COMPREHENSIVE SERVICES

We offer competitive repair and calibration services, as well as easily accessible documentation and free downloadable resources.

SELL YOUR SURPLUS

We buy new, used, decommissioned, and surplus parts from every NI series. We work out the best solution to suit your individual needs.

Sell For Cash Get Credit Receive a Trade-In Deal

OBSOLETE NI HARDWARE IN STOCK & READY TO SHIP

We stock **New, New Surplus, Refurbished,** and **Reconditioned** NI Hardware.

APEX WAVES

Bridging the gap *between the manufacturer and your legacy test system.*

1-800-915-6216

www.apexwaves.com

sales@apexwaves.com

All trademarks, brands, and brand names are the property of their respective owners.

Request a Quote CLICK HERE *VXIpc-860*

VXI

Getting Started with Your VXIpc™ Embedded Controller for VxWorks

June 2002 Edition
Part Number 370552B-01

Worldwide Technical Support and Product Information

ni.com

National Instruments Corporate Headquarters

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

Worldwide Offices

Australia 03 9879 5166, Austria 0662 45 79 90 0, Belgium 02 757 00 20, Brazil 011 3262 3599,
Canada (Calgary) 403 274 9391, Canada (Montreal) 514 288 5722, Canada (Ottawa) 613 233 5949,
Canada (Québec) 514 694 8521, Canada (Toronto) 905 785 0085, China 86 21 6555 7838,
Czech Republic 02 2423 5774, Denmark 45 76 26 00, Finland 09 725 725 11, France 01 48 14 24 24,
Germany 089 741 31 30, Greece 01 42 96 427, Hong Kong 2645 3186, India 91 80 4190000,
Israel 03 6393737, Italy 02 413091, Japan 03 5472 2970, Korea 02 3451 3400, Malaysia 603 9596711,
Mexico 001 800 010 0793, Netherlands 0348 433466, New Zealand 09 914 0488, Norway 32 27 73 00,
Poland 22 3390 150, Portugal 210 311 210, Russia 095 238 7139, Singapore 65 6 226 5886,
Slovenia 3 425 4200, South Africa 11 805 8197, 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 and Professional Services* appendix. To comment on the documentation, send email to techpubs@ni.com.

© 2002 National Instruments Corporation. All rights reserved.

Important Information

Warranty

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

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

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

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

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

Copyright

Under the copyright laws, this publication may not be reproduced or transmitted in any form, electronic or mechanical, including photocopying, recording, storing in an information retrieval system, or translating, in whole or in part, without the prior written consent of National Instruments Corporation.

Trademarks

MITE™, MXI™, National Instruments™, NI™, NI-488™, NI-488.2™, ni.com™, NI-VISA™, NI-VXI™, and VXIpc™ are trademarks of National Instruments Corporation.

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

Patents

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

WARNING REGARDING USE OF NATIONAL INSTRUMENTS PRODUCTS

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

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

Compliance

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 at `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.

Compliance to EU Directives

Readers in the European Union (EU) 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.

To obtain the DoC for this product, click **Declaration of Conformity** at `ni.com/hardref.nsf/`. This Web site lists the DoCs by product family. Select the appropriate product family, followed by your product, and a link to the DoC appears in Adobe Acrobat format. Click the Acrobat icon to download or read the DoC.

* 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

How To Use the Manual Set xi
Conventions xii
Related Documentation xiii

Chapter 1
Introduction

How to Use This Manual 1-1
What You Need to Get Started 1-1
Hardware Description 1-2
Software Description 1-2
    Software Notes for VxWorks 1-4
Files and Directories Installed on Your Hard Drive 1-4
Getting Started with VxWorks 1-4
    Developing for VxWorks 1-5
    Software Included with Your VXIpc Controller 1-5
Reinstalling the NI-VXI Software 1-6

Chapter 2
Setup

Step 1. Configure the Hardware 2-1
Step 2. Install the Hardware 2-1
Step 3. Set up the VXIpc Controller with VxWorks 2-2

Chapter 3
VXI Configuration Utility

Running the vxitedit Configuration Utility 3-1
VXIpc Configuration Editor 3-1
    Update Current Configuration 3-2
    Record Configuration to File 3-2
    Load Configuration from File 3-2
    Revert to Current Configuration 3-2
    Logical Address Configuration Editor 3-3
        Logical Address 3-3
        Device Type 3-3
        Address Space 3-3
        VXI Shared RAM Size 3-4

Shared RAM Pool ... 3-4
Advanced Shared RAM Settings ... 3-4
Resource Manager Delay ... 3-5
Device Configuration Editor ... 3-5
System IRQ Level ... 3-5
Servant Area Size ... 3-6
Number of Handlers ... 3-6
Number of Interrupters ... 3-6
Protocol Register ... 3-7
Read Protocol Response ... 3-7
Bus Configuration Editor ... 3-7
VXI Bus Timeout ... 3-7
Automatic VXIbus Retry Protocol ... 3-7
Automatic VXI Slave Cycle Retry ... 3-8
A24/A32 Write Posting ... 3-8
VXI Transfer Limit ... 3-8
Arbiter Type ... 3-9
Request Level ... 3-9
VXI Fair Requester ... 3-9
Arbiter Timeout ... 3-10
User Window and Driver Window ... 3-10
VXI/VME-MXI-2 Configuration Editor ... 3-11
Logical Address ... 3-11
Address Space and Requested Memory ... 3-12
A16 Write Post and A24/A32 Write Posting ... 3-12
Interlocked Mode ... 3-13
VMEbus System Controller ... 3-14
VXI/VME Auto Retry ... 3-15
VXI/VMEbus Timeout Value ... 3-15
VXI/VME Transfer Limit ... 3-16
Arbiter Type ... 3-16
Arbiter Timeout ... 3-16
Request Level ... 3-16
VXI/VME Fair Requester ... 3-17
MXI Bus System Controller ... 3-17
MXI Auto Retry ... 3-17
MXI Bus Timeout Value ... 3-18
Transfer Limit ... 3-18
MXI CLK10 Signal ... 3-18
Parity Checking ... 3-19
MXI Fair Requester ... 3-19

Chapter 4
Developing Your Application

Configuration 4-1
Device Interaction 4-2
VME Support 4-2
Programming with VXI and GPIB 4-3
Additional Compiler Information 4-5
Compiling Your C Program 4-5
#define Statement Used in NI-VXI 4-5

Appendix A
Default Settings

Appendix B
Common Questions

Appendix C
Technical Support and Professional Services

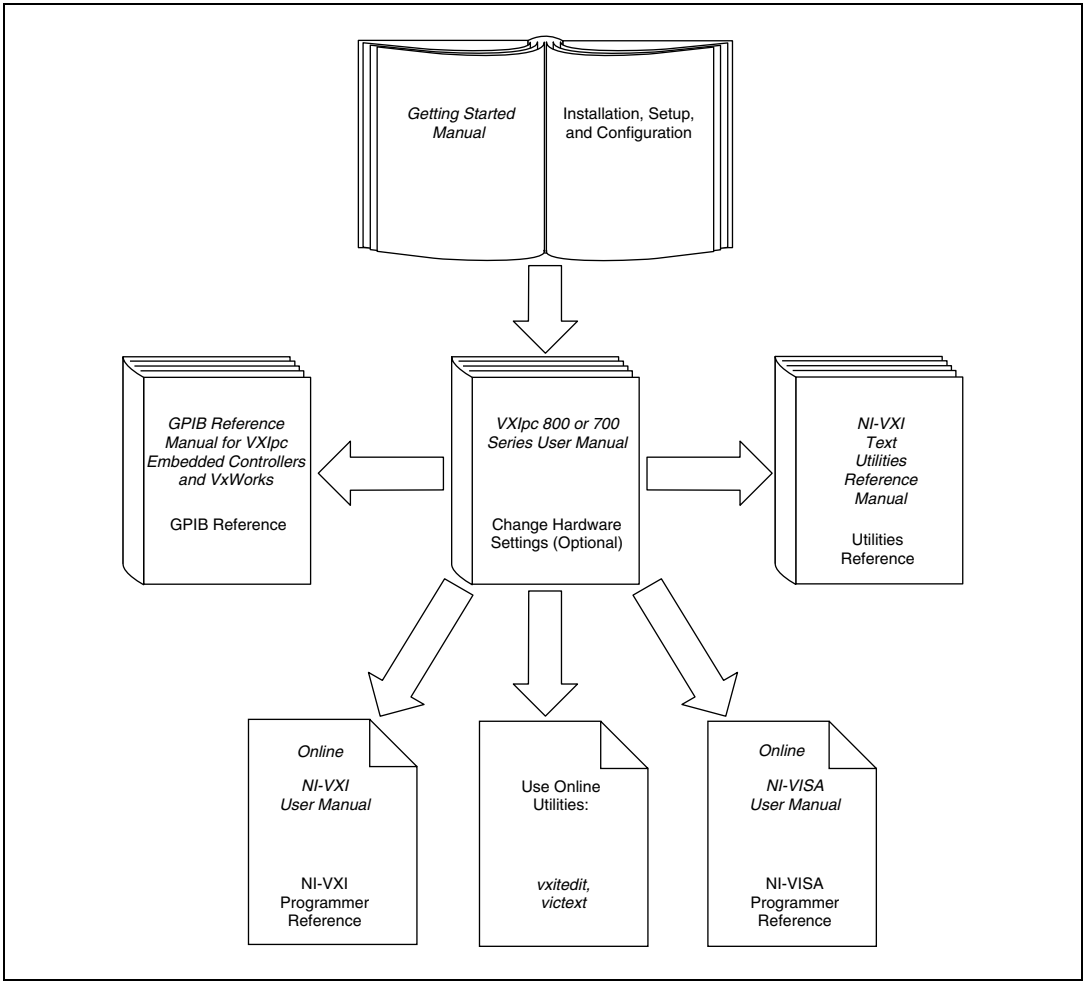
Glossary

Index

About This Manual

Use this manual to get started with the VXIpc embedded computers and the NI-VXI/NI-VISA software for VxWorks. This manual summarizes the setup instructions and default settings for the hardware and software. You may find that these sections contain as much information as you need to get started with your VXIpc kit.

