of a window returned from `MapVXIAddress()`. You should use `GetWindowRange()` to determine the size of a window. The 32-bit value returned from `GetContext()` and passed to `SetContext()` has a new format. Applications that set the context bits directly for use in `SetContext()` may not be compatible with the new format for context. Because the MITE allows more flexible window mapping, extra bits have been added to this field to reflect these new features. Do not manipulate the context bits directly.

High-Level VXIbus Access Functions

For best performance, keep the following in mind when using `VXImove()`:

- Make sure your buffers are 32-bit aligned.
- Transfer 32-bit data whenever possible.
- Using VXI block access privileges significantly improves performance to devices that are capable of accepting block transfers.
- `VXImove()` must lock the user buffer in memory on virtual memory systems, so locking the buffer yourself optimizes `VXImove()`.
- Because `VXImove()` must build a scatter-gather list for the user buffer on paged memory systems, using a contiguous buffer optimizes `VXImove()`.

`VXImemAlloc()` returns 32-bit aligned, page-locked, contiguous buffers, which work efficiently with `VXImove()`, but only if the function returns `MEM_OK (0)`. A status of `MEM_OK_USE_MEMCOPY (1)` means this buffer cannot be used directly with `VXImove()`.

`VXImove()` can also move blocks of data to and from a single VXI address. This is commonly referred to as *FIFO mode*. For more information, refer to the NI-VXI online help.

Local Resource Access Functions

`VXImemAlloc()` does not allocate onboard RAM on the PCI-MXI-2; it only allocates system RAM on the motherboard. If you want to access onboard RAM on the PCI-MXI-2, access it as if it were VXI memory—that is, by using high-level or low-level VXIbus access functions. You can use `GetDevInfo()` on the PCI-MXI-2 device to determine the VXI address space and VXI address of this onboard RAM.

Debugging

NI Spy and VISAIC are useful utilities that can aid in identifying the causes of problems in your application.

NI Spy tracks the calls your application makes to National Instruments T&M drivers including NI-VXI, NI-VISA, and NI-488.2. NI-488.2 users may notice that NI Spy is very similar to GPIB Spy. It highlights functions that return errors, so you can quickly spot which functions failed during your development. NI Spy can log the calls your program makes to these drivers so you can check them for errors at your convenience.

You can also control your instruments interactively using VISAIC. You can use VISAIC to control and communicate with your instruments with NI-VISA without having to write a program. This utility is an excellent platform for quickly testing instruments and learning how to communicate with them.

Refer to the online help for instructions on how to use VISAIC and to learn about their features. In VISAIC, you can right-click to reach *What's This* help and function help.

A

Default Settings

This section summarizes the hardware and software default settings for the hardware and software in your 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 documentation set. Use the *MXI-2 Configuration Reference Manual* for your hardware reference and the MAX online help for your software reference.

PCI-MXI-2/PXI-8320

This section summarizes the hardware and software default settings for the PCI-MXI-2 and PXI-8320.

Hardware Settings

Table A-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 A-2. PXI-8320 Hardware Default Settings

Hardware Component	Default Setting
U6 Switch 1 (FOV)	OFF: PXI-8320 boots off the user-configured half of the EEPROM
U6 Switch 2 (TST)	OFF: Factory configuration of the EEPROM is protected
U6 Switch 3 (POS)	OFF: *Do not alter this setting*
U6 Switch 4 (CT)	ON: *Do not alter this setting*
DRAM SODIMM installed	Per customer order

MAX Settings

Table A-3. Device Tab Default Settings

Editor Field	Default Setting
Logical address	0
Device class	Message-based
Size of Servant area	0
Number of handlers	1
Number of interrupters	0

Table A-4. Shared Memory Tab Default Settings

Editor Field	Default Setting
Memory sharing	Don't share memory
Shared RAM size	0 KB
Reserved physical memory	0 KB
Lower half window byte swapping	Disabled
Lower half window memory selection	System memory
Upper half window byte swapping	Disabled
Upper half window memory selection	System memory
Map upper and lower halves at same PCI address	Disabled

Table A-5. MXI-2 Bus Tab Default Settings

Editor Field	Default Setting
Bus timeout	1 ms
System controller	Auto-detect
MXI-2 auto retry	Enabled
A24/A32 write posting	Disabled
VXImove uses Synchronous MXI	Enabled
MXI transfer limit	Unlimited
MXI CLK10 signal	Receive

Table A-6. PCI Tab Default Settings

Editor Field	Default Setting
Low-level register access API support	Enabled
User window size	64 KB
Expansion ROM	Enabled

VXI/VME-MXI-2

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

Hardware Settings

Table A-7. 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

Table A-7. VXI-MXI-2 Hardware Default Settings (Continued)

Hardware Component	Default Setting
MXIbus termination (U35 switches 1 and 2)	Automatic MXIbus termination: switch 2 set to NO; 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 A-8. VXI-MXI-2/B Hardware Default Settings

Hardware Component	Default Setting
Logical address (U20)	1
VXIbus Slot 0/Non-Slot 0 (W3)	Automatic detection
VXIbus local bus (W2)	Single VXI-MXI-2/B in frame
VXIbus CLK10 routing (W1)	From onboard oscillator
External trigger termination (S5)	OFF: unterminated
SMB CLK10 direction (S7)	OUT: drive CLK10 signal
SMB CLK10 termination (S6)	Ignored; effective only when S7 is set to IN
Polarity of external SMB CLK10 (S3)	Inverted
MXIbus CLK10 signal (S1)	Receive CLK10 from MXIbus

Table A-8. VXI-MXI-2/B Hardware Default Settings (Continued)

Hardware Component	Default Setting
MXIbus termination (U21 switches 3 and 4)	Automatic MXIbus termination: switch 3 set to OFF; switch 4 ignored
Configuration EEPROM (U21 switches 1 and 2)	User-modifiable; factory settings protected: both switches set to OFF

Table A-9. 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

MAX Settings

Table A-10. Device Tab Default Settings

Editor Field	Default Setting
Logical address	Use DIP switch
Address space	A24 *
Requested memory	16 KB *
A24/A32 write posting	Disabled
A16 write posting	Disabled
Interlocked mode	Disabled
* Assumes no DRAM is installed. If DRAM is installed, the **Address space** should be A32, and **Requested memory** should match the amount of DRAM. If you install the DRAM yourself, you must manually specify these changes.	

Table A-11. VXI/VME Bus Tab Default Settings

Editor Field	Default Setting
Bus timeout value	125 µs
Slot 0 configuration	Auto-detect
Auto retry	Disabled
Transfer limit	256
Arbiter type	Priority
Fair requester	Enabled
Arbiter timeout	Enabled
Request level	3

Table A-12. MXI-2 Bus Tab Default Settings

Editor Field	Default Setting
System controller	Auto-detect
Bus timeout value	1 ms
MXI-2 auto retry	Disabled
MXI transfer limit	Unlimited
MXI fair requester	Disabled
Perform parity checking	Enabled
MXI-2 CLK10 signal direction	Switch determines signal direction (VXI-MXI-2 only)

B
Common Questions

This appendix addresses common questions you may have about using the NI-VXI/NI-VISA software on the PCI-MXI-2 or PXI-8320 platform.

What are some of the differences between the old utilities and the new ones?

The old utility components are as follows:

- VXIinit—This utility initializes your National Instruments controller hardware with settings determined in VXIedit.
- Resman—This utility initializes and configures all the other devices in your VXI system.
- VXIedit—This utility configures your National Instruments hardware.
- VXItedit—This is a console-based version of VXIedit.
- VIC—Use this utility to interactively communicate with VXI devices over the VXIbus using the NI-VXI API.
- VICtext—This is a console-based version of VIC.
- VISAconf—This utility configures settings used by NI-VISA
- VISAIC—Use this utility to interactively communicate with VISA devices (GPIB, VXI, serial) using NI-VISA.
- T&M Explorer—Use this utility to configure, view, and initialize your system.

VXI system integration with the old utilities typically proceeded as follows:

1. Install components and boot the system.
2. Configure your hardware with VXIedit.
3. Reboot and run VXIinit to initialize your National Instruments Hardware.
4. Run Resman to initialize the VXIbus.
5. Optionally run VXIedit to configure any extender devices on the VXIbus.

6. Run VIC to verify device operation.
7. Run VISAIC to verify that you can communicate with your system using VISA.

The new utility components are as follows:

- MAX—Use this utility to configure, view, and initialize your system.
- Resman—You can still use this as before. However, you can perform resource manager operations directly from MAX or configure it to run Resman automatically at startup. See *What about running Resman?* later in this section.
- VISAIC—Use as before.
- NI Spy—Use this utility to debug your NI-VXI or NI-VISA application.

Your setup might now include the following steps:

1. Install components and boot the system.
2. Launch MAX. Execute VXI Resource Manager responsibilities (run Resman by clicking the **Run VXI Resource Manager** button in the MAX toolbar).