How To Use the Manual Set



Begin by reading this manual for basic instructions on setting up the hardware and software. This manual describes how to get started with your kit using the default hardware and software settings, and describes optional settings you can configure using the NI-VXI/NI-VISA software.

You received either the *VXIpc 800 Series User Manual*, the *VXIpc 700 Series User Manual*, or the *VXIpc 770/870B Series User Manual* with your kit, depending on the hardware product you purchased. The user manual contains more details about changing the hardware installation or configuration from the defaults, and using the hardware.

When you are familiar with the material in the previous manuals, you can begin to use the *NI-VXI User Manual* or, for VISA users, the *NI-VISA User Manual.* These manuals present the concepts of VXI and describe how to use NI-VXI and NI-VISA. The NI-VXI online help, the NI-VISA online help, the *NI-VXI Programmer Reference Manual* and the *NI-VISA Programmer Reference Manual* contain detailed explanations of NI-VXI and NI-VISA functions. Study the descriptions of each function to fully understand the purpose and syntax. Use the Acrobat Reader program, Version 3 or later, to open, view, and navigate through these manuals online.

GPIB users can refer to the *GPIB Reference Manual for VXIpc Embedded Controllers and VxWorks*.

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.

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

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

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

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

italic	Italic text denotes variables, emphasis, a cross reference, or an introduction to a key concept. 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.
VXIpc 700 Series	The terms *VXIpc 700 Series* and *VXIpc-700* refer to a series of C-size, single-slot, VXI controllers. Currently, this series consists of the VXIpc-740, VXIpc-745, and VXIpc-770.
VXIpc 800 Series	The terms *VXIpc 800 Series* and *VXIpc-800* refer to a series of C-size, dual-slot, VXI controllers. Currently, this series consists of the VXIpc-850, VXIpc-860, VXIpc-870, and VXIpc-870B series with various processors and in different speeds.

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*
- VXI-6, *VXIbus Mainframe Extender Specification*, Rev. 1.0, VXIbus Consortium
- *VxWorks Programmer's Guide, 5.3.1* (or later), Wind River Systems, Inc.
- *VxWorks Reference Manual, 5.3* (or later), Wind River Systems, Inc.

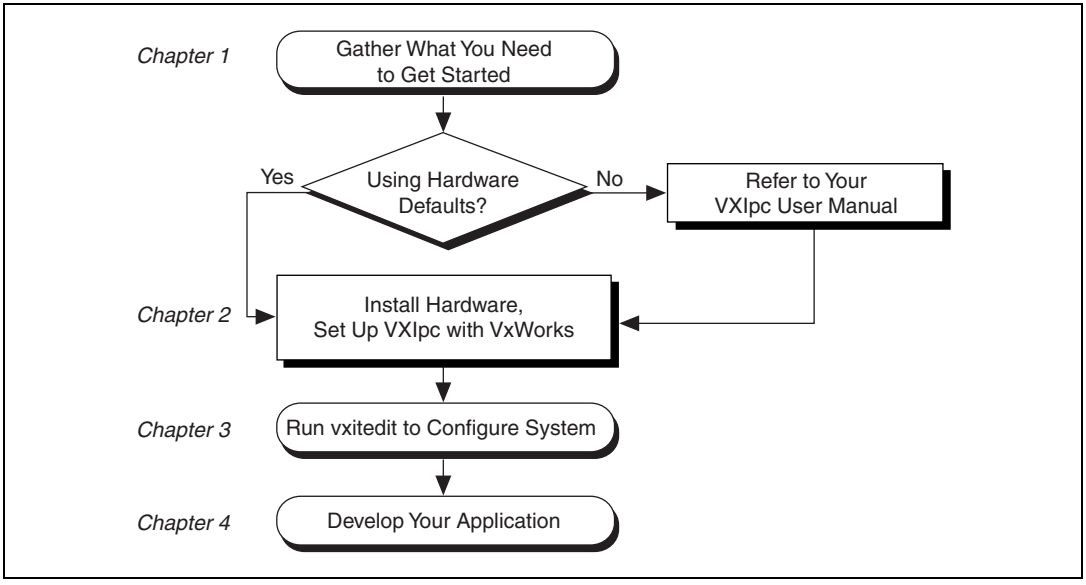
1

Introduction

This chapter describes the VXIpc embedded VXI computers and the NI-VXI software, lists what you need to get started, and gives an overview of the directory structure on your hard drive.

How to Use This Manual

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



What You Need to Get Started

- ❑ VXIpc 800 Series or VXIpc 700 Series embedded controller (hereafter described together as the VXIpc controller)
- ❑ VXIbus mainframe
- ❑ Keyboard (and included adapter cable)

- ❑ Monitor with VGA connector
- ❑ National Instruments software media for the VXIpc embedded controller
- ❑ VxWorks Development System

The NI-VXI software is already installed on your VXIpc computer. It is also included on disk in case you need to reinstall your software.

Order the VxWorks Development System on Windows hosts for *x*86 targets from Wind River Systems, Inc. Install and set up the Tornado development environment for VxWorks according to the documentation that comes with the kit.

Hardware Description

The VXIpc controllers are C-size embedded computers based on the Peripheral Component Interface (PCI) bus. These computers are high-performance, easy-to-use platforms for controlling VXIbus systems, featuring complete VXI functionality through interactive utilities and C function calls. These embedded computers can take advantage of the VXI high-performance backplane capabilities and give you direct control of VXI registers, memory, interrupts, and triggers.

For in-depth details on the VXIpc 800/700 hardware (including a description of the differences between the various models in their respective series), consult the hardware manual that came with your kit—either the *VXIpc 800 Series User Manual*, the *VXIpc 700 Series User Manual*, or the *VXIpc 770/870B Series User Manual*.

Software Description

The NI-VXI bus interface software for the VXIpc embedded controller is already installed on your hard drive. It includes a VXI Resource Manager, an interactive configuration program, libraries of software routines for programming, and an interactive VXIbus control program. You can use this software to seamlessly program multiple-mainframe configurations and ensure software compatibility across a variety of controller platforms. If for some reason you need to reinstall NI-VXI, refer to the *Reinstalling the NI-VXI Software* section.

If you decide to change the NI-VXI software configuration from its default settings, refer to Chapter 3, *VXI Configuration Utility*. This chapter describes each field in the VXIpc Configuration Editor and the VXI/VME-MXI-2 Configuration Editor of the vxitedit software utility. Use the *NI-VXI Text Utilities Reference Manual* to get more information about victext and the other configuration editors in vxitedit. Refer also to the *NI-VXI User Manual*, the *NI-VXI Programmer Reference Manual*, and the NI-VXI online help for thorough details about NI-VXI and the groups of NI-VXI function calls.

NI-VISA is a standard I/O application programming interface (API) for instrumentation programming.

In its full implementation, NI-VISA can control VXI/VME, PXI, GPIB, TCP/IP, or Serial instruments, making the appropriate driver calls depending on the type of instrument being used. NI-VISA uses the same operations to communicate with instruments regardless of the interface type. For example, the NI-VISA command to write an ASCII string to a message-based instrument is the same whether the instrument is Serial, GPIB, or VXI. As a result, NI-VISA gives you interface independence. This makes it easier to switch bus interfaces and means that users who must program instruments for multiple interfaces need learn only one API.

Another advantage of NI-VISA is that it is an object-oriented API that will easily adapt to new instrumentation interfaces as they evolve, making application migration to the new interfaces easy.

VISA is the industry standard for developing instrument drivers. Most current drivers written by National Instruments use NI-VISA and support Windows, Solaris 2, VxWorks, Linux, and Macintosh, as long as the appropriate *system*-level drivers are available for that platform. NI-VISA for VxWorks currently supports only the VXI and Serial interfaces.

The NI-488.2 software for VxWorks kit is also included, which gives you access to the industry-standard NI-488.2 software for controlling external GPIB instruments through the GPIB port on the front panel. The GPIB interface on your VXIpc controller is compatible with the NI-488 driver for a variety of operating systems. Refer to the *GPIB Reference Manual for VXIpc Embedded Controllers and VxWorks* for more information.

Software Notes for VxWorks

The software configuration for NI-VXI for VxWorks offers all the functionality of our NI-VXI drivers for other platforms, within the text-based VxWorks environment. Use vxitedit to reconfigure your VXI hardware and victext to interactively perform NI-VXI operations accessing your VXI devices.

Programs written using the NI-VXI, NI-VISA, or NI-488.2 (board level) function libraries on other platforms are completely portable to VxWorks.

Note VxWorks objects do not contain the symbol `main()`. When porting programs from other platforms, use a more application-specific name in your source code.

Files and Directories Installed on Your Hard Drive

Your hard drive includes a directory called `nivxi` in its root that NI-VXI needs for proper operation. The `nivxi` directory contains several levels of subdirectories that contain help files, tables of information for the driver, and other necessary files for NI-VXI. The `vxipnp` directory includes similar files for NI-VISA.

The hard drive also contains several VxWorks object files, which contain the NI-VXI library and its associated utilities, the NI-VISA library, and the NI-488.2 (GPIB) libraries. Load these files when you boot your VxWorks system so that you can use NI-VXI, NI-VISA, and NI-488.2. Use the VxWorks object loader `ld` to load the files. Refer to Chapter 2, *Setup*, for more information.

Getting Started with VxWorks

The software already installed on your hard drive and on the disks that came with your kit does not include the VxWorks operating system itself. You must contact Wind River Systems, Inc. to order a copy of VxWorks and the Tornado development environment. Use the documentation that comes with VxWorks as your guide for installing, setting up, and learning how to use VxWorks.

Developing for VxWorks

The VxWorks development system is unusual in that the programmer does all coding and compiling on a workstation—the *host* machine—as usual, but then transfers the compiled object files to a *target* machine—in this case, the VXIpc embedded controller. National Instruments supports Windows host machines, although it is possible to use other platforms for this purpose.

The VxWorks development environment, Tornado, makes it easy to manage the unusual host-target configuration. By starting a target server on your host machine, you open connections to the target machine. You use these connections to enter commands on a VxWorks command line, linking and loading your programs as well as issuing the commands from the host to run the application. However, your programs actually execute on the VXIpc target machine.

Software Included with Your VXIpc Controller

The software that comes with your VxWorks-based controller falls under two main categories: host based and target based. All National Instruments software that you need to run NI-VXI on your embedded controller—the development *target*—is already installed on the VXIpc hard drive. The host software is provided separately for you to install on your Windows-based workstation.

The host-side installer installs several subcomponents, including a reference guide for NI-VXI functions, NI-VXI example programs, the VXIpc board support package (BSP), and copies of the NI-VXI library files found on the VXIpc hard drive, as well as NI-488.2 and NI-VISA support.

After you receive your copy of VxWorks from Wind River Systems, use the VXIpc board support package to build the operating system that will run on the VXIpc as explained in Chapter 2, *Setup*.

Note You may never need to use the VxWorks object files installed on the host. This depends on how you use your VxWorks environment.

Reinstalling the NI-VXI Software

The NI-VXI software for VxWorks is already installed on your VXIpc hard drive. However, if for some reason you need to reinstall the software, perform the following steps:

1. Be sure you have up to 5 MB of free space available to accommodate the NI-VXI software.
2. If necessary—for example, if you reformatted your hard drive—set up the VxWorks boot block by using the `vxsys` command on the drive (`vxsys` is part of your Tornado installation). You can find more details on this process in the BSP section in the *Intel x86* appendix of your *VxWorks Programmer's Guide*.
3. Use the diskettes labeled *Target-Side Software for VXIpc with VxWorks* to set up your target hard drive. These disks contain the files required if you need to rebuild your VXIpc hard drive, including the `nivxi` and `vxipnp` directories and the object files for the library and utilities. These disks also contain the VxWorks loader `bootrom.sys`.

2

Setup

This chapter contains basic instructions for setting up the VXIpc embedded controller and the NI-VXI software.

You can use this material as a guide to quickly configure and operate your VXI system using the VXIpc controller. This chapter assumes you intend to perform a basic configuration as follows:

- You have one VXIbus chassis in which you will use the VXIpc embedded controller as the Resource Manager (logical address 0).
- You will use the NI-VXI software for initialization, configuration, and device interaction.
- You will use the default hardware and software settings.

Step 1. Configure the Hardware

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

The *VXIpc 800 Series User Manual,* the *VXIpc 700 Series User Manual*, and the *VXIpc 770/870B User Manual* fully describe the configuration and installation of each embedded controller in their respective series. Refer to your VXIpc user manual if you want to try a different hardware configuration or need more information on a particular setting.

Step 2. Install the Hardware

1. To prevent electrostatic discharge, touch the antistatic plastic package to a metal part of your VXIbus chassis before removing the VXIpc module from the package.

Caution To protect both yourself and the mainframe from electrical hazards, leave the mainframe off until you finish installing the VXIpc module.

2. Plug in your chassis, but leave the power turned off.

3. Install the VXIpc controller in the first slot (Slot 0) of the VXI chassis. In its default configuration, the VXIpc automatically detects whether it should be the VXIbus system controller. The VXIbus system controllers operate certain VXIbus lines as required for VXI systems. Verify that no other VXI devices with system controller capability in the same chassis are configured as system controller.

Caution Having more than one device configured as system controller will damage the VXI system.

4. 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.
5. Also ensure that no VXI devices in your system are configured for logical address 0, which is the default configuration for the VXIpc controller.
6. To complete your installation, attach cables for any devices you want to connect to your system. Refer to your VXIpc user manual if you are uncertain about any of these connections.
7. Turn on power to the VXI chassis.

Step 3. Set up the VXIpc Controller with VxWorks

Install the host-side software on your workstation in this order:

1. Insert the Tornado CD-ROM from Wind River Systems and follow the instructions to install the Tornado development environment.
2. Run the installer on the *Host-Side Software for VXIpc with VxWorks* disk to install the VXIpc board support package (BSP), online help, example files, header files, and other required software from National Instruments into the `target` subdirectory of your Tornado installation.
3. Build the operating system, if you have not done so already, using the VxWorks development software (Tornado) and the VXIpc BSP.

   The current version of the VXIpc BSP requires Tornado II. If you have an earlier version of Tornado, contact Wind River for an upgrade. In the Tornado II environment, the VXIpc BSP can be loaded as a project in your workspace. You can then use the graphical Tornado II tools to configure the parameters of the VXIpc BSP and compile the operating

system. This creates the files `vxWorks` and `vxWorks.sym`, which are required to run the operating system. Refer to your Tornado manuals for more details on how to use these tools. The VXIpc BSP also works in Tornado 1.0.1 compatibility mode.

4. Configure the boot parameters as appropriate for your location. At the VxWorks boot prompt, enter the location of your copy of the VxWorks operating system image, your host computer's IP address, and other applicable parameters.

   Depending on your preference, you can have the VxWorks image loaded from the network or from the VXIpc hard drive. Power on your VXIpc to configure this and other options. After the controller does its normal self-tests, it displays a countdown timer. To change the boot parameters for the controller, abort this timer and enter `c`.

   The parameters you most likely will change are:

   - **boot device** `ata=0,0` for internal IDE or `esmc` (VXIpc-850/860/74*x*), `esyf` (VXIpc-870), or `fxp` (VXIpc-770/870B series) for Ethernet
   - **host** host computer's name on the network
   - **file name** the location of your VxWorks image
   - **inet address** site-specific
   - **host inet** site-specific

   When you finish your changes, the new settings are written to nonvolatile RAM on your controller. This may take a few seconds.

   To permanently change the default values for these options, you can remake the `bootrom.sys` file by following the steps outlined in the BSP section of the *Intel x86* appendix of your *VxWorks Programmer's Guide*.

   For certain applications, you may want to reconfigure parts of the VXIpc BSP. In general, the default settings are taken from the Wind River series of *x*86 BSPs and should be appropriate for your application. VXIpc-specific peripherals configured in the BSP include the network adapter, SCSI controller, and VXI and GPIB interface hardware. Each of these is configured automatically based on the model of VXIpc you specify in Tornado. To change which model is used, navigate to **hardware»BSP configuration variants»VXIpc model selection** in the VxWorks tab of the Tornado graphical project manager and change the selected model (or define the appropriate `INCLUDE_VXIPC_…` macro in the Makefile if you are not using the graphical interface).

If you make changes to the hardware in your system—for example, if you add a new plug-in card or alter the interrupt used by a device—you may need to change the BSP default settings to reflect your change.

Refer to your *VxWorks Programmer's Guide* for full details about installation.

5. Load the NI-VXI object files after booting your controller into the VxWorks operating system. From your `->` prompt (on the host shell by default, or on the target based on your settings in the BSP `config.h` file), use the `ld` command to load the NI-VXI objects, as shown here:

```
-> ld <nivxi.o
-> ld <vxitedit.o
-> ld <resman.o
-> ld <victext.o
```

6. Ensure that your NI-VXI configuration files are available. They are installed on your VXIpc hard drive by default, and the NI-VXI library finds them in `/ide0/nivxi`. To use the default setting if you are not booting from `/ide0`, you need to mount the hard drive as follows:

```
-> usrAtaConfig 0,0,"/ide0"
```

   To change the path the library uses for these files, set the environment variable `NIVXIPATH` as follows:

```
-> putenv("NIVXIPATH=your_path")
```

7. Run the Resource Manager to set up your VXI system by typing `resman` at the prompt. This program automatically finds all other VXI devices in the system configuration and handles all appropriate system setup issues. You must run Resman every time the chassis power is cycled so your application can access devices in the VXI/VME chassis.
8. You may need to run vxitedit to make changes to your system if the default configuration is not suitable for you. Refer to Chapter 3, *VXI Configuration Utility*, for details about each field you can configure through the vxitedit configuration editor.
9. After you finish configuring the system through vxitedit, verify the system configuration through the victext interactive control utility. For an example of how to use victext, refer to the *Device Interaction* section in Chapter 4, *Developing Your Application*.
10. There is no configuration tool for NI-VISA or the NI-488.2 (GPIB) support libraries. To use NI-VISA or NI-488.2, load the objects into memory after performing the above steps to load NI-VXI. To load NI-VISA, type the following:

```
-> ld <visa.o
```

To load GPIB, use the same command, loading the appropriate object file for your platform and application. Normally, this is the PCI-based NI-488.2 (board-level) API:

```
-> ld <nigpib_p.o
```

For older VXIpc controllers with ISA-based GPIB interfaces, or to use the older ESP API, use other GPIB objects (`nigpib.o`, `esp_pci.o`, etc.) instead. Refer to the GPIB `README.TXT` in the BSP for details.

Refer to the NI-VISA and GPIB manuals for VxWorks to learn more details about these libraries.

3

VXI Configuration Utility

This chapter contains instructions for using the VXI Resource Editor to configure the VXIpc embedded computer and the VXI-MXI-2 or VME-MXI-2 chassis extender.

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

Running the vxitedit Configuration Utility

Type `vxitedit` to run the resource editor program. vxitedit presents a list of several configuration editors. Select the VXIpc Configuration Editor from the main menu. Later in this chapter are instructions for using the VXI/VME-MXI-2 Configuration Editor, in case your system contains a VXI-MXI-2 or VME-MXI-2 chassis extender.

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

The rest of this chapter describes only the features of the VXIpc Configuration Editor and the VXI/VME-MXI-2 Configuration Editor. For instructions on using the other editors, refer to the *NI-VXI Text Utilities Reference Manual*.

VXIpc Configuration Editor

The first three options under the VXIpc Configuration Editor are:

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

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

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

Update Current Configuration

Use this option to write the configuration settings to the VXIpc controller EEPROM and files used by NI-VXI. This option configures the VXIpc controller to be consistent with the configuration EEPROM. Notice that some of the configuration settings cannot take effect until you reset the machine, either by using the reset button or by turning the power off and on again.

Record Configuration to File

Use this option to save your configuration settings to a file. Notice that this option does *not* write the configuration settings to the VXIpc controller configuration EEPROM.

If you want to update the VXIpc configuration settings, use the **Update Current Configuration** option instead.

Load Configuration from File

You can use this option to load your configuration settings from a file. This action only updates the configuration settings in your editor. It does *not* write the configuration settings to the VXIpc configuration EEPROM. Use the **Update Current Configuration** option to make the changes take effect.