What happened to VXIinit?

You no longer need to run VXIinit to initialize settings on your hardware. We now take care of loading hardware settings in the driver, completely eliminating the need for VXIinit.

Where do I find the information that VXIinit used to print?

You can view information about your controller from the properties pages and the hardware configuration pages. For example, you can view logical address and user window size in the configuration view in MAX.

What happened to T&M Explorer?

The functionality of this utility has been integrated into a powerful new utility called MAX. This utility starts with a graphical view of the VISA devices (GPIB, VXI, and Serial) that it finds in your system. Right-click on an individual device in the configuration tree to see its properties. You can further configure National Instruments devices by selecting the **Hardware Configuration** option. This includes National Instruments VXI-DAQ cards, which means you can configure these devices from MAX without running the DAQ Configuration Utility.

What about running Resman?

Resman is the name of the utility that performs the duties of a VXI Resource Manager as discussed in the VXIbus specification. When you set a National Instruments controller to Logical Address 0, you will at some point need to run Resman to configure your VXI instruments. If your controller uses a different (non-zero) logical address and is a message-based device, you need to start Resman before running it on the Logical Address 0 computer.

So when do you need to run Resman?

Run Resman whenever you need to configure your VXI instruments. For example, if you power-cycle your VXI chassis, your instruments will be reset, and you will need to run Resman to configure them. You can get into trouble if you run Resman when your devices are not in a reset state. Therefore, if you need to run Resman after running it once, you should reset all of your VXI instruments.

In our new architecture, you can perform resource manager operations from within MAX. Additionally, you can tell MAX to run Resman when the computer first boots. In this case you may never need to run Resman explicitly again. This is common when you use an embedded PC such as the VXIpc-860. You can configure the computer to run Resman at startup, so when you power the chassis, Resman runs. If you power-cycle the chassis, the PC reboots, forcing Resman to run again.

With the PCI-MXI-2 or PXI-8320, you may need to run the Resman utility if you boot your computer before turning on your VXI chassis or if you power-cycle your VXI chassis while the external PC remains on. In these cases, the instruments would have been reset without the computer rebooting. You will need to run the Resman utility or configure your system in MAX to initialize your VXI system.

What if I have a system that requires the old utilities?

The new installer for NI-VXI/NI-VISA gives you the option of installing the old utilities. Thus, if you have a documented procedure for configuring your system that relies on the old configuration utilities, you can install them on your system. Use the **Custom** installer option to explicitly add the old utilities.

Why do you install some of the old utilities?

The VIC and VISAIC utilities still have a lot of functionality that our new utilities do not include. Therefore, it makes sense to distribute these two utilities as part of a normal installation.

How do I handle VME devices?

Although there is no way to automatically detect VME devices in a system, you can add them easily through the **Add Device Wizard** in MAX. Through this procedure, you can reserve resources for each of your VME devices and configure MAX to show VME devices on the screen with all your other devices.

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

There are several ways to find this information.

- Under Windows 2000/NT/Me/9*x*, you can find version information by right-clicking on any component and selecting the **Properties** option. This displays a property sheet with a version tab. This tab has version information about the product (NI-VXI) and the component (`NIVXINT.DLL`, for example).
- You can find version information about the NI-VXI driver by running the VIC utility program. Type `ver` at the prompt, and the utility displays the versions of VIC and NI-VXI, and the latest PCI-MXI-2 or PXI-8320 board revision that this NI-VXI driver supports.
- You can find version information about the VISA driver through VISAIC by selecting **About...** from the **Help** menu.

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

Run MAX and right-click on the name of the MXI-2 board in the configuration tree. Select **Hardware Configuration**, and the dialog box for the MXI-2 board is displayed. The title bar includes the serial number and hardware revision of the board.

Which NI-VXI utility program must I use to configure the PCI-MXI-2 or PXI-8320?

Use MAX to configure the PCI-MXI-2 or PXI-8320. Access MAX from its desktop icon.

Which NI-VXI utility program must I use to initialize the PCI-MXI-2 or PXI-8320?

Windows 2000/NT/Me/9*x* automatically initializes the board at system startup.

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

Use the Resman program to perform startup Resource Manager operations. You can run Resman by clicking on the **Run VXI Resource Manager** button in the MAX toolbar. Resman uses the settings configured in MAX. It initializes your VXI/VMEbus system and stores the information that it collects in the `RESMAN.TBL` file in the `TBL` subdirectory of the `NIVXI` directory.

Through MAX, you can also configure Resman to run automatically at computer startup.