Revert to Current Configuration

If you made changes to the configuration settings without committing those changes—writing to configuration EEPROM—you can use this option to revert the configuration settings to the values they had before you made the changes.

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

Logical Address Configuration Editor

The Logical Address Configuration Editor has options for the device's logical address, device type, address space, VXI shared memory, and the resource manager delay. The following paragraphs describe the options you can select for each field.

Logical Address

This parameter sets the logical address of the VXIpc controller. The following table shows the allowable range of values and the default value.

Logical Address Range	Default Value
0 to 254	0

Device Type

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

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

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

Address Space

This field indicates the addressing mode(s) of the device's operational registers. You can configure the VXIpc controller in one of three ways. The default addressing mode is for A16 space only. Your other options are A16/A24 and A16/A32.

Notice that options relating to VXI shared memory are disabled when the **Address Space** option is set to A16. Only if you select A16/A24 or A16/A32 are the following options relevant:

- **VXI Shared RAM Size**
- **Shared RAM Pool**
- **Upper/Lower Half Window Byte Swapping**
- **Upper/Lower Half Window Address Mapping**

VXI Shared RAM Size

This field indicates the amount of RAM (in bytes) that is shared in either A24 or A32 space. This determines the *total* shared RAM size. Setting this field to –1 detects how much memory you have installed in your VXIpc controller and requests the same amount of A24 or A32 space. However, if you have more than 8 MB installed in your VXIpc, you need to change the Address Space field to use A32 space.

Shared RAM Pool

This field indicates the size of memory in kilobytes that is allocated on NI-VXI startup. This is physically contiguous memory that can be dual-ported on the VXIbus.

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

If you make a change to this setting, you must restart the computer to enable the change.

Memory Range	Default Value
0 to 65535 KB	0 KB

Advanced Shared RAM Settings

The VXI shared RAM is divided into two halves, or *windows*. You can select the byte order and mapping scheme for each half independently. These configuration options are intended for more advanced users.

Upper/Lower Half Window Byte Swapping

This field indicates whether byte swapping should be performed for slave accesses to this half of the VXI shared RAM space. For example, if the

native byte order of the shared RAM is Intel (Little Endian), and you want to present data to the VXIbus in Motorola (Big Endian) byte order, you need to enable byte swapping for the appropriate window half. Byte swapping is disabled for both windows by default.

Upper/Lower Half Window Address Mapping

This field determines if the upper/lower half windows map to the same address or different addresses in system memory.

The default setting maps each half window to a unique local address on the VXIpc controller. If you change this setting, the buffer in system RAM is dual-ported to the VXIbus in both Little Endian and Big Endian byte order. The setting of the **Byte Swapping** option for each half window determines whether the byte order is Little Endian or Big Endian.

Resource Manager Delay

Note This field is effective only when the VXIpc controller is at its default logical address of 0. This logical address is required for the Resource Manager.

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

RM Delay Range	Default Value
0 to 65535 s	5

Device Configuration Editor

The Device Configuration Editor configures options for remote controller communication and local device settings.

System IRQ Level

Remote controllers can report events such as triggers and DMA to the VXIpc through a VXI IRQ line. This field selects which VXI IRQ level the remote controllers should use to report such events.

Interrupt Request Levels	Default Value
1 to 7 or disabled	Disabled

Note When the system IRQ line is disabled, the remote controller functionality is not available. Enable the system IRQ line if you are using a multi-mainframe system.

The VXI IRQ designated as system IRQ line cannot be disabled using the `DisableVXIint()` or `DisableVXItoSignalInt()` functions. The VXIpc controller always acknowledges it automatically when it is the Resource Manager.

Servant Area Size

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

Servant Area Range	Default Value
0 to 255	0

Note If the VXIpc controller is the Resource Manager (Logical Address 0), this setting is irrelevant.

Number of Handlers

This field gives the number of interrupt handlers that the VXIpc controller supports.

Interrupt Handlers	Default Value
0 to 7	1

Number of Interrupters

This field gives the number of interrupters that the VXIpc controller supports.

Interrupters	Default Value
0 to 7	0

Protocol Register

This field specifies the contents of the Protocol register, indicating which protocols the device supports. This field is meaningful only when the VXIpc controller is configured as a message-based device. The default value is 0xFF0 (Commander, Signal Register, Master).

Read Protocol Response

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

Bus Configuration Editor

Use the Bus Configuration Editor to configure VXI bus settings, PCI bus settings, and bus arbitration settings for the VXIpc controller.

VXI Bus Timeout

The Bus Timeout (BTO) is a watchdog timer for transfers on the VXIbus. After the specified amount of time has elapsed, the BTO circuitry terminates a VXIbus cycle if no slave has responded. This feature is applicable only if the VXIpc controller you are configuring is a VXI Slot 0 device. You should disable the BTO of any other non-Slot 0 devices residing in the mainframe.

The lowest value in the allowable range is 15 µs and the highest is 256 ms. The default value is 500 µs.

Automatic VXIbus Retry Protocol

When the **Enable Auto Retry protocol** option is active, the VXIpc controller can recognize and send the VXIbus retry protocol. If you disable this option, a retry is mapped to a bus error response. By default this option is enabled.

Automatic VXI Slave Cycle Retry

- This option is *not* available in the VXIpc 700 Series.

The VXIpc 800 Series has an automatic retry feature for cycles that map from the VXIbus to the PCI bus on the VXIpc 800. You can use the Automatically retry VXI slave cycles field to enable or disable this option. By default this option is enabled on the VXIpc 800 Series and disabled on the VXIpc 700 Series.

Normally, when a cycle maps from the VXIbus to the PCI bus, any retry response received on the PCI bus is passed to the VXIbus. When this feature is enabled, the VXIpc 800 automatically retries any PCI cycle when the PCI host responds to a cycle with a retry. The VXIpc 800 automatically continues to retry the PCI cycle until it receives either a Disconnect or Target-Abort response, which it then passes to the VXIbus. This behavior is the default because many VXIbus masters do not support VXI retries. If the VXIbus master does support retries, you may find it beneficial to disable this feature. With this feature disabled, you can lower the value of the VXI Bus Timeout because there is no delay from the inward cycles being retried.

Note The VXIpc 800 has a limit on the number of automatic retries it will perform on any one cycle. If the limit is exceeded and the VXIpc 800 receives another retry, it will pass a retry or BERR (depending on whether the **Enable Auto Retry protocol** option is active or disabled) to the VXIbus even though the **Automatically retry VXI slave cycles** option is active.

A24/A32 Write Posting

The VXIpc controller can increase performance with its capability to post write cycles from the VXIbus. You should post write cycles only to addresses that cannot return a BERR signal, because the BERR will not be reported to the originating master. By default, this option is enabled.

The A24/A32 write posting field affects write cycles to the VXIpc controller via its requested memory space from the VXIbus. When this option is enabled, the VXIpc controller completes a VXIbus write cycle before writing the data from the cycle to the local destination on the VXIpc.

VXI Transfer Limit

You can use the Transfer Limit field to set how many data transfers the VXIpc controller will perform on the VXIbus before releasing it to another master device that is requesting use of the bus.

The available options you can choose from are 16, 64, and 256 transfers. If you do not want the VXIpc to hold the VXIbus long enough to perform 256 transfers (the default value), you can select a smaller value for this field.

Arbiter Type

You can use the **Arbiter Type** feature to configure the VXIpc controller as either a Priority or Round Robin VMEbus arbiter. This option is applicable only if the VXIpc you are configuring is a VXI Slot 0 device. The default value is **Priority**.

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

Request Level

The VXIpc controller uses one of the four VXIbus request levels (0 to 3) to request use of the VXI Data Transfer Bus (DTB). The VXIpc requests use of the DTB whenever a local cycle maps into a VXIbus cycle.

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

VXI Fair Requester

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

Arbiter Timeout

An arbitration timeout feature is available on the VXIpc controller when it is acting as the VMEbus arbiter. This feature applies only to a VXI Slot 0 VXIpc. By default, this option is disabled.

If you enable this feature, the timer begins when the arbiter circuit on the VXIpc drives one of the BGOUT lines on the backplane. If no device takes over the bus within the timeout limit, the BGOUT is removed and the bus is either idle or granted to another requester.

User Window and Driver Window

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

The windows are mapped to PCI base address registers and determine the amount of PCI memory space the VXIpc requests from the PCI system during initialization. You can set the window base, window size, and whether the window resides above or below the 1 MB address space boundary.

Window Size

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

If you change this setting, you must reconfigure the memory mapping settings in `syslib.c` and recompile the VxWorks image. Refer to the *Step 3. Set up the VXIpc Controller with VxWorks* section in Chapter 2, *Setup*, for details on how to do this.

You can also disable this option. Disabling the user window causes the VXIpc to request the minimum amount of address space on the PCI bus. With the window disabled, you cannot perform any low-level function calls such as `VXIpeek()`, `VXIpoke()`, and `MapVXIAddress()`.

The default setting for the user window is set at 64 KB. National Instruments recommends you have a user window of at least this value.

If you will initiate transfers to a wide variety of addresses in both A24 and A32, you should increase the size of the user window.

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

Below 1 MB

This field is not used by the NI-VXI for VxWorks driver and is disabled. It is used with other operating systems, such as DOS. Ignore this setting if you are using VxWorks.

Window Base

This field is not used by the NI-VXI for VxWorks driver and is disabled. It is used with other operating systems, such as DOS. Ignore this setting if you are using VxWorks.

VXI/VME-MXI-2 Configuration Editor

If your system contains a VXI-MXI-2 or a VME-MXI-2, you can configure it through the vxitedit utility. Before running the VXI/VME-MXI-2 Configuration Editor, you must run Resman.

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

When you start the VXI/VME-MXI-2 Configuration Editor, it prompts you for the logical address of the VXI/VME-MXI-2 that you want to configure. You can use the **Resource Manager Display** option in vxitedit to determine the logical address your VXI/VME-MXI-2 is using.

After finding a VXI/VME-MXI-2, the VXI/VME-MXI-2 Configuration Editor prompts you to match the type of address—VXI-MXI-2 or VME-MXI-2—installed in your system. You can then use normal vxitedit commands such as `help`, `list`, and `modify` to display the current settings for your VXI/VME-MXI-2 and to change the configuration if necessary.

Logical Address

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

The default selection is the **Switch** option. Notice that the Logical Address is read only. In this option you need to change the hardware switch setting on the VXI/VME-MXI-2 itself if you want to change the logical address.

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

Address Space and Requested Memory

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

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

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

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

A16 Write Post and A24/A32 Write Posting

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

Set the appropriate option if you want to use either A16 or A24/A32 write posting. By default, both options are disabled.

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

The A24/A32 write posting option affects write cycles that map through the A24 window and A32 window from the VXI/VMEbus to the MXIbus and vice-versa. This field also affects write cycles to the VXI/VME-MXI-2 itself via its requested memory space from both the VXI/VMEbus and the MXIbus. For more information on the A16, A24, and A32 windows, refer to VXI-6, *VXIbus Mainframe Extender Specification*.

Interlocked Mode

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

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

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

longer possible, and system performance may be lower than in normal operating mode.

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

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

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

VMEbus System Controller

You can use the **System Controller** setting to override the jumper setting on the VXI-MXI-2. (The VME-MXI-2 does not have an onboard jumper setting for this option.) When the **Auto** setting (the default setting) is active, the onboard jumper setting determines if the VXI-MXI-2 is the VXI Slot 0 device. For more information, refer to your hardware user manual or your MXI-2 getting started manual.

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

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

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

VXI/VME Auto Retry

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

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

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

VXI/VMEbus Timeout Value

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

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

VXI/VME Transfer Limit

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

You can choose 16, 64, or 256 transfers. If you do not want the VXI/VME-MXI-2 to hold the VXI/VMEbus long enough to perform 256 transfers (the default value), select a smaller value for this field.

Arbiter Type

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

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

Arbiter Timeout

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

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

Request Level

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

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

VXI/VME Fair Requester

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

MXI Bus System Controller

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

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

MXI Auto Retry

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

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

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

MXI Bus Timeout Value

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

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

Transfer Limit

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

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

MXI CLK10 Signal

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

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

You can use the **Drive** or **Receive** options to override the setting of S7 and control the direction of the MXIbus CLK10 signal. If you set the VXI-MXI-2 to receive the MXIbus CLK10 signal, configure the W3 jumper setting to use the MXIbus as the source for generating the VXIbus CLK10 (applicable only if the VXI-MXI-2 is a Slot 0 device). When you set the VXI-MXI-2 to drive the MXIbus CLK10, it uses the VXIbus CLK10

as the source. In this case, change the jumper setting so that it does *not* use the MXIbus CLK10 as the source for the VXIbus CLK10.

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

Parity Checking

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

MXI Fair Requester

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

4

Developing Your Application

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

After verifying your system configuration, you can begin to develop your VXI application. Be sure to check the `README` file for the latest application development notes and changes.

Your software includes several utilities to assist you in your system development. These include Resman, vxitedit, and victext. You can also access several examples to learn how to use NI-VXI and NI-VISA for certain tasks.

Configuration

The configuration utilities in your kit are Resman and vxitedit. Resman is the application that 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 every time chassis power is cycled to reconfigure them.

After Resman has detected and configured all VXI devices, you can view specific information on each device in your system by using the vxitedit utility. This utility includes a Resource Manager Display, which contains a description for each device, including each VXI device's logical address.

Although the VXIpc controller is configured in hardware and software for the most typical configuration, you may want to change the default settings to best suit your application. Use the VXIpc Configuration Editor or the VXI/VME-MXI-2 Configuration Editor available in vxitedit to view or change the settings for your devices.

Device Interaction

You can interact with your VXI devices by using the victext utility. This utility lets you interactively control your VXI devices.

Try the following in victext.

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 0,0
```

This should return a value, such as:

```
Return Status (0): SUCCESS.
value = 0x9ff6
```

If the value ends with `ff6`, you have successfully read the National Instruments manufacturer ID from the VXIpc controller's 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 Resource Manager Display of vxitedit. In this way, you can verify that your VXIpc can access each of the devices in your VXI system successfully.

You can also access VXI and VME devices that are configured in A16, A24, and A32 address space by using the `vxiin` or `vxiout` commands. For more information regarding victext operation and commands, refer to the *NI-VXI Text Utilities Reference Manual*.

VME Support

The Resman utility identifies and configures VXI devices but does not configure VME devices. The VME specification does not define the initialization and configuration procedures that the VXI specification requires.

To access VME devices in your system, you must configure NI-VXI to see these devices by using the Non-VXI Device Configuration Editor in vxitedit.

Note Be sure to indicate the frame number for each VME device you add. The frame number is the logical address of the controller for the chassis in which the device is located. For a single-frame system, the frame number is 0.

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. VXI and VME devices cannot share interrupt levels.

Resman uses this information to properly configure the various device-specific VME address spaces and VME interrupt lines. For more information on configuring VME devices in your VXI system, refer to the *NI-VXI Text Utilities Reference Manual*.

Programming with VXI and GPIB

National Instruments provides three different programming interfaces for accessing your instruments: NI-VISA, NI-VXI, and NI-488.2. 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, VME, GPIB, and serial devices—because the NI-VISA functions have the same interface.

NI-VXI is the National Instruments proprietary interface for programming VXI/VME 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.

NI-488.2 is the National Instruments industry-standard implementation of the ANSI/IEEE Standards 488.1-1987 and 488.2-1992. The original GPIB (General Purpose Interface Bus) specification, known as ANSI/IEEE Standard 488.1-1987, describes a standard interface for communication between instruments and controllers from various vendors. It contains information about electrical, mechanical, and functional specifications. The GPIB is a digital, 8-bit parallel communications interface with data transfer rates of 1 Mbytes/s and above. The bus supports one System

Controller, usually a computer, and up to 14 additional instruments. The ANSI/IEEE Standard 488.2-1992 extends IEEE-488.1 by defining a bus communication protocol, a common set of data codes and formats, and a generic set of common device commands.

The best way to learn how to program with NI-VXI and 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-VXI examples are called `vxihigh.c` and `vxiws.c`. The NI-VISA examples of these tasks are called `VXI-VME\HighReg.c` and `General\RdWrt.c`. Use the other examples as you try more advanced techniques. Consult the *NI-VXI User Manual*, the *NI-VISA User Manual*, or the GPIB online help for additional information on these topics.

Note By default, the NI-VXI examples and user manual reside in the BSP's `\vxi\examples` or `\manuals` directory (usually `c:\Tornado\target\config\vxipc\vxi\examples` or `\manuals`), and the NI-VISA examples and user manual are in the *`VXIpnp`*`\VxWorks\NIvisa\Examples` or `\Manuals` directory. Use the Acrobat Reader program to open and navigate through the manuals.

Table 4-1 summarizes the topics addressed by the NI-VXI and NI-VISA example programs.

Table 4-1. NI-VXI/NI-VISA Examples

Coverage	NI-VXI Example	NI-VISA Example
Message-Based Access (Word-Serial)	`vxiws.c`	`General\RdWrt.c`
High-Level Register Access (Data Move Operations)	`vxihigh.c`	`VXI-VME\HighReg.c`
Low-Level Register Access	`vxilow.c`	`VXI-VME\LowReg.c`
Sharing Memory	`vximem.c`	`VXI-VME\ShareSys.c`
Interrupt Handling	`vxiint.c`	`VXI-VME\AsyncIntr.c` and `WaitIntr.c`
Trigger Handling	`vxitrig.c`	`VXI-VME\WaitTrig.c`

Additional Compiler Information

When building an application with NI-VISA, you must include `visa.h` in your source code.

When building an application with NI-VXI, you must include `nivxi.h` in your source code.

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.

Compiling Your C Program

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

The easiest way to compile the sample program is to use the makefile included with the NI-VXI/NI-VISA software. The makefile uses `GNU cc`, which comes with the VxWorks development kit from Wind River Systems. The `gcc` executable is `cc386`.

#define Statement Used in NI-VXI

It is necessary to define the `VXIVXWORKS` symbol so that the NI-VXI library can work properly with your program. You can define the symbol using `#define` statements in the source code or you can use the `-D` option in your compiler. If you use a `#define` statement, you must define the symbol before including the NI-VXI header file `nivxi.h`. If you use the makefiles to compile the sample program, the makefile already defines the necessary symbol.

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

```
#define VXIWVXWORKS
.
.
.
#include <nivxi.h>
```

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