What can I do to make sure 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 or VISAIC program to interactively issue NI-VXI functions or NI-VISA operations, respectively. 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, but you can use `VXIin()` and `VXIout()` for both VXI and VME.

Finally, if you are using LabVIEW or Measurement Studio 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 should I do if I get a `Configuration EEPROM is Invalid` message?

There are several reasons why you might get the `Configuration EEPROM is Invalid` message. If you turned off the computer while the configuration update process was still in progress, the board functions normally except when running MAX. To correct these problems, reboot the computer with the Factory Override (FOV) switch set (as described in Appendix B, *EEPROM Configuration,* in the *MXI-2 Configuration Reference Manual)* and update the configuration, or load the configuration from file.

Two other reasons you might receive this error message are that the board might have an incorrect base address assigned for the driver window, or there may be a conflict with another adapter or memory management software.

What should I do if Resman hangs?

1. Ensure that the MXI-2 cable is plugged in and that the end labeled *Connect this end to the device closest to the MXIbus System Controller* is connected to the MXIbus system controller (by default the PCI-MXI-2 or PXI-8320). Because the MXI-2 cables are polarized, it matters which end is connected to which device.
2. Check for bent or broken pins on the MXI-2 connectors.
3. If you are using a VME-MXI-2 in the first slot of a VMEbus chassis, the chassis may be causing problems with the First Slot Detection circuit on the VME-MXI-2. Use MAX to change the **Slot 0 Configuration** setting of the VME-MXI-2 to **Slot 0** rather than **Auto-detect** before running Resman again. You must then enter the logical address of the VME-MXI-2 to configure.
4. If problems persist, run Resman with the VXI/VME-MXI-2 located in the leftmost slot of your chassis (VXI Slot 0) and no other instruments installed. If this works, try adding instruments until the problem occurs again. Contact National Instruments for further assistance.

Where can I find hardware specifications?

Refer to Appendix A, *Specifications*, in the *MXI-2 Configuration Reference Manual*.

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 or PXI-8320 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 VXI-MXI-2 and either the PCI-MXI-2 or PXI-8320 two devices or one with respect to the VXIbus?

The PCI-MXI-2, PXI-8320, and the VXI-MXI-2 are unique VXIbus devices with their own logical addresses. However, the MXIbus allows the PCI or PXI/CompactPCI 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, supply a TTL-level signal for EXTCLK; 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.

What are the user and driver windows?

The PCI-MXI-2 and PXI-8320 driver requires the use of two PCI windows: a user window and a driver window. NI-VXI uses the driver window to perform high-level functions such as `VXIin()` and `VXIout()`, and to access registers on the MXI-2 boards in the system. The user window is reserved for low-level function calls, such as `VXIpeek()`, `VXIpoke()`, and `MapVXIAddress()`. The driver window is system defined and not configurable, but you can increase the size of your user window through MAX if you expect to initiate transfers to a wide variety of addresses in both A24 and A32 address space.

What is shared memory and dual-ported memory?

These terms refer to a block of memory that is accessible to both a client and a server. The memory block operates as a message buffer for communications. Shared memory is applicable only if you are using either A24 or A32 address space.

How should I assign logical addresses in a multiple-mainframe system?

A simple algorithm for a system containing only one level of hierarchy—a single chain of MXI cables—is to use the upper nibble (most significant four bits) as a *frame* number and the lower nibble (least significant four bits) as a *device* number. For example, the *fifth* device in the *third* mainframe would be logical address 35 (hex).

For more detailed information on this topic, refer to the *VXI-MXI-2 User Manual*, the *VME-MXI-2 User Manual*, or *VXI-6, VXIbus Mainframe Extender Specification*.

C

Technical Support Resources

Web Support

National Instruments Web support is your first stop for help in solving installation, configuration, and application problems and questions. Online problem-solving and diagnostic resources include frequently asked questions, knowledge bases, product-specific troubleshooting wizards, manuals, drivers, software updates, and more. Web support is available through the Technical Support section of `ni.com`

NI Developer Zone

The NI Developer Zone at `ni.com/zone` is the essential resource for building measurement and automation systems. At the NI Developer Zone, you can easily access the latest example programs, system configurators, tutorials, technical news, as well as a community of developers ready to share their own techniques.

Customer Education

National Instruments provides a number of alternatives to satisfy your training needs, from self-paced tutorials, videos, and interactive CDs to instructor-led hands-on courses at locations around the world. Visit the Customer Education section of `ni.com` for online course schedules, syllabi, training centers, and class registration.