```
-DVXIVXWORKS
```

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

A

Default Settings

This appendix summarizes the 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 VXIpc user manual for your hardware reference and to Chapter 3, *VXI Configuration Utility*, for your software reference.

Because you can also use vxitedit to configure a VXI-MXI-2 or a VME-MXI-2, this appendix also summarizes the software default settings for the VXI/VME-MXI-2.

VXIpc Controller

This section summarizes the hardware and software default settings for the VXIpc controllers.

Table A-1. VXIpc 770/870B Series Hardware Default Settings

Jumper	Default Setting	Optional Setting
W5	1–2 Normal CMOS operation	2–3 Clear CMOS
W6	2–3 16-bit SCSI termination enabled	1–2 SCSI termination disabled
J20	No jumper	Master/Slave/CSEL*
J17	3–4 Automatic slot zero detection	1–2 Non-slot 0 5–6 Force slot 0
W2	2–3 Enable MITE self-configuration	1–2 Disable MITE self-configuration
W1	2–3 MITE user configuration	1–2 MITE factory configuration
S1	1–2 Internal oscillator	2–3 External oscillator
* These pins are generally defined in a figure on the hard drive cover.		

Table A-2. VXIpc 870 Series Hardware Default Settings

Jumper	Default Setting	Optional Setting
J12	Enable automatic Slot 0 detection	Force Slot 0; Force Non-Slot 0
S1	MITE user configuration	MITE factory configuration
S2	Enable MITE self-configuration	Disable MITE self-configuration
W1, 3, 5, 7	CPU bus factor	**Note:** For more information, refer to the *VXIpc 870 Series User Manual*.
W4	100 MHz CPU bus speed	66 MHz CPU bus speed
W6	Normal CMOS operation	Clear CMOS
W8	Flash write enable	Flash protection
W10	Enable Ethernet Serial EEPROM configuration	Disable Ethernet Serial EEPROM configuration (uses default power on values)
W11–12	Enable 16-bit SCSI termination	SCSI termination
W15	Voltage monitor only required voltages	Voltage monitor all voltages

Table A-3. VXIpc 850/860 Hardware Default Settings

Hardware Component	Default Setting
S1—Ethernet EEPROM	Enabled. *Do not alter this setting*.
S2—Power On Self-Configuration (POSC)	Enabled. *Do not alter this setting*.
S3—CLK10 Source	Source from onboard oscillator
S4—CLK10 SMB Polarity	Not inverted
S5—CLK10 SMB Direction	Receive CLK10
S6—CLK10 SMB Termination	Do not terminate
S7—TrigIn SMB Termination	Do not terminate
S8—GPIB Circuitry Interrupt	Level 11
S9—MITE Configuration EEPROM	Load values from user section

Table A-3. VXIpc 850/860 Hardware Default Settings (Continued)

Hardware Component	Default Setting
W1—SCSI Termination	Enabled
W2—CMOS Clear	CMOS not cleared
W4—Parallel Port DMA Channel	Channel 1
W13—Slot Detection	Automatically detect slot

Table A-4. VXIpc 740/745 Hardware Default Settings

Hardware Component	Default Setting
W1—Slot detection	Automatically detect slot
W3—CMOS Clear	CMOS not cleared
W6—Ethernet EEPROM	Enabled. *Do not alter this setting*.
W7—MITE Configuration EEPROM	Load values from user section
W10—Power On Self-Configuration (POSC)	Enabled. *Do not alter this setting*.
W12—TrigIn SMB Termination	Do not terminate

Table A-5. Logical Address Configuration Editor Default Settings

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

Table A-6. Device Configuration Editor Default Settings

Editor Field	Default Setting
System IRQ Level	Disabled
Servant Area Size	0
Number of Handlers	1
Number of Interrupters	0
Protocol Register	0xFF0
Read Protocol Response	0x8448

Table A-7. Bus Configuration Editor Default Settings

Editor Field	Default Setting
Bus Timeout	500
Automatic Retry Protocol	Enabled
Automatic VXI Slave Cycle Retry	Enabled on the VXIpc 800 Series Disabled on the VXIpc 700 Series
A24/A32 Slave Write Posting	Disabled
VXI Transfer Limit	256
Arbiter Type	Priority
Request Level	3
Fair Requester	Disabled
Arbiter Timeout	Disabled
User Window Base	Auto
User Window Size	64 KB
User Window Below 1 MB	No
Driver Window Base	Auto
Driver Window Size	32 KB
Driver Window Below 1 MB	No

B

Common Questions

This appendix addresses common questions you may have about using the NI-VXI software on the VXIpc platform for VxWorks.

How should I set up my system?

1. Install hardware components and boot the system.
2. Build the VxWorks operating system using the VxWorks Development System and the Board Support Package.
3. Load the NI software and utility object files.
4. Configure your hardware with vxitedit. Use the Logical Address, Bus, and Device Configuration Editors to change aspects of the hardware and software.
5. Reboot to initialize your National Instruments hardware.
6. Reload the tools you need and run Resman to initialize the VXIbus.
7. Optionally run vxitedit to configure any extender devices on the VXIbus.
8. Run victext to verify device operation.
9. Load and run your NI-VXI, NI-VISA, and/or NI-488.2 (GPIB) application.

How do I load the NI-VXI software driver and utilities?

There are two options for loading and linking the NI-VXI software. One option is to load the NI-VXI software into the VxWorks operating system before your application requires it. You can include the following lines in your startup script to load the NI-VXI modules at boot time.

```
ld < path/nivxi.o
ld < path/resman.o
ld < path/vxitedit.o
ld < path/victext.o
```

where *path* is the location where you have installed the utilities.

If you have a good understanding of VxWorks, another option is to modify the VXIpc BSP Makefile to link the NI-VXI libraries into the VxWorks system image. You can do this in the project manager GUI or by changing

MACH_EXTRA to include the object files for NI-VXI. This action ensures that the NI-VXI software is available as soon as the operating system finishes booting.

What is the function of the NI-VXI utilities?

The utilities have the following responsibilities:

- Resman—This utility initializes and configures all the other devices in your VXI system.
- vxitedit—This utility configures your National Instruments hardware.
- victext—This utility allows you to communicate interactively with VXI devices over the VXIbus using the NI-VXI API.

What does Resman do?

The Resman utility 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 (nonzero) logical address and is a message-based device, you need to start Resman before running it on the Logical Address 0 computer.

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.

How do I handle VME devices?

Although there is no way to detect VME devices in a system automatically, you can add them easily through the Non-VXI Device Editor in vxitedit. After you assign a pseudo-logical address and other resource values, Resman can configure the VME devices into your VXI system.

How can I determine the revision of the VXIpc controller that my NI-VXI software supports?

Run the NI-VXI utility program victext. Type `ver` at the prompt. The utility displays the versions of victext and NI-VXI, and the latest VXIpc hardware revision that this NI-VXI driver supports.

How can I determine the serial number and hardware revision of the VXIpc controller?

Run vxitedit and select the **VXIpc Configuration Editor**. The opening screen displays the serial number and hardware revision of the VXIpc.

Which NI-VXI utility program must I use to configure the VXIpc controller?

Use the vxitedit utility to configure the VXIpc. You do not need to run vxitedit if you are satisfied with the default settings. Refer to Chapter 3, *VXI Configuration Utility*, for complete details on using the configuration editors.

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

Use the Resman utility to perform startup Resource Manager operations. This utility uses the settings configured in vxitedit. It initializes your VXI/VMEbus system and stores the information that it collects to the `resman.tbl` file in the `tbl` subdirectory of the `nivxi` directory. You can access this information using the NI-VXI system configuration functions described in detail in Chapter 2, *Function Reference*, of the *NI-VXI Programmer Reference Manual*.

Resman reports errors (such as "Unknown directory") if the `nivxi/tbl` directory is not found. By default, this directory is `/ide0/nivxi` on your VXIpc hard drive. To use the default setting if you are not booting from `/ide0`, you need to mount the hard drive as follows:

```
-> usrAtaConfig 0,0,"/ide0"
```

To change the path NI-VXI uses for these files, set the environment variable `NIVXIPATH` as follows:

```
-> putenv("NIVXIPATH=your_path")
```

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

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

To test individual devices, you can use the victext program to interactively issue NI-VXI functions. You can use the `vxiin()` and `vxiout()` functions or the `vxiinReg()` and `vxioutReg()` functions to test register-based devices by programming their registers. If you have any

message-based devices, you can send and receive messages with the `wswrt()` and `wsrd()` functions. Notice that `vxiinReg()` and `vxioutReg()` are for VXI devices only.

What should I do if I get a `Configuration EEPROM is Invalid` message?

There are several reasons why you could receive this message. If you turn off the computer while the configuration update process is still in progress, the VXIpc functions normally except when using vxitedit. To correct this problem, switch to the factory configuration as described in the configuration and installation chapter of your VXIpc user manual. This requires that you change switch S9 on the VXIpc 800 Series or jumper W7 on the VXIpc 700 Series. Reboot the computer 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 the memory for the MITE's configuration EEPROM may not be mapped correctly. If a mapping error is the cause, follow the instructions in the *Step 3. Set up the VXIpc Controller with VxWorks* section in Chapter 2, *Setup*, to set up the mapping in `sysLib.c`.

What do the LEDs on the front of the VXIpc controller mean?

Refer to the LED indicator descriptions in your VXIpc user manual for a description of the front panel LEDs.

Is something wrong if the red SYSFAIL and FAILED LEDs stay lit after booting the VXIpc controller?

If either the SYSFAIL or FAILED LED remains lit, refer to the LED indicators descriptions in your VXIpc user manual for troubleshooting steps.

Can I access 32-bit registers in my VXIbus system from the VXIpc?

Yes. The VXIpc uses the 32-bit PCI bus to interface to the VXIbus. In fact, its VXIbus circuitry also supports the new VME64 standard for D64 accesses.

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 backplane. However, the oscillator for the VXIpc 800/700 and the EXTCLK input on the VXIpc 800

Series front panel use TTL levels; therefore, you need to supply a TTL-level signal for EXTCLK. Our voltage converters convert the signal to differential ECL. You cannot drive CLK10 externally on the VXIpc 700 Series.

What is the accuracy of the CLK10 signal?

The CLK10 signal generated by all of the VXIpc controllers is ±100 ppm (0.01%) as per the VXIbus specification. If you need a more accurate CLK10 signal on the VXIpc 800 Series, you can use the EXTCLK connector on its front panel.

What kind of monitor can I use with the VXIpc controller?

VXIpc computers that use Super VGA video output work only with monitors having a horizontal scan rate of at least 50 kHz and a vertical scan rate of 60 Hz.

Caution Make sure your monitor meets this specification. Enabling the Super VGA option on a monitor that does *not* meet this specification will damage your monitor.

What should I do if my keyboard connector does not fit into the keyboard port on the VXIpc controller?

You can plug keyboards that have a 6-pin Mini DIN PS/2 type connector directly into the VXIpc. You can use the keyboard adapter cable included with every VXIpc 800 Series kit to adapt the larger AT keyboard connector to the 6-pin Mini DIN connector.

How do I connect an external speaker to get audio capability?

- **VXIpc 800 Series users only**—A twisted-pair cable connects the front panel audio connector to the VXIpc 800 Series motherboard. Connect the external speaker to this front-panel connector. The center pin of the connector provides the audio signal. The shield of the connector is GROUND.

How do I add RAM to the VXIpc? What is the maximum amount of RAM that I can have?

For information about adding RAM to the VXIpc controller, refer to Appendix A, *Specifications*, in your VXIpc user manual.

Which interrupt levels are free to be used by ISA bus boards? Which area of upper memory (adapter space) is free for use by ISA bus boards or expanded memory manager software programs?

Refer to the appendix on VXIpc system resources in your VXIpc user manual for information on the available port I/O register space, upper memory area, interrupts, and DMA channels.

How do I install the VXIpc controller in a slot other than Slot 0?

The VXIpc controller automatically detects whether it is in Slot 0 of a VXIbus mainframe. You do not need to change jumper settings to install the VXIpc in a slot other than Slot 0 unless you have defeated the first slot detector (FSD) circuitry by changing the appropriate jumper setting on the VXIpc.

Refer to the configuration and installation chapter of your VXIpc user manual for information on enabling and defeating the FSD circuitry.

How do I check the configuration of the memory, floppy drive, hard drive, time/date, and so on?

You can view these parameters in the BIOS setup. To enter the BIOS setup, reboot the VXIpc and press <Del> during the memory tests. Refer to Chapter 4, *BIOS*, in your VXIpc user manual for more information.

C

Technical Support and Professional Services

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

- **Support**—Online technical support resources include the following:
  - **Self-Help Resources**—For immediate answers and solutions, visit our extensive library of technical support resources available in English, Japanese, and Spanish at ni.com/support. These resources are available for most products at no cost to registered users and include software drivers and updates, a KnowledgeBase, product manuals, step-by-step troubleshooting wizards, hardware schematics and conformity documentation, example code, tutorials and application notes, instrument drivers, discussion forums, a measurement glossary, and so on.
  - **Assisted Support Options**—Contact NI engineers and other measurement and automation professionals by visiting ni.com/ask. Our online system helps you define your question and connects you to the experts by phone, discussion forum, or email.
- **Training**—Visit ni.com/custed for self-paced tutorials, videos, and interactive CDs. You also can register for instructor-led, hands-on courses at locations around the world.
- **System Integration**—If you have time constraints, limited in-house technical resources, or other project challenges, NI Alliance Program members can help. To learn more, call your local NI office or visit ni.com/alliance.

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

Glossary

Prefix	Meaning	Value
n-	nano-	10^{-9}
µ-	micro-	10^{-6}
m-	milli-	10^{-3}
k-	kilo-	10^{3}
M-	mega-	10^{6}
G-	giga-	10^{9}

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 device 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

address — character code that identifies a specific location (or series of locations) in memory

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
ASIC	Application-Specific Integrated Circuit

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 has two sets of bused connectors called J1 and J2. A D-size VXIbus system has three sets of bused connectors called J1, J2, and J3.
BERR*	Bus Error Signal
BIOS	Basic Input/Output System. BIOS functions are the fundamental level of any PC or compatible computer. BIOS functions embody the basic operations needed for successful use of the computer's hardware resources.
BSP	Board Support Package. A set of files that defines how a VxWorks operating system image is created for a given target.
BTO	*See* Bus Timeout 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.
byte order	how bytes are arranged within a word or how words are arranged within a longword. Motorola ordering stores the most significant byte (MSB) 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.

CMOS	Complementary Metal Oxide Semiconductor; a process used in making chips
Commander	a message-based device that is also a bus master and can control one or more Servants
configuration registers	a set of registers through which the system can identify a module device type, model, manufacturer, address space, and memory requirements. In order to support automatic system and memory configuration, the VXIbus specification requires that all VXIbus devices have a set of such registers.

D

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 address space that is decoded by the VXIpc 800/700 for use by the NI-VXI software
DTB	*See* Data Transfer Bus

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.

F

fair requester	a VXIbus device that will not arbitrate for the VXIbus after releasing it until it detects the bus request signal inactive. This ensures that all requesting devices will be granted use of the bus.
frame number	the frame number is the logical address of the lowest-numbered device in the frame. This is usually the controller (logical address 0) or the parent-side extender in the frame.

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
host	the computer where development of your real-time application takes place. In the VxWorks environment, this is where you install Tornado and the BSP. You write and compile code on the host and transfer the code to a target.
Hz	hertz; cycles per second

I

I/O	input/output; the techniques, media, and devices used to achieve communication between machines and users
IEEE	Institute of Electrical and Electronics Engineers
interrupt	a means for a device to request service from another device
interrupt handler	a VMEbus functional module that detects interrupt requests generated by interrupters and responds to those requests by requesting status and identify information

interrupt level	the relative priority at which a device can interrupt
IRQ*	Interrupt Signal

K

KB	kilobytes of memory

L

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

M

master	a functional part of a VME/VXIbus device that initiates data transfers on the backplane. A transfer can be either a read or a write.
MB	megabytes of memory
MBD	Message-Based Device
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

N

NI-VXI	the National Instruments bus interface software for VME/VXIbus systems
Non-Slot 0 device	a device configured for installation in any slot in a VXIbus mainframe other than Slot 0. Installing such a device into Slot 0 can damage the device, the VXIbus backplane, or both.

P

PCI	Peripheral Component Interconnect. The PCI bus is a high-performance 32-bit or 64-bit bus with multiplexed address and data lines.
POSC	Power On Self-Configuration

R

RBD	Register-Based Device
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 the NI-VXI bus interface software. *See* Resource Manager.
Resource Manager	a message-based Commander located at Logical Address 0 that 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 that is accessible to both a client and a server. The memory block operates as a message buffer for communications.
SIMM	Single In-line Memory Module
slave	a functional part of a 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

Slot 0 device	a device configured for installation in Slot 0 of a VXIbus mainframe. This device is unique in the VXIbus system in that it performs the VMEbus System Controller functions, including clock sourcing and arbitration for data transfers across the backplane. Installing such a device into any other slot can damage the device, the VXIbus backplane, or both.
SMB	Sub Miniature Type B connector with a snap coupling for fast connection

T

target	the embedded computer where your real-time code runs. Code is compiled on a host machine and transferred to the target—for example, a VXIpc controller running VxWorks.
Tornado	an integrated development environment for VxWorks, provided by Wind River Systems, the developers of VxWorks.
trigger	either TTL or ECL lines used for intermodule communication
TTL	Transistor-Transistor Logic

U

user window	a region of address space reserved by the VXIpc 800/700 Series 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

victext	VXI Interactive Control Program, a part of the NI-VXI bus interface software package. Used to program VXI devices, and develop and debug VXI application programs.
VME	Versa Module Eurocard or IEEE 1014

VMEbus System Controller	a device configured for installation in Slot 0 of a VXIbus mainframe or Slot 1 of a VMEbus chassis. This device is unique in the VMEbus system in that it performs the VMEbus System Controller functions, including clock sourcing and arbitration for data transfers across the backplane. Installing such a device into any other slot can damage the device, the VMEbus/VXIbus backplane, or both.