System Integration

If you have time constraints, limited in-house technical resources, or other dilemmas, you may prefer to employ consulting or system integration services. You can rely on the expertise available through our worldwide network of Alliance Program members. To find out more about our Alliance system integration solutions, visit the System Integration section of `ni.com`

Worldwide Support

National Instruments has offices located around the world to help address your support needs. You can access our branch office Web sites from the Worldwide Offices section of `ni.com`. Branch office Web sites provide up-to-date contact information, support phone numbers, e-mail addresses, and current events.

If you have searched the technical support resources on our Web site and still cannot find the answers you need, contact your local office or National Instruments corporate. Phone numbers for our worldwide offices are listed at the front of this manual.

Glossary

Prefix	Meaning	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}

A

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. In VISA, it identifies a resource.
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
API	Application Programming Interface; the direct interface that an end user sees when creating an application
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
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 error	An error that signals failed access to an address. Bus errors occur with low-level accesses to memory and usually involve hardware with bus mapping capabilities. For example, nonexistent memory, a nonexistent register, or an incorrect device access can cause a bus error.
bus master	A device that is capable of requesting the Data Transfer Bus (DTB) for the purpose of accessing a slave device
byte order	How bytes are arranged within a word or how words are arranged within a longword. Motorola ordering stores the most significant (MSB) byte or word first, followed by the least significant byte (LSB) or word. Intel ordering stores the LSB or word first, followed by the MSB or word.

C

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.
Commander	A message-based device that is also a bus master and can control one or more Servants
CompactPCI	An adaptation of the PCI specification for industrial and/or embedded applications that require a more robust mechanical form factor than desktop PCI. CompactPCI provides a standard form factor for those applications requiring the high performance of PCI as well as the small size and ruggedness of a rack-mount system.
configuration registers	A set of registers through which the system can identify a module device type, model, manufacturer, address space, and memory requirements. 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.

DMA	Direct Memory Access; a method by which data is transferred between devices and internal memory without intervention of the central processing unit
DRAM	Dynamic RAM (Random Access Memory); storage that the computer must refresh at frequent intervals
driver window	A region of PCI address space that is decoded by the PCI-MXI-2 or PXI-8320 for use by the NI-VXI software
DTB	*See* Data Transfer Bus.
dynamic configuration	A method of automatically assigning logical addresses to VXIbus devices at system startup or other configuration times
dynamically configured	A device that has its logical address assigned by the Resource
device	Manager. A VXI device initially responds at Logical Address 255 when its MODID line is asserted. A MXIbus device responds at Logical Address 255 during a priority select cycle. The Resource Manager subsequently assigns it a new logical address, which the device responds to until powered down.

E

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

G

GPIB	General Purpose Interface Bus (IEEE 488)

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

IEEE	Institute of Electrical and Electronics Engineers
I/O	input/output; the techniques, media, and devices used to achieve communication between machines and users
instrument driver	A set of routines designed to control a specific instrument or family of instruments, and any necessary related files for LabWindows/CVI or LabVIEW
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

K

KB	Kilobytes of memory

L

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
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*. MXI-2 also defines new higher-performance data transfer protocols.
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-488 or NI-488.2	The National Instruments software for GPIB systems
NI-DAQ	The National Instruments software for data acquisition instruments
NI-VISA	The National Instruments implementation of the VISA standard; an interface-independent software that provides a unified programming interface for VXI, GPIB, and serial instruments
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 MXI-2 board

P

PCI	Peripheral Component Interconnect. The PCI bus is a high-performance 32-bit or 64-bit bus with multiplexed address and data lines.
PXI	PCI eXtensions for Instrumentation; an open implementation of CompactPCI that adds electrical features that meet the high-performance requirements of instrumentation applications by providing triggering, local buses, and system clock capabilities. PXI also offers two-way interoperability with CompactPCI products.

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* 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

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 accessible to both a client and a server. The memory block operates as a message buffer for communications. This is unique to register-based interfaces such as VXI.
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.
SODIMM	Small Outline Dual In-line Memory Module
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 MXI-2 board

T

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

U

user window	A region of PCI address space reserved by the PCI-MXI-2 or PXI-8320 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

VIC	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
VISA	Virtual Instrument Software Architecture. This is the general name given to VISA and its associated architecture. The architecture consists of two main VISA components: the VISA Resource Manager and the VISA Instrument Control Resources.
VISAIC	VISA Interactive Control Program, a part of the NI-VISA software. Used to program devices, and develop and debug 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

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