VXIbus	VMEbus Extensions for Instrumentation
vxiinit	a program in the NI-VXI bus interface software package that initializes the board interrupts, shared RAM, VXI register configurations, and bus configurations
vxitedit	VXI Resource Editor program, a part of the NI-VXI bus interface software package. Used to configure the system, edit the manufacturer name and ID numbers, edit the model names of VXI and non-VXI devices in the system, as well as the system interrupt configuration information, and display the system configuration information generated by the Resource Manager.

W

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

Index

Symbols

#define statement in NI-VXI, 4-5

Numerics

32-bit registers, accessing, B-4

A

A16 write post, 3-12
A24/A32 write post, 3-8, 3-12
additional compiler information, 4-5
address mapping, Upper/Lower Half Window, 3-5
address space, 3-3
address space and Requested Memory
    VXI/VME-MXI-2 Configuration Editor, 3-12
application development, 4-1
    #define statement in NI-VXI, 4-5
    additional compiler information, 4-5
    compiling C programs, 4-5
        #define statement in NI-VXI, 4-5
    for VxWorks, 1-5
    GPIB, 4-3
    interacting with devices using victext utility, 4-2
    NI-VISA, 1-3
    NI-VXI and NI-VISA example programs, 4-4
    programming with VXI and GPIB, 4-3
    reconfiguring hardware, 4-1
    reference manuals, *xii*
    resetting of devices by power cycling (note), 4-1
    VME support, 4-2
    VXIpc Configuration Editor, 4-1
arbiter timeout, setting
    VXI/VME-MXI-2, 3-16
    VXIpc, 3-10
arbiter type, setting
    VXI/VME-MXI-2, 3-16
    VXIpc, 3-9
arbitration mode
    configuring, 3-7
    interlocked, 3-13, 3-14
    priority, 3-9, 3-16
    round robin, 3-9, 3-16
    timeout, 3-10
    timeout on VXI/VME-MXI-2, 3-16
audio capability, B-5
auto retry
    VXI/VME, 3-15
    VXIbus, 3-7
automatic retry
    MXIbus, 3-17
    VXI slave cycle, 3-8

B

below 1 MB field, 3-11
boot parameters, 2-3
BTO. *See* Bus Timeout (BTO) value
Bus Configuration Editor. *See* VXIpc Bus Configuration Editor
Bus Timeout (BTO) value
    MXIbus, 3-18
    VXI/VME, 3-15
    VXIbus, 3-7
byte swapping, Upper/Lower Half Window field, 3-4

C

C programs
compiling, 4-5
CLK10 signal, MXIbus
questions about, B-4
setting, 3-18
common questions, B-1
compiling C programs, 4-5
#define statement in NI-VXI, 4-5
configuration, 4-1
checking BIOS setup parameters, B-6
common questions, B-1
hardware installation, 2-1
Resman, 4-1
setting up VXIpc controller with VxWorks, 2-2
system setup, B-1
testing system, B-3
using default settings, 2-1
VME devices, 4-2, B-2
VXI/VME-MXI-2 Configuration Editor, 3-11
VXIpc Configuration Editor, 4-1
configuration editor
address space, 3-3
Logical Address Configuration Editor, 3-3
VXI shared RAM size, 3-4
configuration EEPROM error message, B-4
configuration settings, VXIpc, B-1
loading drivers and utilities, B-1
updating current configuration, B-2
configuration utility
load configuration from file, 3-2
record configuration to file, 3-2
revert to current configuration, 3-2
running the vxitedit configuration utility, 3-1
VXIpc Configuration Editor, 3-1
contacting National Instruments, C-1
controllers. *See* System Controller; VXIpc controller; VXIpc embedded controller for VxWorks
conventions used in the manual, *xii*
customer
education, C-1
professional services, C-1
technical support, C-1

D

data transfer
transfer limit, 3-18
default settings, A-1
default hardware settings, 2-1
Ethernet driver default settings for VxWorks (table), 2-3
VXI/VME-MXI-2, A-1
VXIpc controller
Device Configuration Editor (table), A-4
Logical Address Configuration Editor (table), A-3
VXIpc 700 Series
hardware default settings (table), A-3
VXIpc 800 Series
hardware default settings (table), A-2
VXIpc Bus Configuration Editor (table), A-4
developing applications. *See* application development
Device Configuration Editor, 3-5
See also VXIpc Configuration Editor
default settings (table), A-4
device interaction using victext utility, 4-2
device type, setting, 3-3
diagnostic resources, C-1
directories pre-installed on hard drive, 1-4

documentation
conventions used in manual, *xii*
how to use manual set, *xi*
online library, C-1
related documentation, *xiii*
DRAM, installing (caution), 3-12
driver window. *See* user window and driver window configuration
drivers
instrument, C-1
software, C-1

E

EEPROM
enable, Ethernet serial, A-2
error message, B-4
Ethernet driver default settings for VxWorks (table), 2-3
example code, C-1

F

FAILED LED, B-4
fair requester
MXI fair requester, 3-19
VXI fair requester, 3-9
VXI/VME fair requester, 3-17
files for NI-VXI, pre-installed on hard drive, 1-4
frequently asked questions, C-1
functions, B-3

G

getting started, 1-4
hardware description, 1-2
software description, 1-2
what you need, 1-1
GPIB
programming with, 4-3
reference manual, *xii*
software, 1-3

H

handlers for interrupts, selecting number of, 3-6
hardware
basic configuration. *See* default settings
checking BIOS setup parameters, B-6
common questions, B-1
description, 1-2
hardware installation, 2-1
setting up VXIpc controller with VxWorks, 2-2
using default settings, 2-1
VME devices, 4-2
help
professional services, C-1
technical support, C-1
how to use manual set, *xi*
how to use this manual, 1-1

I

installation
hardware installation, 2-1
reinstalling NI-VXI software, 1-6
instrument drivers, C-1
interlocked arbitration mode, 3-13
interrupt handlers, selecting number of, 3-6
interrupt levels, B-6
interrupts for ISA bus boards, B-6
introduction, 1-1
IRQ level (system), selecting, 3-5
ISA bus boards, B-6

K

keyboard connector problems, B-5
KnowledgeBase, C-1

L

LEDs on front panel, B-4
load configuration from file, 3-2
logical address configuration
  VXI/VME-MXI-2, 3-11
  VXIpc, 3-3
Logical Address Configuration Editor, 3-3
  See also VXIpc logical address configuration editor
  default settings (table), A-3
  resource manager delay, 3-5
  Upper/Lower Half Window address mapping, 3-5
Lower Half window mapping. *See* address mapping

M

memory
  See also VXI Shared RAM options
  adding RAM, B-5
  address space and Requested Memory fields, 3-12
  DRAM, installing (caution), 3-12
  ISA bus boards or expanded memory manager software, B-6
  user window and driver window configuration, 3-10
monitor (caution), B-5
MXI auto retry, 3-17
MXI bus system controller, 3-17
MXI Bus Timeout Value, setting, 3-18
MXI CLK10 signal, 3-18
MXI fair requester, 3-19

N

National Instruments
  customer education, C-1
  professional services, C-1
  system integration services, C-1
  technical support, C-1
  worldwide offices, C-1
National Instruments software, 1-2, 4-3
NI-488.2
  programming with, 4-3
  software for VxWorks kit, 1-3
NI-VISA
  building applications, 4-5
  definition, 1-3
  example programs, 4-4
  programming with, 4-3
NI-VXI
  building applications, 4-5
  loading driver and utilities, B-1
  programming with, 4-3
  utilities, B-2, B-3
NI-VXI and NI-VISA software
  See also application development
  programming with VXI instruments examples (table), 4-4
NI-VXI bus interface, 1-2
NI-VXI configuration utility, 3-1
NI-VXI software
  See also application development
  common questions, B-1
  components, 1-5
  determining revision of VXIpc controller supported, B-2
  example programs, 4-4
  files pre-installed on hard drive, 1-2
  functions of utilities, B-2
  loading driver, B-1
  notes for VxWorks, 1-4
  overview, 1-2
  reinstalling, 1-6

Non-VXI Device Configuration Editor, 4-3
number of handlers, 3-6
number of interrupters, 3-6

O

onboard DRAM, installing (caution), 3-12
online technical support, C-1

P

parity checking, MXIbus, 3-19
phone technical support, C-1
posting write cycles, 3-12
problems and solutions, B-1
professional services, C-1
programming
  additional compiler information, 4-5
  NI-VXI and NI-VISA examples, 4-4
programming examples, C-1
programming with VXI and GPIB. *See* application development
Protocol Register contents, specifying, 3-7

Q

questions about NI-VXI software, B-1

R

RAM
  See also VXI Shared RAM options
  adding, B-5
Read Protocol response, specifying, 3-7
record configuration to file, 3-2
reinstalling VXI software, 1-6
related documentation, *xiii*
request level, setting
  VME Data Transfer Bus, 3-16
  VXIpc, 3-9
Requested Memory field, 3-12
requirements for getting started, 1-1
resetting of devices by power cycling (note), 4-1
Resman, 2-4, 4-1, 4-2
  definition of, B-2
Resource Manager (Resman), 2-4
  overview, B-2
  performing startup Resource Manager operations, B-3
  running after power cycling (note), 4-1
  setting Resource Manager delay, 3-5
  testing your system, B-3
  when to run, B-2
Resource Manager Display, 4-1
retry
  See also automatic retry
  automatic VXIbus retry protocol, 3-7
  enable auto retry protocol, 3-7, 3-8
  MXI auto retry, 3-17
  RETRY signal, 3-13
  slave cycle, 3-8
  VME64 RETRY, 3-14
  VXI auto retry, 3-17
  VXI/VME auto retry, 3-15
revert to current configuration, 3-2

S

servant area size, setting, 3-6
setup,
  See also hardware
  configure the hardware, 2-1
  install the hardware, 2-1
  set up VXIpc controller with VxWorks, 2-2
  system, B-1
  vxitedit, 2-4
Shared RAM Pool field, VXI Shared Ram, 3-4
shared RAM. *See* VXI Shared RAM options
slave cycle retry, automatic, 3-8

Slot 0 installation considerations
VMEbus System Controller (caution), 3-15
VXIpc controller, B-6
software
See also NI-VXI software; application development
#define statement in NI-VXI, 4-5
compiling C programs, 4-5
default settings for VXI/VME-MXI-2, A-1
description, 1-2
developing for VxWorks, 1-5
files and directories, 1-4
included with VXIpc controller, 1-5
loading, B-1
NI-488.2 software kit, 1-3
NI-VXI and NI-VISA example programs, 4-4
notes for VxWorks, 1-4
pre-installed software, 1-2
programming with VXI and GPIB, 4-3
reference manuals, *xii*
reinstalling NI-VXI software, 1-6
VxWorks host and target software, 1-5
software drivers, C-1
support
technical, C-1
SYSFAIL LED, B-4
System Controller
MXI bus, 3-17
Slot 0 considerations (caution), 3-15
VMEbus, 3-14
system integration services, C-1
system IRQ level, selecting, 3-5
system testing, B-3

T

technical support, C-1
telephone technical support, C-1
testing system, B-3
timeout
arbiter timeout setting, 3-10
arbiter timeout, setting VXI/VME-MXI-2, 3-16
BTO (bus timeout), 3-15
MXI Bus Timeout Value, 3-18
VXI bus timeout, 3-7, 3-8
VXI/VMEbus timeout value, 3-15
training
customer, C-1
transfer limit
MXIbus, 3-18
VXI, 3-8
VXI/VME bus, 3-16
troubleshooting resources, C-1

U

update current configuration, 3-2
Upper/Lower Half Window field
address mapping, 3-5
byte swapping, 3-4
user and driver window configuration
Below 1 MB field, 3-11
Window Base, 3-11
Window Size, 3-10

V

victext utility
definition, B-2
interacting with devices, 4-2
VISA. *See* NI-VISA
VME
device handling, B-2
support, 4-2
VMEbus
system controller, 3-14
VXI auto retry, 3-17
VXI bus timeout, 3-7, 3-8

VXI fair requester, 3-9
VXI Shared RAM options, 3-4
advanced settings, 3-4
memory range (table), 3-4
Upper/Lower Half Window address mapping, 3-5
Upper/Lower Half Window byte swapping, 3-5
VXI Shared RAM size, 3-4
VXI transfer limit, 3-8
VXI/VME auto retry, 3-15
VXI/VME fair requester, 3-17
VXI/VME transfer limit, 3-16
VXI/VMEbus timeout value, 3-15
VXI/VME-MXI-2
software default settings, A-1
VXI/VME-MXI-2 Configuration Editor, 1-3, 3-11, 4-1
address space and Requested Memory, 3-12
logical address, 3-11
VXIpc
classification of controller, 3-3
Configuration Editor, 3-1
device type, setting, 3-3
logical address of controller, 3-3
VXIpc 700 Series
hardware default settings (table), A-3
VXIpc 770/870B Series
hardware default settings (table), A-1
VXIpc 800 Series
audio capability, B-5
hardware default settings (table), A-2
VXIpc 870 Series
hardware default settings (table), A-2
VXIpc Bus Configuration Editor, 3-7
default settings (table), A-4
VXIpc Configuration Editor, 1-3, 3-1, 4-1
VXIpc controller, A-1
classification, 3-3
configuring, B-3
default settings, A-1
definition, 1-2
determining revision of, B-2
monitor, B-5
serial number, B-3
Slot 0, B-6
VXIpc embedded controller for VxWorks, 1-1
VXIpc logical address configuration editor
Upper/Lower Half Window byte swapping, 3-4
VXIpc system
interrupt levels, B-6
vxitedit, 2-4, 3-1, 4-1, A-1
definition, B-2
VxWorks
set up the VXIpc controller with VxWorks, 2-2

W

Web
professional services, C-1
technical support, C-1
window base field, 3-11
window size, 3-10
worldwide technical support, C-1
write posting
A16 write posting, VXI/VME-MXI-2, 3-12
A24/A32 write posting
VXI/VME-MXI-2, 3-12
VXIpc controller, 3-8